Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Мобільний застосунок "Студентський аукціон"»

08-54.БКР.020.00.000.ПЗ

Виконав студент 2 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»
Шкамбула К. М.
(прізвище та ініціали)
Керівник к.т.н. доц. каф. ОТ
Дудник О. В.
(прізвище та ініціали)
Рецензент к.т.н. доц. каф. ПЗ
Чехместрук Р. Ю.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«11» 06 2025 р.

Вінниця ВНТУ — 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 «Інформаційні технології»
Спеціальність — 123 «Комп'ютерна інженерія»
Освітня програма — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«21» березня 2025 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шкамбулі Костянтину Миколайовичу

1 Тема роботи: «Мобільний застосунок "Студентський аукціон"», керівник роботи Дудник О. В. к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року № 97.

2 Термін подання студентом роботи 13.05.2025 р.

3 Вихідні дані до роботи: структуровані JSON-формати лотів, користувачів, ставок, формати запитів і відповідей REST API та Websocket-події для реалізації повідомлень в реальному часі.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної області та огляд аналогічних рішень мобільних аукціонів, дизайн інтерфейсу, технічна реалізація серверної та клієнтської частини мобільного застосунку, тестування додатку, висновки.

5 Перелік графічного матеріалу: структурна схема мобільного застосунку «Студентський аукціон», дизайн сторінок додатку.

6 Консультанти розділів роботи представлені у таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	к.т.н., доц. каф. ОТ Дудник О. В.	21.03.25	13.05.25
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.25	30.05.25

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва стапів кваліфікаційної роботи	Строк виконання	Підпис
1.	Постановка задачі роботи	21.03.25	вик.
2.	Аналіз предметної області	22.03.25 — 25.03.25	вик.
3.	Аналіз потреб користувачів	26.03.25 — 03.04.25	вик.
4.	Вибір технологій та інструментів для розробки	04.04.25 — 06.04.25	вик.
5.	Розробка структурної схеми	07.04.25 — 10.04.25	вик.
6.	Програмна реалізація мобільного застосунку	11.04.25 — 30.04.25	вик.
7.	Тестування мобільного застосунку	01.05.25 — 03.05.25	вик.
8.	Оформлення пояснювальної записки	04.05.25 — 12.05.25	вик.
9.	Попередній захист БКР, доопрацювання, рецензування БКР	13.05.25 — 04.06.25	вик.

Студент Костянтин ШКАМБУЛА

Керівник роботи к.т.н., доц. каф. ОТ Олександр ДУДНИК

АНОТАЦІЯ

УДК 004.42

Шкамбула К. М. Мобільний застосунок «Студентський аукціон». Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, освітня програма — Комп'ютерна інженерія. Вінниця: ВНТУ, 2025. 90 с.

На укр. мові. Бібліогр.: посил.: 26; рис.: 26; табл.: 5.

У бакалаврській кваліфікаційній роботі розроблено мобільний застосунок «Студентський аукціон», який дозволяє студентам брати участь в аукціоні. Проведено аналіз сучасних мобільних аукціонних рішень і визначено ключові потреби студентів як користувачів платформи. Дизайн-система додатку базується на темній палітрі з яскравими акцентами для основних дій, реалізовано єдині компоненти кнопок, форм і карток лотів. Для реалізації клієнтської частини використано React Native з Expo із застосування React Query для ефективного кешування та оновлення даних, а для серверної частини — NestJS із MongoDB. Проведено функціональне тестування платформи та адміністративної панелі. Результати підтверджують стабільність, продуктивність і надійність розробленого рішення.

Ключові слова: мобільний застосунок, аукціон, інтерфейс, реєстрація та авторизація, продуктивність, кешування, безпека, тестування.

ABSTRACT

Shkambula K. M. Mobile application «Student Auction». Bachelor's thesis in the specialty 123 - Computer Engineering, educational program - Computer Engineering. Vinnytsia: VNTU, 2025. 90 p.

In Ukrainian. Bibliography: ref.: 26; fig.: 26: table: 5.

In the bachelor's thesis, a mobile application "Student Auction" was developed, which allows students to participate in the auction. An analysis of modern mobile auction solutions was conducted and the key needs of students as platform users were identified. The application design system is based on a dark palette with bright accents for main actions, and single components of buttons, forms and lot cards were implemented. To implement the client part, React Native + Expo was used using React Query for effective caching and updating of data, and for the server part - NestJS with MongoDB. Functional testing of the platform and administrative panel was conducted. The results confirm the stability, performance and reliability of the developed solution.

Keywords: mobile application, auction, interface, registration and authorization, performance, caching, security, testing.

ЗМІСТ

ВСТУП

У сучасному інформаційному середовищі студентам часто бракує зручних та безпечних можливостей для купівлі й продажу навчальних матеріалів, техніки та інших корисних речей за вигідними цінами. Мобільна аукціонна платформа, адаптована під потреби студентів, здатна вирішити цю проблему, об’єднавши покупців і продавців в одному додатку, доступному у будь-який час. Використання мобільного додатку дозволяє не тільки прискорити комунікацію, а й автоматизувати процес виставлення лотів, прийому ставок та фінансових розрахунків, що підвищує ефективність та довіру користувачів.

Актуальність теми зумовлена зростанням популярності ринку б/в товарів та потреби студентів в недорогих та надійних інструментах для купівлі та продажу різного плану товарів. Також, поширення мобільних телефонів серед сучасного покоління та ситуація дистанційного навчання вимагає, щоб усі сервіси були зручними та доступними на смартфонах.

Метою даної кваліфікаційної роботи є створення сучасного мобільного застосунку «Студентський аукціон» на основі аналізу ринку та UX-рішень, який забезпечить інтуїтивне користування, безпечну клієнт-серверну архітектуру та високу продуктивність. Для досягнення цієї мети поставлено наступні **завдання**:

— провести аналіз існуючих мобільних аукціонних платформ;

— визначити ключові потреби студентів як цільової аудиторії;

— забезпечити зручний UX/UI-дизайн, адаптований до мобільних екранів студентів;

— розробити архітектуру клієнт-серверної взаємодії із застосуванням REST API та WebSocket для реального часу;

— визначити та реалізувати ключові модулі додатка;

— протестувати функціональність і продуктивність, а також оптимізувати безпеку й стабільність роботи.

Об’єктом дослідження є процес розробки мобільного застосунку для проведення онлайн-аукціону, а **предметом дослідження** — методи й технології створення клієнт-серверної архітектури, алгоритми обробки ставок, інтеграція Websocket для миттєвого обміну повідомленнями, а також засоби забезпечення зручності інтерфейсу та безпеки транзакцій.

Таким чином, дана бакалаврська робота спрямована на розробку сучасного мобільного застосунку «Студентський аукціон», який задовольняє потреби студентської спільноти у швидкому, зручному та безпечному методі купівлі-продажу товарів.

Апробацією результатів кваліфікаційної роботи є доповідь на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025).

Комп’ютерна онлайн-система проведення студентських аукціонів у мобільному додатку // Шкамбула К. М., Дудник О. В. Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2025) [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23859/19677 [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГИ

1.1 Огляд існуючих мобільних аукціонних платформ

У сучасному світі мобільні застосунки набули великого поширення. Зокрема, з'явилися рішення для проведення аукціонів в зручному режимі за допомогою смартфонів та інших популярних пристроїв. Саме тому важливо розглянути найпопулярніші платформи, щоб зрозуміти їхні сильні та слабкі сторони та врахувати це при розробці власного рішення.

eBay — це одна з наймасштабніших світових платформ електронної комерції та онлайн-аукціонів, що поєднує продавців і покупців із усього світу. Користувачі можуть як виставити свій товар на аукціон із можливістю торгуватися за найвигіднішою ціною, так і одразу придбати його за фіксованою «Buy It Now» ціною. Інтерфейс мобільного застосунку оптимізований для швидкого перегляду лотів, подачі ставок та відстеження своєї історії, зображено на рисунку 1.1 [2].

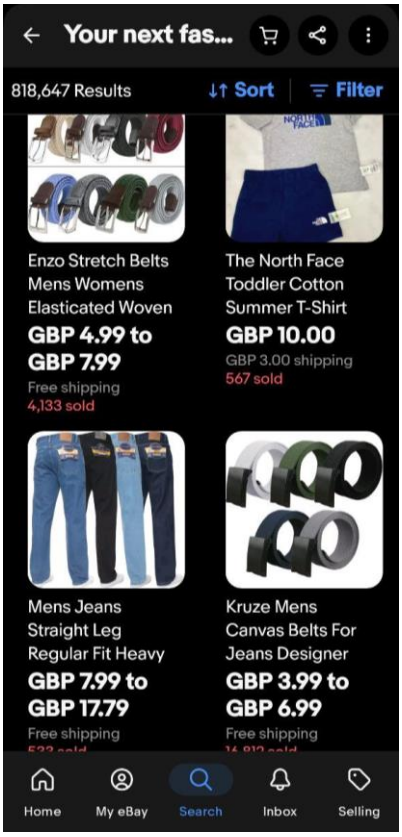

Рисунок 1.1 — Інтерфейс мобільного застосунку eBay

Основні переваги та недоліки платформи eBay наведено в таблиці 1.1.

Таблиця 1.1 — Переваги та недоліки платформи eBay

Переваги eBay	Недоліки eBay
Велика міжнародна аудиторія, що дозволяє знайти покупців із різних країн і отримати кращу ціну	Високі комісії (від 10 % до 12 %), що зменшують вигідність продажів для простих продавців
Інтегровані безпечні платіжні рішення (PayPal та інші), що гарантують захист транзакцій	Складна навігація через велику кількість категорій, що ускладнює пошук конкретних товарів
Програма гарантій компенсує у разі недоставки або несумісності товару з описом	Затримки доставки, особливо при міжнародних відправленнях

Аналіз застосунку eBay демонструє, що основою успішної платформи аукціонів є поєднання максимальної аудиторії користувачів із надійними інструментами оплати та захисту угод. Велика кількість активних покупців і продавців створює різноманітний ринок, де лоти швидко знаходять свій запит, а конкуренція стимулює підвищення цін. Інтеграція з перевіреними платіжними системами, такими як PayPal, гарантує не лише миттєве списання коштів, але й захист обох сторін через можливість відшкодування коштів в критичних ситуаціях.

Catawiki — це спеціалізована онлайн-платформа аукціонів, орієнтована передусім на колекціонерів та поціновувачів рідкісних предметів. Через мобільний додаток користувачі можуть брати участь у щотижневих торгах, переглядати відібрані лоти та купувати або продавати об’єкти від класичних коміксів і монет до антикварних годинників та витончених ювелірних виробів.

Інтерфейс Catawiki виглядає простим та інтуїтивно зрозумілим, зображено на рисунку 1.2. Головний екран представляє клієнту слайдер з найбільш цікавими лотами, швидкий доступ до категорій, а у «картці лоту» відразу видно основну інформацію про товар, з можливістю миттєво перейти до деталей чи зробити ставку [3]. Основні переваги та недоліки платформи Catawiki наведено в таблиці 1.2.

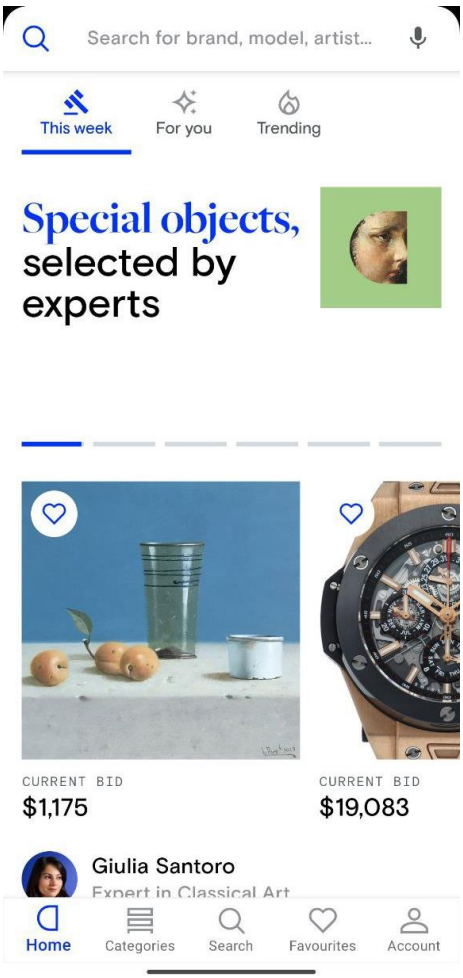

Рисунок 1.2 — Інтерфейс мобільного застосунку Catawiki

Таблиця 1.2 — Основні переваги та недоліки платформи Catawiki

Переваги Catawiki	Недоліки Catawiki
Якість та надійність лотів	Орієнтація на професійних колекціонерів
Професійна підтримка та консультації з питань зберігання й доставки колекційних предметів	Відносно високі комісії та додаткові витрати на експертну оцінку й організацію торгів
Експертна оцінка та гарантія автентичності	Високий поріг входу через велику стартову ціну лотів
Широкий асортимент товарів	Обмежене коло потенційних покупців через вузьку спеціалізацію платформи

Catawiki поєднує в собі найкращі практики класичних аукціонів із сучасним цифровим досвідом. Платформа пропонує ретельно відібрані лоти, кожен із яких проходить експертну оцінку, що забезпечує впевненість в

автентичності та якості товару. Таким чином, Catawiki ідеально підходить для досвідчених колекціонерів, але не завжди є оптимальним вибором для широкої аудиторії, яка шукає дешевші або більш універсальні лоти.

