

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій»

(тема роботи)

Виконав: студент 2-го курсу групи ІІСТ-22м

(шифр групи)

спеціальності 126 – Інформаційні системи та технології

(шифр та назва спеціальності)

Дмитро МАЛЬОВАНИЙ

(ПІБ студента)

Керівник: к.т.н., доц. каф. АІТ

Ілона БОГАЧ

(науковий ступінь, вчене звання / посада, ПІБ керівника)

« 4 » чрудня 2023 р.

Опонент: к.т.н., доц. каф. КН

Людмила КРИЛИК

(науковий ступінь, вчене звання / посада, ПІБ опонента)

« 7 » чрудня 2023 р.

Допущено до захисту

Завідувач кафедри АІТ

д.т.н., проф. Олег БІСІКАЛО

« 11 » 12 2023 р.

Вінниця ВНТУ – 2023 рік

Vinnitsia national technical university
Faculty of intellectual informational systems and automatics
Department of automation and informational technologies

MASTER'S QUALIFICATION PAPER

to the topic:

«Architectural and functional optimization of program module for probabilistic calculations and gamification for financial industries»


(topic of the paper)

Performed by: student of 2nd course, group 1IST-22m

(group cipher)

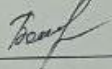
of specialty 126 – Informational systems and
technologies

(cipher and naming of specialty)

Dmytro MALIOVANYI 

(full name of student)

Supervisor: PhD, docent of AIIT department

Ilona BOGACH 

(scientific degree, title / position, full name of supervisor)

« 4 » iprognuz 2023 y.

Opponent: PhD, docent of KN department

Liudmyla KRYLYK

(scientific degree, title / position, full name of supervisor)

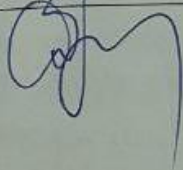
« 7 » iprognuz 2023 y.

Accepted to defense

Head of AIIT department

PhD, prof. Oleh BISIKALO

« 11 » 12 2023 y.



Vinnitsia VNTU – 2023 year

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти _____ II-ий (магістерський)
Галузь знань – _____ 12 – Інформаційні технології
Спеціальність – _____ 126 Інформаційні системи та технології
Освітня програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

Завідувач кафедри АІТ

д.т.н., проф. Олег БІСІКАЛО

« 20 » _____ 09 _____ 2023 р.

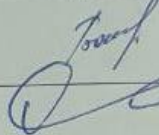

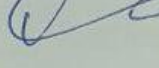
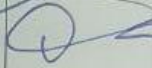
ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мальованому Дмитру Вадимовичу

1. Тема роботи: Architectural and functional optimization of program module for probabilistic calculations and gamification for financial industries
Керівник роботи: к.т.н., доцент каф. АІТ Ілона БОГАЧ
Затверджені наказом ВНТУ від «18» 09 2023 року №247.
2. Строк подання студентом роботи до «20» листопада 2023 року.
3. Вихідні дані до роботи: операційна система Ubuntu 20.04; мова програмування Java, база даних PostgreSQL; середовище AWS Cloud, тип розгортання Fargate deployment; 8 ГБ оперативної пам'яті; 100 МБ місця на запам'ятовуючому пристрої; багатоядерний процесор з базовою тактовою частотою до 4 ГГц і не більше 12 ядер.
4. Зміст текстової частини: вступ; аналіз предметної області та системи, що слід оптимізувати; вибір інструментів розробки; дизайн архітектури та імплементація компонентів системи; аналіз та оптимізація системи; економічне обґрунтування; висновки; список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: діаграма класів модуля генерації випадкових чисел; функціональна діаграма модуля обрахунків; діаграма класів модуля обрахунків; діаграма залежностей бази даних.

Консультанти розділів магістерської кваліфікаційної роботи


Розділ	Прізвище, ім'я та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ілона БОГАЧ, к.т.н., доцент каф. АІТ		
5	Володимир КОЗЛОВСЬКИЙ, к.е.н., доцент каф. ЕПтаВМ		

6. Дата видачі завдання «20» 09 2023 р.

КАЛЕНДАРНИЙ ПЛАН


№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та системи, що слід оптимізувати	20.09.23 - 02.10.23	виконано
2	Вибір інструментів розробки	02.10.23 - 10.10.23	виконано
3	Дизайн архітектури та імплементація компонентів системи	11.10.23 - 01.11.23	виконано
4	Аналіз та оптимізація системи	02.11.23 - 13.11.23	виконано
5	Економічний розділ	15.11.23 - 17.11.23	виконано
6	Оформлення пояснювальної записки і графічного матеріалу	17.11.23 - 20.11.23	виконано
7	Попередній захист роботи	21.11.23	виконано
8	Остаточний захист роботи	11.12.23 - 22.12.23	виконано

Студент


(підпис)

Дмитро МАЛЬОВАНІЙ

Керівник роботи


(підпис)

Ілона БОГАЧ

АНОТАЦІЯ

УДК 004.42 + 519.2

Дмитро МАЛЬОВАНИЙ. Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 109 с.

На англ. мові. Бібліогр.: 33 назви; рис.: 27; табл.: 6.

У роботі розроблено архітектуру та виконано програмну реалізацію модуля обчислень та системи взаємодії для гейміфікації комерційних проєктів, зокрема закладів торгівлі, а також модель оптимізації мережевої архітектури та програмної реалізації модуля. Робота містить математичне та алгоритмічне обґрунтування принципів роботи системи.

Ключові слова: теорія ймовірностей, гейміфікація, система прийняття рішень, архітектура програмного забезпечення.

ABSTRACT

Dmytro MALIOVANYI. Architectural and functional optimization of program module for probabilistic calculations and gamification for financial industries. Master's qualification paper of a specialty 126 – Informational systems and technologies, curricula – Informational technologies of data and image analysis. Vinnytsia: VNTU, 2023. 109 p.

In English language. Bibliography: 33 titles; fig.: 27; tabl.: 6.

In the work, an architecture of calculations' module and interaction system for commercial projects, trade establishments in particular, have been designed and implemented; also, a network optimization architecture and a codebase optimization has been performed. The paper contains mathematical and algorithmic reasoning for of system working principles.

Keywords: probability theory, gamification, decision support system, software architecture.

TABLE OF CONTENTS

INTRODUCTION	4
1 GENERAL CHARACTERISTIC OF THE PROBLEM AND A SYSTEM TO OPTIMIZE	7
1.1 Basic concepts and approaches	7
1.2 Description of target system functionality	9
1.3 Description of target system structure, its pros and cons	10
1.4 Review of analogues	12
1.4.1 BlueRibbon Software	13
1.4.2 SoftSwiss Software	13
1.5 Conclusions to Section 1	14
2 REVIEW OF TOOLS AND APPROACHES.....	16
2.1 Programming language of choice	16
2.2 IDE of choice	18
2.3 Version control system of choice.....	19
2.4 Conclusions to Section 2.....	20
3 MODULES' DESIGN AND IMPLEMENTATION	21
3.1 Design of PRNG module	21
3.1.1 Defining legal & technical requirements	21
3.1.2 Architecture design.....	22
3.2 Design of decision support module.....	23
3.2.1 Deriving mathematical component for volatility-based entropy	24
3.2.2 Deriving mathematical component for impact of the input parameters ...	30
3.2.3 Architecture choices	40
3.3 Implementation of modules	41
3.3.1 DSS implementation	41
3.3.2 PRNG module implementation	45
3.3.3 PRNG module certification.....	47
3.3.4 DSS module integration with PRNG	48

3.3.5	Statistic collection module implementation	49
3.3.6	Containerization support implementation	51
3.4	Conclusions to Section 3	53
4	BOTTLENECKS OPTIMIZATION AND ARCHITECTURE AMENDMENTS .	55
4.1	Subnet traffic optimization problem	55
4.2	Database interaction optimization problem	57
4.3	Load balancing optimization problem	59
4.4	Calculations optimization consideration.....	61
4.4.1	Probability function intermediary calculations optimization.....	61
4.4.2	Dependency inversion of calculations and participant requests	62
4.5	Conclusions to Section 4.....	62
5.	THE ECONOMIC SECTION.....	64
5.1	Technological audit of the developed the retail gamification system	64
5.2	Calculation of the expenses incurred in the development of the retail gamification system.....	69
5.3	Calculation of the economic effect from the potential commercialization of the development	74
	CONCLUSIONS	82
	LIST OF REFERENCED LITERATURE.....	85
	APPENDICES.....	88
	Appendix A (mandatory) Technical task.....	89
	Appendix B (mandatory) Graphical section	92
	Appendix C (mandatory) Code listing	98
	Appendix D Act of incorporation	108
	Appendix E (mandatory) Plagiarism check protocol.....	109

INTRODUCTION

Relevance of the problem. Nowadays humanity is striving towards delegating different aspects and complications of life to the automatons as much as possible. This also implies automating different complex systems and even making decisions. Speaking of that, there are already areas where computers' fraction is overwhelming the human one in aspect of making decisions. Such areas are generally sharing a couple of common characteristics: they are simple enough to be defined for solution which can be done in a set number of operations; they are tedious and time-consuming enough that automating such processes is promising in terms of saving time and money. Most of the implementations of decision-making systems and modules also have one thing in common: they heavily rely on a context and historical data, e.g., a financial decision-making system relies on a huge amount of historical data, and may outperform human operating in the same area because of faster computational rate. [1]

But sometimes, there's a need, due to limitations either specific to area or related to law, which define a hard requirement that such decision-making system should perform in stateless systems, while keeping general fashion in making decisions same as for stateful systems. This raises a problem which needs to be solved. [2]

Also, when talking specifically regarding financial and gambling industries, which are heavily governed by law and have strict definitions and limitations. This means there are systems which require the feature of making decisions based on a true entropy, to make games progress fairly and yet to provide predictable general fashion of the data as a number of decisions rapidly increasing. There is a set of limitations for such components, which vary slightly in each country where a gambling law applies, but generally such regulations include such decision-making component being separated into a random number generator (hereinafter referred to as the RNG), and a decision-making system itself. Both components are obliged to be designed in a manner to withstand high load, provide failsafe mechanism and operate in stateless environment. [3] Both components then must be tested and certified by an assessed audit-conducting body, which will undertake a full audit of the component and it's

functioning. Another complication is both above-mentioned components must be otherwise stateless (excluding storing the data produced) [4]. Development process of such system faces a problem to define an efficient and performance-cheap way to produce decisions under high load, with following the general distribution while neither interacting with it nor gathering any other data about it except the initial parameters.

The purpose of the work is to improve a performance and optimize resource utilization of a stateless decision-making module, to adapt it to better suit applications in stateless systems, work under high load and providing a satisfactory entropy degree while following general distribution tendency with the rule of big numbers. Since conventional statistical formulas rely on the distribution which is to be analyzed, and do not provide required levels of entropy out of the box, another approach needs to be discovered and applied, efficiently providing the product system with all the required features.

The following problems should be solved to achieve the goal:

- 1) Conduct an analysis of existing analogues and techniques applied to achieve the said goal.
- 2) Review the tools which are to be used in the development process.
- 3) Implement a random number generator module compliant with restrictions and undergo certification of the components.
- 4) Define a mathematical solution to the problem for providing convergent entropy-based decision-making generator and implement it programmatically.
- 5) Conduct analysis and perform system & architecture optimization.
- 6) Check the performance & resource utilization gains.

The object of work is a process of gamification of retail and other commercial industries via application of stateless decision support system and an analysis of optimization process of existing systems.

The subject of work are instruments and approaches to building decision support systems as well as methodologies of software architecture and codebase analysis and optimization.

Research methods applied in this work include, but are not limited to: analysis, modelling, classification, generalization, observation, prognostication, experiment, pragmatic model of scientific research etc.

Scientific novelty is granted by implementing of a new event dependency chain for a decision support system, which, in contrast to existing ones, leads to structure improvements and is characterized by more than 70% CPU load reduction for specified system, as well as reduction in average response awaiting time for an end user by 50% on average, and in implementing a tailored loadbalancer which takes into account custom logic and system-specific metrics for an optimal distribution of decision support system modules among physical instances, which would allow to save up to 20% of resources in cases when more than 1 game is being launched.

Practical value of the work lays in providing a ready-to-deploy system, compliant with law-enforced and domain-specific restrictions[2-4]. This system should be certified by a legitimate body. It will allow to provide reliable and endorsed service for supplying decisions to different games and other areas of applications. System will operate based on the random numbers generated by a cryptographically strong PRNG implementation [5-6]. Specified system should be optimized according to found defects. Improvement of bottlenecks' handling and performance and resource usage optimization should be justified and quantified.

Approbation and publications of the work results. Preliminary results of this paper has been presented on LI Scientifically-technical conference of the units of the intellectual informational technologies and automatization department, VNTU, and published in the form of abstracts of the report [7]. The part of the codebase has been copyrighted under name «server-metric-processor» №110271 from December 13, 2021; a DSS module codebase has been accepted for a copyright procedure at October 2023. Also, results of the work were incorporated by the Pragmat Tech Ltd. (Appendix D).

1 GENERAL CHARACTERISTIC OF THE PROBLEM AND A SYSTEM TO OPTIMIZE

1.1 Basic concepts and approaches

Decision making system generally means any kind of a computational system able to make decisions based on the input. They can be implemented either by an artificial intelligence or by a strictly predefined algorithm. Such systems very often heavily rely on the data, and very narrowly specialized. The two most spread subdivisions of these systems are expert systems and decision support systems [8-9].

Expert systems are general-purpose services which are meant to free people from not very expertise-demanding yet time-consuming tasks, e.g., answering questions of patients. In most cases symptoms are typical, and can be answered by an automaton if provided with sufficient knowledge base, which gives an opportunity for skilled doctors to focus on more tough cases, where physical involvement is required. On the current step of development, although providing very good and promising results and being widely integrated in many areas, these systems do face a set of limitations and drawbacks. First of all, such systems require a knowledge base, which usually requires huge amounts of data and obviously, a lot of effort to collect and validate it. Such systems also give a superficial answer, they do not understand a case in a way human does, which means such systems sometimes may give the most inefficient or even irrelevant solution [10].

Decision support systems, abbreviated as DSS, are meant for helping out in making financial solutions, often applied to stock markets or banking industry. These systems may or may be complex and do work from just collecting data to building annual financial strategies. They also may or may not involve human in the process, so some of these systems are not fully automatic. Another narrow application of subclass of these systems are in gambling or another area, where both individual entropy and general convergency are required [11]. In some cases, in gambling and other applicable areas, limitations for systems to be as independent and stateless as possible are applied,

especially in countries and regions where such domains are strictly regulated by the law. This raises a problem to develop a standalone-able system, viable for application in stateless environment while meeting the business needs. Such systems also often require certification or kind of endorsement, both making the company eligible on the market and giving the customers an assurance that the process is fair and they aren't fooled. All the big gambling companies in countries with law-regulated gambling constantly undergo licensing, and present the result of it to the world, making them more attractive to customers [12].

Also, approaches of implementation of decision-making systems vary. Data science is applied in areas, where either some human-like perception is required, or the area of applications is very wide to be set in a defined list of rules, or a general approach to solution is so vague that it is hard or impossible to be formalized from the human point of view. Another approach are strictly defined algorithms. They outperform data science in most of the times in both performance and accuracy of the decision, but have two significant drawbacks: firstly, they are very narrow-specialized and unable to “learn” to make better decisions without human interference in the algorithm implementation, and secondly, they cannot be applied to every aspect where a decision-making system is required, making data science approach the only viable solution for such cases.

When considering DSS for entropy-based decisions for gambling, the formalized set of rules seems to be a more suitable solution. But when talking about stateless system, it is unclear how to supply convergence to general distribution (which is inaccessible from a stateless system) yet keep entropy of individual decisions. Machine learning can provide most straightforward solution to the problem, but in stateless system, it might be not that “random”, because once the system is trained and put into the workflow, it shouldn't be able to store previously processed data, raising hazard of failing the randomness certification. So, there is a need to find a way to derive a mathematical approach which would allow to implement an algorithm which will perform well in stateless systems.

1.2 Description of target system functionality

System for gamification of financial industries and sales optimization (hereinafter referred to as the gamification system) has providing services on a business-to-business model which in serve consumers in any fashion on paid basis as its aim. Key principle of functioning system's work

The principle of the system's operation lies in introducing material and non-material incentives for end users of the business system that utilizes gamification services (hereinafter referred to as the client) to engage in the use and/or purchase on the client's platform.

The implementation of such incentives is carried out by placing an additional mechanism on target goods or services, which will allow all users or buyers of the aforementioned goods or services to participate in a draw for material or non-material rewards. The prize draw has a fundamental similarity to a lottery system but has several key differences, as well as much greater configuration flexibility to satisfy even very specific client's requirements.

Material rewards can take the form of cash sums, physical gifts, electronic objects that are subjects of licenses for commercial distribution, tokens for using commercial systems (e.g., such as Dall-E tokens), and so on. Non-material rewards can be presented as verbal wishes, predictions, and so forth. The client also has the opportunity to integrate their own type of incentive into the system.

An ability to create, configure and manage aforementioned games should be provided. It is suggested to implement it in the form of the Web interface with ability to create new and edit existing games, as well as monitor their state, progress and other analytical parameters related to target metrics of the customer.

1.3 Description of target system structure, its pros and cons

Gamification system has a row of components and layers. The following ones should be highlighted as functionally significant:

- 1) frontend – responsible for visual representation of functions on the client’s web resource in the form of addon/widget, or on an own web resource;
- 2) segmentation handler (backend) – responsible for events’ probability definition and automatical distribution of lottery ticket applications, filtering appropriate end user segments in case of tag system utilization;
- 3) auth service – responsible for providing external API endpoints, user pool management, notification subscription and propagation management; also provides authentication, filtering and API access levels segregation;
- 4) jackpot service – responsible for applications’ handling as well as lottery drawing, creation of new lotteries’ instances (also referred to as jackpots), managing existing jackpots and propagation of win and state notifications to auth and backend layers, storing data in the database and sending the auditorial statistics;
- 5) randomizer – responsible for providing random numbers generated with cryptographically strong algorithm, subject for audits and certification;
- 6) payment service – responsible for handling payment requests in case of monetary rewards; can operate in tandem with both bank accounts and client-specific bonus program system accounts;
- 7) backoffice – responsible for providing the client with web functionality for creating and managing jackpots and other aspects of bonus programs applied to their services.

Thus, any event for participation in the jackpot draw must go through the frontend, be validated on the backend, processed by the jackpot system, stored in the database, and the processing result should be announced to all end users participating in the jackpot draw through the auth service. Notification distribution is implemented using the WebSocket Secure protocol and STOMP (an extension over WSS that provides heartbeat and HTTP handshake functionality). In the case of a winning event,

notifications will also be sent to the frontend through auth, and in the case of a monetary prize, a notification will be sent to the payment system.

For clarity on the operation principle of the described structure, a schematic sequence diagram is provided in Figure 1.1. This diagram also includes the Loadbalancer element, currently fulfilled by the Elastic Load Balancer (a component of the AWS cloud environment). The load is distributed using a round-robin algorithm. The most resource-intensive components are the backend and jackpot, as they must process a significant number of requests and filter them. These components are implemented with scalability in mind and are capable of achieving it.

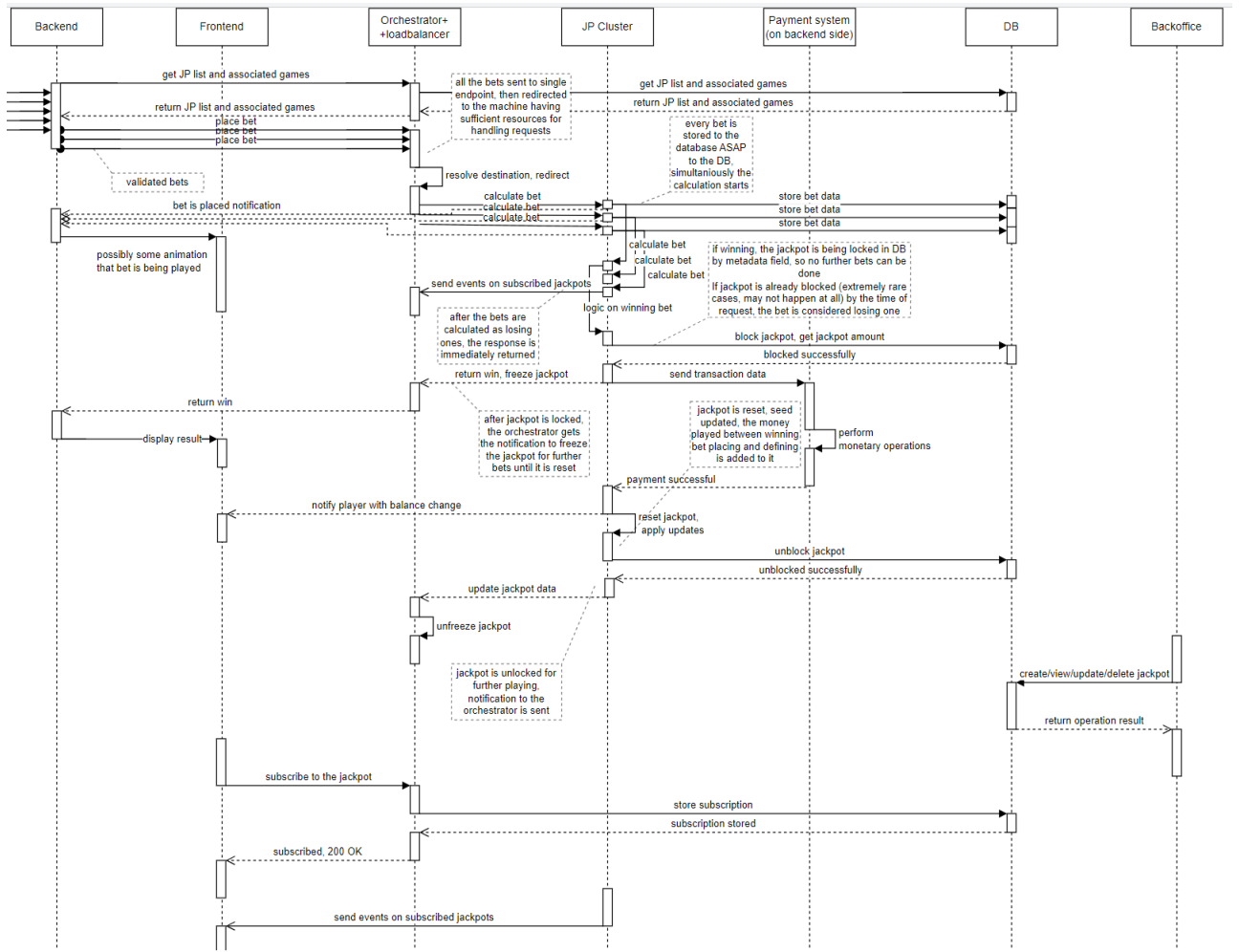


Figure 1.1 – Sequence diagram of the key processes in the system

The advantages of such a system architecture in its current implementation include the speed of processing and notification of results without unnecessary delays,

a reactive approach to event processing, high intensity, efficiency, and optimization tailored to the specific tasks of performing mathematical and logical operations according to formulas for determining draw results. The architecture also demonstrates flexibility for integration with client systems and scalability.

Despite the aforementioned benefits, a thorough analysis has detected a presence of a number of bottlenecks and places which could potentially create obstructions for system's operations, in peak situations in particular:

- traffic speed inside the cloud's subnet, in particular between pairs jackpot-randomizer and backend-jackpot;
- jackpot layer's database session pool size;
- request distribution using round robin algorithm and AWS-provided orchestrator implementation do not suffice in taking into account system's specifics, thus is suboptimal and can potentially be a source of an unjustified extra load.

1.4 Review of analogues

Due to complexity of the task and high demand from industry, a number of companies propose such kind of services. Most of them are businesses of considerable size, and their business model orients serving big customers, thus not providing a detailed description or pricing, limiting publicly available information to general roster, motivating requesting a personalized consultation instead. In addition to this, to be eligible for aforementioned consultation, a seeker must provide proofs of being licensed gambling company and complies with standards and permits, so far rendering next to impossible getting all the required data for ordinary users. Nonetheless, from available information, the following description of some analogues can be drawn.

1.4.1 BlueRibbon Software

BlueRibbon provides services of certified gaming options, including DSS for making decisions granting enough entropy and general convergency. Their system is certified to comply with all the required limitations, including statelessness of RNG and DSS, and passed all the entropy and other statistical tests, and has a license to operate in EU.

BlueRibbon provides services based on the annual subscription paid per-month, but one cannot subscribe less than a year. Company doesn't provide a clear information about pricing, instead providing a mail to contact sales department. It's not a viable option to contact sales unless one has a registered gambling enterprise. But in general, a pricing is defined based on the size, scale, number of jurisdictions (different countries etc.) and revenue of the company requesting the services.

Since it is a large company, they do not provide a DSS itself, but rather integrate it inside their products – game engines. Such game processing engines are pre-defined, and it is required to integrate with provided API and SDK. Also, if some company would like to implement or change some aspect of a certain game, it would render either impossible or will cost additional money, since BlueRibbon doesn't provide a way to extend functionality, and can only review company's request and implement a particular solution for this company “on demand” [13].

1.4.2 SoftSwiss Software

SoftSwiss is another company providing DSS-based solutions. In analogy to BlueRibbon, pricing policy isn't public and is personalized for individual customers. Services are also provided in the form of game engines with DSS integration. As an advantage, SoftSwiss provides cryptocurrency support, significantly extending the number of potential clients. In addition, company provides a customer with ability to define a desired functionality and implements it on demand, thus making their solutions

much more flexible and tailored to the client. However, implementation is still up to the SoftSwiss discretion, so it still has ownership over it and incurs additional billing in invoices for such a services. As an additional advantage, the service infrastructure is cloud-based, which in general case means more confidence in stable operations.

As of drawbacks, the pricing information is also not available without direct online consultation, and for that, an EU-licensed business is required. SoftSwiss also doesn't provide neither DSS nor PRNG 'as-is', in contrast making them available only as a part of bigger products.. It's also worth mentioning that company provides its own payment system as well as account system, integrated directly into game engines, meaning impossibility to use custom-provided systems and also routine and risk of transferring all the customer's users' data into an external system. Not all businesses are keen to do that, rendering this step an additional complications and also creating a hard dependency on external provider for vital parts of the system [14]. This solution looks lucrative for smaller businesses, but big ones are likely to be very reluctant on sharing users' data, considering this an unreasonable exposure of law-protected data.

1.5 Conclusions to Section 1

An extensive analysis and a review of concepts and approaches used in a development process of decision-making systems has been performed. Two main ways classes: expert systems and decision support systems were characterized, along with two main approaches to development. One mentioned use of data science, while another one relied on use of strictly defined algorithm, both approaches having their advantages and drawbacks. Specifics of the gaming & gambling domain, requiring DSS to comply with restrictions and run inside stateless systems, has been reviewed.

Along this, an analysis and review of existing solutions has been performed, which defined BlueRibbon Software and SoftSwiss as two leading enterprises providing services to the market. After a review, it has been summarized that while these companies provide fine-grained services, are certified that their systems comply

with restrictions and meet the business requirements for almost every application in the domain and licensed to be eligible to be run under jurisdictions of most countries regulating the gambling, such solutions also apply a set of limitations. Almost all gambling DSS service providers do not provide DSS itself, instead offering a wide specter of games and game engines which require to be integrated into service receiver's system, limiting them to the provided functionality, and increasing the cost of DSS monetization.

It renders a need for a new, customizable DSS along with RNG to be implemented and certified, to satisfy needs of many gambling companies which seek a solution, which would give them an opportunity to buy the bare DSS service and adopt it to their specific requirements without needing to make extra expenses and requests to DSS provided to extend desired functionality.

2 REVIEW OF TOOLS AND APPROACHES

2.1 Programming language of choice

As a programming language of choice, Java has been taken. Such decision is conditioned by several factors. Java is an object-oriented programming language with strong typing. Its syntax is similar to C++, but in contradiction to C++, Java provides common root of inheritance (Object), applies more type safety, provides out-of-box memory allocation management called Garbage Collection (GC for short), provides cross-platform support thanks to use of Java Virtual Machine, more human-readable and significantly less verbose. In addition to it, Java provides error handling and very good backward compatibility [15-16].

Java is designated for building enterprise solutions which are meant to reliably work in any kind of situations, withstand high load and provide sufficient performance. Java pays attention to early error detection and excluding error-prone cases. Java compiler tends to detect a vast majority of possible errors, many of which are detected by other languages' compilers only in runtime. Also, there is a new virtual machine for Java, called GraalVM, which allows compiling Java code not into traditional bytecode, but directly into a machine executable, making it platform-dependent, but reaching exact same performance as C-written applications. Although it worth to note, this requires closed-world assumption and won't allow any dynamic class loading, thus excluding reflection entirely. Regarding generics, they are not a problem, since GraalVM compiler is smart enough to find all usages of generic methods and classes and generate hard typed derivative class for each [17].

To summarize Java advantages:

- 1) Good human readability. The code is kept simple yet efficient. It lowers entry threshold for newcomers trying to code in Java, and is comprehensible even for people not familiar with programming at all.

2) Object-oriented approach allows to abstract from hardware-specific details and focus totally of business model elements while development process. This adds to the final product's quality and simplifies the process.

3) Automatic memory management, including Garbage Collector, lifts off the developer needs to define boundaries of objects' existence and lifespan. Virtual machine will define and clean sections of memory which are out of scope.

4) Strong typing adds towards reliability and performance. It allows developer to pre-define objects, models and interfaces which would take part in interaction. This applies a requirement to developer to pay more attention in cases of serialization/deserialization or marshalling/unmarshalling processes, but with correct implementation, it makes environment defined before the launch fully or for the most part (Ahead-of-Time or Just-in-Time compilation respectively).

5) Java is a web-oriented programming language. There is a plethora of a very developed frameworks for convenient development of web applications, which provide most of web part with almost no specific configuration in general case. More specific needs require more configuration to be done, but generally, these frameworks allow to minimize time spent on Web API and endpoints configuration, allowing to focus on business logic.

6) Very good debuggability. Java is a static programming language, thus, along with modern IDEs and built-in tools such as jdb, allows for a great variety of options while debugging, making debugging process much more comprehensible and simple.

7) Facilitated code support. Compared to most other programming languages, Java provides features which make code support much easier. It owes to cross-platform support and backward compatibility, resulting in from little to no changes needed to adapt application to different platform. It also allows building platform-independent and hardware-independent architecture.

8) Backward compatibility. In general case, this implies the code written for the current version of Java will compile into the same bytecode in all further Java Development Kit versions. Java has been considered a world leader in the area of backward

compatibility, where C or Python, in contradiction to it, have differences for each version and environment, making code portation a potentially effort-consuming task.

9) Cross-platform support. Java follows concept “written once – run anywhere”, guaranteeing Java code can be run in an exactly same manner on virtually platform (actually every platform for which there is a Java Virtual Machine, will support it).

After a consideration, it can be concluded that reviewed programming language fully meets development needs and is a very suitable solution for development both RNG and DSS, and will contribute towards their load resistance and reliability, and making web part working very smoothly. Inheritance and hierarchy of classes, along with clean and comprehensible interfaces will help to stick to SOLID principles (Single responsibility, Open-closed, Liskov substitution principle, Interface segregation, Dependency inversion) and make project more flexible, giving a lot of points for future modifications on demand.

Another principles and paradigms, except SOLID, followed while development are: KISS, YAGNI and DRY.

The libraries and frameworks used while the development process include, but are not limited to: Spring Boot, projectlombok, apache-commons-lang3, springdoc-openapi, openfeign, flyway, jackson-databind, Junit 5.

It has been decided to use Apache Maven as a dependency manager and assembly automatization tool, particularly because of declarative approach to dependency definitions and usage of xml-like code style, and good readability.

2.2 IDE of choice

For the convenience of the development process, it has been decided to make use of IntelliJ IDEA IDE (Integrated Development Environment).

The first version of IntelliJ IDEA has been released in the January of 2001 and has rapidly spread among developers, becoming more and more popular. It is often

considered as the first IDE with the wide set of integrated tools for refactoring and debugging, with also integrated tools for Version Control Systems. Design of the environment is designed to increase programmers' productivity [18].

IDE is distributed in the form of Software as a Service model. Subscription offers monthly, annual or 3-years plans. Also, starting from version 9.0, there is a community edition of IntelliJ IDEA with open code. Source code of IntelliJ IDEA is distributed under Apache 2.0 license. Executable binary packages are present for Linux, Mac OS X and Windows. [19]

2.3 Version control system of choice

Among version control systems, it has been decided to choose Git as it is one of the most powerful tools for version tracking.

Git allows not only to securely and conveniently store versioning history of the project; it also provides tools that will help to keep project history clean and comprehensible. Such tools include squashing, also known as interactive rebasing, and cherry-picking. Git also provides all the basic features a VCS should provide, such as versioning, branches, merging, rebasing etc. [20].

One of the most important advantages of Git among the analogues is that it provides out-of-box integrations with cloud-based remote repository storages, such as GitHub and GitLab, both of them being the most famous repositories in the world and being de-facto standard in the domain of programming and cloud-based VCS solutions.

All the Git, GitHub and GitLab provide most of the required functionality for free. Git is an opensource project. GitHub and GitLab are powerful tools, providing not only support of VCS, but services such as CI/CD, teamwork, project spaces, statistic tracking and much more, making them very convenient for the development. The one downside here is that many of these powerful and convenient features require either paid subscription or a tedious configuration, e.g., one can buy a subscription and conveniently configure CI, paying attention only to the logic of CI itself, or can go on

with installing and configuring own CI computational environment, forwarding ports, and other parts of routine, making such approach “cheap yet angry”.

2.4 Conclusions to Section 2

In this section, a review of the technologies to be used during the development process has been performed. The advantages of chosen programming language, frameworks, libraries, IDE, VCS and other tools have been described. To conclude, the following main stack will be used in the development process:

- Java 11;
- Spring Boot;
- Lombok;
- Flyway;
- JPA;
- Apache Maven;
- IntelliJ IDEA;
- Git;
- GitLab.

An extensive analysis showed this stack fully meets development requirement and will contribute towards the final product’s quality.

3 MODULES' DESIGN AND IMPLEMENTATION

3.1 Design of PRNG module

The first step to development of the module is defining its constraints and limitations, as well as expected performance. For this module, in addition to some programmatical limitations, there are also legal compliance requirements, which imply additional restrictions.

3.1.1 Defining legal & technical requirements

After a research, it has been defined that RNG has to comply with a set of restrictions to render viable for gambling industry. Among them, random number generator module must be stateless, must rely on a cryptographically strong core PRNG algorithm and must produce results which should pass a set of tests. Among them are: entropy test, correlation test, shuffledeck entropy test and diehard test. In order to receive a certification, an audit should be conducted by a certified body. For the audit, said body will require the following data sets:

- 3 samples 3 million of data each with numbers for each of the ranges: 0-33;0-36;0-51;0-66;0-99;0-500;0-999;
- 3 samples with 3 million rows with 16 digits each (48 million data in total per sample) in range 0-255;
- 3 samples with 3 million shuffled decks (for decks of 52 cards).

Another requirement is to provide an ability to the auditing party to perform independent tests for gaining such samples independently. During audit, the auditor will perform a full review of the source code and deployment environment, to emulate exact same situation and check against possible hardware issues which might interfere randomness of generated data.

The system should communicate over any TCP-compliant protocol and has to be deployed either on the same physical instance, or inside the subnet of the main system cluster in order to keep latency as low as possible.

The system should be audited and tested on randomness inside its deployment environment in order to assure the same behaviour and eliminate any possibility of impacting the program outcomes.

3.1.2 Architecture design

It has been decided to develop a module in a form of a standalone web application, with the only external API endpoint, allowing to retrieve a random number generator. The system is planned to be designed with an extensive use of interfaces, allowing for abstraction and making possible future substitution of components a very easy process, requiring only to implement said modules and inject them, with no other code being changed.

A core PRNG algorithm implementation chosen for the project is Oracle's SHA1PRNG implementation.

It has been decided to provide out-of-the-box ability for containerization, along with support of AWS ECS (Amazon Web Services Elastic Container System) or AWS Fargate (Amazon Web Services container system with computational resources-as-a-service) deployment.

Such approach will make the module an easily scalable, and will grant a wide set of options for extension.

The service will provide simple header-based authorization with predefined key, allowing for easy integration yet granting some defense against DDoS attacks, which are otherwise virtually impossible, since the system itself is designated to be run inside private subnetworks and not be otherwise accessible.

It has been decided to make service configuration externalized, to provide even better flexibility and making it possible to re-configure some parts without modifying the code.

It has been decided to cover vital functionality, such as number generations and boundaries acceptance, with tests. Junit 5 will be used for testing purposes.

Modules for data samples' generations will be shipped along the main RNG module, including necessary libraries and scripts, to make a convenient use of them.

3.2 Design of decision support module

To be eligible for application in financial and gambling industries, a DSS must comply with a certain set of rules and constraints, which comprise the following:

- system must be stateless whenever related to the process of making decisions;
- system must be able to withstand 10 000 concurrent requests;
- system must make use of certified RNG;
- system must take in account a set of parameters when making a decision;
- system must guarantee an entropy of each individual decision;
- system should store every input and output it processes in order to gather historical data to be sent to legal authorities for review of correct functioning of the system over time;
- system must keep convergence of the total of individual decisions to the general distribution (without having access to said distribution since being stateless).

With all these constraints being defined, it has become clear that the most significant requirement to proceed is to develop a mathematical approach which would allow to solve the problem of supplying decisions which are random individually yet convergent to the general distribution on big scales.

3.2.1 Deriving mathematical component for volatility-based entropy

The problem raised here is how to guarantee totally independent from each other so the play is always fair and in the same time provide a house advantage in the long run, while knowing nothing about previous decisions [21-22]. It is hard to guarantee it if there is no insight on a historical data, so system cannot keep and define current state of a general distribution, whereas general distribution is comprised of total collected historical data. Conventional statistical methods do not provide ready solutions to implement a convergency to a distribution according to the big numbers rule without knowing nothing about one and keep an entropy on a high level simultaneously [23]. One of the parameters characterizing a degree of possibility of deviation of the future data from the historical data distribution used in financial domain is a volatility [24].

Speaking generally, volatility is an index of randomness, describing fluctuation range for time series. It means the more volatility is, the less predictable the future outcome will be. In the current implementation, volatility will be used as an abstract reversed value, instead of traditional percentage representation. Possible volatility values will vary from 1 to 99 inclusive, and will describe maximally possible deviation of result from the expected one [25]. For all the probabilistic calculations, including multiplications, a Bayes rule for probabilities will be used [26].

In general case, volatility of the distribution X can be expressed in the following way:

$$X_{mean} = \sum_1^n x_i/n \quad , \quad (3.1)$$

$$\sigma = \sqrt{(1/(n-1) * \sum_1^n (x_i - X_{mean})^2)}, \quad (3.2)$$

where σ – volatility,

n – number of elements in distribution,

x_i – i-th member of distribution,

X_{mean} – mean of the distribution.

One of the approaches which might work for generating local values satisfying convergence to the general distribution is to generate candidates being potentially eligible to become members of the general distribution. With such approach, it seems it could be possible to keep general tendency towards the target function or historical data distribution, while also granting maximum possible differentiation of individual results.

In case of DSS for gambling, the characteristics one can operate with include winning parameters. These usually include desired average, and desired minimum and maximum wins the system should perform. This should be not a hard limit, but rather a value to which the system should aim.

Let's define the following constraints:

let $X_{mean} = averageWin$, and $minWin < x_i < maxWin$, and n is a number of values to generate for a sample, then:

$$n * \sigma^2 = \sum_1^n (x_i - averageWin)^2 \quad (3.3)$$

Assume there is a simple case for better understanding. The least value of n to make formula meaningful and satisfactory for full explanations is 3:

let $n=3$, then:

$$3 * \sigma^2 = (x_1 - averageWin)^2 + (x_2 - averageWin)^2 + (x_3 - averageWin)^2 \quad (3.4)$$

Then it is required to define, that each value comprising the sample will be generated the following way:

$$x_1 = generateRandom(minWin, maxWin) \quad (3.5)$$

For every next value we must take into account all previously generated values, to ensure the sample will have statistical behaviour and parameters close to the ones of

the general sample. Taking this into account, the formula for x_2 will have the following form:

$$x_2 = generateRandom(minWin, maxWin - x_1) \quad (3.6)$$

It is also required to have a different equation for calculating the last element. Here it is 3rd one, but for $n=10$ it will be the 10th one. It should be calculated as the theoretical remainder after generating all previous numbers in the sample's sequence. The formula will have the following form:

$$x_3 = \sqrt{(3 * \sigma^2 - (x_1 - averageWin)^2 - (x_2 - averageWin)^2)} + averageWin, \quad (3.7)$$

To mention, the following constraint applies for sample of size of 3:

$$(x_1 - averageWin)^2 + (x_2 - averageWin)^2 < 3 * \sigma^2 \quad (3.8)$$

The solution can be transformed the following way:

given the maximum and minimum desired boundaries and average value, it is required to generate a sequence of eligible deviations for each element in the sample. If apply a sequence of generated deviations to the average, a series will be received, where each member will comply with all restrictions to claim a place in the general distribution. In other words, these series will form eligible subset of the general distribution, satisfying all the constraints, yet providing enough randomness by using random number while generation, which means even on big distributions it is unlikely for each individual value to be exact result of using correlated values of the distribution, granting independence and uniqueness for each particular value.

Said deviation can be expressed in the following form:

$$(x_i - averageWin)^2 = \eta_i \quad (3.9)$$

For $n=3$ the following will be fair:

$$3 * \sigma^2 = \eta_1 + \eta_2 + \eta_3, \underbrace{\eta_1 > 0, \eta_2 > 0, \eta_3 > 0}_{(3.10)} \quad (3.10)$$

From (3.5), (3.6), (3.7) and (3.10), the following equations for each of the elements for $n=3$ can be derived:

$$\eta_1 = \text{generateRandom}(0, 3 * \sigma^2) \quad (3.11)$$

$$\eta_2 = \text{generateRandom}(0, 3 * \sigma^2 - \eta_1) \quad (3.12)$$

$$\eta_3 = 3 * \sigma^2 - \eta_1 - \eta_2 \quad (3.13)$$

This allows to generalize formulas to define universal equations for every i^{th} element of the sample where n can take any value on plane of natural numbers. Taking this into account, every η_i except $i=n$ for every $n \in N$ the following is true:

$$\eta_i = \text{generateRandom}\left(0, (3 * \sigma^2 - \sum_1^k \eta_k)\right), \quad (3.14)$$

$$i \in [1, n - 1], k \in [1, i - 1]$$

And for cases $i=n$ the following equation should be used:

$$\eta_n = 3 * \sigma^2 - \sum_1^{(n-1)} \eta_i \quad (3.15)$$

Having a need to define a deviation, a (3.9) can be transformed into the following form:

$$|x_i - \text{averageWin}| = \sqrt{\eta_i}, \quad (3.16)$$

Which gives two roots:

$$x_i - averageWin = \sqrt{\eta_i}; x_i - averageWin = -\sqrt{\eta_i} \quad (3.17)$$

or

$$x_i = averageWin \pm \sqrt{\eta_i} \quad (3.18)$$

From (3.9) and $minWin < x_i < maxWin$, additional constraints can be formulated:

$$0 < \eta_i < (maxWin - averageWin)^2 \quad (3.19)$$

and

$$0 < \eta_i < (minWin - averageWin)^2 \quad (3.20)$$

(3.19) and (3.20) must be used to ensure values are generated in the correct fashion, but a very important note here not to just narrow the possible diversity of solutions to (3.20), a sign of the resulting value should be taken into account (applying (3.18)).

With that, mathematical problem for defining random individually decisions simultaneously convergent to the general distribution with the big numbers rule in stateless systems has been solved. There is a similar approach for applications in an investment area [27], but it doesn't satisfy all the constraints a problem comprises of, because it only operates based on historical data, and outputs pure entropic values, resulting in it cannot guarantee passing condition of sufficiency of each returned element to belong to the general distribution.

Also, there is an approach with creating self-regulated series of numbers [28], but mentioned implementation doesn't take into account a set of constraints which apply to each individual element over the course of changing its ordinal number, or with increasing size of the generated sample. However, this method considers usage of different parameters to be guided by while generating the series. Part of these

parameters, significant to the area of application, such as average and boundaries have been successfully applied in the developed algorithm of series generation.

The result of applying the following set of equations has been extensively tested, with also minor modifications in coefficients and sizes of samples. Resulting graph with impact of volatility over time and convergence to the general distribution is shown on Figure 3.1.

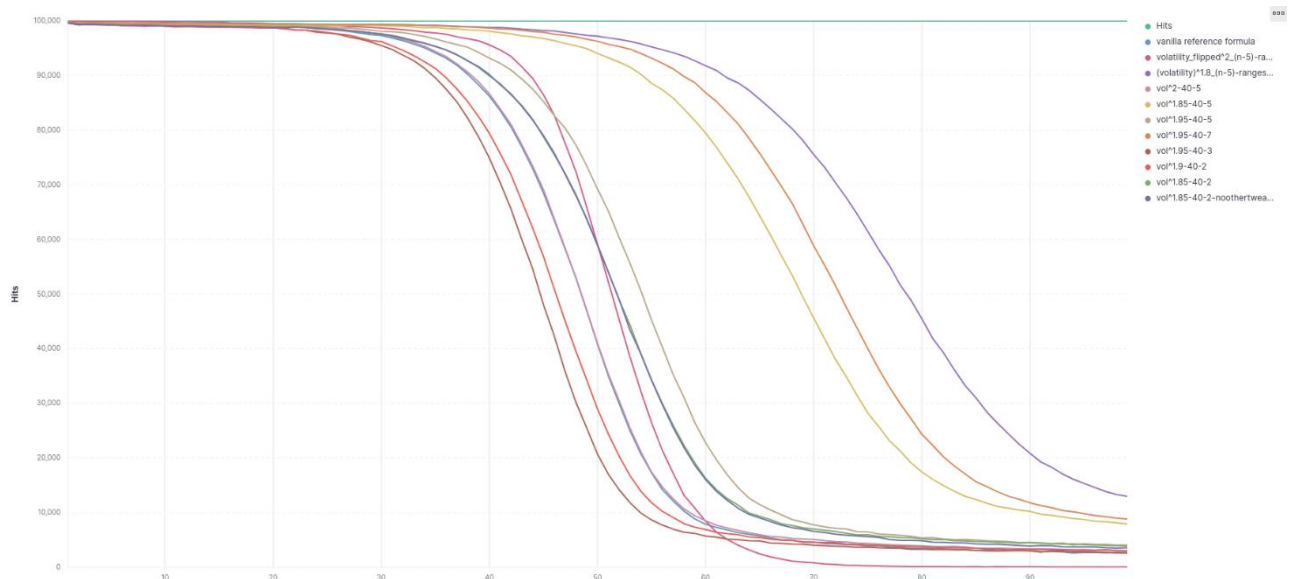


Figure 3.1 – Example of statistical decrease of probability of the positive decision due to increasing value of volatility. Tests are performed for a set of different candidates to the final formula, each set containing 10'000'000 unique generated data

This graph has been plotted using Kibana. Kibana is a very powerful tool which allows to process huge amounts of data with relative ease [29]. Legend to the right provides mathematical notation of each plotted graph, making it easier to differentiate on most suitable solution for a hands-on application. All of these formulae are intended to achieve a smooth non-linear sigmoidal inverse dependency between volatility in range of 1 to 99 both inclusive, and a probability to grant a positive decision in case of previous submodule qualified an event for a positive decision.

3.2.2 Deriving mathematical component for impact of the input parameters

After a solution to provide individually independent yet generally convergent sequence of decisions has been derived, there is a need to define a way to apply an impact of the input parameters. A detailed description of the problem is explained below.

First thing to do is to define is a way a final impact should look like. Having the following constraints:

- minWin – minimal amount of winning value possible;
- maxWin – maximal amount of winning value possible;
- avgWin – average amount of winning value. The overall winning value mean of all the winning events should approach to this value.

- the average wins should lay inside bounds of $avgWin \pm avgWin * 15\%/100\%$

Main concerns here include:

- Mathematical expectation should be followed without context, i.e., it is not allowed to store the context, and the formula should be implemented to approach specified mathematical expectation programmatically.