LiveAuctioneers — це онлайн-платформа для участі в прямих аукціонах у реальному часі по всьому світу, інтерфейс зображено на рисунку 1.3. Користувачі можуть робити ставки на унікальні лоти в режимі реального часу, отримувати миттєві оновлення про перебіг торгів і сповіщення про перебиття ставок. Платформа також забезпечує надійну обробку платежів та гарантію безпеки транзакцій [4]. Основні переваги та недоліки платформи LiveAuctioneers наведено в таблиці 1.3.

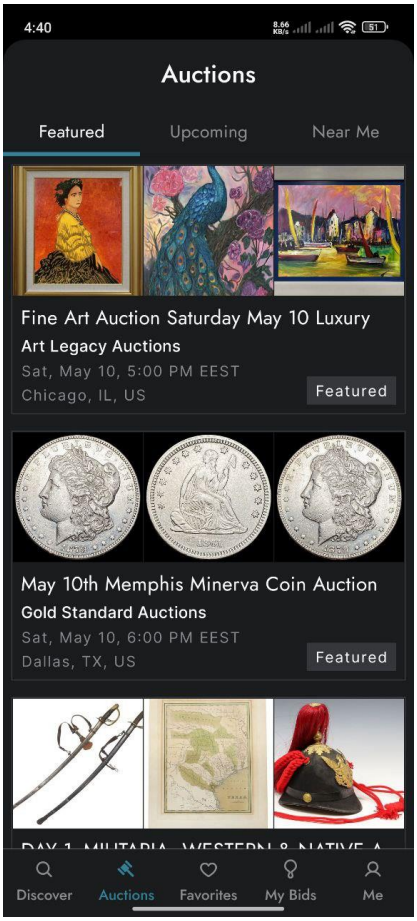

Рисунок 1.3 — Інтерфейс мобільного застосунку LiveAuctioneers

Отже, LiveAuctioneers поєднує переваги прямих торгів у реальному часі з широким вибором унікальних лотів і високим рівнем безпеки транзакцій, завдяки чому користувачі можуть брати участь у аукціонах, що відбуваються в

усьому світі, і миттєво отримувати сповіщення про перебиття ставок. Інтерфейс платформи простий у використанні: під час торгів оновлення ставок відображаються майже без затримок, а система захисту платежів гарантує, що кошти списуються лише після завершення угоди.

Таблиця 1.3 — Основні переваги та недоліки платформи LiveAuctioneers

Переваги LiveAuctioneers	Недоліки LiveAuctioneers
Прямі торги в реальному часі з відображенням усіх ставок онлайн	Висока конкуренція серед учасників
Унікальні лоти, часто рідкісні або антикварні речі	Різні комісії, що можуть підвищувати вартість торгів
Миттєві сповіщення про перебиття ставки та хід аукціону	Деякі аукціони вимагають мінімальної ставки для участі
Надійність і безпека платежів	Обмежена доступність у деяких регіонах

Спеціалізовані мобільні аукціони (StockX, GOAT та інші) фокусуються на вузьких категоріях товарів — кросівках, одязі, прикрасах та інших подібних предметах. Ключова особливість таких платформ це обов’язкова професійна перевірка кожної одиниці товару перед відправкою, що мінімізує ризики підробки товару і забезпечує максимально безпечні торги для обох сторін. Основні переваги та недоліки таких платформ наведено в таблиці 1.4.

Таблиця 1.4 — Переваги та недоліки спеціалізованих мобільних аукціонів

Переваги	Недоліки
Зручний та інтуїтивний інтерфейс	Обмежені категорії товарів
Обов’язкова професійна перевірка якості	Висока комісія за перевірку та обробку товару
Швидкі та надійні платежі	Не підходить для універсальних лотів

Незважаючи на різноманіття функціоналу, усі розглянуті платформи мають низку суттєвих обмежень, які не відповідають специфічним потребам студентської аудиторії.

По-перше, високі комісійні збори роблять ці сервіси малоефективними для користувачів із обмеженим бюджетом.

По-друге, складна навігація та велика кількість категорій у глобальних маркетплейсах ускладнюють пошук вузькопрофільних лотів, якими найчастіше торгують студенти.

По-третє, жодна з платформ не пропонує вбудованого real-time чату та локалізованих фільтрів за регіоном, що значно підвищило б довіру та швидкість угод у межах однієї локації.

Крім того, більшість існуючих сервісів розроблені для колекціонерів або професійних продавців і не передбачають швидких, локалізованих торгів із мінімальними кроками ставок та максимально спрощеним інтерфейсом, необхідних студентам для оперативної участі у невеликих аукціонних угодах.

Отже, мобільні аукціони розширюють традиційні формати торгів, що зацікавлює потенційних покупців. Найпопулярніші платформи поєднують простий інтерфейс, високу якість товару та надійність платежів. Під час розробки студентського аукціону важливо врахувати основні переваги та мінімізувати можливі недоліки.

1.2 Визначення потреб студентів як цільової аудиторії

Для успішного створення мобільного додатку студентських аукціонів необхідно чітко усвідомити бажання та очікування студентів як основних користувачів системи. Проаналізувавши наявні методи проведення торгів серед студентів і виявивши їхні ключові недоліки, можна виокремити основні потреби та вимоги до функціоналу.

Студенти, як правило, мають обмежений бюджет, тому для них важлива не лише доступність товарів за ціною, що не перевищує середньоринкову вартість, але й мінімальні комісії за проведення платежів, щоб загальна сума угоди залишалася максимально вигідною.

Оскільки сучасні студенти цінують час, мобільний застосунок має забезпечувати миттєву взаємодію без затримок: інтуїтивно зрозумілий інтерфейс, високу швидкість роботи та оновлення даних у реальному часі.

Враховуючи різноманіття пристроїв, на яких працюють студенти, додаток повинен коректно відображатися та однаково стабільно функціонувати на різних платформах і розмірах екранів, одночасно не навантажуючи ресурси смартфона.

Особливу увагу було приділено соціальній взаємодії та формуванню довіри між користувачами: система рейтингу продавців і відгуків після кожної угоди підвищує прозорість торгів і зменшує ризики шахрайства, а вбудовані локаційні фільтри дозволяють орієнтуватися на лоти від студентів із університетів розташованих поряд.

Для оперативного обговорення деталей використовується внутрішній чат, що запобігає необхідності переходу в сторонні месенджери.

Невід’ємною частиною зручності також є можливість гнучкого та розширюваного опису лотів: фотографії, детальні характеристики товару й чітка структура інформації допомагають покупцям приймати зважені рішення.

Отже, можна підсумувати, що мобільний аукціон для студентів має бути зручним, швидким та безпечним, зосереджений на економію коштів та комфортне користування додатком з повною довірою до торгів.

1.3 Вимоги до застосунку

Для коректної роботи студентського аукціону та задоволення виявлених потреб студентів, як основних користувачів системи, застосунок має реалізувати відповідні вимоги. Спочатку розглянемо функціональні вимоги.

Для коректної роботи мобільного аукціону користувачі реєструються за допомогою електронної пошти, а у своєму профілі можуть змінювати ім’я користувача, пароль та зображення.

Створення та управління лотами виконано через просту форму, у якій продавець задає заголовок, опис, початкову ціну, завантажує фотографії та вказує додаткові параметри товару; за бажанням лот можна видалити.

Подача ставки здійснюється з урахуванням мінімального кроку й поточної найвищої ціни — система відхиляє некоректні значення й миттєво оновлює інформацію про найвищу пропозицію.

Щоб полегшити пошук цікавих лотів, користувачі можуть додавати їх до «обраного» та знімати з цього списку в будь-який момент.

Протягом торгів застосунок надсилає сповіщення про перебиття ставки, завершення аукціону та інші важливі події, в профілі відображаються всі зроблені й отримані ставки, а також створені й завершені лоти.

Значну увагу приділено продуктивності: дані завантажуються та оновлюються з мінімальною затримкою, а кешування локально знижує навантаження на мережу.

Надійність і масштабованість реалізовані через обробку запитів у реальному часі та стійкі механізми збереження транзакцій, а для захисту облікових записів і передачі даних використовуються JWT-токени та HTTPS.

Інтерфейс додатка адаптовано під Android та iOS із урахуванням різних розмірів екранів, тому він стабільно працює на широкому колі пристроїв.

Навіть за нестабільного інтернет-з’єднання користувачі можуть ознайомитися з даними, а зрозуміла навігація та мінімалістичний дизайн гарантують інтуїтивно зрозумілий досвід як для новачків, так і для досвідчених користувачів.

Відповідність всім вимогам забезпечить створення надійного, зручного та безпечного мобільного аукціону, який задовольнить потреби студентів як у ролі покупців, так і продавців.

1.4 Проблеми та виклики на ринку мобільних аукціонів

Ринок мобільних аукціонів стрімко розвивається, однак разом із новими можливостями він стикається з низкою суттєвих обмежень та труднощів.

Однією з головних проблем є технічні вимоги й обмежені ресурси користувацьких пристроїв, що можуть призвести до повільного завантаження зображень, затримок у оновленні ставок та збоїв у роботі інтерфейсу. Для

вирішення цього завдання необхідно впроваджувати поступове завантаження контенту, оптимізувати розмір медіафайлів та використовувати локальне кешування, аби мінімізувати залежність від пропускної здатності мережі.

Економічна складова робить студентів особливо чутливими до розміру комісій. Високі збори за обробку платежів або додаткові сервісні платежі можуть зробити угоди невигідними, що знизить активність аудиторії. Платформі слід дотримуватися оптимального балансу: забезпечити сталий дохід для розробника, водночас утримувати сумарні витрати користувача на мінімальному рівні.

Питання довіри й безпеки стоять гостро: студентські спільноти часто не мають усталених відносин, тому будь-який негативний досвід (шахрайство, несправний товар, затримка з оплатою) швидко поширюється серед потенційної аудиторії. Важливо впровадити багаторівневу верифікацію користувачів, систему оцінок та відгуків, прозорі правила повернення коштів і механіку внутрішнього чату для уточнення деталей угоди.

Відповідність правовим та фінансовим нормам у різних країнах вимагає гнучкого підходу: обробка персональних даних повинна відповідати місцевим законам, а інтеграція платіжних шлюзів враховувати регуляторні обмеження на переказ коштів та комісії.

Жорстка конкуренція з боку неймовірно популярних платформ (eBay, OLX) та локальних стартапів змушує постійно шукати відмінні переваги — локалізацію для конкретних університетів, інтеграцію з внутрішніми студентськими чатами, гейміфікацію застосунку, ексклюзивні механізми захисту транзакцій.

Успішному входу на цей ринок сприятиме комплексний підхід до технічної оптимізації, зваженої монетизації, посилення система довіри та юридичної відповідності додатку, а також постійне залучення аудиторії через унікальні сервіси та партнерства в межах студентських спільнот.

Отже, ринок мобільних аукціонів стикається із рядом взаємопов’язаних викликів — від технічних обмежень і оптимізації роботи в умовах мобільного

зв’язку до формування довіри та дотримання регуляторних норм. Використання сучасних підходів до кешування, оптимізації зображень і реал-тайм технологій дозволяє уникнути дані перешкоди.

1.5 Обґрунтування вибору технологій

Розробка мобільного застосунку студентських аукціонів потребує сучасного й ефективного технологічного стеку, який забезпечить максимально надійну, швидку розробку, високу продуктивність та простоту для подальшої підтримки проєкту. З огляду на ці вимоги, для клієнтської частини обрано React Native (Expo), для серверної — NestJS, а також інші допоміжні технології (Zustand, Typegoose, ReactQuery тощо). У цьому підрозділі розглянемо обґрунтування вибору технологій та їх переваги.

1.5.1 Вибір клієнтської платформи (React Native + Expo)

React Native — це відкритий фреймворк від Facebook, який дає змогу писати мобільні застосунки одразу під iOS і Android, використовуючи знайомі веб-технології. Він побудований компонентній моделі React, тому розмітка UI також описується у вигляді JSX-компонентів. Замість HTML-елементів застосовуються справжні нативні компоненти (View, Text, TextInput тощо), що гарантує таку ж поведінку, ніби інтерфейс створено на Swift чи Kotlin [5].

JSX (JavaScript XML) — це розширення синтаксису JavaScript, яке дозволяє писати структуру UI-компонентів у вигляді схожого на HTML коду в JavaScript-файлах. Під час компіляції JSX перетворюється на відповідні React-елементи, завдяки цьому опис інтерфейсу стає більш зрозумілим і декларативним [6].

Для написання коду на React Native використовується JavaScript — динамічна мова програмування зі стандартом ECMAScript, що підтримується в усіх сучасних браузерах і середовищах Node.js. Саме JavaScript відповідає за логіку застосунку, обробку подій та взаємодію з серверними запитами[7].