- The probability of impact on positive decision result:
 - should be lowest around minimal positive decision value;
 - should positively and non-linearly increase inside the average positive decision bounds;
 - should have the peak probability in the value of average win;
 - whilst should not be high enough to make other areas of probabilities non-considerable;
 - should gradually decrease after 15% past average win;
 - should rapidly increase around maximal win (some predefined small range related to the whole range of possible values just before maximal win should exponentially increase the probability of positive decision);

◦ the positive decision probability should be affected by volatility (this point here could be redundant as the volatility already defines a formula which affects positive decision probability in its own fashion) The point of concern here is that it is not defined whether volatility will be passed into or defined based on the win capping & conditions;

- From the above, the resulting ruleset is very likely to consist of at least two formulas, possibly more. The formulas will need to describe following fashions:

- increase from minWin to $(1-0.15)*\text{avgWin}$;
- increase, peak and decrease from $(1-0.15)*\text{avgWin}$ to $(1+0.15)*\text{avgWin}$;
- decrease from $(1+0.15)*\text{avgWin}$ to $(1-x)*\text{maxWin}$;
- rapid increase from $(1-x)*\text{maxWin}$ to maxWin ;

These points raise a set of issues which will be faced while development and must be resolved:

1) What to do if either maxWin or minWin are in range of avgWin ? To use one of their formulas (and which?) or define specific ones for such cases?

2) There is a contradiction: how to assure the positive solution will happen before or on the maximal allowed value, but never exceed it, along with avoiding reaching 100% probabilities on any numbers from range, since it might add unfair advantage and be used to predict a certain positive decision if properly analyzed by players or insider information is gained?

3) If the positive solution didn't occur on the whole range, what to do? How to handle situations if the accumulated amount actually exceeding maxWin , and hence, allowed amount?

The principal sketch of ranges of input values impact is depicted on Figure 3.2. It describes an intended distribution of probability over the game progression, and is based on breakpoints derived from the initial configuration, making use of parameters such as minimal expected win, maximal expected win and average expected win, basically producing a graph with similarities to a bell curve, but adjusting areas where it is intended by business and/or legal requirements.

For some parameters to take place and effectively affect resulting graph in terms of computer calculations, we'll add constraints that will ensure a number's calculation precision can be treated without considerable loss of accuracy:

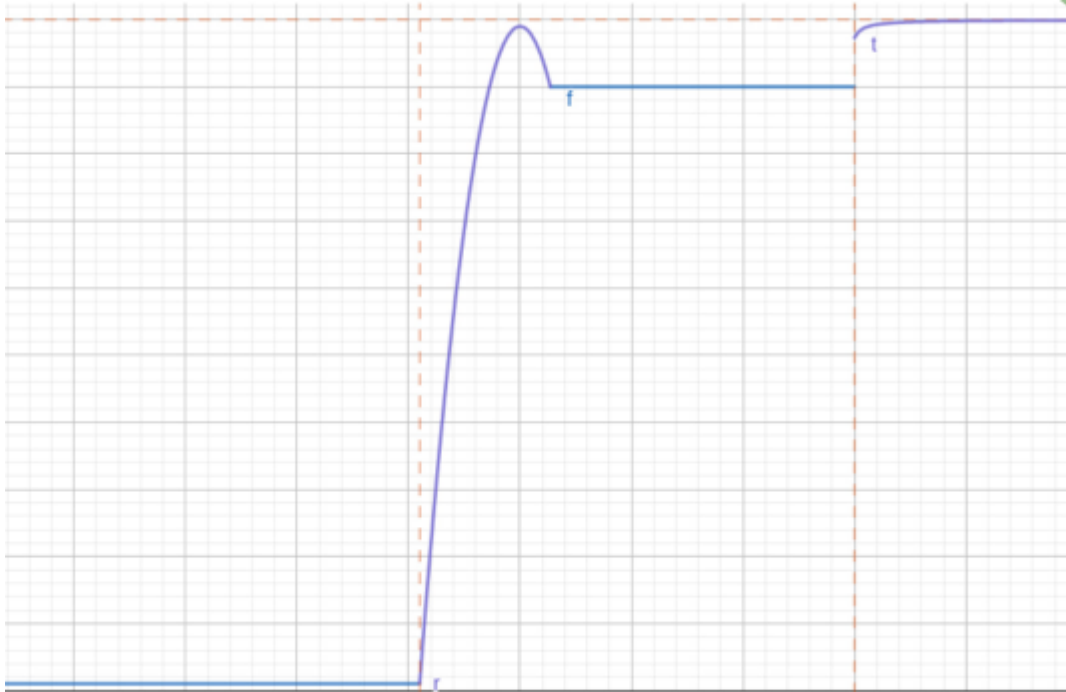


Figure 3.2 – Sketch of the planned impact of input on the decision

1) minWin, maxWin and avgWin are all positive integer numbers (natural numbers), or positive numbers with floating point (rational numbers).

2) The minWin will be treated a hard cap, the value of minWin will have the lowest probability to happen, the values less than minWin will have 0% probability to happen.

3) The maxWin will be treated a soft cap, i.e., the probability from start of maxWin's neighbourhood to maxWin will drastically increase almost to 100%, but less. Then the function will approach 100% probability, but will never hit 100% probability.

4) If minWin is inside the avgWin neighbourhood with size of 15% of range, the function for avgWin will be used, the minWin function and gradual ascent function will be neglected, but the values before minWin, even if inside avgWin-15% range, will be treated as zero probability.

5) If maxWin is inside the avgWin neighbourhood with size of 15% of range, the function of gradual descent from avgWin+15% to maxWin's neighbourhood will be omitted, and the function of exponential ascent to almost 100% probability and then infinite approach to 100% with no actual reaching.

6) The neighbourhood for minWin and MaxWin will be defined as 1% of total range (so if the range is 1000 then the neighbourhood will be defined as 10), but should be never less than 2, i.e., if the range is 100, the neighbourhood will be 2, not 1.

7) The neighbourhood of avgWin will be calculated as 15% from avgWin, meaning if avgWin is 100, then the neighbourhood will be from 85 to 115 inclusively.

The mathematical expectation will look in the following manner:

$$E(x) = \sum_1^n x_i p_i, i \in [1; n], \quad (3.21)$$

where x_i is a partial case of all positive values of x , $x \in Q; x \geq 0$;

p_i is a probability of x_i ; $p_i \in [0; 1]$.

It can also be represented as

$$E(x) = \sum_1^n x_i F(x_i), \quad (3.22)$$

where F is a probability function defined on range $[x_1; x_n]$

In a general case, the mathematical expectation for equal probabilities of all values, will have the form of

$$E(x) = \frac{1}{n} \sum_1^n x_i, \quad (3.23)$$

actually, being the mean value of all the individual values.

But for the formula to work correctly, it cannot be reduced to mean. It is required to define the following functions:

- $f(x)$ – is a probability function for range $[minWin; minWin+1%*range]$;

- $g(x)$ – is a probability function for range $(minWin + 1%*range; avgWin-15\%);$
- $h(x)$ – is a probability function for range $[avgWin-15%; avgWin+15\%];$
- $i(x)$ – is a probability function for range $(avgWin+15%; maxWin-1%*range);$
- $j(x)$ – is a probability function for range $[maxWin-1%*range; maxWin];$
- $k(x)$ – is a probability function for range $(maxWin; +\infty).$

Assuming these definitions, the final mathematical expectation function can be represented for this case (assuming minimal step is 0.01):

$$\begin{aligned}
 E(x) = & \sum_{minWin}^{minWin+0.01*range} x_i f(x_i) + \sum_{minWin+0.01*range+0.01}^{(1-0.15)avgWin-0.01} x_i g(x_i) + \\
 & + \sum_{(1-0.15)avgWin}^{(1+0.15)avgWin} x_i h(x_i) + \sum_{(1+0.15)avgWin+0.01}^{maxWin-0.01*range-0.01} x_i i(x_i) + \\
 & + \sum_{maxWin-0.01*range}^{maxWin} x_i j(x_i) + \sum_{maxWin+0.01}^{+\infty} x_i k(x_i);
 \end{aligned}
 \tag{3.24}$$

The next step is to define each of the probability functions listed above.

To do this, we need to also define the border probabilities each of the ranges would be allowed to have.

That is, for $f(x)$ the probability would be set from 0% to 10%;

- for $g(x)$ the probability would go from 10% to 30%;
- for $h(x)$ the probability would go from 30% to 85%, then again to 30%;
- for $i(x)$ the probability would go from 30% to 10%;
- for $j(x)$ the probability would go from 10% to 95%;
- for $k(x)$ the probability would go from 95% to 99.99%.

After the measures are defined, the formulas can be chosen. Important to note, there is no rule or law which can be used to define which formula to use. The formulas will be chosen empirically.

For empiric tests to be actually graphically represented, let the parameters have been set to the following values:

$$minWin = 8.5 ; avgWin = 30; maxWin = 45 ; (range = 45 - 8.5 = 36.5)$$

Here is described a proposal of general function form with some optimization to lift off CPU load. For $f(x)$, it is proposed to use a quadratic equation in the form

of $f(x) = ax^2 + bx + c$, where $a < 0$, and $a(x-f(x)) = 0.1 - 2x^2 + x$; to make formula more suitable to required constant translations, it will be represented in form $\pm(ax-b)^2$. It will have the final form:

$$f(x) = -(ax - a(\text{minWin} + 0.01\text{range}))^2 + 0.1, \quad (3.25)$$

where a is a free parameter which will straightly affect steepness of function gradient;

x is an actual value of the current accumulated value;

minWin is the least value at which positive decision should be possible;

Also, the function should cross the point $(\text{minWin}; 0)$, resulting in

$$-(a * \text{minWin} - a(\text{minWin} + 0.01\text{range}))^2 + 0.1 = 0;$$

$$(a * \text{minWin} - a(\text{minWin} + 0.01\text{range}))^2 = 0.1;$$

$$(a * \text{minWin} - a * \text{minWin} - a * 0.01\text{range})^2 = 0.1;$$

$$(-a * 0.01\text{range})^2 = 0.1;$$

$$a^2 * 0.0001 * \text{range}^2 = 0.1;$$

$$a^2 = \frac{0.1}{0.0001 * \text{range}^2};$$

$$a = \frac{\sqrt{0.1}}{0.01\text{range}};$$

thus, resulting in the final formula will take the form:

$$f(x) = -\left(\frac{\sqrt{0.1}}{0.01\text{range}}(x - (\text{minWin} + 0.01\text{range}))\right)^2 + 0.1 \quad (3.26)$$

The formula for the $g(x)$ would likely be $g(x) = kx + b$ in general form; for the needs of the task, it will be transformed to the form of

$$g(x) = -a * (\text{minWin} + 0.01\text{range}) + ax + 0.1, \quad (3.27)$$

where a is a free coefficient which will affect the slope of the actual line;

Since it is known that the function should end at the point $((1-0.15)\text{avgWin}; 0.3)$, a can be calculated as following:

$$-a * (\text{minWin} + 0.01\text{range}) + a * (1 - 0.15)\text{avgWin} + 0.1 = 0.3;$$

$$a(0.85\text{avgWin} - (\text{minWin} + 0.01\text{range})) = 0.2;$$

$$a = \frac{0.2}{0.85\text{avgWin} - (\text{minWin} + 0.01\text{range})};$$

With this, the final formula for $g(x)$ takes the following form:

$$g(x) = \frac{0.2(x - (\text{minWin} + 0.01\text{range}))}{0.85\text{avgWin} - (\text{minWin} + 0.01\text{range})} * +0.1; \quad (3.28)$$

The next function is likely to be a parabola which crosses points $(0.85*\text{avgWin};0.3)$, $(\text{avgWin};0.85)$ and $(1.15*\text{avgWin};0.3)$; that is, the vertex will be in $(\text{avgWin};0.85)$, and it is enough to ensure function will cross $(0.85*\text{avgWin};0.3)$; so, having derived similar formulas, the formula will be

$$h(x) = -a^2(x - \text{avgWin})^2 + 0.85, \quad (3.29)$$

where a is a parameter which would fit formula to cross $(0.85*\text{avgWin};0.3)$

It can be derived in the following manner:

$$-a^2(0.85\text{avgWin} - \text{avgWin})^2 + 0.85 = 0.3;$$

$$-a^2(-0.15\text{avgWin})^2 = -0.55;$$

$$|0.15\text{avgWin}|a = \sqrt{0.55};$$

$$a = \frac{\sqrt{0.55}}{|0.15\text{avgWin}|} = \frac{\sqrt{0.55}}{\pm 0.15} \text{avgWin} = \frac{\sqrt{0.55}}{0.15\text{avgWin}}$$

Having that, and taking into account that negative coefficient value isn't possible due to constraints, the final parabolic formula can be represented in the following way:

$$\begin{aligned} h(x) &= \frac{-0.55}{(0.15avgWin)^2} (x - avgWin)^2 + 0.85 = \\ &= \frac{-24.(4)}{avgWin^2} * (x - avgWin)^2 + 0.85 \end{aligned} \quad (3.30)$$

The $i(x)$ can be made very similar to $g(x)$, i.e., it can be represented as

$$i(x) = \frac{0.2(x - (maxWin - 0.01range))}{1.15avgWin - (maxWin - 0.01range)} * +0.1, \quad (3.31)$$

It will cross points $(1.15avgWin; 0.3)$ and $(maxWin - 0.01 * range; 0.1)$;

After a sequence of trial and error, it has come that functions $j(x)$ and $k(x)$ can be united into one formula, which can be described as

$$\begin{aligned} j(x) &= \frac{-1}{100(x - (maxWin - 0.01range))} + 1, \\ \lim j(x) &= 1, x \in [maxWin - 0.01range; +\infty]. \end{aligned} \quad (3.32)$$

After all the formulas for different ranges were defined, they had to be plotted. GeoGebra [30] has been selected a plotting tool, since it allows flexible configuration and definition of sets of different formulas, with ability to specify ranges for each formula. Resulting graph is represented on the Figure 3.3.

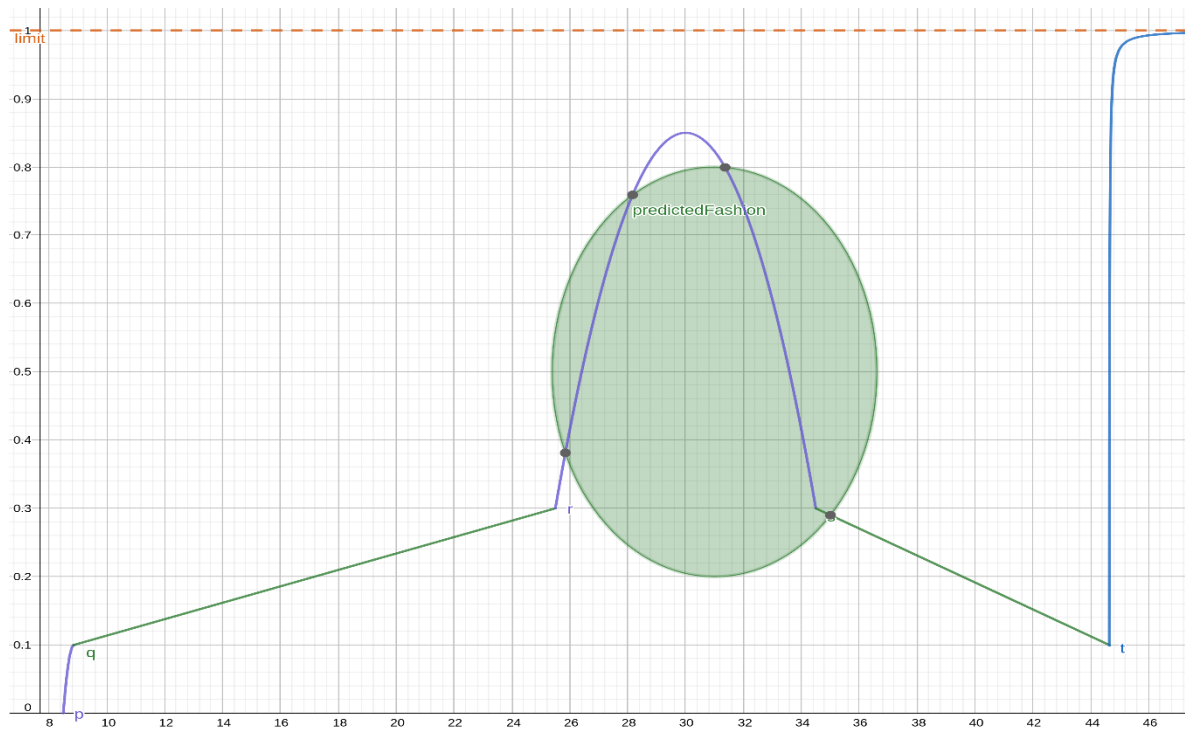


Figure 3.3 – Scaled graph of united functions restricted to defined ranges

Reduced functions will have the form with ranges:

- $$\minWinNbh = \minWin \vee seed + 0.01range;$$
- $$f(x) = 0.1 - 1000 \left(\frac{x - \minWinNbh}{range} \right)^2; x \in [\minWin \vee seed; \minWinNbh];$$
- $$\minWinNbh = \minWin \vee seed + 0.01range;$$
- $$g(x) = \frac{0.2}{0.85avgWin - \minWinNbh} * (x - \minWinNbh) + 0.1;$$

$$x \in [\minWinNbh; 0.85avgWin];$$
- $$h(x) = -24.4444 \left(\frac{x - avgWin}{avgWin} \right)^2 + 0.85; x \in [0.85avgWin; 1.15avgWin];$$
- $$\maxWinNbh = \maxWin - 0.01range;$$
- $$i(x) = \frac{0.2}{1.15avgWin - \maxWinNbh} * (x - \maxWinNbh) + 0.1;$$

$$x \in [1.15avgWin; \maxWinNbh + 0.015range];$$
- $$j(x) = 1 - \frac{0.01}{x - (\maxWin - 0.01range)}; \lim j(x) = 1;$$

$$x \in [\maxWinNbh + 0.025range; +\infty]$$

After an implementation of these formulas by means of programming language, a way to collect and visualize data has been implemented, the data was collected, and then analyzed, then fixes applied, analyzed, and after iterative fixes of all the errors, the final data has been visualized. The diagram of the plot received by applying the final formula set to real data is shown on Figure 3.4.

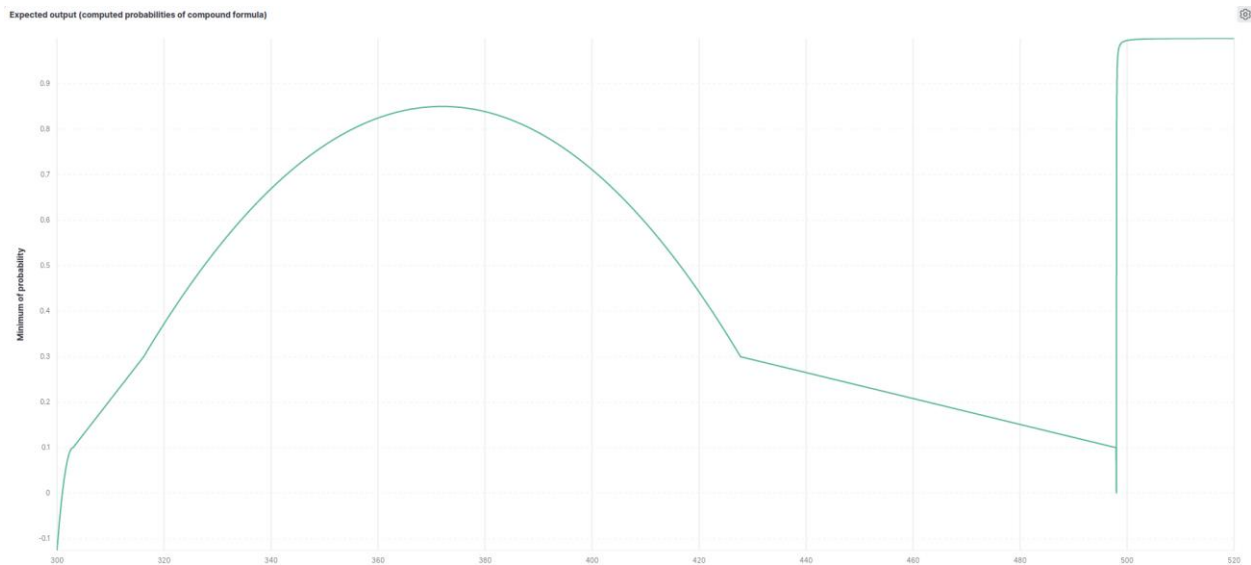


Figure 3.4 – Graph of empiric data representing positive decisions to negative decisions ratio according to the final formula set

To check if final mean values are in range of expected, they should be calculated as following:

$$E(x) = \frac{\sum x_i n_i}{N} = \frac{1}{N} \sum x_i n_i; \sum n_i = N; \quad (3.33)$$

where n_i is a total amount of occurrence of x_i

and N is a total number of all the occurrences

Said formula is easily derived from (3.23).

Calculation of mathematical expectation showed it is in the expected range, meaning this set of formulae fully meets the requirements.

3.2.3 Architecture choices

It has been decided to develop a module in a form of a web application with a set of API endpoints allowing to process individual inputs. The system is planned to be designed with an extensive use of interfaces, allowing for abstraction and making possible future substitution of components a very easy process, requiring only to implement said modules and inject them, with no other code being changed.

The developed system must store all the data it processes for further reporting to legal authorities and certificating bodies, but it must be avoided from usage in the process of making decisions. To solve this problem, it has been decided to make components of the module highly isolated, and store all the required data in the database.

The system will be designed to provide the same ability for containerization and deployment on AWS as the RNG module does. Such approach will make the module an easily scalable, and will grant a wide set of options for extension, also making it easier to integrate with RNG.

The service is meant to be run only inside private networks with no external interaction, lifting off the need to implement security module.

It has been decided to make the service configuration externalized, to provide even better flexibility and making it possible to re-configure some parts without modifying the code. The main functionality will be covered with tests, via usage of Junit 5 for testing purposes. The designed system should be able to withstand high load, up to 10000 concurrent requests, and be suitable for horizontal scaling.

It has been defined that the orchestrator module will also be implemented with a primary focus for AWS integration, and will perform statistic checking and scaling tasks issuing with the use of AWS CLI or AWS Development Kit. It's role implies managing redirection of events and requests to specific jackpots modules according to specific games, as well as deploying new or removing old instances of jackpot from environment in order to optimize resource utilization.

3.3 Implementation of modules

In this section, an implementation process of the modules described above is provided. It contains main points of interests, as well as required diagrams utilized in the development process. It also contains a section describing an implementation of a statistic module. There is not design section for it, since it is a straightforward from architecture and programming solutions point of view, but it's database relational structure and some specifics of implementation are still relevant.

3.3.1 DSS implementation

Since all the mathematical blockers have been resolved, it is possible to develop a programming implementation of sought solution. Since the solution does not bear the business value on itself, it was designed to be a part of a real-life flow. The workflow diagram for the module will have the logic as shown on Figure B.1. Full component diagram is represented on Figure B.2. Full class diagram for the DSS module is represented on Figure B.3.

From the diagram, it can be observed that the service is planned to be run inside the cluster, with orchestrator/loadbalancer handling requests and delegating them to the least loaded instances of DSS. While processing the request DSS requests a set of random numbers from RNG, defines the result for the request, stores it to the database for historical data reporting purposes, and then returns a notification to the requester in case system has made a positive decision. Otherwise, system just stores the data, with no notifications being sent.

In case of successful decision, DSS also triggers backend to activate payment module in order to process transaction.

The DSS module itself will hence consist of several submodules with distinct purpose each. Among these:

- core module, designated to request processing and decision making;

- feign RNG web module, designated to request random numbers from RNG on demand from core module;
 - web module providing API for interactions with the service;
 - database module for operating with database and storing historical data;
 - feign notification web module for sending notification to the requester.
- Also, since the DSS module needs a way to be configured directly by customers and clients, there is a requirement to develop the following submodules:
- web module providing API for DSS configuration; this one should be called 'Backoffice API';
 - client-side module for operating with 'Backoffice API'.

If refer to constraints to the DSS module, it requires a way to gather statistical data to compute an accuracy and authenticity of current state of the system and decisions it makes. This implies an additional list of submodules is required to be implemented:

- module for real input emulation;
- module for collection of the results and performing statistical calculations.

Provided all the modules for development are listed, they were grouped where applicable to create a full and harmonical view of the system to be developed and a diagram of principal architecture of the DSS module is represented on Figure 3.5.

According to the diagram, all the submodules have been grouped into 3 most significant categories. Main module is responsible for any data processing, handling requests, internal representation of different objects and models and all other internal logic. Web interface module actually includes all the communication interfaces used by the service. This includes Web API for input requests placing & processing, Web API for Backoffice, Feign for placing requests to RNG module, Feign for sending notifications to requester, and also includes JPA module for interacting with models from the AWS RDS (Amazon Web Services Relational Database Service).

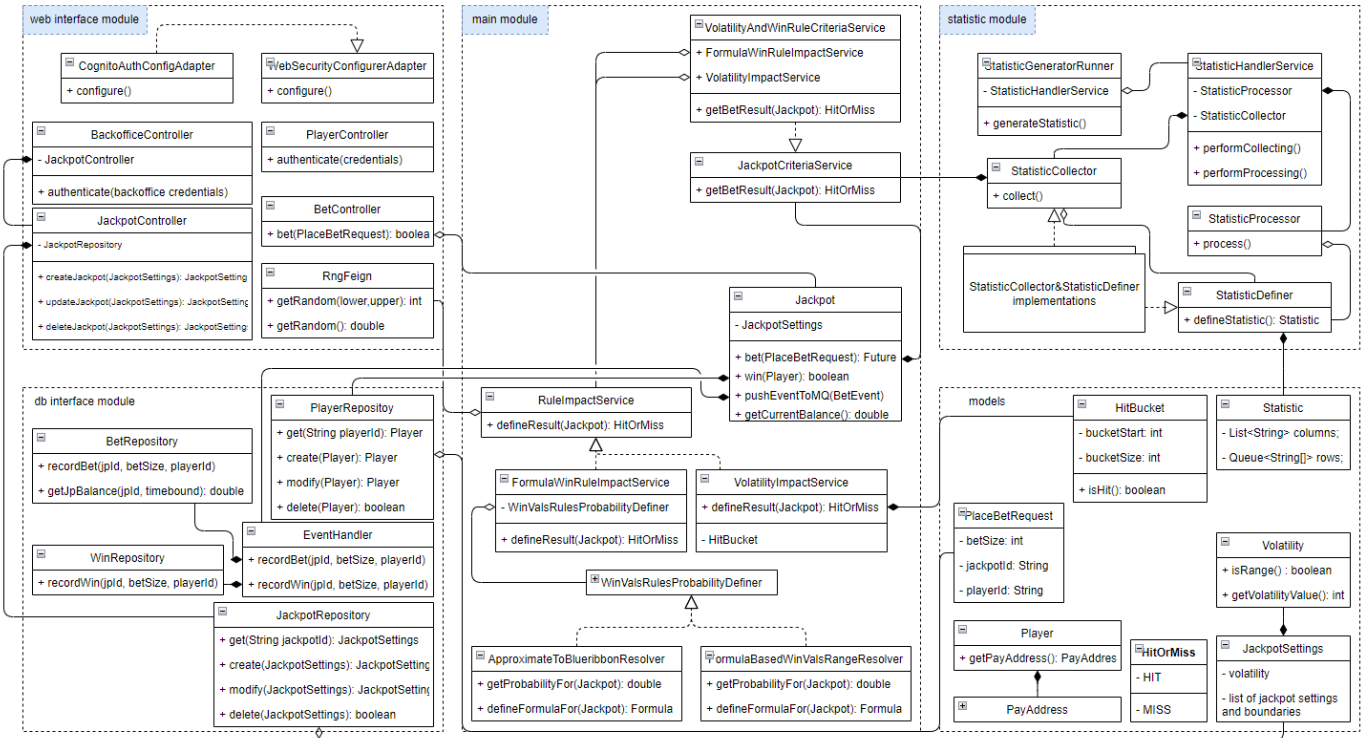


Figure 3.5 – Architectural diagram of the DSS module with submodules grouped into three most significant categories

It also contains Web API for interaction and convenient view of internal database models and representations. A Web API documentation module has been developed for the interface section, via use of the powerful library called springdoc-openapi. The example of the resulting Web API design is shown on Figure 3.6.

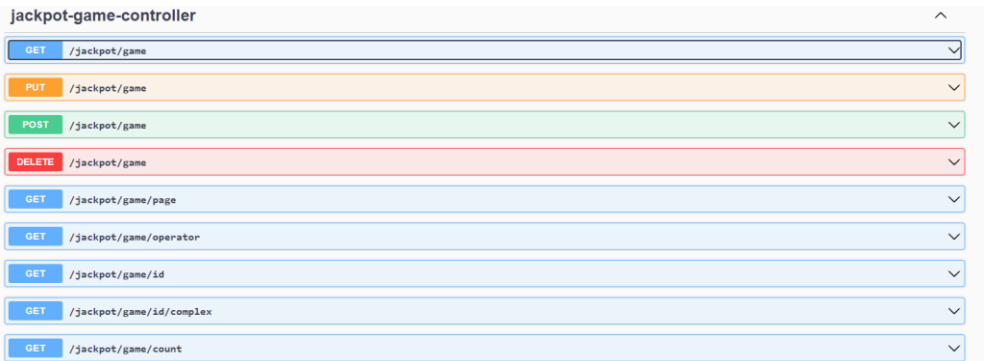


Figure 3.6 – Springdoc-openapi-powered Web API design for DSS module

This implementation of Web API design is interactive, meaning one can expand every listed method, check available description, parameters and the expected structure

of body, as well as expected structure of response The example of usage of interactive API is depicted on Figure 3.7.

The screenshot shows a web interface for an interactive API. At the top, it indicates a GET request to the endpoint `/jackpot/game/page`. Below this, there is a 'Parameters' section with a 'Cancel' button. The parameters are listed in a table with columns for 'Name' and 'Description'. Each parameter has a corresponding input field.

Name	Description
search string (query)	search
operatorid * required integer(\$int32) (query)	operatorid
page integer(\$int32) (query)	1
elements * required integer(\$int32) (query)	elements
sortBy string (query)	sortBy
descending boolean (query)	false

At the bottom of the interface, there is a large blue 'Execute' button.

Figure 3.7 – Example of springdoc-openapi interactive method usage

As an additional feature, this implementation provides a cURL analogue for each request it performs, which allows for these requests to be put into automation scripts or easily run from inside the environments with only CLI present, with no fear request somehow differs from the one which worked via springdoc-openapi. Example of cURL-transformed API request is shown on Figure 3.8.

```

Curl
curl -X 'GET' \
  'https://[redacted]/jackpot/settings/many?limit=2' \
  -H 'accept: */*'

Request URL
https://[redacted]/jackpot/settings/many?limit=2

```

Figure 3.8 – Example of the request automatic transformation into cURL request by springdoc-openapi

For the database connections, Spring Data JPA has been used, much simplifying the development of database queries themselves, allowing to focus more on logic of the service. However, complex queries were implemented manually.

As the next step, a core module has been implemented. It consists of three main parts: part responsible for volatility impact; part responsible for input parameters and configuration impact; internal service part, responsible for data preprocessing, models, representations, data marshalling & unmarshalling, and logic for web requests processing. A diagram showing the classes for volatility impact and impact of input parameters along with configuration is shown on the Figure 3.9. A showcase of the Web module for managing games' instances and creating new ones is presented on Figure B.5.

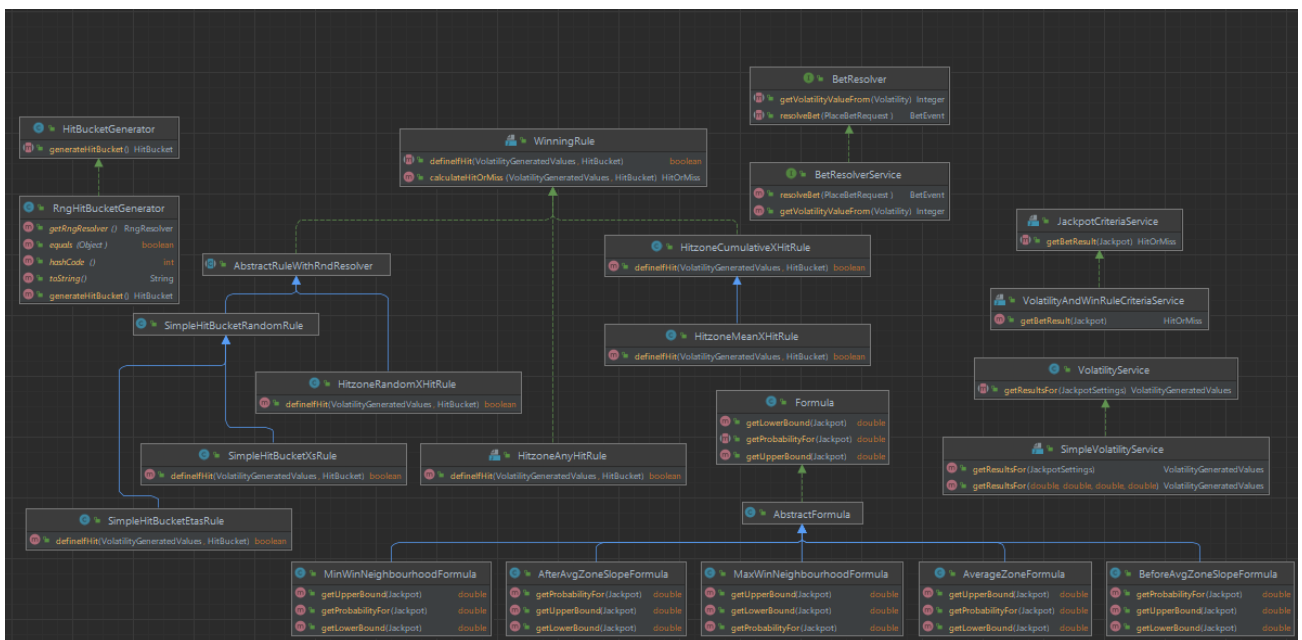


Figure 3.9 – Class and relations diagram for logic responsible for volatility impact and input parameters and configuration impact

3.3.2 PRNG module implementation

With interfaces implemented, the system will have the flow, depicted on the Figure 3.10.

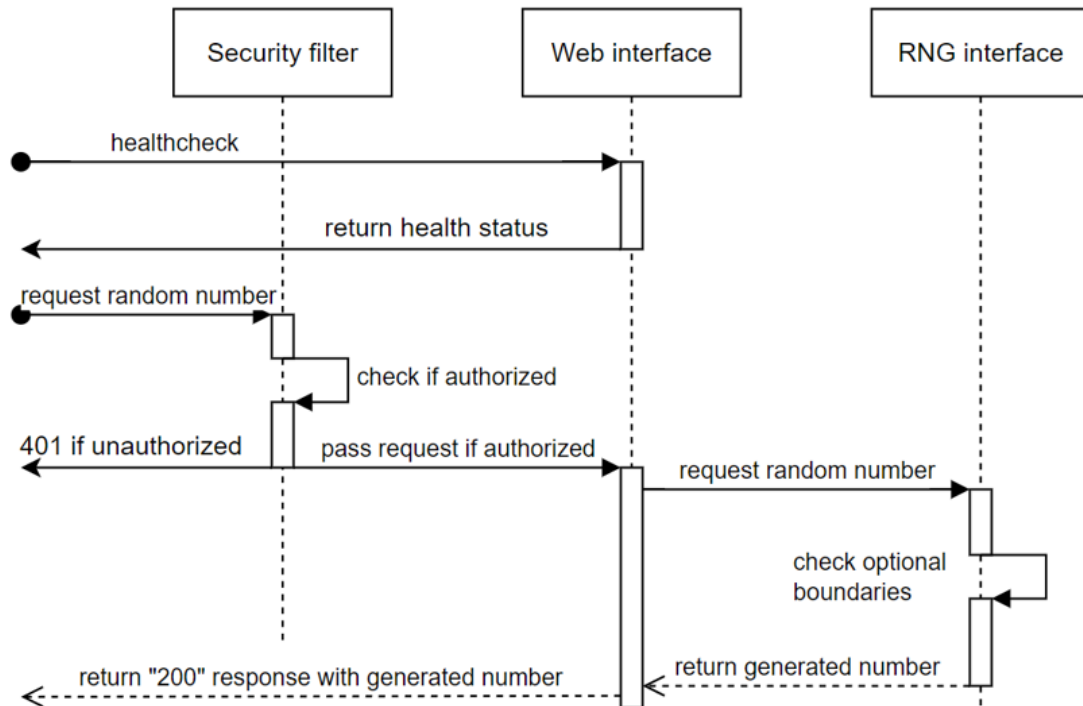


Figure 3.10 – Workflow diagram of RNG module

So, the principle of work is simple: if a healthcheck request is received, it bypasses security filter and is directly put to the web interface. It returns the state of the service and is generally required to make use of autoscaling and AWS deployment, where deployment system will gather information about the state of the service's instance from this endpoint. If a random number request is received, it first put against check of authorization header validity. If the auth header either is missing or doesn't match the required one after decoding, the service returns "401 Unauthorized" response. In case the request is successfully authorized, it is passed to the web interface, which marshals it and delegates further to RNG core interface. Here the request is checked against optional boundaries, since request supports a possibility to specify boundaries for generated values. Next step, RNG interface generates a random number inside specified boundaries and returns it to the web layer, from where the response "200 Success" with body containing the resulting random value is returned to the requester.

The class diagram for the RNG module is placed is depicted on Figure B.4. The diagram shows all the functional classes, including ones used for data samples generation. Functionality interfaces for each class are depicted where applicable.

3.3.3 PRNG module certification

After the service has been done, a certification process took place, including an audit and full RNG testing by a certified authority. The body conducting the audit is known as eCOGRA Limited [31].

Audit finished successfully, resulting in the RNG certificate being received. An extract from the certificate proving correlation tests have successfully passed threshold is added on Figure 3.11.

a. REZULTATE TESTE DIEHARD

TEST DIEHARD		DATE GENERATE DE CLIENT			DATE GENERATE DE eCOGRA		
Nume test	Dimensiune eşantion	Eşantionul 1	Eşantionul 2	Eşantionul 3	Eşantionul 1	Eşantionul 2	Eşantionul 3
TESTUL BIRTHDAY SPACINGS	48 000 000	0.0309	0.0337	0.2497	0.0738	0.1573	0.1330
TESTUL OVERLAPPING 5-PERMUTATION	48 000 000	0.6924	0.7250	0.3961	0.0858	0.0225	0.1367
TESTUL BITSTREAM	48 000 000	0.0065	0.2119	0.0060	0.1279	0.0028	0.0376
TESTUL COUNT-THE-1s	48 000 000	0.0011	0.0355	0.0215	0.0142	0.0564	0.0082
TESTUL DISTANŢĂ MINIMĂ	48 000 000	0.8082	0.9548	0.8364	0.1160	0.5274	0.3193
TESTUL SQUEEZE	48 000 000	0.5526	0.3548	0.1704	0.2152	0.5312	0.4630
TESTUL RUNS	48 000 000	0.1517	0.4014	0.2113	0.3579	0.0945	0.0195
TESTUL CRAPS	48 000 000	0.3060	0.3485	0.4450	0.7806	0.4100	0.1363

b. SUMAR SUCCES (✓) SAU DEFICIENŢĂ (✗) AL TESTULUI DIEHARD

TEST DIEHARD		DATE GENERATE DE CLIENT			DATE GENERATE DE eCOGRA		
Nume test	Dimensiune eşantion	Eşantionul 1	Eşantionul 2	Eşantionul 3	Eşantionul 1	Eşantionul 2	Eşantionul 3
TESTUL BIRTHDAY SPACINGS	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL OVERLAPPING 5-PERMUTATION	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL BITSTREAM	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL COUNT-THE-1s	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL DISTANŢĂ MINIMĂ	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL SQUEEZE	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL RUNS	48 000 000	✓	✓	✓	✓	✓	✓
TESTUL CRAPS	48 000 000	✓	✓	✓	✓	✓	✓

Figure 3.11 – Results of diehard test on provided samples passed the threshold

3.3.4 DSS module integration with PRNG

For the system to work properly and in a full scale, it is necessary to integrate service with the RNG module, in order to receive true random number for calculation of decision outcome.

For better scalability, a RNG interaction class has been abstracted to the interface, with its implementation providing required functionality. Since the flexible design, RNG interaction implementation class can be easily substituted if required. Interface is shown on the Figure 3.12.

```
@Service
public interface RngResolver {

    RngResponse getRngRandom(RngRequest rngRequest);
}
```

Figure 3.12 – RNG interaction interface

This implementation utilizes feign connection, implemented with the use of openfeign library. It implements a method to send a request to RNG module. Implementation of feign class is shown on Figure 3.13. It also worth to mention, that the URL that specifies RNG module location is externalized in order to keep service's flexibility.

```
@FeignClient(value = "rngFeign", url = "${rngservice.url}")
public interface CertifiedRNGFeign {

    @GetMapping
    Map<String, String> getRandom(@RequestHeader(name = "Authorization") String authHeader
        @RequestParam(required = false) Long bound);
}
```

Figure 3.13 – Implementation of the feign class to send requests to RNG module

An implementation of RngResolver interface is provided on Figure 3.14.

```

@Slf4j
@Service
@AllArgsConstructor
public class CertifiedRNGGenerator implements RngResolver {

    private CertifiedRNGFeign rngFeign;

    @Override
    public RngResponse getRngRandom(RngRequest rngRequest) {
        long bound;
        if (rngRequest.getUpperBound() != null) {
            bound = (long) rngRequest.getUpperBound();
        } else {
            bound = Integer.MAX_VALUE;
        }
        Integer random = Integer.parseInt(rngFeign.getRandom(aut
        if (rngRequest.getLowerBound() != null) {
            random += rngRequest.getLowerBound();
        }
        RngResponse rngResponse = new RngResponse(Math.toIntExact
        log.trace("Random to be returned: {}", rngResponse);
        return rngResponse;
    }
}

```

Figure 3.14 – Implementation of RNG interaction interface

3.3.5 Statistic collection module implementation

Statistic module is required to emulate a massed requests from real users, and collect a data received after processing these requests. It is a vital component, because without this module it cannot be told for certain if the service produces values of satisfactory entropy and convergence. This module is subdivided into two main interfaces: StatisticDefiner and StatisticCollector.

StatisticProcessor here used as a substitution of requester, receiving all the processed data and putting it into the analyzable format. In the current implementation of collector, it has been decided to use collection to the .csv file format. This format is quite easily produced and marshalled, and can be easily processed by Kibana, using the correct plugin for Logstash. In case of CSV files, Logstash requires not too many configuration and provides enough flexibility to optimize data before putting to

Kibana, to make it actually possible to produce visualization even for huge amounts of data (hundreds of millions of rows, a couple gigabytes each index). An all-round statistic processor implementation is shown on Figure 3.15.

```

@Service
public class CsvStatisticProcessor extends AbstractStatisticProcessorWriter {

    @Getter
    private final Path fileToWriteTo;
    private final PrintWriter writer;

    public CsvStatisticProcessor(StatisticCollectingConfig statisticCollectingConfig,
                                StatisticDefiner statisticDefiner) throws IOException {
        super(statisticDefiner.defineStatistic());
        this.fileToWriteTo = statisticCollectingConfig.getWriteFile();
        try {
            if (!Files.exists(fileToWriteTo)) {
                Files.createFile(fileToWriteTo);
            }
            writer = new PrintWriter(fileToWriteTo.toFile(), StandardCharsets.UTF_8.name());
        } catch (IOException e) {
            log.error("Error either creating file to write statistic to or opening it for wr
                throw e;
        }
        setupStatisticResource(statistic);
    }

    protected void setupStatisticResource(Statistic statistic) { writer.println(rowToCsv(sta

    @Override
    protected void logNWrite(List<String[]> rowsToWrite) {
        log.trace("Writing statistics to file {}", fileToWriteTo);
        rowsToWrite.stream().map(this::rowToCsv).forEach(this::write);
    }

    private synchronized void write(String row) { writer.println(row); }

    private String rowToCsv(String... row) { return String.join( delimiter: ",", row); }

    @Override
    @PreDestroy
    protected void preDestroy() {
        super.preDestroy();
        writer.close();
    }
}

```

Figure 3.15 – Implementation of StatisticProcessor which writes the data to the CSV

StatisticCollector is responsible for creating the required DSS components, configuration and setup, with the following emulation of user requests and processing

them. There is a plethora of implementations of `StatisticCollector`, in order to collect and analyze different aspects of decision-making process. A class diagram of different `StatisticCollector` implementations is presented on Figure 3.16.

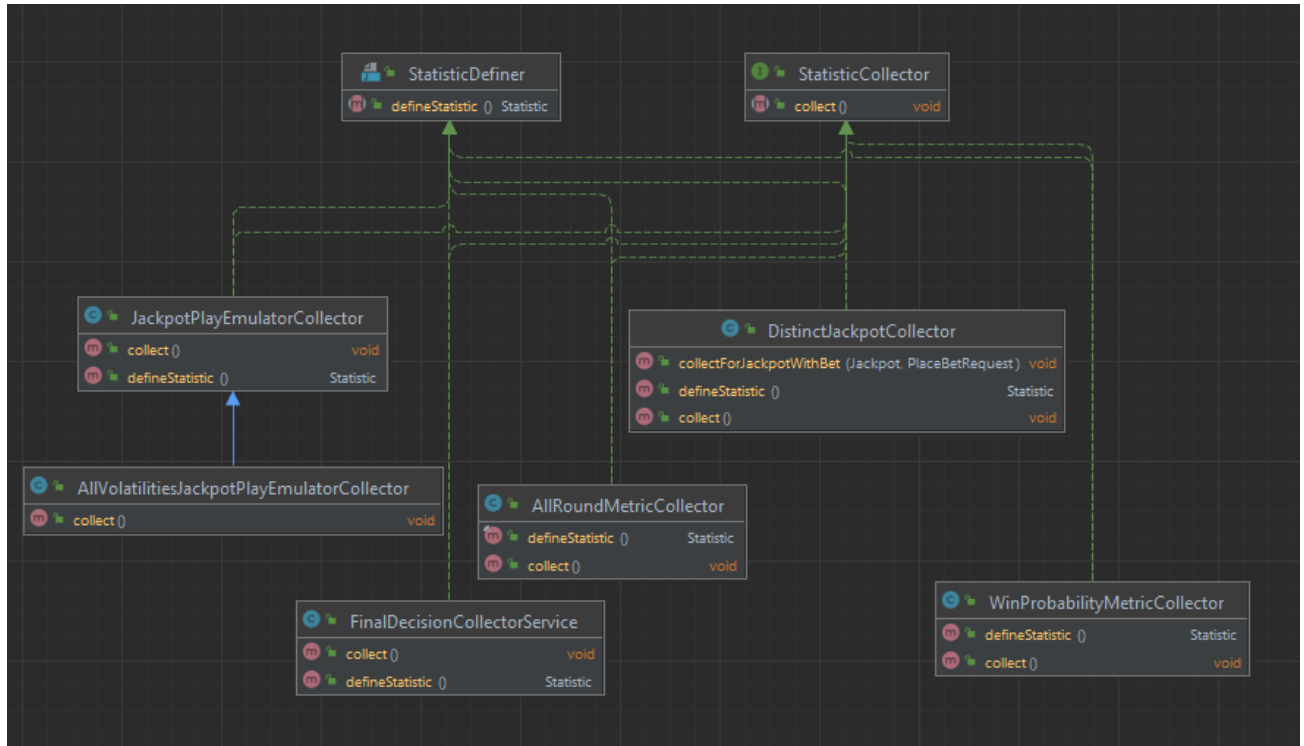


Figure 3.16 – Class diagram of different implementations of `StatisticCollector`

3.3.6 Containerization support implementation

After the main functionality for the system has been developed, it has been decided to provide a fully functional way for containerization, making service deployment much more easy and quick.

To achieve this goal, all the required externalized configuration parameters have been delegated to the Dockerfile, to make service fully customizable. As a base image, `openjdk-11` has been chosen. The final Dockerfile contains the following instructions, as shown on the Figure 3.17.