React — бібліотека від Facebook, забезпечує механізм створення й оновлення компонентів через віртуальний DOM [8]. У React Native віртуальний DOM адаптований до мобільної платформи, але принципи роботи залишаються однаковими.

Щоб уникнути складнощів з налаштуванням середовищ розробки, обрано Expo — надбудову над React Native, яка з одного боку приховує тонкощі конфігурації Xcode і Android Studio, а з іншого надає готові модулі й інструменти для швидкої збірки та публікації застосунку [9].

Основні переваги використання React Native (Expo):

— крос-платформеність, що дозволяє одним кодом створювати одночасно iOS та Android додатки;

— використання нативних компонентів (View, TextInput, Text тощо), завдяки чому інтерфейс працює плавно;

— швидкий цикл розробки з Expo Go, який миттєво відображає зміни в коді на реальному пристрої;

— наявність готових модулів у Expo SDK (камера, геолокація, Push-сповіщення, медіа-обробка);

— розвинена екосистема та велика спільнота React Native і Expo з безліччю готових компонентів, плагінів і рішень на GitHub і форумах;

— повна підтримка TypeScript, яка забезпечує сувору типізацію, автодоповнення в IDE та виявлення помилок під час розробки;

— інтегровані інструменти CI/CD через Expo EAS Build та EAS Submit для автоматизації збірки, тестування та публікації мобільних додатків.

Таким чином, поєднання React Native із Expo забезпечує баланс між швидким стартом та надійністю розробленого застосунку, дозволяючи зосередитися саме на розробці додатку, а не налаштуванні нативних компонентів та інших базових речей. Завдяки цьому, гарантується надійність, масштабованість та легкість в підтримці розробленого проєкту.

1.5.2 Вибір серверного середовища (NestJS)

Враховуючи необхідність чіткої модульної архітектури, підтримки Typescript для строгої типізації та зручної роботи з REST-інтерфейсами було вирішено використовувати NestJS.

REST (Representational State Transfer) — це архітектурний стиль для побудови сервісів, що базується на ресурсно-орієнтованій моделі обміну даними. Кожен ресурс має унікальний шлях і доступний через стандартні HTTP-методи [10].

HTTP (Hypertext Transfer Protocol) — це протокол прикладного рівня для обміну даними в мережі Інтернет за моделлю «запит-відповідь». Клієнт надсилає серверу HTTP-запит із вказаним методом і адресою ресурсу, а сервер повертає статусний код та тіло відповіді [11].

NestJS — це прогресивний Node.js-фреймворк, побудований поверх Express з повною інтеграцією TypeScript та архітектурними патернами, притаманними великим корпоративним додаткам. Він забезпечує декларативний підхід через декоратори для контролерів, сервісів, модулів та провайдерів, автоматично вирішує залежності та дозволяє структурувати код у чіткі, ізольовані блоки [12].

Основні переваги використання NestJS:

— модульна архітектура із чітким розділенням на модулі, сервіси та контролери, що спрощує масштабування та підтримку коду;

— повна підтримка TypeScript для суворої типізації, автодоповнення у редакторах коду та виявлення помилок;

— декоратори (@Controller, @Injectable, @Module) для декларативного опису бізнес-логіки, що підвищує читабельність і спрощує навігацію по проєкту;

— вбудовані засоби для створення REST-ендпоінтів та real-time комунікації через WebSocket (@nestjs/websockets), що дозволяє швидко налаштувати API та обмін даними в реальному часі;

— висока продуктивність завдяки роботі поверх Express і оптимізованій обробці HTTP-запитів;

— широка спільнота та екосистема з сотнями плагінів, розширень та прикладів на GitHub.

API (Application Programming Interface) — це набір визначених методів і правил взаємодії між різними програмними компонентами чи сервісами. За допомогою API клієнт може надсилати запити до сервера, отримувати дані або виконувати певні операції, такі як створення, читання, оновлення, видалення тощо [13].

Таким чином, NestJS є потужним інструментом для побудови надійного, масштабованого та тестованого бекенду, поєднуючи популярні практики розробки з простотою та гнучкістю в написанні коду.

1.5.3 Вибір бази даних (MongoDB)

При виборі бази даних важливим елементом було гнучкість моделі даних та простоту в інтеграції з Javascript-стеком, тому було прийняте рішення використовувати саме базу даних MongoDB.

MongoDB — це NoSQL СУБД, що зберігає інформацію у вигляді JSON-подібних документів, дозволяючи гнучко змінювати структуру даних без складних міграцій та забезпечуючи високу швидкість операцій читання і запису [14].

Основні переваги використання MongoDB:

— гнучка схема даних, що дозволяє документам містити довільний набір полів і легко змінювати структуру без складних міграцій;

— висока продуктивність завдяки відсутності JOIN-операцій і зберіганню всього об’єкта в одному документі;

— MongoDB можна розгорнути на багатьох серверах одночасно, розподіливши навантаження й уникаючи затримок при великій кількості запитів;

— підтримка транзакцій, яка гарантує цілісність даних;

— розвинена екосистема інструментів: Mongoose (Typegoose) для зручної роботи з TypeScript, графічний інтерфейс Compass, а також хмарний сервіс Atlas із вбудованим моніторингом і автоматичними бекапами;

— швидке розгортання й гнучкі тарифи: безкоштовні локальні кластери для розробки та можливість поступового масштабування в Atlas без змін у коді.

Отже, вибір MongoDB як основи зберігання даних для «Студентського аукціону» показує себе якнайкраще з точки зору гнучкості й продуктивності. Інтеграція з TypeScript через Typegoose спрощує розробку й перешкоджає потенційним помилкам типізації. Завдяки широкому набору інструментів і можливості розгортання як локально, так і в хмарі Atlas, MongoDB є ефективним рішенням для підтримки подальшого зростання та розвитку застосунку.

2 ДИЗАЙН ІНТЕРФЕЙСУ

Розробка дизайну мобільного застосунку є одним із найголовніших етапів створення проєкту. Сучасні користувачі звикли до надзвичайно креативних рішень, тому дизайн має бути інтуїтивно зрозумілим та максимально привабливим, що забезпечить зручну взаємодія користувача з основною логікою платформи. Всі макети розроблено в Figma — потужному інструменті для прототипування та спільної роботи над дизайном [15].

2.1 Кольорова палітра

Кольорова палітра мобільного застосунку відіграє ключову роль у формуванні візуального сприйняття, встановленні емоційного зв’язку з користувачем та забезпеченні читабельності інтерфейсу. Під час розробки дизайну студентського аукціону було обрано стриману темну тему з яскравими акцентами, що поєднують сучасний вигляд і зручність використання як у денних, так і у нічних умовах. Темний фон дозволяє знизити навантаження на зір у погано освітленому середовищі та підкреслити вміст карток лотів, тоді як акцентні кольори направляють увагу на ключові елементи управління та дії користувача [16].

Основний колір використовується для фону основних екранів і становить насичений глибокий чорний (#0B0C10). Він забезпечує високу контрастність із тілом тексту та іншими графічними елементами й створює основу для виділення карток, полів вводу та кнопок. Допоміжний колір — темно-сірий, використовується для підкреслення на основному фоні поверхонь карток лотів, панелей навігації, коментарів тощо. Різниця в яскравості між першим і другим шарами фону надає інтерфейсу відчуття глибини й ієрархічної структури, допомагаючи користувачеві швидко орієнтуватися на екрані.

Акцентні кольори вибрані з урахуванням їхнього психологічного впливу та асоціацій із фінансовими та торговими операціями. Основним акцентом слугує насичений зелений (#006E00), який застосовується для кнопок «Створити лот», «Зробити ставку», а також для підсвічування активних

елементів навігації. Цей відтінок символізує позитивні дії, зростання та успіх, спонукаючи користувача до взаємодії. Як допоміжний акцент використовується червоний (#E53935) для підкреслення інформаційних повідомлень, пов’язаних із помилками, відхиленням ставок, недоступністю операцій та видалення лотів.

Текстові елементи виконано в білому кольорі (#FFFFFF) та світло-сірому (#C5C6C7) для менш важливої інформації. Білий текст на чорному тлі гарантує максимальну читабельність основного контенту, тоді як світло-сірий застосовується для підзаголовків, допоміжних підписів і позначень стану. Різниця яскравості тексту допомагає створити візуальний ієрархічний порядок, не відволікаючи при цьому увагу від ключових елементів управління.

Всі компоненти мають визначені правила застосування відтінків, наприклад, кнопки за замовчуванням — великі, із заокругленими краями та тінню, акцентний зелений фон із білим текстом, при натисканні фон затемнюється. Поле вводу має тонку сіру рамку і світло-сірий плейсхолдер, що підкреслює область заповнення інформації, при помилці в даному полі, з’являється текст-підказка червоного кольору.

У підсумку, розроблена кольорова палітра мобільного застосунку «Студентський аукціон» забезпечує єдність стилю, виразність ключових дій і комфортну взаємодію в різних умовах. Завдяки ретельно підібраним відтінкам та дотриманню стандартів доступності, інтерфейс залишається інтуїтивно зрозумілим, привабливим і функціональним для широкої аудиторії студентів.

2.2 Базові компоненти застосунку

Першим етапом розробки дизайну будь-якого додатку є визначення і реалізація базових компонентів, тобто таких, які будуть використовуватися на багатьох екранах.

Жодна платформа не існує без кнопки, це компонент завдяки якому користувач може здійснити певні маніпуляції, наприклад перейти на інший екран, відкрити модальне вікно, підтвердити власну дію тощо. Розроблена кнопка може бути двох різних розмірів: велика і маленька, а також трьох різних

кольорів: сіра, блакитна та зелена. Стандартно, вона велика, зелена та з білим кольором тексту, дизайн кнопки зображено на рисунку 2.1.

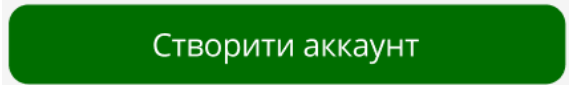

Рисунок 2.1 — Дизайн стандартної кнопки

Ще одним важливим компонентом являється поле для запису користувацьких даних на екрані авторизації та інших, при необхідності. Це поле має сіру рамку та текст-плейсхолдер по замовчуванню, а також підтримує стани фокусу та помилки з відповідними змінами. Дизайн цього компоненту наведено на рисунку 2.2.

Рисунок 2.2 — Дизайн поля для вводу тексту

Звичайно, одним із ключових компонентів додатку студентських аукціонів є картка лоту, яка поєднує зображення, базову інформацію, та кнопку яка вже згадувалася раніше. Для інформації, яка знаходиться поверх картинки добавлено напівпрозорий оверлей, який дозволяє роботи текст зручним для читання, коли картинка білого кольору. Дизайн картки лоту наведено на рисунку 2.3.

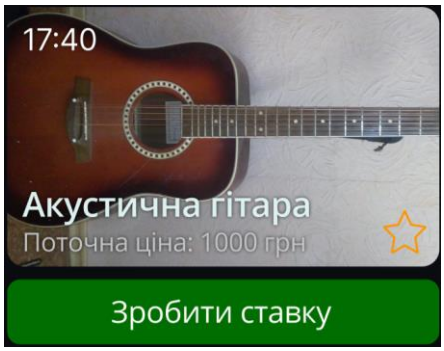

Рисунок 2.3 — Дизайн картки лоту

Ще одним компонентом, який використовується на всіх головних екранах є шапка(хедер). В шапці зображено логотип платформи, поточний баланс користувача та кнопку, яка відкриває та закриває невеликий блок з двома посиланнями для маніпуляцій з балансом, а також іконка на якій зображується кількість непрочитаних сповіщень, та яка є посиланням на екран сповіщень. Дизайн шапки зображено на рисунку 2.4.

Рисунок 2.4 — Дизайн шапки застосунку

В застосунку також присутній інший варіант шапки, яка здебільшого використовується на другорядних екранах. Даний компонент складається з стрілочки, яка веде користувача на попередню сторінку, а також заголовка, який описує екран. На сторінці «Профіль» додатково розташовується іконка для переходу на сторінку «Налаштування». Дизайн наведено на рисунку 2.5.

Рисунок 2.5 — Дизайн шапки другорядних сторінок

Ще одним невід’ємним базовим компонентом у дизайн-системі є навігаційна панель, яка забезпечує швидкий доступ до основних розділів застосунку. Вона фіксована внизу екрана та складається з п’яти іконок: «Головна», «Каталог лотів», «Створення лоту лот», «Чати» та «Профіль». Для кращої візуальної орієнтації поточна вкладка підсвічується насиченим зеленим кольором, решта іконок залишаються базового білого кольору. Дизайн навігаційної панелі наведено на рисунку 2.6.

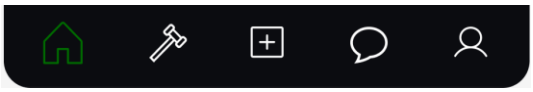

Рисунок 2.6 — Дизайн навігаційної панелі

Така навігаційна панель гарантує єдиний, послідовний спосіб перемикання між основними секціями застосунку, зберігає простір екрана під контент і забезпечує інтуїтивно зрозумілий доступ до ключових функцій.

Таким чином, набір базових компонентів забезпечує одночасно однорідний вигляд, зручність користування та швидку реалізацію дизайну за допомогою коду.

2.3 Дизайн сторінки «Реєстрація»

Екран реєстрації відкриває додаток з яскравого логотипу «Student Auction» на тлі глибокого чорного, що одразу налаштовує користувача на бренд і головну мету — участь в аукціоні, дизайн даної сторінки наведено на рисунку Г.1.

Під логотипом розташовано заголовок «Створіть безкоштовний акаунт» та підзаголовок «Перший крок до вигідних угод!», які мотивують пройти процедуру реєстрації.

Далі йдуть чотири однорядкові поля вводу: «Ім’я користувача (username)», «Ім’я та прізвище», «Email» та «Пароль». Кожне поле має чітку рамку із закругленими кутами, плейсхолдер і підсвічується при фокусі, забезпечуючи користувачу зрозумілий відгук. Застосовано валідацію на рівні форми: обов’язкові поля підсвічуються червоним при порожньому або некоректному введенні, наприклад невірний формат email чи занадто короткий пароль.

Після заповнення всіх полів з’являється можливість створити акаунт. Для цього потрібно натиснкути на кнопку «Створити акаунт» зеленого кольору. Під нею розташовано текстовий рядок із закликом для тих, хто вже має профіль: «Вже маєш акаунт? Увійти», де слово «Увійти» виконано яскравим зеленим кольором і є посиланням на сторінку входу.

Після натискання на кнопку «Створити акаунт» користувачу на вказану пошту прийде повідомлення для підтвердження реєстрації, приклад даного повідомлення наведено на рисунку 2.7. Після підтвердження можна увійти в додаток та користуватися усіма його функціями.

Рисунок 2.7 — Повідомлення для підтвердження акаунту

Отже, сторінка «Авторизація» розроблена адаптованою під усі мобільні екрани із читабельними шрифтами та інтервалами між полями, що забезпечує зручність використання та створює привабливий інтерфейс першого контакту користувача з додатком.

2.4 Дизайн сторінки «Головна»

Одним із головних екранів будь-якого застосунку є перша сторінка на яку користувач попадає після авторизації. Цей екран є важливим елементом і поєднує в собі ключові елементи навігації, інформаційні блоки та заклики до відповідних дій на платформі, дизайн наведено на рисунку Г.2.

Після успішної авторизації користувача зустрічає головний екран, у верхній частині якого розташована шапку. Під нею знаходиться велика привітальна фраза з підзаголовком «Купуй, продавай, вигравай — легко та вигідно!», що підсилює емоційний зв’язок із користувачем.

Нижче розміщена секція «Нові лоти» у вигляді горизонтального слайдера, де кожна картка поєднує прев’ю зображення товару з напівпрозорим оверлеєм для кращої читабельності тексту. У верхньому лівому куті картки показується час завершення, а внизу — назва лоту та підпис з поточною ціною. Кнопка «Зробити ставку» виконана в яскравому зеленому кольорі з м’якими заокругленими кутами, що одразу привертає увагу. Для швидкого переходу до

повного каталогу під слайдером є кнопка «Всі лоти», обведена тонкою зеленою рамкою і з текстом та стрілкою всередині.

За ним іде блок «ТОП продавці», що демонструється у вигляді круглих аватарів у каруселі. Кожен аватар складається з картинки профілю та повного імені даного користувача. Такий формат дозволяє ознайомитися з надійними користувачами, а також надає мотивації широкого користування додатком, що надасть можливість користувачу потрапити в даний блок на головній сторінці.

Наприкінці головного екрану розміщено блок із закликом «Бажаєте щось продати?» і коротким текстом про переваги платформи. Під ним знаходиться велика кнопка «Створити лот» у тому ж зеленому стилі, що й на інших екранах, для прямого переходу до сторінки «Створення лоту».

Отже, сторінка «Головна» виконує роль не лише вітального екрану, але й основу для взаємодії користувача з усіма ключовими функціями застосунку. Збалансоване розташування інформаційних блоків — нових лотів і топ-продавців, у поєднанні з чіткими кнопками доступу до каталогу та створення лоту забезпечує швидкий і зручний доступ до основних дій.

2.5 Дизайн сторінки «Каталог лотів» та «Деталі лоту»

Звичайно, що в додатку студентських аукціонів необхідно реалізувати сторінки, які будуть безпосередньо вміщувати в собі список усіх лотів, їх фільтрацію, а також можливість взаємодіяти з ними. Такими сторінками є «Каталог лотів» та «Деталі лоту».

На сторінці «Каталог лотів» розміщені основні елементи, такі як шапка та навігаційна панель, також користувач одразу бачить рядок пошуку з плейсхолдером «Введіть запит…», що дозволяє швидко відфільтрувати товари за ключовими словами, а поруч розміщені іконки фільтра та сортування для доступу до вікон з відповідними налаштуваннями пошуку та фільтрації, дизайн наведено на рисунку Г.3.

Під ними розташовується список карток лотів із зображеннями товарів, коротким описом, поточною ставкою та кнопкою «Зробити ставку», яка

дозволяє перейти на екран «Деталі лоту», де детально описана вся інформація про відповідний лот.

При скролі внизу автоматично підвантажуються наступні лоти, а жест «потягнути вниз» дозволяє моментально оновити увесь список актуальними даними, при цьому зберігаючи вибрані користувачем параметри пошуку та фільтрації.

Вікно «Фільтр» включає в себе можливість вибрати одну або кілька категорій за допомогою чекбоксів, задати діапазон ціни, а також обрати стан товару — новий або вживаний. Для зрчуності біля кожної категорії наведено кількість таких лотів в даний момент на аукціоні. Додатково в даному блоці наявні дві кнопки, для застосування усіх фільтрів та скидання усіх фільтрів до початкового стану, а також вони закривають дане вікно, дизайн даного вікна наведено на рисунку 2.8.

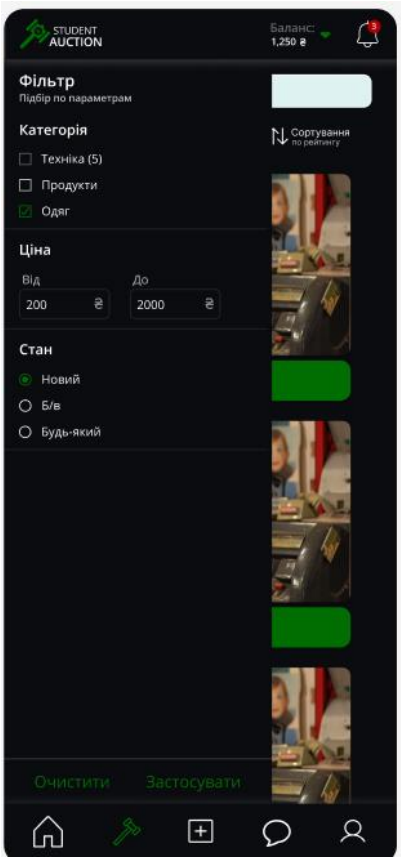

Рисунок 2.8 — Дизайн вікна «Фільтр»

Вікно «Сортування» надає можливість сортування лотів в потрібному порядку, наприклад по рейтингу, новинки, за збільшенням чи зменшенням ціни лоту. Після чого потрібно натиснути кнопку «Закрити» і список лотів оновиться у відповідно заданому порядку, дизайн даного вікна наведено на рисунку 2.9.

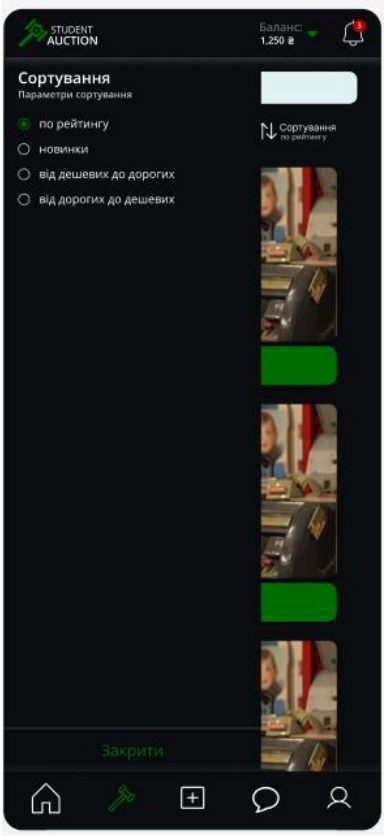

Рисунок 2.9 — Дизайн вікна «Сортування»

Екран «Деталі лоту» відкривається при натисканні на кнопку «Зробити ставку» під будь-якою карткою каталогу, дизайн наведено на рисунку Г.4. На даній сторінці детально описується інформація про кожний лот. Спочатку йде заголовок, після чого користувач може переглянути картинки даного товару та прочитати детальний опис від продавця, а також додати чи видалити лот з «вибране».

Після загальної інформації наводиться додаткові данні для зручності користувачів — ім’я користувача та можливість написати йому, де знаходиться

продавець, скільки часу залишилось, мінімальний крок ставки, поточна найвища ставка, а також можливість ввести суму ставки.

Коли користувач ознайомився з усією інформацією, він має можливість ввести суму бажаної ставки (щонайменше сума поточної ставки та мінімальний крок ставки) та натиснути кнопку «Зробити ставку» для участі в аукціоні.

Обидва екрани побудовано за єдиною системою відступів, кольорової палітри а також стилями кнопок і полів для введення інформації. Такий підхід гарантує однорідність інтерфейсу, полегшує орієнтацію користувача під час важливих дій та робить простим реалізацію даних сторінок.

2.6 Дизайн сторінки «Створення лоту»

Щоб повністю реалізувати функціонал аукціону необхідна сторінка для створення та публікації нового лоту, дизайн зображено на рисунку Г.5.

Спочатку йде заголовок лоту — однорядкове текстове поле з плейсхолдером «Назва лоту», яке має обмеження в 50 символів і валідується на обов’язковість та мінімальну довжину від 4 символів. Під ним розташовано великий багаторядковк поле для розширеного опису з підказкою «Опис».

У наступному блоці дві сусідні числові кнопки: «Початкова ціна» (гривні) і «Тривалість» (години). Ці поля дозволяються вводити лише числові дані. Мінімальна ціна — 50 грн, максимальна — 50 000 грн, тривалість — від 5 до 50 годин. При виході за межі допустимого інтервалу під полем з’являється червоне повідомлення із підказкою.

Далі розміщено два дропдаун-поля: «Категорія» та «Локація». Після натискання на таке поле з’являється список із можливих варіантів вибору, які були попередньо завантажені з серверу.

Між ними розташований блок «Стан товару», який реалізовано у вигляді радіокнопок: «Б/в» і «Новий», тобто надається можливість обрати лише один із наведених станів товару.

Найбільшу увагу привертає блок для завантаження зображень. При натисканні на відповідну іконку відкривається вікно для вибору картинки, а

також можливість відредагувати її розміри. Вибрані картинки відразу відображаються одразу після кнопки вибору у вигляді маленьких прев’ю з кнопкою для видалення.

Внизу екрану знаходиться велика кнопка зеленого кольору «Створити лот». Після того як продавець натисне на неї, вверху екрану з’явиться відповідне повідомлення, що підтверджує коректність введеної інформації, а на сервері лот буде моментально оброблений та стане доступний для перегляду та створення ставок.

Отже, було створено єдину дизайн-систему мобільного застосунку, яка забезпечує однорідність інтерфейсу, зручність використання всіх екранів та визначає базові компоненти платформи. Крім того, дотримання єдиних принципів стилю й елементів інтерфейсу забезпечує комфортне користування застосунком на різних пристроях і при різних рівнях освітленості.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Серверна частина (API та логіка)

Реалізація серверної частини є одним із ключових етапів створення швидкого та надійного застосунку. Основним завданням бекенду є надійне і ефективне обслуговування клієнтських запитів та забезпечення коректної роботи головних функцій: реєстрація користувачів, створення лотів, прийом ставок тощо. Створення чіткої архітектури та використання сучасних технологій гарантує масштабованість, захищеність та простоту підтримки проєкту.

3.1.1 Проєктування REST API

Проєктування REST API це важливий крок під час реалізації серверу, він має бути розроблений відповідно до рекомендацій RESTful-архітектури, де кожен логічний ресурс має чітко визначений шлях та використовує семантичні HTTP-методи [17].

Усі запити та відповіді структуровані через DTO-класи, які проходять валідацію за допомогою вбудованого ValidationPipe у NestJS. Це гарантує, що логіка на сервері завжди буде працювати тільки з правильно заповненими та коректними даними.

DTO (Data Transfer Object) — це клас, який описує структуру даних, що передаються між клієнтом і сервером. У ньому визначаються поля та їхні типи, а також можуть задаватися додаткові правила валідації. Завдяки використанню DTO чітко фіксуються дані, які надходять, а також створюється автоматична перевірка вхідних запитів, що значно підвищує безпеку та передбачуваність роботи серверу [18].

Приклад DTO-класу наведено в лістингу 3.1.