To make service better applicable for local testing and to automate Docker container creation, which otherwise requires configuration of container's volumes,

variables and port forwarding directives, a docker-compose.yml has been composed. The full docker-compose.yml contents are depicted on Figure 3.18.

```
FROM openjdk:11

RUN mkdir -p /statistic-emulate

COPY target/rng-jackpot-1.0.1-SNAPSHOT.jar jackpot.jar

EXPOSE 8080

CMD ["java", "-jar", "/jackpot.jar"]
```

Figure 3.17 – Contents of Dockerfile

```
version: '3.1'

services:
  rn-jackpot:
    build:
      context: .
      dockerfile: Dockerfile
    restart: always
    environment:
      JACKPOT_DATABASE_USER: [REDACTED]
      JACKPOT_DATABASE_PASSWORD: [REDACTED]
      JACKPOT_DATABASE_HOST: host.docker.internal
      JACKPOT_DATABASE_PORT: 5432
      JACKPOT_DATABASE_NAME: [REDACTED]
    ports:
      - "8080:8080"
      - "5432:5432"
```

Figure 3.18 – Contents of docker-compose.yml file

It can be observed, that this docker-compose file specifies all the variables required to properly function. It also forwards port 8080 from inside container to 8080 of the machine's one, to make Web API accessible. Another port it forwards is 5432, which is a port to maintain database connection

3.4 Conclusions to Section 3

In this section, a detailed description and explanation of the decision support system designed for use in financial and gambling industries development process has been provided. The description includes a complete definition of the problem and domain-specific requirements.

Next, it describes the flow of derivation of the mathematical component for the problem. Existing similar approaches are also reviewed, with detailed description which their characteristics render them not viable for this specific case.

Following mathematical derivation, a process of implementation the mathematical algorithm and formula set into the programming language in the form of fully standalone stateless system has been depicted. Diagrams and architecture graphs are provided where applicable.

A process of RNG module design and development has been depicted, and approaches to its design and development have been explained and substantiated. A workflow diagram depicting the principle of the service flow has been provided along the explanation to it. Also, a class diagram has been provided and explained.

As a result, a fully functional RNG module has been developed and certified by a licensed authority, resulting in gaining the certificate that RNG is compliant with all the restrictions applicable to this kind of services used in gambling and financial industries and, in the same time, satisfies all the domain-specific business requirements, rendering the service a viable option to be applied in such systems.

An extract from the certification document is added depicting that the module has successfully passed statistical diehard test.

After the RNG integration, a process of statistic module development is shown. This implementation provides a flexible and convenient way for extension and substitution of the existing functionality. The data processed by it is designated to be stored in the CSV format, with Logstash configuration to process this data and put in Kibana in a form that will allow plotting huge amounts of data, such as several

gigabytes at the time, has been also described. As it already mentioned, ELK stack (ElasticSearch-Kibana-Logstash) has been used for data processing and analysis.

The final step was a description of development of containerization support. The service has out-of-the-box support to be deployed as a Docker container via Docker or Docker-Compose, with docker-compose.yml also present for convenience. Thanks to this, the service is able to be deployed on AWS ECS or AWS Fargate.

This contributes for a description of the final product, which finally represents a complete decision support system for gambling industries. Said system provides a functionality required at the domain, and is certified to prove that the services are provided at satisfactory level, and don't violate legal regulations. System is designed to withstand high load and is tested to be run under 10'000 simultaneous requests. It also provides a great flexibility for extension in case of some very specific client's needs, and provides a comprehensible and convenient REST API, which, in turn, contributes for easy integration.

Ability for containerized support greatly increases a compatibility of cloud-based deployment and makes the deployment simple and quick. Also, containerization allows detailed and up-to-date resource monitoring, with great ability for both vertical and horizontal scaling.

4 BOTTLENECKS OPTIMIZATION AND ARCHITECTURE AMENDMENTS

Some of the potential problems of the system have already been described in Section 1. Here, let's take a look at some of them and consider what possible actions to improve the system.

4.1 Subnet traffic optimization problem

Regarding AWS subnet traffic speed, there's no real way to influence it directly. It depends solely on the physical location of the services inside the data center, quality of hardware (which is of high grade in AWS services) and a current load of the data center. Taking into account deployment inside a single subnet approach, it's already can be considered second best possible connectivity, the first one being deploying on own physical servers directly connected to the target facility via Ethernet or optical fiber.

Thus, only ways to increase interservice communication is either eliminate this data exchange at all (by unifying a number of services into a bigger one), or change protocols on either network or transport layers of OSI model [32], or limit information being sent to essential only, or remove a direct dependency between response await time and service idling.

Protocol change can be considered in the form of moving from HTTP over TCP to WebSocket over UDP or plain UDP. Transitioning to WSS is not a viable option as the current data structure requires information transmission through request headers. Moreover, maintaining the connection session could be detrimental to service operation, as it eliminates the possibility of independently scaling components and introduces issues with load distribution – the load balancer loses control over the direct number of requests to each of the services. Plain UDP is also unsuitable for the task's requirements since it significantly diminishes the ability to inform the sender about the

processing result of a request and leads to the loss of guaranteed message delivery, contradicting market regulator rules. It is also clearly stated in PRNG restrictions that it should not rely on any protocols which may not guarantee information on delivery status, limiting choices to TCP-based protocols.

The option of reducing the volume of transmitted information is not available, as the current communication is already minimized to only essential data. Potentially, data compression could be applied before transmission, but this might be effective only in the communication scenario between jackpot-backend, as the entropy of data in the communication with jackpot-randomizer is very high, and compression may not yield significant gains. Additionally, the compression process itself is resource-intensive for the processor: in a system with thousands of requests per second, increasing the processor load for compression may be impractical.

It is also impossible to eliminate jackpot to backend communication since these two systems are purposefully built for principally different tasks, have totally different responsibilities and database access levels. To note, they are also both resource-demanding. Also, since they are implemented using different technological stack, the process of their unification would take a considerable amount of time and effort, along with reducing maximal individually available resources for each of these components. This resource contention will inevitably impact performance and efficiency, since vertical scaling always has a hardware limit and current top-notch hardware could not content the services in their peak load viably. Such resources would impact pricing severely, to the extent of the service not being monetarily attractive anymore, thus leading to competition disadvantages. Also, a unification of jackpot and randomizer will also raise its own problems, including the fact that jackpot service is a subject of licensing and auditing, meaning changing its environment would lead to a new licensing process each time, rendering development process of both very cumbersome and also greatly increasing a cost with every new update deployed. It can also sabotage an ability to deploy critical system fixes or amendments, as deployment of unlicensed module is considered illegal, while the licensing itself can take weeks upon weeks.

Eliminating data exchange between jackpot-backend and jackpot-randomizer is not possible. These two systems are implemented in different programming languages, have fundamentally different responsibilities and databases, and are resource-intensive individually. Combining them would lead to increased competition for resources and result in a loss of productivity. Vertical scaling is limited, and at the current stage of technological development, it may not provide sufficient capabilities at a cost that would keep the system competitive. Merging the jackpot and randomizer services is also problematic, as randomizer is subject to licensing and auditing. Any changes to the code or hardware of this service would require re-licensing, a costly and time-consuming procedure that would hinder active development of functionality on the jackpot service.

However, there is a way to improve inter-service communication in both cases by decoupling the direct dependency between network transmission speed and response delay. This can be achieved by adding queues to the jackpot service to store intermediate requests. Thus, the jackpot service won't need to wait for a response from the randomizer, as the service will have internally stored a certain amount of data sets necessary for performing intermediate computations. This will help mitigate the increase in response delay during peak periods. It's worth noting that such a mechanism is already implemented for communication between jackpot and backend modules.

4.2 Database interaction optimization problem

Improving interaction with databases can be achieved through several approaches. Some of the most effective methods include:

- 1) specific queries: writing specific queries that take into account the system's specifics and execute as quickly as possible;
- 2) database structure: creating the right database structure and indexing necessary fields to optimize data retrieval;

3) connection pool arbitration: ensuring proper arbitration of the connection pool sessions with the database, preventing exhaustion of the connection pool.

After the system analysis, it can be concluded that the method of writing specific queries for performance optimization has been implemented during the creation of the system's core functionality. This approach, combined with proper database structuring and connection management, contributes to the efficiency and speed of database interactions.

Regarding database structure, it contains only the most essential data, which is structured in a way to be easily retrieved with minimal computational load. Everything easily derivable from the core data is calculated in situ on modules, to avoid extra database session usage. Database structure also sacrifices some storage space in sake of more indexing for faster queries. Every field used in comparison or search is indexed. Indexed are kept to HASH and BRIN where applicable, but every nonbinary comparison with naturally chaotic data required BTREE indexes. Application of decomposition and simplification to database models also helped keep it concise and efficient. However, for such a complex system, it is hard to define or argue on an optimality criteria, for real-world problems often implying some sort of compromise. The full entity relation diagram of the system placed under Appendix B on Figure B.6.

Connection pools can be scaled vertically, but this also has its limitations. However, to prevent pool depletion, a mechanism can be implemented to regulate a number of concurrent connections from service to the database and to manage database transactions. Communication with the database is implemented with aid of Spring JPA, which is a quality standard for the industry, and it already provides automatical connection pool management on the most part. A couple of significant configuration options include connection limit per physical instance, a mechanism of transaction handling and a blocking level for concurrent access with at least one thread trying to modify some records.

4.3 Load balancing optimization problem

Current loadbalancer implementation utilizes AWS variation, with distribution algorithm being round robin. Its principle can be basically reduced to cyclic distribution among all addresses for the alias in the DNS table. It provided an ability to filtering via CPU and/or RAM utilization metrics. It also provides orchestrator functionality by default and allows for autoscaling based on these parameters.

Still, it has no option to orchestrate/balance the load on custom metrics, neither does it have an option to extend it to include them.

In the essence, each jackpot creates some CPU load by its operation, as well as requests it handles also do create load. This means, each jackpot has some operational overhead on the machine it's being deployed onto. Thus, in the example case of 2 physical servers and 2 jackpots, it's far more beneficial to have each jackpot on its distinct server, rather than having both ones on both servers, avoiding double the overhead. This shows that with the specified problem, each new server will add lower performance boost to the system, also rendering vertical scaling (increasing single instance in terms of computational power). Instead, it is proposed to add a request distribution mechanism in such a way that, if possible, distribute the responsibility of servers for specific jackpots. In the example mentioned above, events for jackpot 1 will exclusively go to server 1, and events for jackpot 2 will go to server 2 accordingly. In case a specific server reaches its limit, scaling will occur for a specific set of jackpots.

Since existing out-of-the-box solutions don't provide required mechanisms to implement this idea, it creates a requirement to implement a custom loadbalancer. Estimated principal workflow is depicted on Figure 4.1.

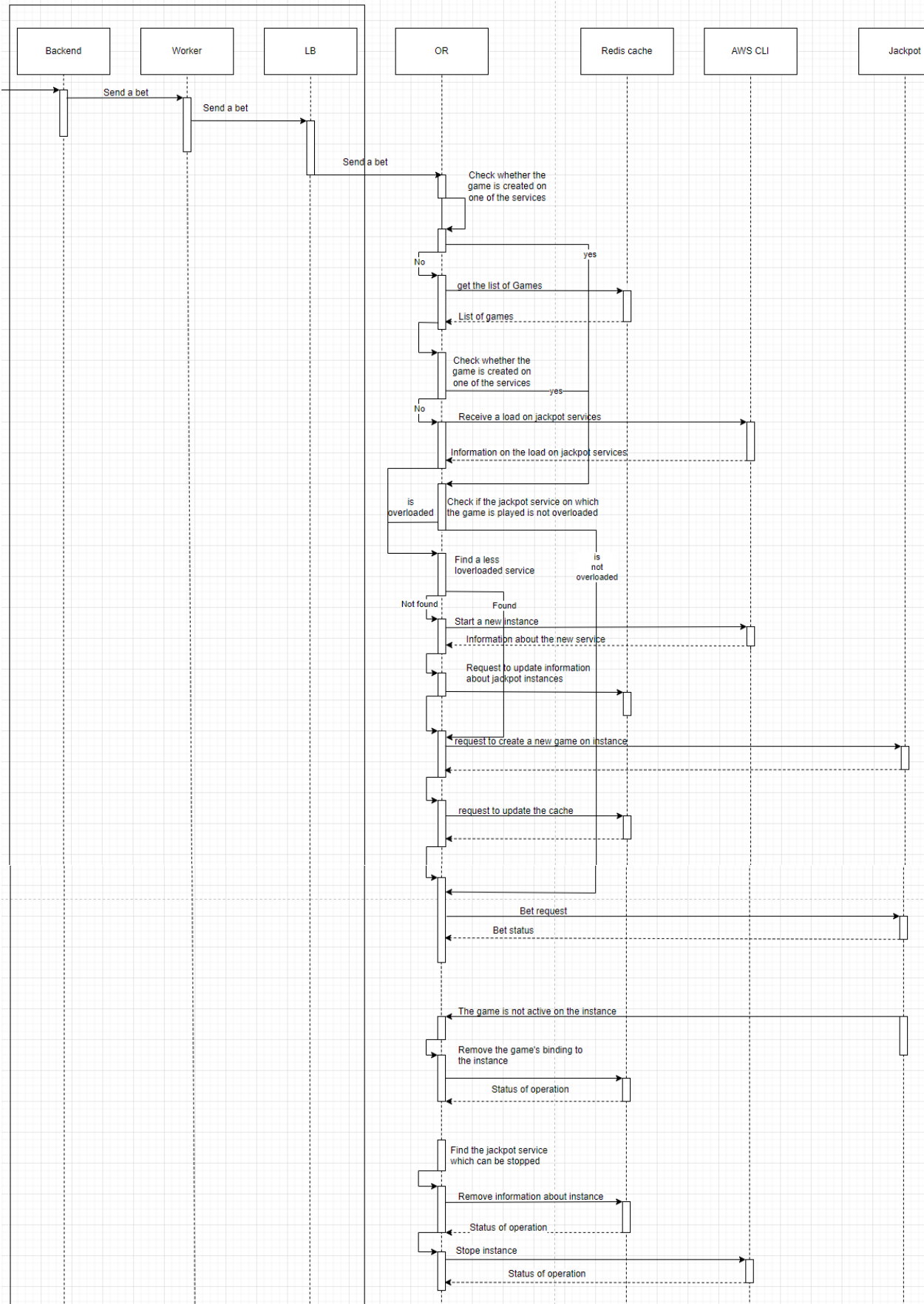


Figure 4.1 – Principal sequence diagram of expected orchestrator workflow

4.4 Calculations optimization consideration

Jackpot service handles a considerable number of computational operations and also handles a lot of lottery application requests. It renders a consideration for simplification or avoidance of some of these calculations a viable option to decrease load. A significant portion of the calculations are mandatory for each user's request, and are mathematically complex, as can be deduced from Section 3's derived formulae. They have been optimized for computer application during development, but calculation saturation reduction should be considered anyways, as it has a lot of potential.

4.4.1 Probability function intermediary calculations optimization

Probability function formula for each specific win uses a number of intermediary calculations. Codebase analysis has provided the evidence that some of them can be moved into a 'temporary constants' subclass, i.e., being constant precalculated values for a row of conditions, unless such conditions change (on winning or on configuration update for example).

For this purpose, it is suggested to create an additional container class for such constants. The lifespan of the constants is one jackpot iteration (until the next win) or until the first change in the settings of the specified jackpot. To synchronize this data across all physical instances of the service, these data should be stored in the database. It is also necessary to introduce a mechanism that will initiate the update of this data on all physical instances in case of updating these constants in the database by one of the services. Ensuring sequential access is not necessary since any of the proposed candidate models, although different from others (calculations dependent on random numbers), will be equally probable.

4.4.2 Dependency inversion of calculations and participant requests

Current system implementation implies that on each lottery application request eligible for participation with one user having an option to participate multiple times, a system conducts a row of calculations for determining the probabilistic outcome if this specific request is a winning one. It actually means that amount of participation applications generally equals to amount of calculation iterations.

However, for some jackpot types, periodical and time-driven in particular, it is irrelevant as win probability is much more tightly tied to the period (hour, day, week etc.), so that calculation dependency can be effectively inverted. That means for each period there will be a preliminary calculation to define a rough approximation of a winning time, and once a jackpot reaches such a threshold, only then will it start making more of heavy calculations, leading to skipping a significant chunk of uptime. In case of appropriate implementation, it can potentially lower the load in tens to hundreds times, dependent on period type.

Implementation of such an approach implies significant complications however, since current architecture design has its own limitations, and concurrency also contributing to the development complexity, effectively mandating implementation of a thread-safe mechanisms.

4.5 Conclusions to Section 4

This section depicts a process of analysis of bottlenecks and methodologies applied to it. It also shows a detailed review of possible solutions for each found problem and shows evaluation of viability of application for each one. It is comprised of such problems as subnet traffic speed optimization, database interactions optimization, load balancing optimization and calculations' optimization.

For a subnet traffic, it has been proven that not much can be dealt in terms of increasing speed. The main reasons are AWS subnet providing virtually the best speeds

possible, communication being cut to essential only already, and limitations in protocol choices applied by PRNG certification requirements.

Considering database interactions optimization, it has been described that most part of connection pool management is already present, as well as transactions kept concise and as fast as possible and required fields are indexed. Regarding database access frequency, there is already caching present where applicable, and increasing cache time would compromise data precision to the unacceptable levels, thus, current frequency can be considered minimal.

Regarding load balancing optimization, it has been shown that stock options provided do not take into account specific metrics of the system built, and to make the most use of resource, it is required to build a custom load balancer with orchestration functionality. Such service should balance loading not only according to CPU and RAM metrics, but also comply with amounts of requests received, as well as try to segregate different logical game instances on different physical instances to gain the most out of horizontal scaling. A sequence diagram of such a system in place has been provided.

From calculations' perspective, it has been defined that some calculations are effectively constant unless some environment and/or configuration change related to the games' entities they relate to. A system to keep a map of locally stored constants which would only change when required, and this way will remove around 20% of calculations from each lottery application to only being performed on environment change. Also, for periodic game types, these dependency for calculations has been inverted, so that for the most part there will be no load unless a time is close to estimated win period, thus saving as much as 90% of calculation load for games with periods bigger than or equal to one week. Aforementioned changes have been reviewed by legal entities and have been considered acceptable for not modifying average values and randomness of wins happening.

5. THE ECONOMIC SECTION

5.1 Technological audit of the developed the retail gamification system

As mentioned earlier, currently, a majority of human activities witness the dominance of computers in decision-making aspects. These areas can be categorized as follows: a) either they are simple enough to isolate for decision-making that can be executed within a specific number of operations (well-suited for algorithmization), or b) they are sufficiently routine and laborious, thus making automation of such processes' perspective in terms of time and financial savings.

Nowadays, most implementations of decision-making systems and modules heavily rely on context and historical data, thereby surpassing human productivity due to higher computational speed and the ability to retain a large dataset in unaltered form within operational memory. However, sometimes due to various constraints associated with legislation, there is a need to comply with established requirements. The decision-making system must function independently of context while maintaining the same decision-making module as systems with state retention.

Therefore, the analysis focuses on relevance of application of decision support system to a retail industry and its potential to optimize sales. To establish the level of commercial potential for the software developed, the three prominent experts were invited: Doctor of Technical Sciences, Professor Kvetny R.N., Candidate of Technical Sciences, Associate Professor Harmash V.V., and Candidate of Technical Sciences, Associate Professor Kulyk Y.A.

The assessment of the commercial potential of the application was conducted based on the criteria summarized in Table 5.1, with each expert evaluating certain points of system's viability and potential for profitability.

Table 5.1 – Assessment criteria for evaluating the commercial potential of any development and their score rating (on a scale of 0 - 1 - 2 - 3 - 4 points)

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
Technical Feasibility of the Concept:					
1	The credibility of the concept is not confirmed.	Concept validated by expert opinions	Concept validated by calculations	Concept tested in practice	Product's operational capability verified in real-world conditions
Market Advantages (Disadvantages):					
2	Numerous analogs in a small market	Few analogs in a small market	Several analogs in a large market	One analog in a large market	No analogs for the product in a large market
3	The price of the product is significantly higher than that of analogs	The price of the product is slightly higher than that of analogs	The price of the product is approximately equal to the prices of analogs	The price of the product is slightly lower than that of analogs	The price of the product is significantly lower than that of analogs
4	Technical and consumer properties of the product are significantly worse than those of analogs	The technical and consumer properties of the product are slightly worse than those of analogs	The technical and consumer properties of the product are on par with those of analogs	The technical and consumer properties of the product are slightly better than those of analogs	The technical and consumer properties of the product are significantly better than those of analogs

Continuation of Table 5.1

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
Market Prospects					
5	Operational costs are significantly higher than those of analogs	Operational costs are slightly higher than those of analogs	Operational costs are on par with the operational costs of analogs	Operational costs are slightly lower than those of analogs	Operational costs are significantly lower than those of analogs
6	The market is small and lacks positive dynamics	The market is small but shows positive dynamics	Medium-sized market with positive dynamics	Large and stable market	Large market with positive dynamics
7	Active competition from major companies in the market	Active competition	Moderate competition	Slight competition	No competitors
Practical Feasibility					
8	Lack of experts in both technical and commercial implementation of the idea	Requires hiring experts or significant investment of time and money in training existing personnel	Minor training required for staff and slight expansion of the team	Minor training required for staff	Experts available both technically and commercially

Continuation of Table 5.1

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
9	Significant financial resources needed. Lack of funding sources for the idea	Slight financial resources needed, but no funding sources available	Substantial financial resources needed, funding sources exist	Slight financial resources needed, funding sources exist	No need for additional funding
10	Requires development of new materials	Materials required are used in military-industrial complex	Expensive materials needed	Accessible and inexpensive materials needed	All materials for idea implementation are well-known and have long been used in production
11	Implementation timeline exceeds 10 years	Implementation timeline exceeds 5 years. Return on investment period exceeds 10 years	Implementation timeline from 3 to 5 years. Return on investment period exceeds 5 years	Implementation timeline is less than 3 years. Return on investment period from 3 to 5 years	Implementation timeline is less than 3 years. Return on investment period is less than 3 years
12	Necessary development of regulatory documents and acquiring numerous permits for production	Requires acquiring permits for production and implementation, significant costs and time	The process of obtaining permits for production and product implementation requires minor costs and time	Only notification to relevant authorities about production and product implementation is necessary	No regulatory constraints on production and product implementation

Invited experts have evaluated developed system as followed in table 5.2:

Table 5.2 – Evaluation Results of the Commercial Potential of the Development

Criterion	Last name, initials of the expert		
	Kvetny R.N.	Harmash V.V.	Kulyk Y.A
	Scores given by the experts:		
1	3	4	3
2	4	3	3
3	3	4	3
4	4	4	3
5	3	4	3
6	3	4	4
7	3	4	3
8	4	4	3
9	4	3	4
10	4	4	3
11	4	3	3
12	4	4	4
Sum of grades	СБ ₁ = 43	45	39

The arithmetic mean (\overline{CB}), of the scores assigned by the experts was:

$$\overline{CB} = \frac{\sum_{i=1}^3 B_i}{3} = \frac{43+45+39}{3} = \frac{127}{3} = 42,33.$$

The overall level of commercial potential for any development was determined based on the criteria outlined in Table 5.3 [33].

Guided by the recommendations in Table 5.3, it can be concluded that the developed system was evaluated by experts at 42,33 points, indicating that named development possesses a commercial potential categorized as "high".

Table 5.3 – Levels of Technical and Commercial Potential of the Development

The arithmetic mean of scores calculated based on the experts' conclusions.	The level of technical and commercial potential of the development.
0 – 10	Low
11 – 20	Below average
21 – 30	Average
31 – 40	Above average
41 – 48	High

This is explained by the fact that the development has prospects and advantages of modern scientific solutions (simplicity, functionality, flexibility, efficiency), while eliminating several drawbacks (complexity in interface and functionality, use of complex algorithms, the necessity of support and certification, etc.). Additionally, for further improvement of the development, it can be expanded by supplementing the software with new modules.

5.2 Calculation of the expenses incurred in the development of the retail gamification system

During the work, the following expenses were incurred:

1) Primary salary of executors 3_o :

$$3_o = \frac{M}{T_p} \cdot t \text{ UAH}, \quad (5.1)$$

Where M represents the monthly base salary of a specific executor in UAH; In 2023, the salary ranges for researchers fall within (6700...26000) UAH/month; T_p – indicates the number of working days in a month; let's assume $T_p = 20$ days.