Лістинг 3.1 — Вміст файлу SetReviewDto

```
export class SetReviewDto {
  @IsObjectId({ message: 'Id користувача не коректний' })
  userId: Types.ObjectId;
```

```
  @IsNumber()
  @Min(0)
  @Max(5)
  rating: number;
  @IsString()
  comment: string;
}
```

Для налаштування правильних маршрутів, потрібно правильно використовувати основні HTTP-методи:

— GET — отримання ресурсу або списку ресурсів. Повертає статус 200 OK та тіло відповіді з необхідними даними.

— POST — створення нового ресурсу. Тіло запиту містить дані нового об’єкта, сервер повертає статус 201 Created.

— PUT — повне оновлення існуючого ресурсу. Тіло запиту замінює всі властивості об’єкта, відповідає 200 OK із новим станом ресурсу.

— PATCH — часткове оновлення ресурсу. Тіло запиту містить лише ті поля, які треба змінити, повертає 200 OK із оновленим об’єктом.

— DELETE — видалення ресурсу. Після успішного видалення сервер повертає 204 No Content без тіла відповіді [19].

Серверна частина мобільного застосунку «Студентський аукціон» надає декілька основних ресурсів через REST-ендпоінти, наведено в таблиці 3.1. Усі запити створені під префіксом /api.

Таблиця 3.1 — Основні REST-ендпоінти серверної частини

Ресурс	Метод	Шлях	Опис
Реєстрація	POST	/api/auth/register	Створює нового користувача. Після валідації полів userName, email, password повертає JWT-токени
Логін	POST	/api/auth/login	Перевіряє облікові дані, повертає об’єкт користувача та пару токенів accessToken і refreshToken
Отримання профілю	GET	/api/user/:id	Повертає дані профілю користувача за його id
Оновлення профілю	PUT	/api/user/:id	Оновлює дані профілю

Продовження таблиці 3.1

Список лотів	GET	/api/auction	Повертає перелік усіх лотів із можливістю фільтрації та пошуку за назвою
Деталі лоту	GET	/api/auction/:id	Повертає повні дані одного лоту за його id
Створення лоту	POST	/api/auction	Приймає параметри нового лоту (title, description, startPrice, images тощо) та створює його
Створення ставки	POST	/api/bid	Приймає auctionId та amount, виконує валідацію й повертає об’єкт створеної ставки
Отримання сповіщень	GET	/api/notification/:id	Повертає список усіх повідомлень користувача (нові ставки, завершення аукціонів тощо)

Таким чином, спроектована структура REST API забезпечує послідовну й передбачувану роботу клієнтської частини мобільного застосунку. Чітке розмежування маршрутів, використання HTTP-методів та семантичних статус-кодів, а також наявність валідації вхідних даних і захисту JWT-токенами гарантують надійну, розширювану та ефективну частину бекенду.

3.1.2 Авторизація та безпека (JWT)

Використання JWT дозволяє відмовитися від серверних сесій і зберігати всю інформацію про автентифікацію в самому токені, що спрощує масштабування та знижує навантаження на бекенд.

JWT (JSON Web Token) — це відкритий стандарт для компактного й безпечного представлення інформації як JSON-об’єктів, підписаних цифровим підписом, що дозволяє автентифікувати користувача без використання сесій на сервері [20].

JSON (JavaScript Object Notation) — це текстовий формат обміну даними, заснований на синтаксисі об’єктів JavaScript. Він складається з певної кількості

пар типу «ключ-значення» та масивів, легко читається та генерується, тому широко використовується для передачі структурованої інформації між клієнтом і сервером [21].

Авторизація починається з моменту успішної автентифікації: формується пакет даних для токена, який містить мінімально необхідну інформацію. Конфігурація модуля в NestJS наведено в лістингу 3.2.

Лістинг 3.2 — Конфігурація JwtModule у NestJS

```
export class JwtStrategy extends PassportStrategy<any>(Strategy) {
  constructor(
    private readonly configService: ConfigService,
    @InjectModel(UserModel) private readonly UserModel: ModelType<UserModel>,
  ) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: configService.get('JWT_SECRET'),
    });
  }

  async validate({ _id }: Pick<UserModel, '_id'>) {
    return await this.UserModel.findById(_id).exec();
}}
```

Після цього у сервісі AuthService при успішній перевірці даних викликається метод this.jwtService. signAsync (), який повертає підписаний токен. У відповіді клієнту віддаються два токени: короткоживучий accessToken та довгоживучий refreshToken. Метод який генерує токени наведено в лістингу 3.3.

Лістинг 3.3 — Генерація Access та Refresh токенів

```
async issueTokenPair(userId: string) {
  const data = { _id: userId };

  const refreshToken = await this.jwtService.signAsync(data, {
    expiresIn: '15d',
  });

  const accessToken = await this.jwtService.signAsync(data, {
    expiresIn: '1h',
  });
```

```
  return { refreshToken, accessToken };}
```

Після отримання токенів клієнт записує їх у AsyncStorage, використовуючи прості ключі «accessToken» і «refreshToken».

AsyncStorage — це асинхронне сховище для рядкових значень на пристрої, яке не шифрує дані й доступне будь-якому JavaScript-коду додатку. У зв’язку з цим існує ризик викрадення токенів. Щоби обмежити наслідки, застосовується схема з двома токенами.

Для відправлення запитів використовується accessToken, який дійсний протягом однієї години. Для генерації нового токена без повторної авторизації використовується refreshToken. Такий підхід гарантує, що навіть у разі перехоплення accessToken зловмисник матиме змогу здійснювати запити лише обмежений час.

Усі захищені маршрути позначаються декоратором @Auth(), який залежно від переданого параметра («user» або «admin») автоматично застосовує потрібні Guard. Для звичайного користувача достатньо перевірки валідності JWT-токена через JwtAuthGuard, а для адміністратора додатково виконується перевірка ролі в OnlyAdminGuard. У разі відсутності або недійсності токена сервер повертає код 401 Unauthorized, а при відсутності прав адміністратора — 403 Forbidden. Реалізація @Auth() наведено у лістингу 3.4.

Лістинг 3.4. — Декоратор @Auth() і відповідні Guards

```
import { applyDecorators, UseGuards } from '@nestjs/common';
import { TypeRole } from '../auth.interface';
import { OnlyAdminGuard } from '../guards/admin.guard';
import { JwtAuthGuard } from '../guards/jwt.guard';

export function Auth(role: TypeRole = 'user') {
  return applyDecorators(
    role === 'admin'
      ? UseGuards(JwtAuthGuard, OnlyAdminGuard)
      : UseGuards(JwtAuthGuard),
  );
}
```

Окрім базової авторизації, реалізовано обмін refresh-токена для оновлення access-токена без необхідності повторного логіну. Клієнт надсилає refresh-токен по маршруту POST /api/auth/login/access-token, сервер його

перевіряє та, у разі успіху, повертає новий access-токен. Це знижує ризик втрати доступу при короткому життєвому циклі access-токена.

Для підвищення безпеки секретний ключ JWT_SECRET не зберігається безпосередньо в коді, а передається через змінні середовища (файл .env), який виключається з системи контролю версій.

Таким чином, використання JWT із чітко визначеними стратегіями видачі, перевірки та оновлення токенів гарантує безпечний доступ до захищених маршрутів, спрощує масштабування та підтримку бекенд-частини застосунку.

3.1.3 Обробка бізнес-логіки аукціонів

Ключова складова системи — реалізація логіки проведення аукціонів, що включає управління створенням лотів, створенням та валідацією ставок, визначенням переможця, завершенням торгів тощо.

Логіка створення лоту реалізована в методі create() сервісу AuctionService, який викликається з контролера AuctionController. Для забезпечення коректності вхідних даних використовується клас CreateAuctionDto, що описує всі необхідні поля лоту та перевіряє їх через декоратори з бібліотеки class-validator, наприклад поля є обов’язковими, картинки мають бути масивом рядків тощо. Частину коду даного класу наведено в лістингу 3.5.

Лістинг 3.5 — Частина коду класу CreateAuctionDto

```
export class CreateAuctionDto {
  @IsString({ message: "Заголовок є обов'язковим полем" })
  @IsNotEmpty({ message: 'Заголовок не може бути порожнім' })
  title: string;

  @IsString({ message: "Опис є обов'язковим полем" })
  @IsNotEmpty({ message: 'Опис не може бути порожнім' })
  description: string;
  @IsArray({ message: 'Поле зображень має бути масивом' })
  @ArrayNotEmpty({ message: 'Додайте хоча би одне зображення' })
  @IsString({ each: true, message: 'Кожне зображення має бути рядком' })
  images: string[];

  @IsObjectId({ message: 'Категорія має бути валідним ObjectId' })
  category: Types.ObjectId;
```

Одна з найскладніших частин бізнес-логіки аукціону пов’язана з обробкою ставок. Для забезпечення коректності ставок з’являється необхідність в багатьох перевірках користувача та лоту, код наведено в лістингу 3.6.

Лістинг 3.6. — Методи валідації користувача та лоту

```
private validateAuction(auction) {
    if (auction.status === 'completed') {
      throw new BadRequestException('Лот завершено');
    }
  }
  private validateUser(newBider, bidAmount, auction) {
    if (!newBider) {

      throw new BadRequestException('Користувача не знайдено');
    }
    if (newBider.balance < bidAmount) {
      throw new BadRequestException('Недостатньо коштів');
    }
    const neededNextBid = auction.currentBid + auction.step;
    if (bidAmount < neededNextBid) {
      throw new BadRequestException('Ціна повинна бути більшою за
поточну');
      }}
```

Після усіх перевірок оновлюється баланс, поточна ставка та створюється новий об’єкт типу Bid. Також в методі handlePreviousBidder() перевіряється наявність попередньої ставки та повертаються кошти тому, чия ставка була перебита. Код даної логіки наведено в лістингу 3.7.

Лістинг 3.7. — Оновлення балансу, поточної ставки і створення Bid

```
await this.handlePreviousBidder(auction, newBider);

  await this.userService.updateUser(userId, {
    balance: newBider.balance - dto.amount,
  });

  await this.auctionService.updateCurrentBid(
    userId,
    dto.auctionId,
    dto.amount,
  );

  await this.updateAuctionIfNeeded(auction);

return await this.bidModel.create({ ...dto, userId });
}
```

Під час участі в аукціоні, сповіщення є важливим елементом зручності та реагування на певні події. Створення нового сповіщення відбувається в методі createNotification() сервісу NotificationService, який приймає DTO з описом сповіщення: id користувача, тип сповіщення та id лоту, валідація полів для створення сповіщення наведено в лістингу 3.8.

Лістинг 3.8 — Валідація полів для створення сповіщення

```
export class CreateNotificationDto {
  @IsObjectId({ message: 'Id користувача не коректний' })
  userId: Types.ObjectId;
  @IsEnum(
    [
      'auction_ended',
      'auction_won',
      'auction_ended_no_buyer',
      'auction_lost',
      'new_bid',
    ],
    {
      message: 'Type не коректний',
    },
  )
  type: TypeNotification;

  @IsObjectId({ message: 'Id лоту не коректний' })
  auction: Types.ObjectId;
}
```

Для завершення дії лоту після закінчення часу, у сервісі AuctionService налаштовано Cron-задачу, яка кожну хвилину виконує шукає всі активні аукціони з endTime, що вже минув, і відразу змінює їхній статус на completed, щоб нові ставки не могли потрапити до обробки. Для кожного завершеного лоту перевіряються наявні ставки, якщо жодної немає, власник отримує лише повідомлення про відсутність покупців, а якщо є хоча б одна ставка, то найвища з них автоматично визнається переможцем.

Після цього створюються записи в колекції notification для продавця і переможця, а через WebSocket миттєво надсилаються відповідні події, щоб клієнтські інтерфейси оперативно відобразили результати торгів. Частину коду з використанням Cron та пошуку таких лотів наведено в лістингу 3.9.

Cron — це стандартна утиліта в Unix-подібних операційних системах для планування повторюваних завдань за заданим розкладом. У NestJS-проєктах дану утиліту можна використовувати через модуль @nestjs/schedule, який надає декоратор @Cron() для зручного опису таких завдань [22].

Лістинг 3.9 — Пошук завершених лотів з використанням Cron

```
@Cron('*/1 * * * *')
async checkAndCompleteAuctions() {
  const now = new Date();

  const expiredAuctions = await this.auctionModel
    .find({
      endTime: { $lte: now },
      status: { $ne: 'completed' },
    })
    .exec();
```

Таким чином, реалізовано повну бізнес-логіку аукціону, створення лоту з валідацією вхідних даних через DTO-класи, перевірку коректності ставок і оновлення балансів, автоматичне завершення торгів з визначенням переможця та розсилкою сповіщень у реальному часі.

3.1.4 Зберігання даних у MongoDB

Організація даних у MongoDB закладає фундамент для швидкого доступу, масштабованості та гнучкого розширення функціональності. У контексті NestJS для роботи з базою даних застосовано бібліотеку Typegoose, яка побудована поверх Mongoose і дозволяє описувати схеми через TypeScript-класи, без додаткового коду для визначення схем, приклад наведено в лістингу 3.10.

Mongoose — це популярна бібліотека ODM для Node.js, яка дозволяє описувати схеми даних, виконувати валідацію полів, формувати запити та працювати з колекціями MongoDB через інтуїтивний JavaScript-інтерфейс [23].