The calculations of the primary salary of the executors will be summarized in Table 5.4:

Table 5.4 – Calculation of the primary salary of executors (developers)

Position title of the executor	Monthly base salary, UAH	Payment per working day (or per hour), UAH	Number of working days	Remuneration costs, UAH	Notes
1. Scientific supervisor of the Master's qualification work	20000	1000	20 hrs	3333,33 ≈ 3334	6 hrs per day
2. Student developer - Master's student	6700	335	70 days	23450	
3. Consultant in the economic section	19000	950	1,5 hrs	237,5 ≈ 238	6 hrs per day
4. Other consultants	16000	800	3 days	2400	
Total				29422	

2) Additional remuneration of the executors 3_d is calculated as (10...12 of the primary salary of the executors, which means:

$$3_d = (0,1...0,12) \cdot 3_o. \quad (5.2)$$

For this case:

$$3_d = 0,101 \times 29422 = 2971,62 \approx 2972 \text{ UAH.}$$

3) Accruals to the payroll H_{3n} are calculated by the formula:

$$H_{3n} = (3_o + 3_d) \cdot \frac{\beta}{100}, \quad (5.3)$$

where 3_o – primary salary of the executors, UAH;

3_d – additional remuneration of the executors, UAH;

β – The rate of the unified social security contribution for mandatory state social insurance; $\beta = 22\%$.

Then:

$$H_{3\text{II}} = (29422+2972) \times 0,22 = 7126,68 \approx 7127 \text{ UAH.}$$

4) Material expenses M are calculated per each material type:

$$M = \sum_1^n H_i \cdot \Pi_i \cdot K_i - \sum_1^n B_i \cdot \Pi_B \text{ UAH,} \quad (5.4)$$

where H_i – material expenses per i designation, kg; Π_i – cost of a material i , UAH /kg.; K_i – transportation expenses coefficient, $K_i = (1,1\dots1,15)$; B_i – material disposal mass per material i , kg; Π_B – material waste price per material i , UAH /kg; n – total number of used materials.

5) The expenses on components K are calculated by the formula:

$$K = \sum_1^n H_i \cdot \Pi_i \cdot K_i \text{ UAH,} \quad (5.5)$$

where H_i – the quantity of components i -th type, pcs.; Π_i – price per component of i -th type, UAH; K_i – transportation expenses coefficient, $K_i = (1,1\dots1,15)$; n – total number of components.

Following the analogy with other developments, the cost of all utilized material resources is approximately 900 UAH.

6) Depreciation (A) of equipment, computers, and premises A can be calculated by the formula:

$$A = \frac{\Pi \cdot H_a}{100} \cdot \frac{T}{12} \text{ UAH,} \quad (5.6)$$

where Π – the total book value of fixed assets in UAH;

H_a – the annual depreciation rate: $H_a = (2\dots25)\%$;

T- is the period of equipment, premises, etc. the usage, in months.

The calculations made have been summarized in Table 5.5:

Table 5.5 – Calculation of depreciation deductions

Equipment, premises, etc.	Book value, UAH.	Depreciation rate, %	Period of usage, months.	Depreciation deductions, UAH
1. Personal computers, printers, etc	62000	25	3,2 (50%)	2066,67
2. Department and faculty premises	52000	2,5	3,2 (50%)	173,33
Total				A = 2240

7) Expenses for electrical power B_e are calculated using the formula:

$$B_e = \frac{B \cdot \Pi \cdot \Phi \cdot K_{\Pi}}{K_{\Delta}},$$

(5.7)

where B – price of 1 kilowatt-hour. Electricity in 2023 $\approx 4,5$ UAH/kWh;

Π – The installed capacity of the equipment kWh; $\Pi = 1,05$ kWh

Φ – actual number of equipment operating hours, hours.

Assume, that $\Phi = 315$ hours;

K_{Π} – power usage coefficient; $K_{\Pi} < 1 = 0,83$.

K_{Δ} – Useful action coefficient $K_{\Delta} = 0,76$.

Then the expenses for electrical power:

$$B_e = \frac{B \cdot \Pi \cdot \Phi \cdot K_{\Pi}}{K_{\Delta}} = \frac{4,5 \cdot 1,05 \cdot 315 \cdot 0,83}{0,76} = 1625,46 \approx 1626 \text{ UAH.}$$

8) Other expenses B_{iH} can be estimated as (50...300)% from the initial salary of the performers:

$$B_{iH} = K_{iH} \times 3_o = (0,5..3,0) \times 3_o. \quad (5.8)$$

In this case let's assume that $K_{iH} = 0,75$. Then:

$$B_{iH} = 0,75 \times 29442 = 22081,5 \approx 22082 \text{ UAH.}$$

9) Total sum of all the previous expenses gives the total expenses of the current stage execution by the Student developer - Master's student – B.

In this case:

$$B = 29422 + 2972 + 7127 + 900 + 2240 + 1626 + 22082 = 66369 \text{ UAH.}$$

10) The calculation of the total costs for the development and final refinement of the work that have been done is carried out according to the formula:

$$3B = \frac{B}{\beta}, \quad (5.9)$$

Where β –coefficient characterizing the stage of completion of this work. Since the development still requires slight refinement, it can be assumed that $\beta \approx 0,87$.

Then:

$$3B = \frac{66369}{0,87} = 76286,21 \text{ UAH or approximately } 77000 \text{ UAH.}$$

So, the projected total expenses for the development of the gamification system amount to approximately 77 000 UAH.

5.3 Calculation of the economic effect from the potential commercialization of the development

The market analysis indicates that the developed the gamification system will have significant demand among companies engaged in retail sales, particularly in supermarket chains like Silpo, ATB, Metro, Epicenter, online marketplaces such as Hotline and Rozetka, online stores, online gaming, gambling establishments, and similar domains.

So, if the development is implemented from January 1, 2024, its results will manifest during 2024, 2025, and 2026. The projected increase in demand for the development per year is as follows:

- a) 2023 - 1 unit (development);
- b) 2024 - +5 units from the base year(i.e., 5 clients);
- c) 2025 - +10 units from the base year(i.e., 10 clients);
- d) 2026 - +15 units from the base year(i.e., 15 clients).

According to expert conclusions, the potential market price for the development in the current market is approximately \$6,000 or around 240,000 UAH, while similar developments that partially perform the functions mentioned above cost up to 200,000 UAH in the market. The potential increase in net profit $\Delta\Pi_i$, from taking the product to the market will amount to:

$$\Delta\Pi_i = \sum_1^n (\Delta\Pi_o \cdot N + \Pi_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{v}{100}\right), \quad (5.10)$$

Where $\Delta\Pi_o$ – an improvement in the primary qualitative indicator from implementing the outcomes of the development in this year. In the case, this is:

$$\Delta\Pi_o = 240 - 200 = + 40\,000 \text{ UAH};$$

N – the main quantitative indicator that defines the scope of activities in the year before the implementation of the development results; $N = 1$ pcs.;

ΔN – improvement of the main quantitative indicator due to the implementation of the development results.

This improvement will be as follows: in 2024 – $\Delta N = 3$ pcs., in 2025 $\Delta N = 10$ pcs., and in 2026 $\Delta N = 15$ pcs.;

Π_0 - the primary qualitative indicator (i.e., the price) determining the scope of activity in the year following the implementation of the development results UAH; $\Pi_0 = 240\,000$ UAH;

n – total number of years, during which the positive results from development implementation is expected; in this scenario $n = 3$;

λ – The coefficient that takes into account the value-added tax (VAT) payment; $\lambda = 0,8333$;

ρ – The coefficient that considers the product's profitability. It is recommended to assume $\rho = (0,2...0,5)$; set $\rho = 0,5$;

υ – the corporate tax rate. In 2023-26 years $\upsilon = 18\%$ (assumption).

The potential increase in net profit $\Delta \Pi_1$ for a potential investor during the first year after the possible implementation of the development (2024), it would be:

$$\Delta \Pi_1 = [40 \cdot 1 + 240 \cdot 5] \cdot 0,8333 \cdot 0,5 \cdot \left(1 - \frac{18}{100}\right) \approx 424 \text{ thousand UAH.}$$

For the potential investor during the second year after the possible implementation of the development (2025), it would be calculated similarly:

$$\Delta \Pi_2 = [40 \cdot 1 + 240 \cdot 10] \cdot 0,8333 \cdot 0,5 \cdot \left(1 - \frac{18}{100}\right) \approx 834 \text{ thousand UAH.}$$

The potential increase in net profit $\Delta \Pi_3$ for a potential investor during the first year after the possible implementation of the development during the third year (2026) is totaled:

$$\Delta\Pi_3 = [40 \cdot 1 + 240 \cdot 15] \cdot 0,8333 \cdot 0,5 \cdot \left(1 - \frac{18}{100}\right) \approx 1244 \text{ thousand UAH.}$$

The total value of the increased net profits from the potential implementation and commercialization of the development:

$$\text{ПП} = \sum_1^{\tau} \frac{\Delta\Pi_i}{(1 + \tau)^t}, \quad (5.11)$$

where $\Delta\Pi_i$ – the increase in net profit in each of the years when the results of the completed and implemented work are manifested is as follows;

τ – the time period during which the results of the implemented work are manifested is for 3 years, represented by $t=3$ years.;

τ – the discount rate. Let's assume $\tau = 0,10$ (10%);

t – the period of time from the initiation of the development of the retail gamification system when potential net profits are obtained by the potential investor.

Then, the present value of the growth of all potential net profits (PP) that a potential investor can gain from the commercialization of the development would be:

$$\text{ПП} = \frac{424}{(1+0,1)^2} + \frac{834}{(1+0,1)^3} + \frac{1244}{(1+0,1)^4} \approx 350 + 627 + 850 = 1827 \text{ thsnd. UAH.}$$

The present value of investments (PV) that should be allocated towards the implementation of the development would be: $PV = (1,0 \dots 5,0) \times B_{3ar}$.

In this case, $PV = (1,0 \dots 5,0) \times 73 = 5,0 \times 77 = 385$ thousand UAH.

The absolute effect of potential investments made in the implementation of the development would be E_{a6c} .

$$E_{a6c} = \text{ПП} - PV, \quad (5.12)$$

where $\Pi\Pi$ – the present value of the increase in all net profits for the potential investor from the potential commercialization of the development, UAH;

PV – The present value of investments (PV) amounts to $PV = 385$ thousand UAH.

The absolute effect from the potential implementation of the development will be:

$$E_{a\delta c} = 1827 - 385 = 1442 \text{ 000 UAH.}$$

Next, let's calculate the internal rate of return (E_B) of the invested capital:

$$E_B = \sqrt[T_{\text{ж}}]{1 + \frac{E_{a\delta c}}{PV}} - 1, \quad (5.13)$$

where $E_{a\delta c}$ – the absolute effect of the invested capital; $E_{a\delta c} = 1442 \text{ 000 UAH}$;

PV – the present value of the initial investments $PV = 385 \text{ 000 UAH}$;

$T_{\text{ж}}$ – development lifecycle, years.

$T_{\text{ж}} = 4$ years (2023, 2024, 2025, 2026 years)

In this case it would be:

$$E_B = \sqrt[4]{1 + \frac{1442}{385}} - 1 = \sqrt[4]{1 + 3,7455} - 1 = \sqrt[4]{4,7455} - 1 = 1,476 - 1 = 0,476 = 47,6\%.$$

Next step is to figure out the minimum profitability. If it is less than that, a potential investor wouldn't want to commercialize the development.

The following formula is used to determine the minimum profitability, or barrier discount rate τ_{MiH} :

$$\tau_{\text{MiH}} = d + f, \quad (5.14)$$

where d – the weighted average rate on deposit operations in commercial banks; in 2022-2023 in Ukraine $d = (0,10...0,12)$;

Assume $d = 12\%$;

f – indicator characterizing the riskiness of investments

$f = (0,1...0,50)$. Assume $f = 0,30$.

In this case:

$$\tau_{\text{MIH}} = 0,12 + 0,30 = 0,42 \text{ або } \tau_{\text{MIH}} = 42\%.$$

Given the magnitude $E_B = 47,6\% > \tau_{\text{MIH}} = 42\%$, then a potential investor may indeed be interested in financing and commercializing the development.

. Next step is to calculate the payback period for the funds invested in the potential commercialization of the system.

The payback period T_{OK} is calculated by formula:

$$T_{\text{OK}} = \frac{1}{E_B}. \quad (5.15)$$

In this case the payback period T_{OK} :

$$T_{\text{OK}} = \frac{1}{0,476} = 2,10 \text{ years} < 3 \text{ years},$$

this indicates the potential viability of commercializing the developed system of the retail gamification system.

Further, a simulation was conducted to model the relationship between the internal rate of return of potential investments and the inflation rate in the country.

If the inflation rate in the country increases to 20%, then:

$$\text{III} = \frac{424}{(1+0,2)^2} + \frac{834}{(1+0,2)^3} + \frac{1244}{(1+0,2)^4} \approx 294 + 483 + 600 = 1377 \text{ 000 UAH}.$$

The absolute effect from the potential implementation of the development will amount to:

$$E_{a\delta c} = 1377 - 385 = 992\ 000 \text{ UAH.}$$

Next, the calculation of internal rate of return (IRR) of the invested investments E_B is provided:

$$E_B = T_{\text{ж}} \sqrt[4]{1 + \frac{E_{a\delta c}}{PV}} - 1,$$

where $E_{a\delta c}$ – absolute effect of the invested investments; $E_{a\delta c} = 992\ 000 \text{ UAH}$;

PV – the present value of the initial investments $PV = 385\ 000 \text{ UAH}$;

$T_{\text{ж}}$ – development lifecycle, years.

$$E_B = \sqrt[4]{1 + \frac{992}{385}} - 1 = \sqrt[4]{1 + 2,5766} - 1 = \sqrt[4]{3,5766} - 1 = 1,375 - 1 = 0,375 = 37,5\%.$$

Given magnitude $E_B = 37,5\% < \tau_{\text{миг}} = 42\%$, then the potential investor might have doubts in financing and commercializing the development in principle.

If the country's inflation rate rises to 30%, then:

$$\text{ПП} = \frac{424}{(1+0,3)^2} + \frac{834}{(1+0,3)^3} + \frac{1244}{(1+0,3)^4} \approx 251 + 380 + 436 = 1067\ 000 \text{ UAH.}$$

The absolute effect from the potential implementation of the development will be:

$$E_{a\delta c} = 1067 - 385 = 682\ 000 \text{ UAH.}$$

Next, let's calculate the internal rate of return of the invested investments E_B :

$$E_B = T_{\text{ж}} \sqrt[4]{1 + \frac{E_{a\delta c}}{PV}} - 1,$$

where $E_{a\delta c}$ – the absolute effect from the potential implementation; $E_{a\delta c} = 682$ 000 UAH;

PV – the present value of the initial investments $PV = 385$ 000 UAH;

$T_{ж}$ development lifecycle, years.

In this case:

$$E_B = \sqrt[4]{1 + \frac{682}{385}} - 1 = \sqrt[4]{1 + 1,7714} - 1 = \sqrt[4]{2,7714} - 1 = 1,29 - 1 = 0,29 = 29,0\%.$$

Given magnitude $E_B = 29,0\% < \tau_{\text{мін}} = 42\%$, then a potential investor may not be interested in the commercialization of the development.

The calculations made in the form of graphs are presented in Figure 5.1.

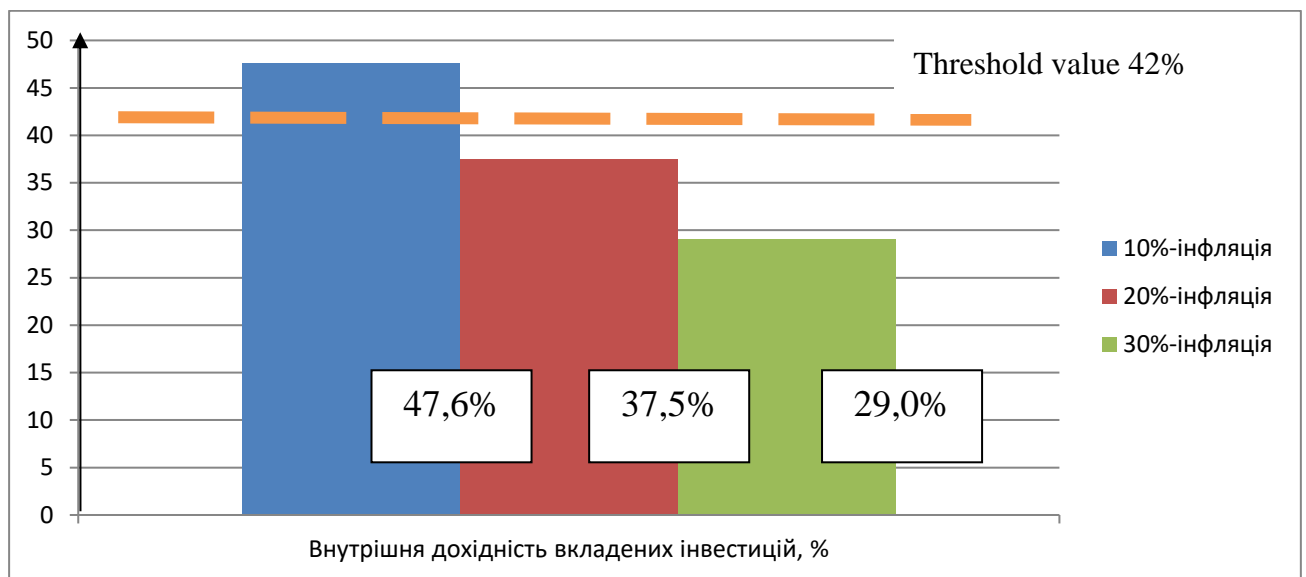


Figure 5.1 – Modeling the relationship between the internal rate of return of potential investments and the inflation rate in the country (10%, 20%, 30%, and 40%)

The analysis of the charts in Figure 5.1 shows that at an inflation rate of 10%, the internal rate of return of investments is $E_B = 47,6\% > \tau_{\text{мін}} = 42\%$, therefore, the commercialization of the development may be worthwhile. The analysis reveals that at an inflation rate of 20% and 30%, the internal rate of return of investments is

respectively $E_B = 37,5\%$ i $E_B = 29\%$ $\tau_{\text{minH}} = 42\%$, hence, the commercialization of the development can be in question. However, a final decision on this matter requires additional calculations (possibly reducing the investment risk level, increasing the demand for the development, enhancing the selling price of the development, etc.).

The outcomes of the performed economic part of the master's qualification theses are summarized in the Table 5.6:

Table 5.6 – Summary of financial analysis of master's qualification theses

Indicators	Defined in Technical task	Attained in the Master's thesis	Conclusion
1. Development expenses	Less than 80 000 UAH	77 000 UAH.	Achieved
2. Absolute effect from implementing the development, thousands of UAH	1400-1500 000 UAH	1442 000 UAH (with 10%-inflation)	Completed
3. Internal Rate of Return on Investments, %	More than 42%	47,6% (with 10%-inflation)	Achieved
4. Payback Period of Investments, years	Less than in 3 yrs.	2,10 yrs.	Completed

Therefore, the key technical and economic indicators of the developed the retail gamification system as defined in the technical task, have been achieved.

CONCLUSIONS

In this master's qualification work, a complete process of stateless DSS ecosystem development, with its further certification, analysis and optimization has been described, along with methods to apply aforementioned system to solving problems of commercial industries' gamification. An extensive analysis has rendered a development of a new DSS system, compliant with all the restrictions and limitations, and applicable to real-world problems as a viable and demanded option on the market. A full process of development and certification of cryptographically strong random numbers generator module has also been shown. All the complications and limitations have been described and explained in detail.

For a decision support module, a mathematical problem has been formalized and solved, providing a new method for implementing decision making algorithm compliant with requirements and able to operate in stateless environment. This method was then implemented into a program module via use of Java programming language, Spring Boot framework and accompanying libraries. As a result, a fully functional decision support system with random numbers generator module has been implemented and integrated into a real-world environment. Said system is designed to be stateless, shows good performance, is able to withstand high load, where it was tested against 10 000 concurrent requests. The system also provides a statistical module for extensive analysis of its statistical parameters. The results of the collected statistic can be put into Kibana for visualization of them, for convenient analysis and processing. Average amount of requests per second converges to 2 500.

To achieve the result, a number of steps were completed during the process. An extensive analysis of existing analogues has been conducted, showing up there are no direct competitors, as most services provide RNG and DSS as a part of their bigger services, built over said technologies. This have led to the conclusion that implementing a service providing bare RNG & DSS will have a great demand on the market. It have been also researched that implementing a number of built-in features, including a Web interface, a module for Web interface integration in customer's

content management system, a number of predesigned game types, along with segmentation possibilities, will be to a significant benefit. The exploration has also shown that one of the industries with most prospects would be a commerce sector. Thus, a gamification of purchasing process have been described as one of the main priorities. As a competitive edge, it has also been decided to make a billing on a per-transaction basis to encourage small and uncertain businesses to try out the solution. It has also been defined that making pricing a public information would greatly increase a number of potentially interested customers, as majority of competitors do not provide this data in any form unless contacted directly, severely limiting information spread.

Programming paradigms and tools were reviewed; those chosen for the development have been extensively described. The system was implemented with Java programming language, using Spring Boot framework as a basic one. Random numbers generation module has been implemented. All the limitations and regulations applied by law and by the business demand, including statelessness, were complied with. As a proof of the service is following all the rules and limitations, RNG has been certified by a British accredited body conducting audits and certifications. A part of a certification report with diehard statistical test compliance has been provided in a corresponding section.

A mathematical solution for following a general convergency yet keeping enough individual entropy has been developed and documented in detail. With that solution, a DSS system has been implemented. After an integration of RNG and DSS, a task of this work can be considered complete. During the process of performing this bachelor's thesis, all the tasks defined in the introduction were fulfilled, resulting in a fully compliant and working DSS system.

The system has been thoroughly analyzed and checked for bottlenecks and potential flaws. After a full review of the codebase and architecture diagrams and documentation, a number of points with potential to improve have been defined. For each of these, an analysis of the root of the problem along with multiple possible ways to solve them has been provided. Then, for each of the proposed improvement approaches, an evaluation has been given based on possibility to apply, viability to the

current state of the system, current level of introduction of such a method in the system, implementation complexity and performance improvement potential. The analysis has discovered that a lot of optimization methods proposed for application have already been applied to varying levels of extent. Nonetheless, a number of optimizations proposals have been defined as viable and implemented, with a reasoning for each provided in detail. The result of optimization process is an improvement in performance for single instance deployment of 70% on CPU usage. Optimization of inter-service communication and better request handling by orchestrator have led to average response time reduction to handle peak situations almost twice as fast, decreasing to 56% from prior metrics (exact numbers are still client- and environment-dependent, but testing conditions have shown change from 247 ms to 139 ms on average to remote AWS cluster). Architecture changes and orchestrator implementation have also provided a benefit of average savings in instance uptime of around 20%, taking into account resources utilized by orchestrator instances. The results are provided for a system with 2 orchestrator modules of AWS m7g.medium and 3 modules of jackpots with configuration of m5.xlarge, with total of 6 multilevel games being present simultaneously on the environment with 1 of jackpot modules only being active during peak periods (around 8 hours each day). The previous configuration was 6 full-time jackpot instances of same configuration (1 per game). The result of the database optimization is a reduction of the most CPU-heavy queries average load by waits metric from 0.79 to a 0.05 per active game (average CPU load, with database providing maximum possible 4 vCPUs for db.m5.large configuration). Data for comparison is taken from AWS RDS Performance insights dashboard.

The resulting product of this qualification work is a working gamification system which has been incorporated in a real business solutions and rendered itself as a successful one. The optimization have also provided significant boost in performance, leading to better customer satisfaction and saving costs on AWS resource billing.

LIST OF REFERENCED LITERATURE

1. Hurwitz, Judith. Smart or Lucky: How Technology Leaders Turn Chance into Success. John Wiley & Son, 2013. 164p. ISBN 978-1118033784.
2. ISO/IEC 27001:2013 Information technology — Security techniques — Information security management systems — Requirements. 2019. ISO, 23p.
3. ISO/IEC 18031:2011 Information technology — Security techniques — Random bit generation. ISO, 2020. 142p.
4. eCOGRA Certification Lab Homepage . eCOGRA ISO 27001 Certification Lab: website. URL: <https://ecogra.org/> (Date of access: 01.12.2023).
5. NIST. NIST's March 2006 Policy on Hash Functions. National Institute on Standards and Technology Computer Security Resource Center. 2006. URL: <https://csrc.nist.gov/projects/hash-functions/nist-policy-on-hash-functions> (Date of access: 01.12.2023).
6. FIPS 140-3 Security Requirements for Cryptographic Modules. National Institute of Standards and Technology. 2019, 11p.
7. Bogach I.V., Maliovanyi D.V. Забезпечення передбачуваності ймовірнісний рішень згідно закону великих чисел у системах із відсутністю стану. Proceedings of LI VNTU Conference of technical science, may 30-31, 2022. URL: <https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15620> (Date of access: 01.12.2023).
8. Daniel Johnson. What is Expert System in AI. Guru99: website. URL: <https://www.guru99.com/expert-systems-with-applications.html#1> (Date of access: 01.12.2023).
9. Investopedia. Decision Support System (DSS). Investopedia: website. URL: <https://www.investopedia.com/terms/d/decision-support-system.asp> (Date of access: 01.12.2023).
10. Cornelius T. Leondes. Expert systems: the technology of knowledge management and decision making for the 21st century. 2002. pp. 1–22. ISBN 978-0-12-443880-4.

11. George Marakas. Decision Support Systems In The 21st Century. Pearson College Div., 2002. 611p. ISBN 978-8120323766.
12. Lawtrust: Worldwide RNG Certification Company. Lawtrust: website. URL: <https://lawstrust.com/en/licence/gambling/rng-certificate> (Date of access: 01.12.2023).
13. BlueRibbon Homepage. BlueRibbon LTD: website. URL: <https://www.bluerbn.com/> (Date of access: 01.12.2023).
14. SoftSwiss Homepage. SoftSwiss: website. URL: <https://www.softswiss.com/licensing/> (Date of access: 01.12.2023).
15. Eckel Bruce. Thinking in Java 4th Edition. Pearson — Education, Inc., 2006. 1057p. ISBN: 0-13-187248-6.
16. Richardson L., Amundsen M. RESTful Web APIs. O'Reilly Media, Inc.: 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2013. 406p. ISBN: 978-1-449-35806-8.
17. Oracle. GraalVM Release Notes. Oracle Corporation: website. URL: https://www.graalvm.org/release-notes/22_1/ (Date of access: 01.12.2023).
18. Martin Fowler. Crossing Refactoring's Rubicon. Martin Fowler: website. URL: <https://martinfowler.com/articles/refactoringRubicon.html> (Date of access: 01.12.2023).
19. JetBrains. IntelliJ IDEA Homepage. JetBrains: website. URL: <https://www.jetbrains.com/idea/> (Date of access: 01.12.2023).
20. Richard Silverman. Git Pocket Guide: A Working Introduction. O'Reilly Media, 2013. 231p. ISBN 978-1449325862.
21. How Gambling Really Works. Addictions Foundation of Manitoba: website. URL: <http://getgamblingfacts.ca/how-gambling-really-works/> (Date of access: 01.12.2023).
22. What is House Advantage. Addictions Foundation of Manitoba: website. URL: <http://getgamblingfacts.ca/how-gambling-really-works/what-is-house-advantage/> (Date of access: 01.12.2023).

23. Robert C. Hannum. A Guide to Casino Mathematics. Gaming Studies Research Center, University of Nevada, Las Vegas., 2005. 12p.
24. Cabot A., Hannum R. Practical Casino Math, 2nd edition. Reno NV, Institute for the Study of Gambling & Commercial Gaming, University of Nevada., 2005. 247p. ISBN 0-942828-53-4.
25. What is volatility? Investopedia: website. URL: <https://www.investopedia.com/terms/v/volatility.asp> (Date of access: 01.12.2023).
26. Simon DeDeo. Bayesian Reasoning for Intelligent People. Social and Decision Sciences, Carnegie Mellon University & the Santa Fe Institute. 2018, 26p.
27. René Garciaa , Abraham Liouib, Patrice Poncetc. The Myth of Long Horizon Predictability: An Asset Allocation Perspective. Hau-Commissariat Au Plan., 2011. 68p.
28. Abraham Lioui, Patrice Poncet. Long horizon predictability: An asset allocation perspective. European Journal of Operational Research, Elsevier., 2019. 34p., pp.961 - 975. hal-03484415.
29. Yuvraj Gupta. Kibana Essentials. Packt Publishing, 2015, 206p. ISBN 978-1-784-39493-6.
30. Geogebra Homepage. Geogebra: website. URL: <https://www.geogebra.org/> (Date of access: 01.12.2023).
31. eCOGRA Homepage. eCOGRA Limited: website. URL: <https://ecogra.org/> (Date of access: 01.12.2023).
32. What is the OSI Model?. Cloudflare: website. URL: <https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/> (Date of access: 01.12.2023).
33. Kozlovskiy V.O., Les'ko O.Y., Kavetskiy V.V. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Vinnytsia, VNTU, 2021. 42 p.

APPENDICES

Appendix A (mandatory)**Technical task**

ЗАТВЕРДЖУЮ

Завідувач кафедри АІТ

д.т.н., проф. Олег БІСІКАЛО

« 12 » 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій»

08-31.МКР.008.02.000 ТЗ

Керівник: к.т.н., доц. каф. АІТ

_____ Ілона БОГАЧ

« 12 » жовтня 2023 р.

Розробив студент гр. ІСТ-22М

_____ Дмитро МАЛЬОВАНИЙ

« 12 » жовтня 2023 р.

1. Назва та галузь застосування

Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій. Інформаційні системи та технології. Галузь застосування: сфера торгівлі, сфера послуг, автоматизація процесів.

2. Підстава для розробки

Підставою для виконання роботи є наказ № 247 по ВНТУ від «18» 09 2023р., та індивідуальне завдання на МКР, затверджене протоколом № 1 засідання кафедри АІТ від «30» 08 2023р.

3. Мета та призначення розробки

Метою роботи є розробка програмного забезпечення для гейміфікації індустрії продажів задля оптимізації кількості продажів та управління фокусом продажів конкретних категорій чи позицій товарів шляхом заохочення споживачів.

4. Джерела розробки

- 1) Hurwitz, Judith. Smart or Lucky: How Technology Leaders Turn Chance into Success. John Wiley & Son, 2013. 164p. ISBN 978-1118033784.
- 2) Cornelius T. Leondes. Expert systems: the technology of knowledge management and decision making for the 21st century. 2002. pp. 1–22. ISBN 978-0-12-443880-4.
- 3) Richardson L., Amundsen M. RESTful Web APIs. O'Reilly Media, Inc.: 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2013. 406p.

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: ОС Ubuntu 20.04, процесор AMD Ryzen 3600, оперативна пам'ять 8 GB, місце на диску 100 MB доступного місця.

Методи дослідження:

В даній роботі використовуються методи аналізу, моделювання, класифікації, спостереження, прогнозування, експерименту та прагматичної моделі наукового дослідження.

Результати роботи програми: створення та редагування розіграшів; прийняття заявок на участь у розіграшах; визначення переможців; здійснення нарахування бонусів переможцям; надсилання нотифікації про виграш у розіграші.

6. Економічні показники

До економічних показників входять:

- витрати на розробку не більше 80 тис. грн
- приведена вартість прибутку за 3 роки 500 тис. грн.
- мінімальна дохідність не менше 42%
- термін окупності не більше 3 років

7. Стадії розробки

а) Аналіз предметної області та системи, що слід оптимізувати	<u>20.09 – 02.10</u>
б) Вибір інструментів розробки	<u>02.10 – 10.10</u>
в) Дизайн архітектури та імплементація компонентів системи	<u>11.10 – 01.11</u>
г) Аналіз та оптимізація системи	<u>02.11 – 13.11</u>
д) Економічна частина	<u>15.11 – 17.11</u>
е) Оформлення матеріалів до захисту МКР	<u>17.11 – 20.11</u>

8. Порядок контролю та приймання

Рубіжний контроль провести до «01» 11 2023 р.

Попередній захист МКР провести до «21» 11 2023 р.

Захист МКР провести до «18» грудня 2023 р.

Розробив студент групи ІСТ-22м _____ Дмитро МАЛЬОВАНІЙ

Appendix B (mandatory)

Graphical section

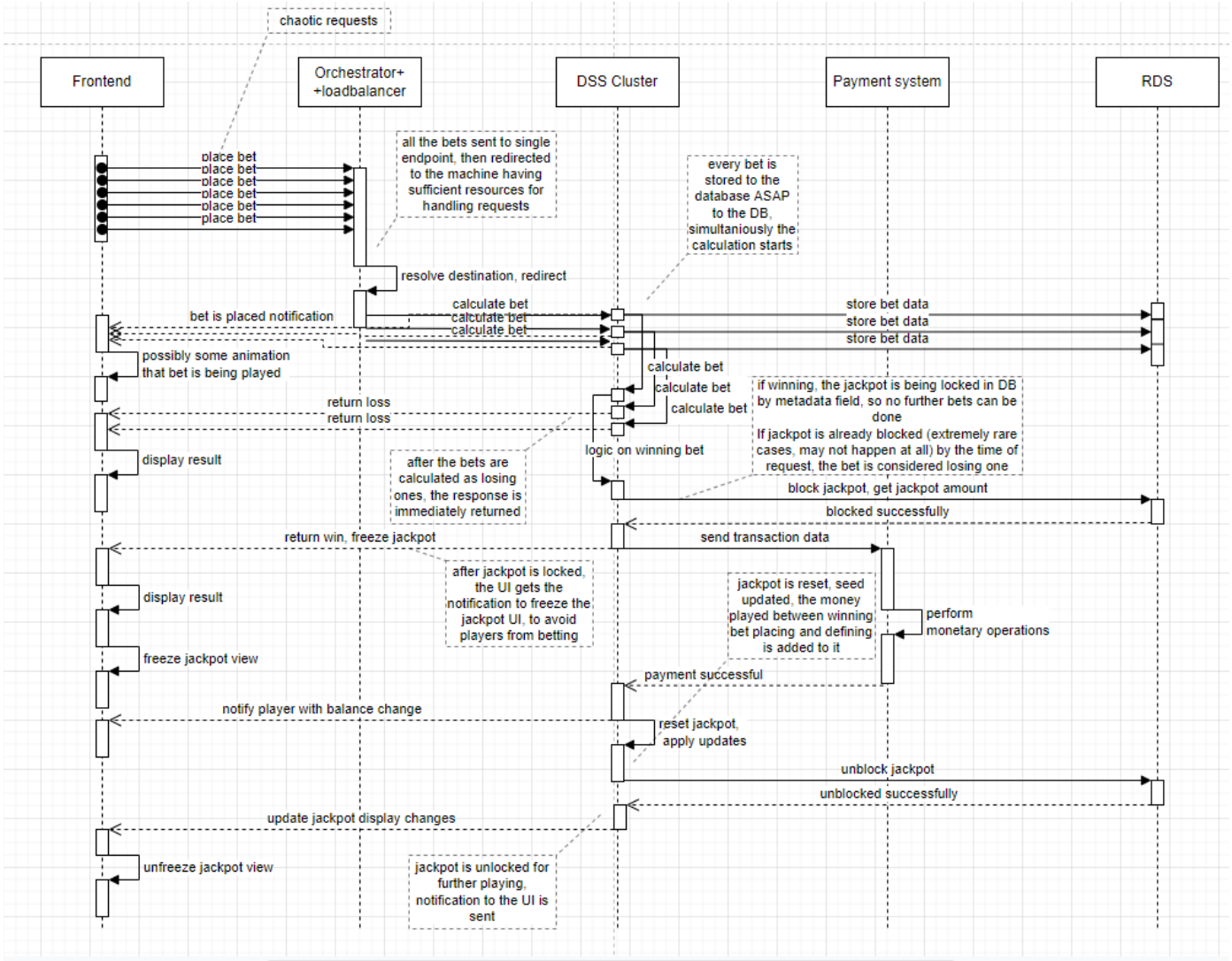


Figure B.1 – Workflow diagram of the DSS module and its application in the system

Continuation of Appendix B

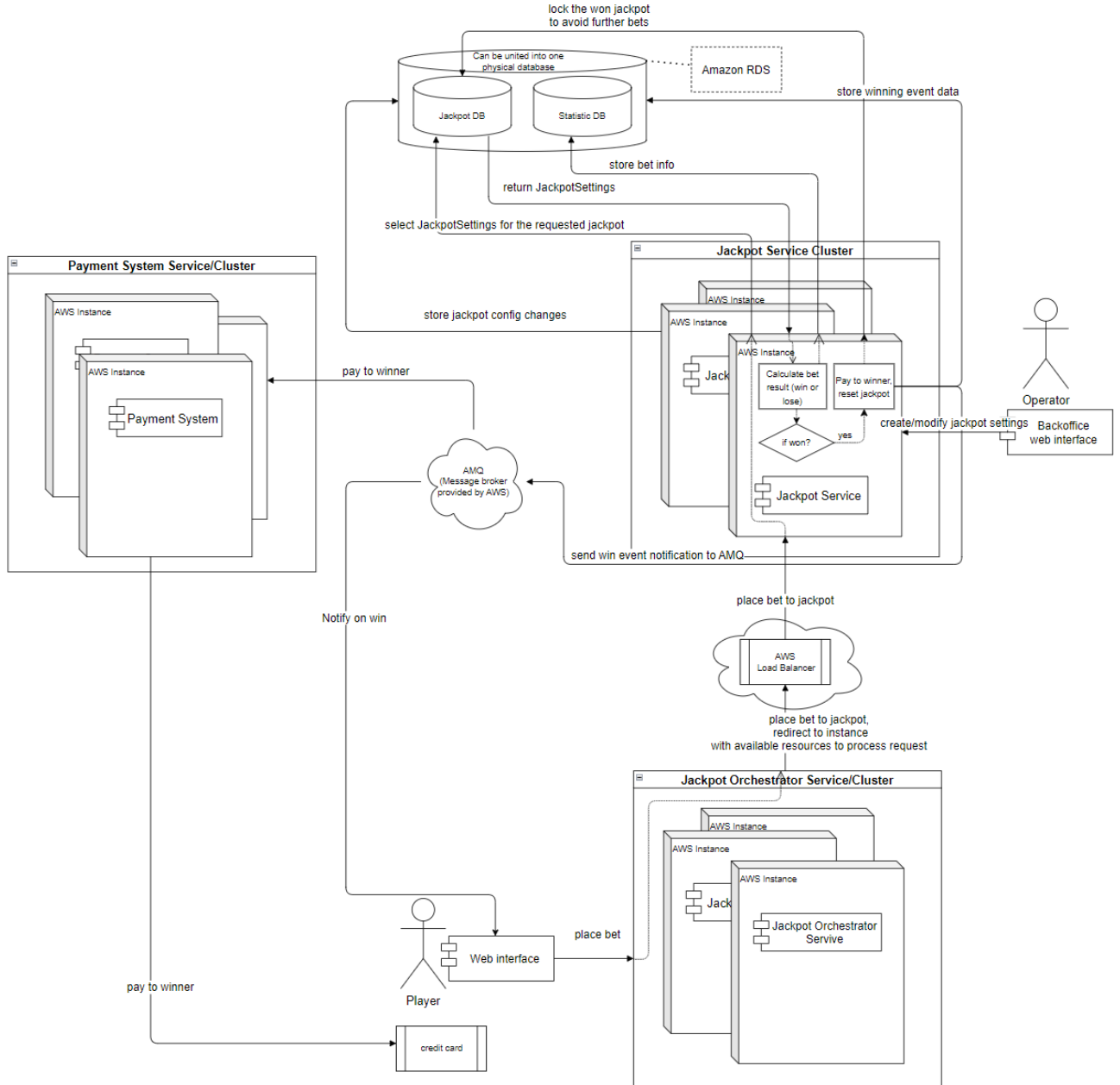


Figure B.2 – Component diagram of jackpot system for commerce gamification

Continuation of Appendix B

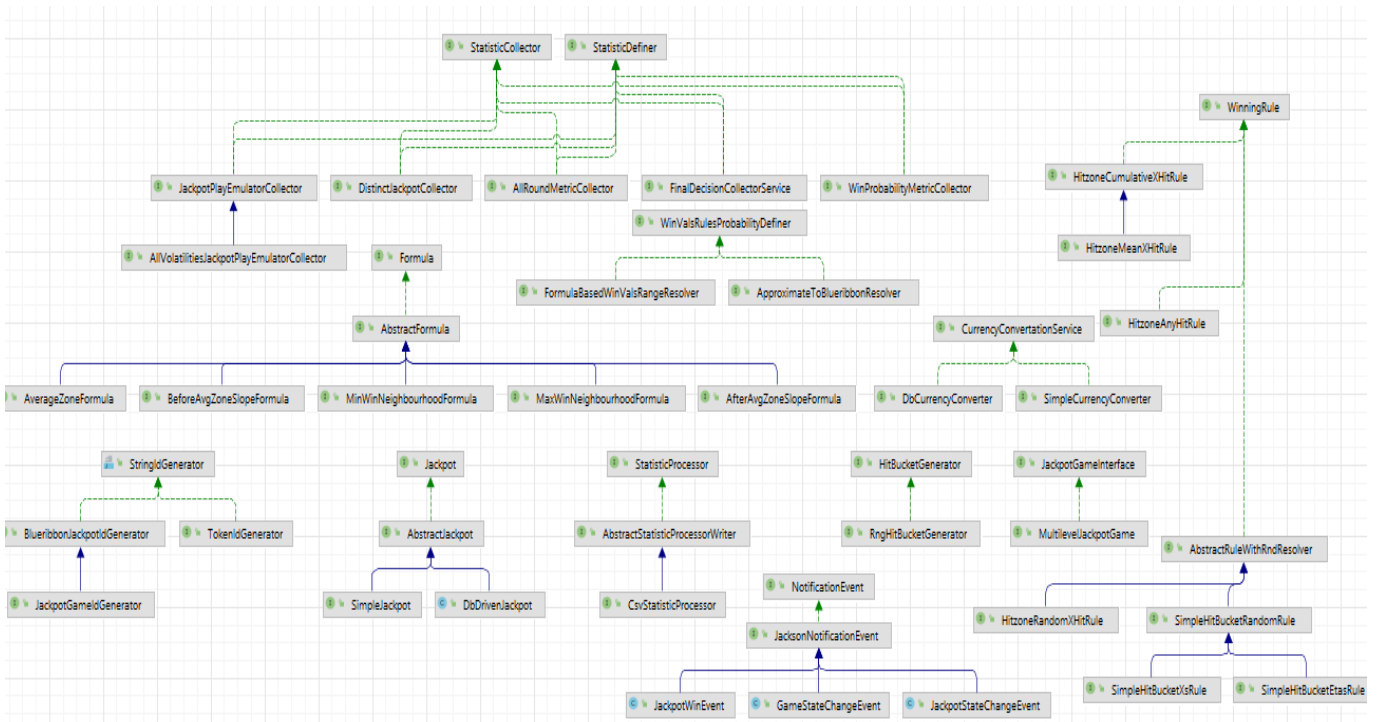


Figure B.3 – Class diagram of DSS module

Continuation of Appendix B

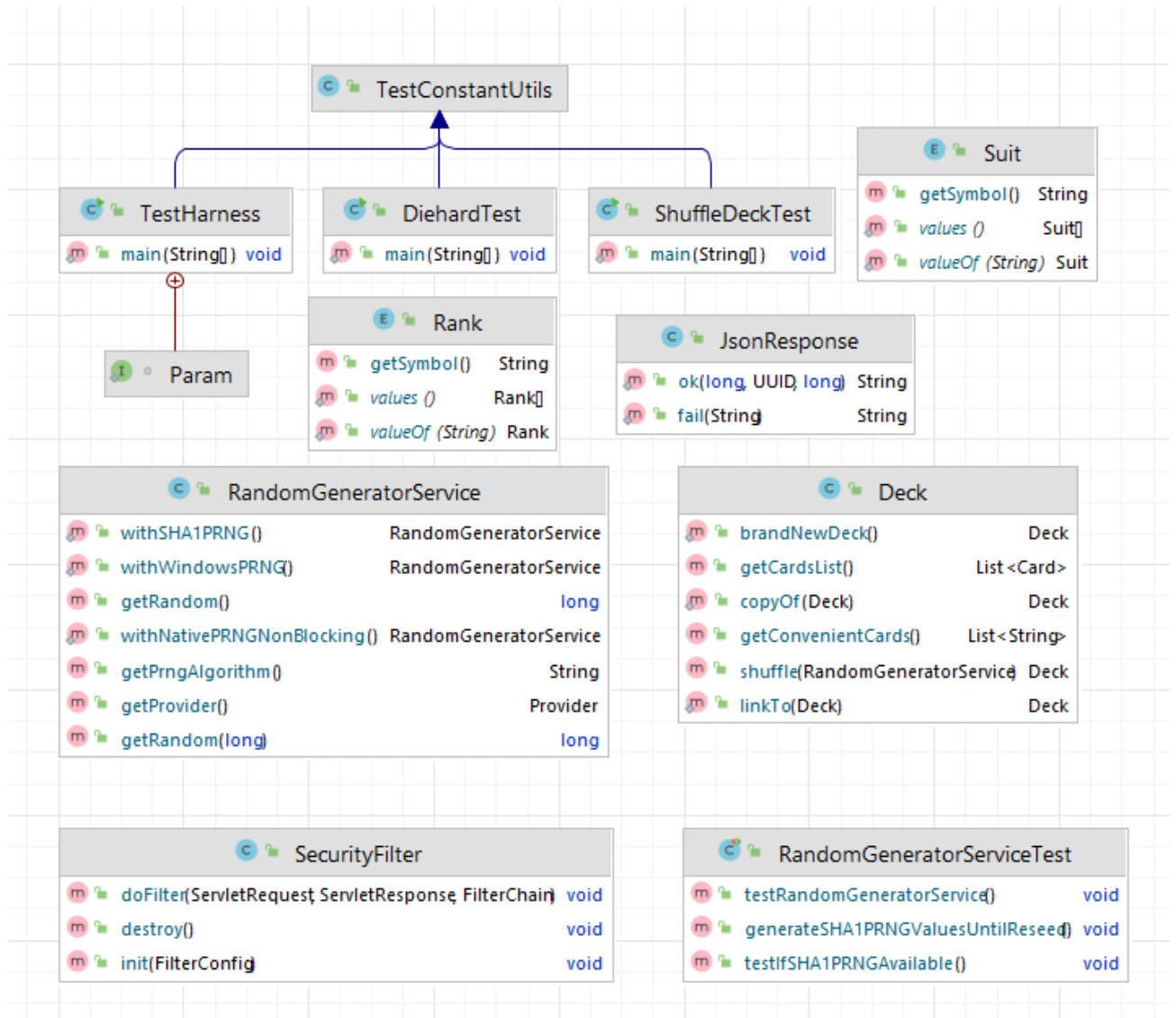


Figure B.4 – Class diagram of RNG module

Continuation of Appendix B

[+ Add New](#)

Update

<input type="checkbox"/>	Status	Name	Jackpot ID	Type	Created	Pot	Opt-ins	Wins	Actives	
	Active	First jackpot	z9x8c7v6b5n4	Time driven	02/12/2021	5,103,205\$	27743	21	16232	
	▼ Init	First jackpot	z9x8c7v6b5n4	Multilevel	02/12/2021		27743	21	16232	
<input type="checkbox"/>	Finished	First jackpot	z9x8c7v6b5n4	Single	02/12/2021		27743	21	16232	
	▼ Active	First jackpot	z9x8c7v6b5n4	Multilevel	02/12/2021	5,103,205\$	27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Single	02/12/2021	5,103,205\$	27743	21	16232	
	▼ Active	First jackpot	z9x8c7v6b5n4	Multilevel	02/12/2021	5,103,205\$	27743	21	16232	
	▼ Init	First jackpot	z9x8c7v6b5n4	Multilevel	02/12/2021	5,103,205\$	27743	21	16232	
<input type="checkbox"/>	▼ Finished	First jackpot	z9x8c7v6b5n4	Multilevel	02/12/2021	5,103,205\$	27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Single	02/12/2021	5,103,205\$	27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Single	02/12/2021	5,103,205\$	27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Time driven	02/12/2021	5,103,205\$	27743	21	16232	
<input type="checkbox"/>	Finished	First jackpot	z9x8c7v6b5n4	Single	02/12/2021		27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Single	02/12/2021	5,103,205\$	27743	21	16232	
	Active	First jackpot	z9x8c7v6b5n4	Single	02/12/2021	5,103,205\$	27743	21	16232	
	Init	First jackpot	z9x8c7v6b5n4	Single	02/12/2021		27743	21	16232	