ODM (Object Data Modeling) — це підхід і набір інструментів для відображення об’єктів прикладного коду на документи в NoSQL-базах даних. ODM бібліотека дозволяє описувати структуру колекцій як класи або схеми в коді, автоматично конвертувати між об’єктами в пам’яті та записами в базі,

виконувати валідацію полів, підтримувати зв’язки між документами та спрощувати формування запитів через API.

Typegoose суттєво спрощує опис моделей даних у NestJS, дозволяючи задавати їх у вигляді класів із декораторами, що автоматично генерують схему Mongoose. Завдяки цьому всі поля моделі мають статичну типізацію і перевіряються вже під час компіляції, що мінімізує ризик помилок. При використанні опцій клас-декораторів досить легко додати стандартні поля createdAt і updatedAt. Інтеграція з NestJS відбувається через модуль TypegooseModule.forFeature() і звичний механізм ін’єкцій @InjectModel(), що є досить зручно [24].

Ці переваги роблять Typegoose оптимальним рішенням для побудови чіткої, підтримуваної архітектури даних у NestJS-проєктах.

Лістинг 3.10 — Оголошення моделі BidModel через Typegoose

```
export interface BidModel extends Base {}

export class BidModel extends TimeStamps {
  @prop({ ref: () => UserModel })
  userId: Ref<UserModel>;

  @prop({ ref: () => AuctionModel })
  auctionId: Ref<AuctionModel>;
  @prop()
  amount: number;
}
```

Приклад інтеграції конфігурації TypegooseModule.forFeature() наведено в лістингу 3.11.

Лістинг 3.11 — Конфігурація TypegooseModule.forFeature()

```
  TypegooseModule.forFeature([
    {
      typegooseClass: BidModel,
      schemaOptions: {
        collection: 'Bid',
      },
    },
  ]),
```

Таким чином, серверна частина мобільного застосунку «Студентський аукціон» побудована за допомогою NestJS із чистою модульною архітектурою, що поєднує в собі декларативне проєктування REST API, надійну авторизацію через JWT, ефективну бізнес-логіку аукціонів та типізоване й масштабоване зберігання даних у MongoDB з використанням бібліотеки Typegoose.

3.2 Реалізація сторінок мобільного застосунку

У цьому підрозділі розглянемо реалізацію інтерфейсу екранів мобільного застосунку «Студентський аукціон» з допомогою React Native та Expo. На основі затверджених дизайн-макетів і базових компонентів проаналізуємо структуру кожного екрану, опишемо ключові компоненти та хуки, що забезпечують відображення даних, обробку подій користувача та взаємодію з сервером. Кожний екран реалізовано як окремий функціональний компонент із чітко визначеними зонами відповідальності, що сприяє простоті підтримки коду і масштабуванню проєкту. Структурна схема застосунку наведена на рисунку Г.6.

3.2.1 Сторінка «Реєстрація»

Сторінка «Реєстрація» призначена для створення нового облікового запису у застосунку. Вона дає можливість ввести необхідні дані: ім’я користувача (username), повне ім’я, електронну пошту та пароль. Крім того, на цій сторінці реалізовано перехід на екран входу для тих, хто вже має акаунт.

Першим кроком створюється форма для реєстрації за допомогою бібліотеки react-hook-form з усіма необхідними полями та включаємо режим валідації на зміну.

React Hook Form — легка й продуктивна бібліотека для роботи з формами в React, яка мінімізує повторні рендери і забезпечує високу швидкість валідації за рахунок використання неконтрольованих компонентів під капотом [25].

Виклик useForm повертає об’єкт control, який передається в кожен <Controller>, функцію handleSubmit для обробки відправки, а також formState. Ініціалізація форми та валідація полів в коді наведено в лістингу 3.12.

Лістинг 3.12 — Ініціалізація форми та валідація полів

```
const {
  control,
  handleSubmit,
  formState: { errors },
} = useForm<ISignUp>({
  defaultValues: {
    email: "",
    password: "",
    name: "",
    userName: "",
  },
});
```

Для того щоб зв’язати поле «email» із контролем форми, задаючи правила обов’язковості та перевірки формату через регулярний вираз використовується контролер. Сам UI-компонент FormInput отримує пропси для оновлення значення та обробки втрати фокусу, а у разі помилки нижче виводиться повідомлення із errors.email.message, код наведено в лістингу 3.13.

Лістинг 3.13 — Зв’язування поля «email» через Controller

```
<Controller
  control={control}
  name="email"
  rules={{
    required: "Поле обов'язкове",
    pattern: {
      value: /^[^\s@]+@[^\s@]+\.[^\s@]+$/,
      message: "Некоректний email"
    }
  }}
  render={({ field: { onChange, onBlur, value } }) => (
    <FormInput
      placeholder="Email"
      keyboardType="email-address"
      onBlur={onBlur}
      onChangeText={onChange}
      value={value}
    />
  )}
/>
```

```
{errors.email && (
  <Text className="text-red text-xs">
    {errors.email.message}
  </Text>
)}
```

Сторінка реєстрації об’єднує валідацію зручної форми, повторно використовувані UI-компоненти та чітку бізнес-логіку авторизації. Використання бібліотеки react-hook-form у поєднанні з компонентами Controller і власними полями вводу забезпечує гнучкий контроль над станом і повідомленнями про помилки. Повний лістинг даної сторінки наведено в лістингу Д.1.

3.2.2 Сторінка «Головна»

Сторінка «Головна» служить точкою входу до мобільного застосунку одразу після авторизації. Вона вітає користувача, показує найсвіжіші лоти, перелік топ-продавців, а також дає швидкий доступ до повного каталогу аукціонів і до створення нового лоту, дані завантажуються із використання бібліотеки React Query.

React Query — це бібліотека для керування асинхронними запитами та серверним станом у React-додатках, що надає ефективне завантаження, кешування та синхронізації даних із сервером [26].

Використання хука useAuctions, який реалізований з допомогою бібліотеки react-query, а також механізм оновлення жестом наведено в лістингу 3.14.

Лістинг 3.14 — Ініціалізація даних і механізм оновлення

```
const user = useAuth();
const { auctions, isLoading, refetch: refetchAuctions } = useAuctions();
const { topSellers, isLoading: topSellersLoading, refetch: refetchSellers } = useGetTopSellers();

const [refreshing, setRefreshing] = useState(false);

const onRefresh = useCallback(async () => {
  setRefreshing(true);
  await Promise.all([
    refetchAuctions(),
```

```
    refetchSellers(),
  ]);
  setRefreshing(false);
}, [refetchAuctions, refetchSellers]);
```

У даному фрагменті коду використовується хук useAuth для отримання імені та даних поточного користувача. За допомогою кастомних хуків useAuctions та useGetTopSellers завантажується список аукціонів і рейтинг продавців, а також методи refetch для повторного запиту. Через локальний стан refreshing та onRefresh із RefreshControl реалізовано жест «потягнути вниз», що одночасно оновлює обидва списки.

Перед тим, як відобразити самі картки аукціонів, виводиться заголовок секції та перевіряється, чи триває ще завантаження даних. Якщо isLoading істинний, показуються плейсхолдери-скелетони, інакше — реальні аукціони. Реалізацію даного механізму наведено в лістингу 3.15.

Лістинг 3.15 — Рендеринг секції «Нові аукціони»

```
<StyledText className="text-xl font-opensmedium mb-4">
  Нові аукціони
</StyledText>
{isLoading ? (
  <Slider
    data={["", ""]}
    renderItem={() => <HomeAuctionsLoader />}
  />
) : (
  <Slider
    data={auctions.slice(0, 8)}
    renderItem={(item, index) => <Auction {...item} key={index} />}
  />
)}
```

Отже, сторінка «Головна» об’єднує вітальний блок, дві ключові секції — «Нові аукціони» та «ТОП продавці», а також навігаційні кнопки для переходу до повного каталогу та створення лоту. Механізм оновлення даних гарантує актуальність інформації в будь-який момент, а повторне використання карусельних компонентів і лоадерів забезпечує єдиний інтерфейс застосунку. Повний лістинг сторінки «Головна» наведено в лістингу Д.2.

3.2.3 Сторінка «Каталог лотів»

Сторінка «Каталог лотів» дозволяє користувачу переглядати всі доступні аукціони з можливістю пошуку, фільтрації та сортування. Вона інтегрує пошукове поле, кнопки для відкриття бокових панелей «Фільтр» і «Сортування», а також підтримує жест для оновлення списку з урахуванням поточних параметрів запиту.

Перед рендерингом списку лотів ініціалізуємо стан фільтрів, сортування та запиту, а також налаштовуємо жест «pull to refresh» для оновлення даних у будь-який момент, код наведено в лістингу 3.16.

Лістинг 3.16 — Ініціалізація стану та оновлення

```
const [query, setQuery] = useState<AuctionParams>(initialParams);
const { auctions, isLoading, refetchWithParams } = useAuctions(query);

const onRefresh = async () => {
  setRefreshing(true);
  await refetchWithParams(query);
  setRefreshing(false);
};
```

Для відображення потрібних лотів користувачу надається можливість шукати та фільтрувати їх у реальному часі. Зміна пошукового рядка автоматично оновлює параметри запиту, а поки дані завантажуються, показуємо скелетони замість карток. Реалізація пошуку та рендеру карток наведено в лістингу 3.17.

Лістинг 3.17 — Пошук і рендер карток аукціонів

```
<SearchInput handleSubmit={text => setQuery({ ...query, search: text })} />

{isLoading
  ? <AuctionLoader count={3} />
  : auctions.map(a => (
      <Auction key={a._id} {...a} isBig />
    ))
}
```

Перед тим як користувач зможе застосувати нові параметри фільтрації, обробник changeQuery збирає поточні значення категорії, стану та діапазон цін,

формує новий об’єкт AuctionParams і передає його через функцію handleSubmit, після чого закриває панель. Обробник застосування фільтрації наведено в лістингу 3.18.

Лістинг 3.18 — Обробник застосування фільтрації

```
const changeQuery = () => {
  const price = priceRange[0] || priceRange[1]
    ? `${priceRange[0]}-${priceRange[1]}`
    : "";
  handleSubmit({
    ...query,
    category,
    condition,
    price
  });
  close();};
```

Таким чином, сторінка «Каталог лотів» поєднує інтуїтивний інтерфейс пошуку, зручну фільтрацію та сортування з автоматичним оновленням даних за допомогою жесту. Використання компонентів SearchInput, AuctionLoader, Filter і Sort забезпечує відповідний інтерфейс і передбачувану поведінку при взаємодії. Плавна анімація відкриття бокової панелі та мінімалістичний дизайн карток гарантують легкість орієнтування, швидкий і зручний доступ до лотів для користувачів. Повний лістинг сторінки «Каталог лотів» наведено в лістингу Д.3.

3.2.4 Сторінка «Деталі лоту»

Сторінка «Деталі лоту» призначена для відображення повної інформації про окремий аукціонний лот та надання користувачу всіх необхідних інструментів для взаємодії: перегляду фотографій, ознайомлення з описом, спостереження за таймером, додавання в «обране», участі у торгах або видалення лоту власником. Розглянемо ключові фрагменти реалізації даної сторінки.

Перед відображенням контенту отримуються дані лоту через хук useGetAuction, налаштовується стан модальних вікон для ставок і видалення, а

також жест для оновлення інформації на екрані. Ініціалізацію даних та їх оновлення наведено в лістингу 3.19.

Лістинг 3.19 — Ініціалізація даних та оновлення

```
const { data, isLoading, refetch } = useGetAuction(id as string);
const [isBidModalVisible, setIsBidModalVisible] = useState(false);
const [refreshing, setRefreshing] = useState(false);
const onRefresh = useCallback(async () => {
  setRefreshing(true);
  await refetch();
  setRefreshing(false);
}, [refetch]);
```

Для зручного перегляду картинок реалізовано динамічний слайдер за допомогою універсального компонента Slider, код наведено в лістингу 3.20. Повний лістингу компоненту ImageSlider наведено в лістингу Д.4.

Лістинг 3.20 — Галерея зображень лоту

```
<Slider
  isPagination
  height={200}
  data={data.images}
  renderItem={(uri) => <ImageSlider imageUri={uri} />}
/>
```

Блок ставок складається з відображення поточної ціни, мінімального кроку та поля введення нової ставки. Коли користувач натискає «Зробити ставку», відкривається модальне вікно підтвердження, реалізацію даного блоку наведено в лістингу 3.21.

Лістинг 3.21 — Поле введення ставки та кнопка

```
<View style={{ width: 100 }}>
  <TextInput
    value={bid.toString()}
    onChangeText={t => setBid(parseInt(t) || 0)}
    keyboardType="numeric"
    placeholder="0"
  />
  <CurrencyIcon />
</View>
<StyledButton onPress={() => setIsBidModalVisible(true)} color="green">
  Зробити ставку
</StyledButton>
<MyModal
```

```
        title={`Зробити ставку (${bid} грн)?`}
        onConfirm={() => createBid({ auctionId: data._id, amount: bid })}
visible={isBidModalVisible}
/>
```

Отже, сторінка «Деталі лоту» поєднує медіа-галерею, інформативні блоки з описом, таймером та даними про продавця, а також інтерактивні елементи для участі в аукціоні чи його адміністрування. Використання компонентів Slider, AuctionTime, MyModal та коректна обробка станів гарантують відповідний інтерфейс та спрощують подальший супровід і розширення функціоналу.

3.2.5 Сторінка «Створення лоту»

Сторінка «Створення лоту» надає користувачу інтерфейс для введення всієї необхідної інформації про новий аукціонний лот: від назви й опису до завантаження фотографій і вибору категорії, стану та місцезнаходження. Вона забезпечує валідацію полів на стороні клієнта та трансформацію даних перед відправкою на сервер. Розглянемо ключові фрагменти коду, що забезпечують реалізацію форми.

Першим кроком ініціалізується об’єкт форми за допомогою бібліотеки react-hook-form та створюється метод який відправляє запит на сервер для створення лоту, код наведено в лістингу 3.22.

Лістинг 3.22 — Ініціалізація React Hook Form та обробник надсилання

```
        const { control, handleSubmit, formState: { errors }, reset } =
useForm<IAuctionForm>({
          defaultValues: { title: "", description: "", startPrice: 0, endTime: 0, category:
"", condition: "new", location: "", images: [] },
        });

        const onSubmit = async (data: IAuctionForm) => {
          const endDate = new Date(Date.now() + data.endTime *
3600000).toISOString();
          await mutateAsync({ ...data, endTime: endDate, step:
Math.max(Math.round(data.startPrice * 0.01 / 10) * 10, 10) });
          reset();
        };
```

Для текстових полів використовується компонент-обгортка FormField, а для числових значень — LabeledInputField, що дозволяє реалізувати єдину

обробку валідації полів, відображення відповідних заголовків та помилок для кожного поля форми. Код даного фрагменту наведено в лістингу 3.23.

Лістинг 3.23 — Поля введення тексту, ціни та часу

```
<FormField
  name="title"
  control={control}
  rules={{
    required: { value: true, message: "Поле обов'язкове" },
    minLength: { value: 4, message: "Мінімум 4 символи" },
    maxLength: { value: 50, message: "Максимум 50 символів" },
  }}
  InputComponent={FormInput}
  inputProps={{ placeholder: "Назва лоту" }}
/>

<LabeledInputField
  label="Початкова ціна:"
  name="startPrice"
  control={control}
  rules={{
    required: { value: true, message: "Поле обов'язкове" },
    min: { value: 50, message: "Мінімальна ціна 50 грн" },
    max: { value: 50000, message: "Максимальна ціна 50 000 грн" },
  }}
  InputComponent={FormPriceInput}
/>
```

За допомогою компонента CustomSelect та RadioGroupField користувач має можливість вибрати категорію, стан та місцезнаходження лоту, причому передані масиви даних отримуються з бекенду через відповідні хуки. Реалізацію даної логіки наведено в лістингу 3.24.

Лістинг 3.24 — Вибір категорії, стану та локації

```
<Controller
  control={control}
  name="category"
  rules={{ required: { value: true, message: "Поле обов'язкове" } }}
  render={({ field: { onChange, value } }) => (
    <CustomSelect
      items={categoryItems}
      selectedValue={value}
      onValueChange={(val) => onChange(val)}
      placeholder="Виберіть категорію"
    />)}/>
<RadioGroupField
  name="condition"
```

```
        control={control}
        rules={{ required: { value: true, message: "Поле обов'язкове" } }}
        label="Стан товару:"
 options={radioOptions}
 />
```

Таким чином, сторінка «Створення лоту» об’єднує різноманітні механізми введення інформації — від звичайних текстових полів до компонентів для вибору файлів. Використання react-hook-form разом із кастомними UI-компонентами забезпечує чистоту коду, повторне використання та зрозумілий інтерфейс при додаванні нових лотів.

3.2.6 Сторінка «Чати»

Сторінка «Чати» призначена для відображення списку всіх активних діалогів користувача та швидкого доступу до кожного з них. Вона забезпечує можливість оновлення списку жестом і виводить індикатор завантаження в разі очікування відповіді від сервера, графічний інтерфейс сторінки «Чати» наведено на рисунку Г.7.

Кожен елемент списку представлений компонентом ChatItem, який містить аватар співрозмовника, його ім’я та короткий фрагмент останнього повідомлення. Навігація до конкретного чату відбувається при натисканні на відповідний елемент.

В лістингу 3.25 наведено фрагмент коду, що відповідає за отримання даних, обробку оновлення та рендеринг списку чатів.

Лістинг 3.25 — Отримання та відображення списку чатів

```
const {data, isLoading, refetch} = useChats();
const [refreshing, setRefreshing] = useState(false);

<ScrollView className="bg-black pt-5 px-4">
  <View>
   {data.map(chat => <ChatItem key={chat.chatId} id={chat.chatId}
avatar={chat.otherUser.avatar} name={chat.otherUser.name}
message={chat.lastMessage ? chat.lastMessage.type === "text" ?
chat.lastMessage.text || "" : "Надіслано фото" : "Напишіть перше повідомлення"}
/>)}
  </View>
</ScrollView>
```

В даному коді використовується хук useChats() для отримання чатів та інших змінних для маніпуляцій з даними, onRefresh керує показником індикатора оновлення та повторним запитом до сервера.

Отже, сторінка «Чати» забезпечує інтерфейс для обміну повідомленнями між користувачами платформи. Завдяки зв’язку з сервером через useChats і можливості оновлення жестом вона гарантує, що користувач завжди бачитиме актуальний список діалогів. В результаті реалізації цей екран сприяє своєчасній комунікації між покупцями й продавцями та підвищує зручність використання додатку.

3.2.7 Сторінка «Профіль»

Сторінка «Профіль» призначена для відображення персональної інформації користувача та його активності на платформі. Вона дозволяє швидко переглянути аватар, ім’я, рейтинг, баланс, а також статистику ставок, виграних і проданих лотів, графічний інтерфейс сторінки «Профіль» навдено на рисунку Г.8.

В лістингу 3.26 наведено фрагмент коду, який відповідає за завантаження даних профілю, відображення аватара та основної інформації.

Лістинг 3.26 — Отримання та вивід даних користувача

```
const { data, isLoading, refetch: refetchProfile } = useProfile(user?._id || "");
if (isLoading) return <Loader />;
<Image
  source={
    data?.avatar
      ? { uri: `${API_SERVER_URL}${data.avatar}` }
      : require("@/assets/images/avatar.png")
  }
  className="w-36 h-36 rounded-full"
/>
<StyledText className="text-xl">{data?.name}</StyledText>
```

У даному фрагменті коду використовується хук useProfile, що повертає інформацію про користувача, поле isLoading, що вказує на завантаження даних, та метод refetch для можливості оновлення даних в потрібний момент. Компонент Loader показується під час завантаження даних, компонент Image

використовується для відображення аватара користувача або стандартний аватар профіля.

Після чого відображається поточний баланс із кнопкою «Поповнити?», яка веде на відповідний екран, а нижче — три інформаційні картки із кількістю ставок, виграних і проданих лотів.

Також, на сторінці є таби, які складаються з: «Мої лоти», «Мої ставки» та «Вибрані», що дозволяє швидко знаходити потрібні лоти з можливістю швидкого переходу до сторінки «Деталі лоту». Отримання даних і використання MyTabs для відображення представлено в лістингу 3.27.

Лістинг 3.27 — Отримання списку лотів та використання MyTabs

```
const { isLoadingMyAuctions, myAuctions, isLoadingMyFavoriteAuctions, myFavoriteAuctions, refetchAllAuctions, isLoadingBidedAuctions, bidedAuctions } = useTabsInfo();

<View className="mb-4">
  <MyTabs favorites={myFavoriteAuctions} myBids={bidedAuctions} myLots={myAuctions} isLoadingBids={isLoadingBidedAuctions} isLoadingFavorites={isLoadingMyFavoriteAuctions} isLoadingLots={isLoadingMyAuctions} />
</View>
```

Внизу сторінки зображено відгуки про користувача, які він отримує після закінчення процесу торгів, для цього використовується компонент Review, повний лістинг якого наведено в лістингу Д.5.

Отже, сторінка «Профіль» об’єднує в собі всі ключові відомості про користувача та його діяльність на платформі. Використання кастомних хуків для запитів, компонента-скелетону для завантаження та єдина стильова система забезпечують плавність роботи та інтуїтивну зрозумілість інтерфейсу.

4 ТЕСТУВАННЯ ЗАСТОСУНКУ

Тестування мобільного застосунку «Студентський аукціон» є критично важливим етапом, що гарантує коректну роботу всіх його функціональних компонентів, стабільність у реальних умовах та задоволеність користувачів. У цьому розділі розглянемо тестування ключових сценаріїв використання додатку: створення лоту, участі в торгах (ставки), роботи з профілем та авторизації, а також адміністративної частини системи.

4.1 Тестування створення лоту

Сторінка «Створення лоту» є одним із найважливіших елементів мобільного застосунку, адже від правильної роботи її форм та контролів залежить створення нових оголошень і загальна активність користувачів на платформі. У цьому підрозділі перевіримо, як працює валідація полів, коректність вибору і завантаження зображень та відправка запиту на сервер, а також переконаємося, що інтерфейс коректно реагує на успішне та неуспішне створення лоту.

Одним із головних і складних елементів сторінки для створення лоту є компонент завантаження зображень. Користувач натискає на кнопку, яка знаходить навпроти тексту «Виберіть зображення» та обирає кілька файлів зі свого пристрою. На рисунку 4.1 показано, що після вибору трьох зображень вони відразу відображаються в полі форми у вигляді попереднього перегляду. Це підтверджує, що компонент ControlledImageInput успішно приймає та відображає фото перед відправкою.

Після заповнення всіх полів і завантаження зображень користувач натискає кнопку «Створити лот», після чого біля полів введення буде описано про невідповідність введених даних, а якщо інформація вірна, то з’явиться повідомлення про відповідь від сервера: успішно або помилка. На рисунку 4.2 показано, як після натискання на кнопку з’являється коротке повідомлення «Успішно створено», що дає знати про те, що запит успішно відправлено на сервер.

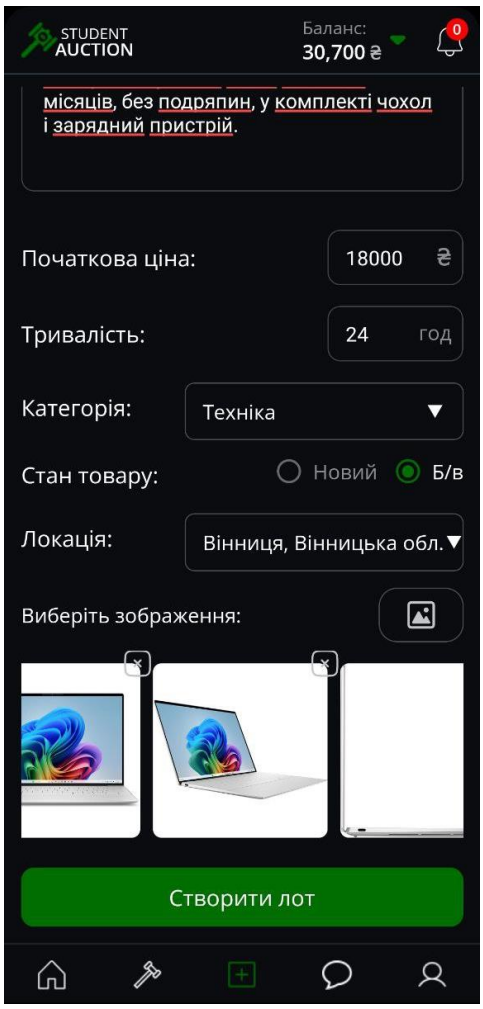

Рисунок 4.1 — Попередній перегляд обраних зображень

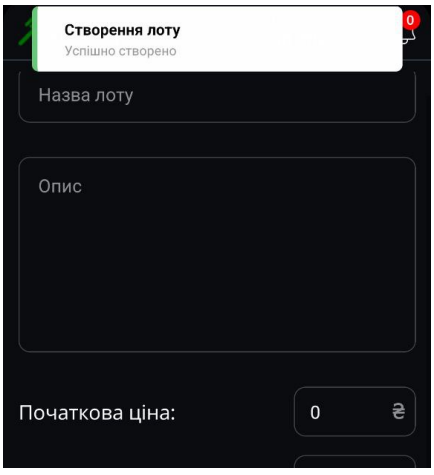

Рисунок 4.2 — Повідомлення про успішно відправлений запит

Після успішного виконання запиту інтерфейс автоматично анулює заповнені дані та повідомляє користувача про створення нового лоту. Наступним кроком, можна переглянути створений лот в каталозі та перейти на

стоірнку «Деталі лоту», де буде описана вся зазначена інформація під час створення, окрема сторінка даного лоту зображена на рисунку 4.3. Це є остаточним свідченням того, що як клієнтська валідація, так і взаємодія з сервером працюють коректно.