1-15 out of 83 < 1 2 3 ... 6 >

Figure B.5 – Design of web component for monitoring and configuring games

Continuation of Appendix B

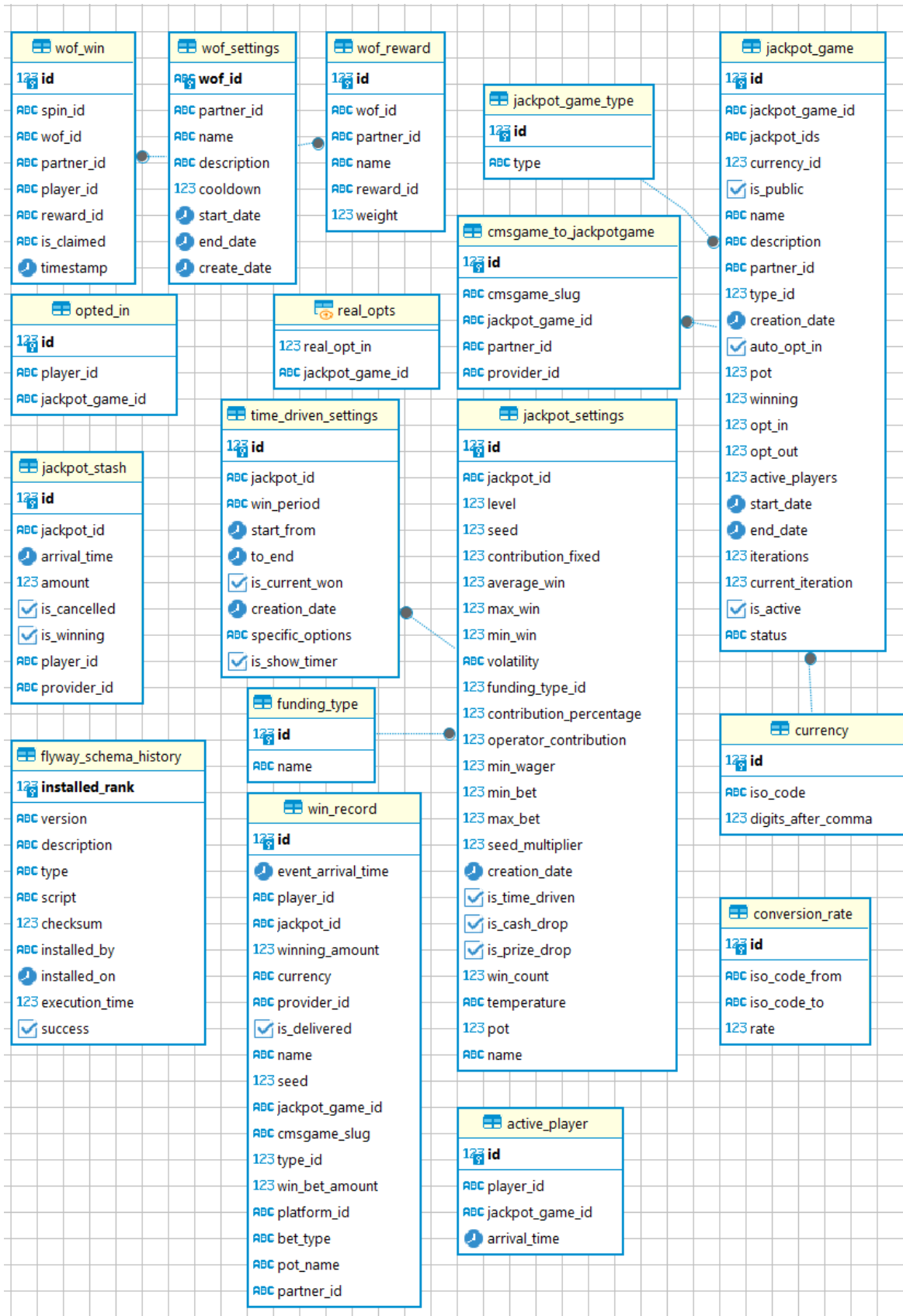


Figure B.6 – Entity relations diagram of the database

Appendix C (mandatory)

Code listing

```

@Slf4j
public class DbDrivenJackpot extends AbstractJackpot {

    private final JackpotStashService stashService;
    private final JackpotSettingsService jackpotService;
    private final JackpotCriteriaService criteriaService;
    private final CurrencyService currencyService;
    private final CurrencyConversionService conversionService;
    private final WinService winService;
    private final NotificationSender notificationSender;

    private volatile long lastStoredIteration;

    private final AtomicInteger activeStateCalls = new AtomicInteger(0);
    private final AtomicInteger jpSumUpdateCalls = new AtomicInteger(0);

    public DbDrivenJackpot(JackpotGame jackpotGame,
        JackpotSettings jackpotSettings,
        JackpotStashService stashService,
        JackpotSettingsService jackpotService,
        JackpotCriteriaService criteriaService,
        CurrencyService currencyService,
        @Qualifier("dbCurrencyConverter")
        CurrencyConversionService conversionService,
        WinService winService,
        NotificationSender notificationSender) {
        super(jackpotGame, jackpotSettings, null, jackpotSettings.getSeed(), true);
        lastStoredIteration = jackpotGame.getCurrentIteration();

        this.stashService = stashService;
        this.jackpotService = jackpotService;
        lastRetrievedJackpotSum = stashService.getCurrentJackpotSum(jackpotSettings.getJackpotId());

        this.criteriaService = criteriaService;
        this.currencyService = currencyService;
        this.conversionService = conversionService;
        this.winService = winService;

        this.notificationSender = notificationSender;
    }

    /**
     * Wins jackpot and sends notification on which sum by which player was won.
     * It also IMPLIES that the winning row has been put to the database.
     *
     * <p>//TODO: currently logic doesn't check if operations such as updating jackpot seed and other parameters have succeeded

```

```

*
* @param player player data who won the jackpot
* @return true if the jackpot win notification succeeded
*/
@Override
public boolean winJackpot(Player player) {
    log.info("Jackpot {} is won", jackpotSettings.getJackpotId());

    double winningAmount = getWinStash();
    jackpotSettings.setSeed(jackpotSettings.getSeed() * jackpotSettings.getSeedMultiplier());
    jackpotSettings = jackpotService.update(jackpotSettings);

    winService.saveWinRecord(getWinRecord(player, winningAmount));

    //Force update of the state and iteration
    updateActiveState();
    getCurrentJPSum();
    updateIteration();

    stashService.deleteAllButCurrentStashFor(jackpotSettings.getJackpotId());

    return sendWinNotification(player.getPlayerId(), winningAmount, jackpotGame.getCurrency());
}

/**
 * Calculates number of iterations as total of won bets for specific jackpot.
 */
protected final void updateIteration() {
    lastStoredIteration = winService.getIterationsCountFor(jackpotSettings.getJackpotId());
    jackpotGame.setCurrentIteration(lastStoredIteration);
}

protected final WinRecord getWinRecord(Player player, Double winningAmount) {
    WinRecord winRecord = new WinRecord();

    winRecord.setJackpotId(jackpotSettings.getJackpotId());
    winRecord.setCurrency(jackpotGame.getCurrency().getIsoCode());

    winRecord.setEventArrivalTime(new Date(System.currentTimeMillis()));

    winRecord.setPlayerId(player.getPlayerId());
    winRecord.setWinningAmount(winningAmount);

    return winRecord;
}

/**
 * Defines if the currently written row should be the winning one.
 *
 * @return true if the contribution is winning, false otherwise
 */
protected boolean contributionIsWinning() {
    return criteriaService.getBetResult(this) == HitOrMiss.HIT;
}

```

```

}

/**
 * Aggregates a sum of between the last and the penultimate winning rows,
 * last winning row inclusive, and adds to a seed amount.
 * //TODO: agree with customer, if they want to store a part of the won jackpot
 * //TODO: as a seed for a next one, if so, re-implement
 *
 * @return a sum that is won
 */
private double getWinStash() {
    double lastWonJackpotSum = stashService.getLastWonJackpotSum(jackpotSettings.getJackpotId());
    if (lastWonJackpotSum < 0) {
        return 0;
    } else {
        return lastWonJackpotSum + jackpotSettings.getSeed();
    }
}

/**
 * Returns the balance of the current jackpot.
 * Also fetches it from the database before returning.
 *
 * @return the most fresh balance of the jackpot
 */
@Override
public double getCurrentBalance() {
    //TODO: temporally activates once per 1000 requests (for hist data testing). Change to update every turn in prod
    getCurrentJPSumUpdateIterationsIfRelevant();

    return jackpotSettings.getSeed() + lastRetrievedJackpotSum;
}

/**
 * Updates private field responsible for the last stored jackpot sum (only sum of contributions, excluding seed).
 */
protected void getCurrentJPSumUpdateIterationsIfRelevant() {
    //TODO: rework with caching
    if (jpSumUpdateCalls.incrementAndGet() >= 10_000) {
        getCurrentJPSum();
        String jackpotId = jackpotSettings.getJackpotId();
        notificationSender.sendEvent(
            new JackpotStateChangeEvent(jackpotGame.getCurrency().getIsoCode(),
                jackpotGame.getJackpotGameId(),
                winService.getLastWinRecordForPot(jackpotId).getEventArrivalTime(),
                jackpotId,
                getCurrentBalance()));
    }
}

protected void getCurrentJPSum() {
    jpSumUpdateCalls.set(0);
    log.debug("Retrieving current jackpot sum for jackpot: {}", jackpotSettings.getJackpotId());
}

```

```

        lastRetrievedJackpotSum = stashService.getCurrentJackpotSum(jackpotSettings.getJackpotId());
        log.info("Jackpot: {}, iteration is: {}, sum is: {}",
                jackpotSettings.getJackpotId(), lastStoredIteration, lastRetrievedJackpotSum);
    }

/**
 * Updates operated fields of the Jackpot.
 * Updates only those fields that require to be the most fresh values.
 */
protected void updateJackpotSettingsToActualValues() {
    updateActiveStateIfRelevant();
    getCurrentJPSumUpdateIterationsIfRelevant();
}

/**
 * Changes `is_active` state of the jackpot, based on the result of { @link JackpotGame#checkInactivateRule()}.
 * Inactivates jackpot if it returns true, activates otherwise.
 * If the { @link JackpotGame#getIsActive()} already equals to the desired condition, no database update will occur.
 */
protected void updateActiveStateIfRelevant() {
    //TODO: rework with caching
    if (activeStateCalls.incrementAndGet() >= 100_000) {
        updateActiveState();
    }
}

protected void updateActiveState() {
    activeStateCalls.set(0);
    log.info("Retrieving active state for jackpot: {}", jackpotSettings.getJackpotId());

    boolean isActive = jackpotGame.getIsActive();
    updateJackpotActiveToOnCondition(!jackpotGame.checkInactivateRule(), isActive);
}

/**
 * Changes a value of `is_active` to a { @code valueToSet} if it is not already has such value.
 *
 * @param valueToSet value to set status to
 * @param currentValue condition value
 */
private void updateJackpotActiveToOnCondition(boolean valueToSet, boolean currentValue) {
    if (valueToSet != currentValue) {
        jackpotSettings = jackpotService.updateIsActive(jackpotSettings.getJackpotId(), valueToSet);
    }
}

/**
 * Sends notification via FeignClient to the auth service, which, in turn, sends it to all the frontend sockets in pool.
 *
 * @param playerId id of player who won
 * @param amount won amount
 * @param currency currency of won amount

```

```

* @return true
*/
@Override
protected boolean sendWinNotification(Integer playerId, double amount, Currency currency) {
    JackpotWinEvent jackpotWinEvent = new JackpotWinEvent(playerId,
        jackpotGame.getJackpotGameId(),
        jackpotSettings.getJackpotId(),
        amount,
        currency.getIsoCode(),
        false);

    try {
        notificationSender.sendEvent(jackpotWinEvent);
    } catch (RetryableException e) {
        log.error("Failed to send win notification! Event: {}. Stacktrace: ", jackpotWinEvent, e);
    }
    return true;
}

/**
 * Resulting Callable puts { @link JackpotStash } entity into a database
 * and returns { @literal true } if insertion succeeds.
 *
 * <p>{ @link JackpotStash } in this case has the following fields:
 * <ul>
 * <li>jackpot id</li>
 * <li>amount, which is defined as:
 * <ul>
 * <li>if jackpot funding type set to { @literal FIXED },
 * as a fixed amount stored in the jackpot config</li>
 * <li>if jackpot funding type set to { @literal PERCENTAGE },
 * as a specified percentage from the following bet</li>
 * </ul>
 * </li>
 * </ul>
 *
 * @param betRequest bet request received in request to the service
 * @return { @link Callable } of { @link Boolean } with function putting bet request
 * @throws UnsupportedCurrencyConversionException if conversion failed
 */
@Override
protected Callable<Boolean> processBetRequest(PlaceBetRequest betRequest) {
    updateJackpotSettingsToActualValues();

    if (!jackpotGame.getIsActive()) {
        log.warn("The jackpot {} is inactive and cannot accept new bets", jackpotSettings.getJackpotId());
        return getBetNotPlacedCallable();
    }

    Currency originCurrency =
        currencyService.getCurrencyByIsoCode(betRequest.getWagerDetails().getCurrencySymbol());
    Currency targetCurrency = jackpotGame.getCurrency();
    double originAmount = betRequest.getWagerDetails().getAmount();

```

```

double convertedAmount;

try {
    convertedAmount = conversionService.convertCurrency(originCurrency, targetCurrency, originAmount);
} catch (IllegalArgumentException e) {
    log.warn("Specified amount is not greater than 0! Message: {}", e.getMessage());
    return getBetNotPlacedCallable();
} catch (UnsupportedCurrencyConversionException e) {
    log.warn("Conversion operation not supported! Message: {}", e.getMessage());
    return getBetNotPlacedCallable();
}

try {
    jackpotSettings.checkAgainstBetConstraints(convertedAmount);
} catch (IllegalArgumentException e) {
    log.warn("Bet is not compliant with the constraints! Stacktrace: ", e);
    log.warn("Placed bet: {} {} was converted to: {} {} and didn't pass constraints! Stacktrace: ",
        originAmount, originCurrency.getIsoCode(), convertedAmount, targetCurrency.getIsoCode(), e);
    return getBetNotPlacedCallable();
}

JackpotStash jackpotStash = new JackpotStash();
jackpotStash.setJackpotId(jackpotSettings.getJackpotId());
jackpotStash.setPlayerId(betRequest.getPlayerDetails().getPlayerId());

if (jackpotSettings.getFundingType().getName().equals("FIXED")) {
    jackpotStash.setAmount(jackpotSettings.getContributionFixed());
} else {
    jackpotStash.setAmount(convertedAmount * jackpotSettings.getContributionPercentage());
}

//Temporal assignment is used to avoid non-atomic operation on volatile field
//noinspection UnnecessaryLocalVariable
double temp = jackpotStash.getAmount() + lastRetrievedJackpotSum;
lastRetrievedJackpotSum = temp;

if (contributionIsWinning() && !isJackpotBeingWon) {
    isJackpotBeingWon = true;
    lastRetrievedJackpotSum = 0;
    jackpotStash.setWinning(true);
}

return new JackpotStashPlacer(jackpotStash, betRequest.getPlayerDetails().getPlayerId());
}

protected final Callable<Boolean> getBetNotPlacedCallable() {
    return () -> false;
}

/**
 * Checks if the bet placed into is winning, and if it is, runs a winning sequence and updates jackpot.
 * If not winning, nothing is performed.
 *
 * @param jackpotStash jackpot stash to check against winning condition
 * @param playerId id of player

```



```

*/
protected void performWinIfRelevant(JackpotStash jackpotStash, Integer playerId) {
    if (jackpotStash.isWinning()) {
        winJackpot(-> playerId);
        isJackpotBeingWon = false;
    }
}

private volatile boolean isJackpotBeingWon = false;

/**
 * Supplementary wrapper on {@link Callable} of {@link Boolean}
 * to perform additional checks and logic on the entity placement into the database.
 */
@AllArgsConstructor
private class JackpotStashPlacer implements Callable<Boolean> {

    private JackpotStash jackpotStash;
    private Integer playerId;

    @Override
    public Boolean call() {
        try {
            performWinIfRelevant(stashService.placeJackpotContribution(this.jackpotStash), playerId);
            return true;
        } catch (IllegalArgumentException e) {
            return false;
        }
    }
}

}

}

@Slf4j
public abstract class AbstractJackpot implements Jackpot {

    @Getter
    protected final JackpotGame jackpotGame;
    protected ExecutorService executorService;
    @Getter
    protected JackpotSettings jackpotSettings;
    protected BetResolver betResolver;
    protected volatile double lastRetrievedJackpotSum;
    private LinkedBlockingDeque<PlaceBetRequest> pendingBets;
    private int executorCounter;
    private boolean isBetProcessingStarted = false;

    public AbstractJackpot(JackpotGame jackpotGame, JackpotSettings jackpotSettings,
        BetResolver betResolver, double lastRetrievedJackpotSum,
        boolean isToStartProcessingImmediately) {
        this.jackpotGame = jackpotGame;
        this.jackpotSettings = jackpotSettings;

```

```

this.betResolver = betResolver;
this.lastRetrievedJackpotSum = lastRetrievedJackpotSum;
this.executorService = Executors.newSingleThreadExecutor();

if (isToStartProcessingImmediately) {
    startBetProcessingCycle();
}
}

/**
 * A method to run in the background and put bets to database.
 */
protected final void startBetProcessingCycle() {
    if (!isBetProcessingStarted) {
        //actually this block will be activated at maximum only once for every instance normally
        isBetProcessingStarted = true;
        pendingBets = new LinkedBlockingDeque<>();
        //starts a new thread to operate on requests simultaneously
        new Thread() -> {
            ExecutorService betsProcessor = Executors.newCachedThreadPool();
            List<PlaceBetRequest> requests;
            //actually loop is infinite and persists until the object is destroyed
            while (true) {
                requests = new ArrayList<>();
                pendingBets.drainTo(requests, 50);
                try {
                    betsProcessor.invokeAll(requests.stream().map(req -> (Callable<Void>) () -> {
                        performSingleBetOperation(req);
                        return null;
                    }).collect(Collectors.toList()));
                } catch (InterruptedException e) {
                    log.error("Executor interrupted for jackpot: {} ! Stacktrace: ",
                        jackpotSettings.getJackpotId(), e);
                }
            }
        }).start();
    } else {
        log.warn("For jackpot: {}, the bet processing is already started!", jackpotSettings.getJackpotId());
    }
}

protected void performSingleBetOperation(PlaceBetRequest betRequest) {
    if (betRequest == null) {
        log.warn("Passed request is null!");
        throw new IllegalArgumentException("Null bet request cannot be processed!");
    } else {
        int showLiveEveryNRequests = 50_000;
        if (executorCounter++ >= showLiveEveryNRequests) {
            //TODO: remove if not needed. For now serves a purpose for correlation testing
            String humanReadableCount = String.valueOf(executorCounter);
            int offset = humanReadableCount.length() % 3;

```

```

        humanReadableCount = offset != 0
            ? (humanReadableCount.substring(0, offset) + "")
            : ""
            + String.join("", humanReadableCount.substring(offset).split("(?<=\\G\\d{3})"));
        log.info("Another {} requests accepted.", humanReadableCount);
        executorCounter = 0;
    }
    try {
        processBetRequest(betRequest).call();
    } catch (Exception e) {
        log.error("Unexpected error while handling bet request: {}. Stacktrace: ", betRequest, e);
    }
}

@Override
public Future<Boolean> bet(PlaceBetRequest betRequest) {
    return executorService.submit() -> pendingBets.add(betRequest);
}

/**
 * This method defines how to treat bet request received.
 * It must be implemented according to the logic and requirements of each specific implementation.
 *
 * @param betRequest bet request received in request to the service
 * @return {@link Callable} of {@link Boolean} indicating if the operation was successful
 */
protected abstract Callable<Boolean> processBetRequest(PlaceBetRequest betRequest);

protected void onSendMoneyFailover() {
    log.error("Failed to send money! Started failover!");
}

protected boolean sendWinnerInvoice(Player player) {
    return sendWinNotification(player.getPlayerId(), getCurrentBalance(), jackpotGame.getCurrency());
}

@Override
public BetEvent processBet(PlaceBetRequest betRequest) {
    return betResolver.resolveBet(betRequest);
}

protected boolean sendWinNotification(Integer playerId, double amount, Currency currency) {
    log.trace("Received arguments: {}, {}, {}", playerId, amount, currency);
}

/**
 * Represents an interface which logical entities of Jackpot should provide.
 *
 * @author dmaliiovanyi
 * @since 06.12.2021
 */
public interface Jackpot {

```

```

/**
 * Starts placing bet into specified {@link Jackpot}.
 * Returns true if request passed check and has been taken into processing. Placing the bet itself is asynchronous.
 *
 * @param betRequest event to place
 * @return future with boolean, which will be true if request passed restriction checks; false otherwise. Evaluated once a thread of this
event happened
 * @throws IllegalArgumentException if null passed
 * @see #processBet(PlaceBetRequest)
 */
Future<Boolean> bet(PlaceBetRequest betRequest);

/**
 * Performs logic to make client side notified about the jackpot is won,
 * and perform all the required actions to set a jackpot to a playable state.
 *
 * @param player player who won the jackpot
 * @return true if the operation succeeded
 */
boolean winJackpot(Player player);

/**
 * Processes bet request and returns the result of operation.
 *
 * @param betRequest bet request to process
 * @return resulting {@link BetEvent}
 */
BetEvent processBet(PlaceBetRequest betRequest);

double getCurrentBalance();

JackpotSettings getJackpotSettings();
}

```


Appendix D

Act of incorporation

pragmat.tech

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ "ПРАГМАТТЕК" / LLC "PragmatTech"
21036, Україна, м. Вінниця, вул. І. Бойка, буд.15, офіс 85; ЄДРПОУ 44420499;
pragmat.tech.ua@gmail.com

Затверджую
Директор ТОВ «ПРАГМАТТЕК»
О.О. Бабієць
«01» 12 2023 р.



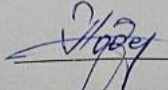
АКТ
Впровадження результатів магістерської дипломної роботи
Мальованого Дмитра Вадимовича на тему:
«Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій»

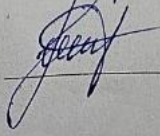
Комісія ТОВ «ПРАГМАТТЕК» у складі залучених фахівців з розробки програмного забезпечення Побережника Станіслава Васильовича та Пшеничної Галини Володимирівни розглянула магістерську дипломну роботу Мальованого Дмитра Вадимовича та встановила, що використання запропонованого програмного забезпечення, разом із покращеннями до існуючого програмного забезпечення, запропонованих у роботі, дозволяє втілити покращення до системи гейміфікації сфери продажів, збільшує ресурсоефективність використовуваної інфраструктури та пропонує кращу стійкість до навантаження і обробки ситуацій з піковим навантаженням.

Актуальність та практична цінність роботи Мальованого Д.В. не викликає сумнівів. Запропоноване програмне забезпечення дозволяє гнучкість подальшої розробки та підтримки та простоту інтеграції за рахунок архітектури, а також задовольняє усім поточним вимогам, передбачаючи можливість розширення функціоналу on demand.

Беручи до уваги вищевказані переваги, результати роботи було впроваджено до системи обслуговування бізнесів, що отримують від компанії послуги системи гейміфікації.

Члени комісії:

 С.В. Побережник

 Г.В. Пшенична

Appendix E (mandatory)
Plagiarism check protocol

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Архітектурна та функціональна оптимізація програмного модуля для імовірнісних розрахунків та гейміфікації для фінансових індустрій

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

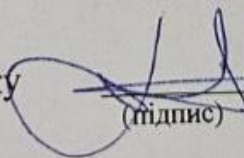
Показники звіту подібності Unicheck

Оригінальність 99.9% Схожість 0.1%

Аналіз звіту подібності (відмітити потрібне):

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку

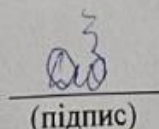


(підпис)

Роман МАСЛІЙ
(прізвище та ім'я)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

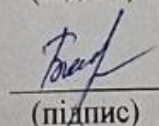
Автор роботи



(підпис)

Дмитро МАЛЬОВАНІЙ
(прізвище та ім'я)

Керівник роботи



(підпис)

Ілона БОГАЧ
(прізвище та ім'я)