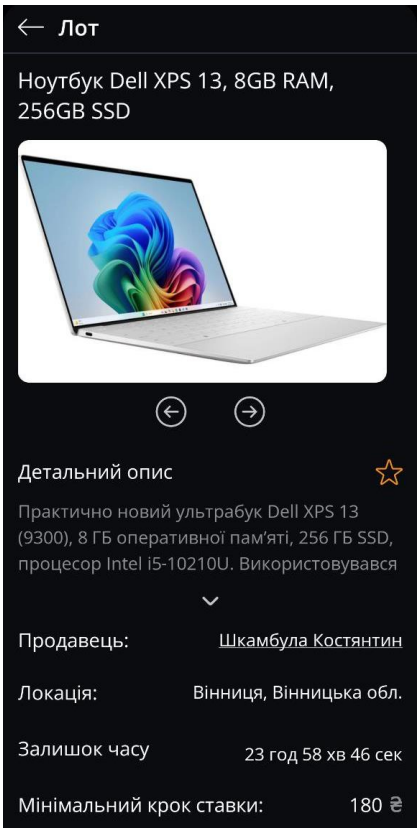

Рисунок 4.3 — Окрема сторінка створеного лоту

Під час заповнення форми всі поля відображаються одразу без затримок, а при виборі зображень індикатор завантаження з'являється лише на короткий час (від 1 до 1.5 секунд), в залежності від розміру вибраного файлу. Після натискання на кнопку «Створити лот» від сервера приходить підтвердження успішного створення лоту за час, що не перевищує 1 с.

Таким чином, тестування сторінки «Створення лоту» охоплює перевірку клієнтської валідації, роботи компоненту завантаження зображень, коректної активації кнопки відправки, успішного створення лоту на сервері та обробки помилок. Це гарантує, що кінцевий користувач отримає зворотний зв'язок і зможе безперешкодно виставляти нові лоти на аукціон.

4.2 Тестування участі в торгах (ставки)

Можливість здійснення коректних ставок необхідна для реалізації надійного механізму аукціону. У цьому підрозділі розглянемо два основних аспекти: валідації мінімальної суми ставки на боці клієнта та підтвердженні успішної відправки ставки з оновленням інтерфейсу. Це дозволяє гарантувати, що користувачі не зможуть випадково або навмисно поставити менше, ніж дозволяє логіка аукціону, і що кожна коректна ставка миттєво відображається в системі.

Важливим елементом онлайн-аукціону є правильна організація створення ставок та їх обробки сервером, протестуємо валідацію введення некоректної суми. При спробі ввести суму, нижчу за поточну найвищу ставку плюс мінімальний крок та підтвердити дану дію, користувач одразу побачить спливаюче повідомлення зверху екрану від сервера про помилку «Ціна повинна бути більшою за поточну». На рисунку 4.4 зображено помилку при спробі введеня меншої суми ставки ніж це вимагає система.

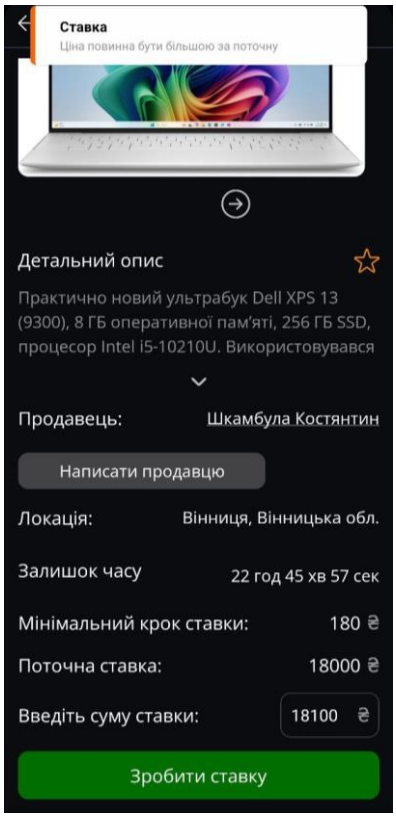

Рисунок 4.4 — Помилка при введенні некоректної суми ставки

Обробка некоректної ставки працює правильно та повідомляє користувача про відповідним повідомленням, тепер перевіримо успішну відправку коректної ставки. Користувач вводить коректну суму, натискає кнопку «Зробити ставку» та підтверджує модальне вікно. Після цього з’являється спливаюче повідомлення про успішну операцію — «Успішно відправлено».

Якщо сервер повертає успішний статус-код, тобто ставка була коректно створена користувачем та оброблена на стороні бекенду, то інтерфейс оновлює відображення поточної найвищої ставки, і нова сума з’являється в розділі «Поточна ставка». Також, якщо поточна ставка користувача є найвищою, то перед кнопкою буде текст — «Ваша ставка лідирує», це значить, що в даний момент його ставка вища за всі попередні. На рисунку 4.5 продемонстровано повідомлення про успіх, а на рисунку 4.6 — оновлений індикатор поточної ставки.

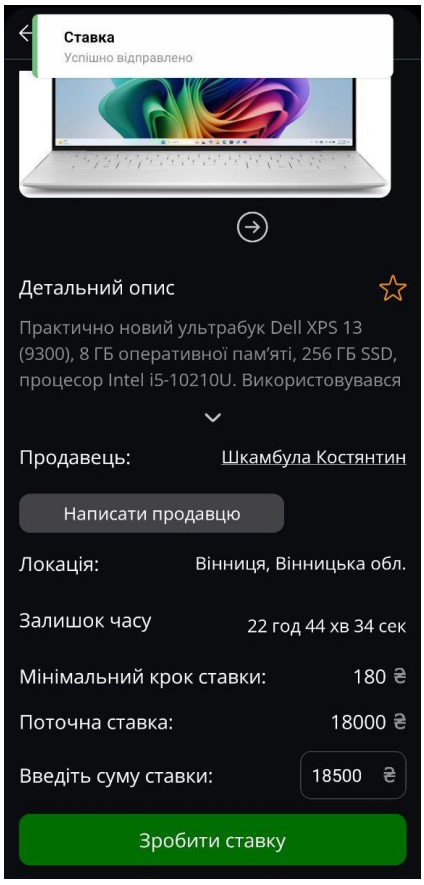

Рисунок 4.5 — Повідомлення про успішну відправку ставки

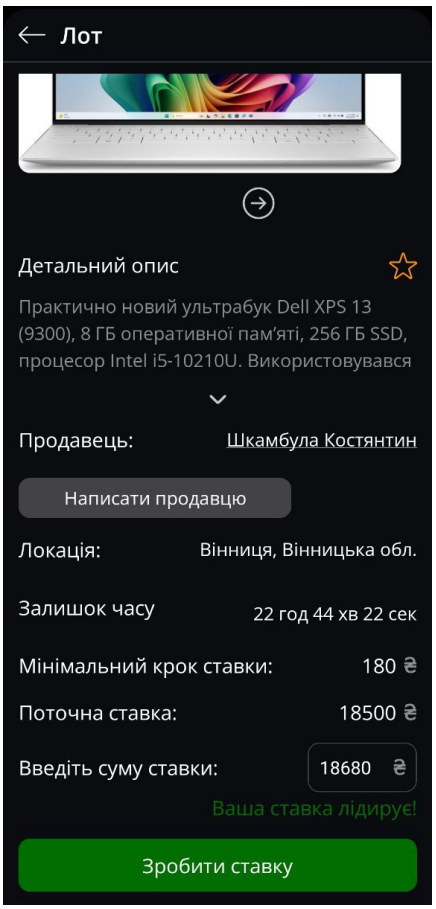

Рисунок 4.6 — Оновлення поточної найвищої ставки після відправки

В умовах одночасної роботи двох клієнтів обробка ставки та отримання відповідного повідомлення про підтвердження відбувається за короткий термін від 300 мс до 500 мс. Після зміни даних лоту інформація оновлюється не пізніше ніж за 3 с. Навіть при активному навантаженні системи всі запити успішно приходять без помилок та дублювань.

Отже, тестування підтвердило, що система коректно обробляє обидва ключові сценарії участі в торгах, вона відображає помилки при занадто низькій ставці та успішно приймає правильні ставки з миттєвим оновленням інтерфейсу. Крім того, модальне вікно підтвердження та спливаючі повідомлення працюють без збоїв, що мінімізує ризик випадкових помилок користувача. Автоматичне оновлення даних через гарантує, що інформація завжди залишається актуальною, а повторні запити не створюють зайвого навантаження на сервер.

4.3 Тестування профілю та авторизації

Для тестування функцій профілю та авторизації потрібно переконатися, що користувач може не лише безпомилково увійти в систему, а й оновити свої персональні дані і побачити ці зміни в своєму профілі. Для цього в даному підрозділі розглянемо наступні три сценарії: авторизація користувача, редагування профілю з введенням нового повного імені та завантаженням аватару, а також відображення оновлених даних на екрані профілю.

Перевіримо коректність авторизації користувача. Для цього потрібно у відповідну форму на сторінці авторизації ввести електронну пошту та пароль і натиснути кнопку «Увійти в аккаунт». Після успішного запиту на сервер відкривається головна сторінка та з’являється повідомлення «Ви успішно авторизувались». На рисунку 4.7 наведено стан екрану відразу після авторизації та повідомлення.

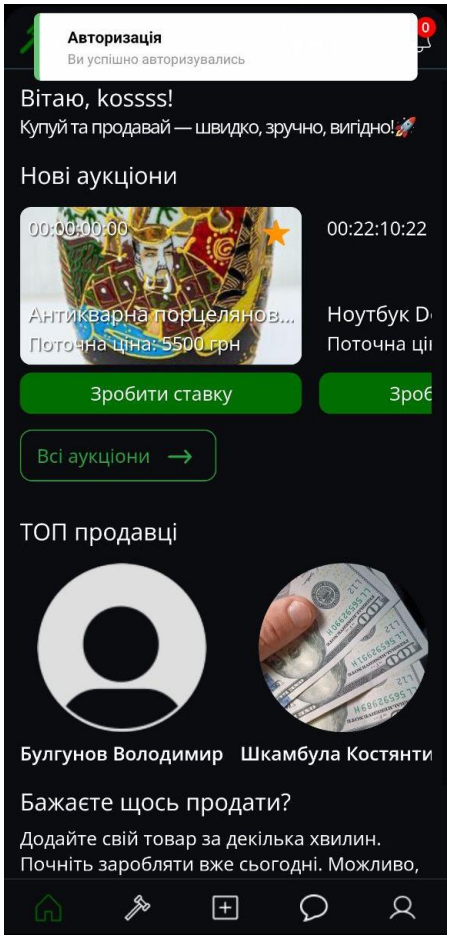

Рисунок 4.7 — Екран після успішної авторизації користувача

Після успішної авторизації, користувач має можливість користуватися усіма функціями додатку, однією з яких є редагування власного профілю. Для цього потрібно зайти в налаштування профілю, для тестування змінюємо поля «Ім’я та прізвище» та «Аватар».

У відповідних полях вводимо нове ім’я та обираємо потрібний файл з фотографією, при потребі його можна відформатувати. Після натискання кнопки «Підтвердити» система повертає повідомлення для підтвердження успішного оновлення — «Успішно змінено». На рисунку 4.8 показано екран редагування профілю з полями, заповненими новими значеннями, й повідомленням про успіх.

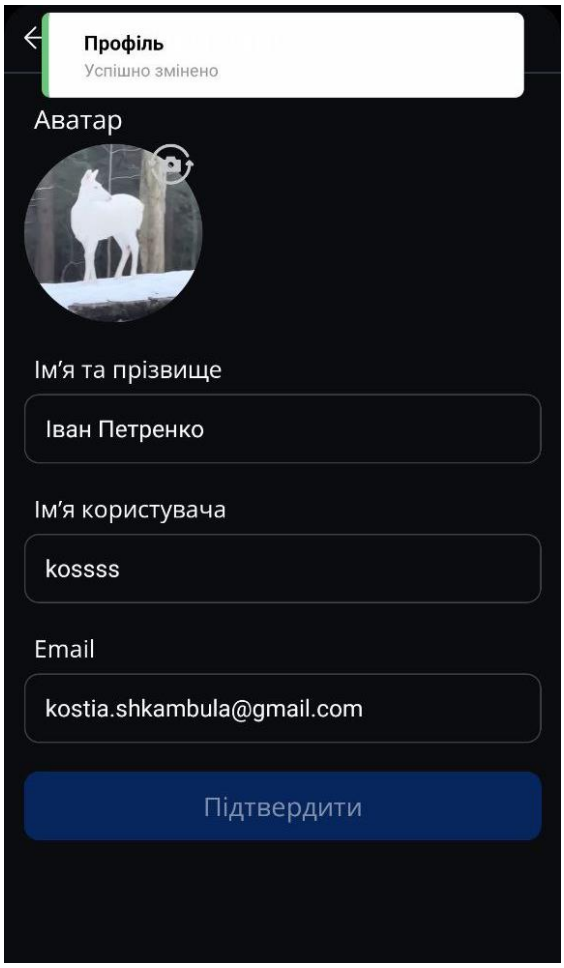

Рисунок 4.8 — Екран редагування профілю

А також перевіримо, що оновлені дані дійсно відображені у профілі. Переходимо на екран «Профіль» — ім’я та аватар замінені на нові. На

рисунку 4.9 зображено фінальний вигляд сторінки профілю з оновленими даними.

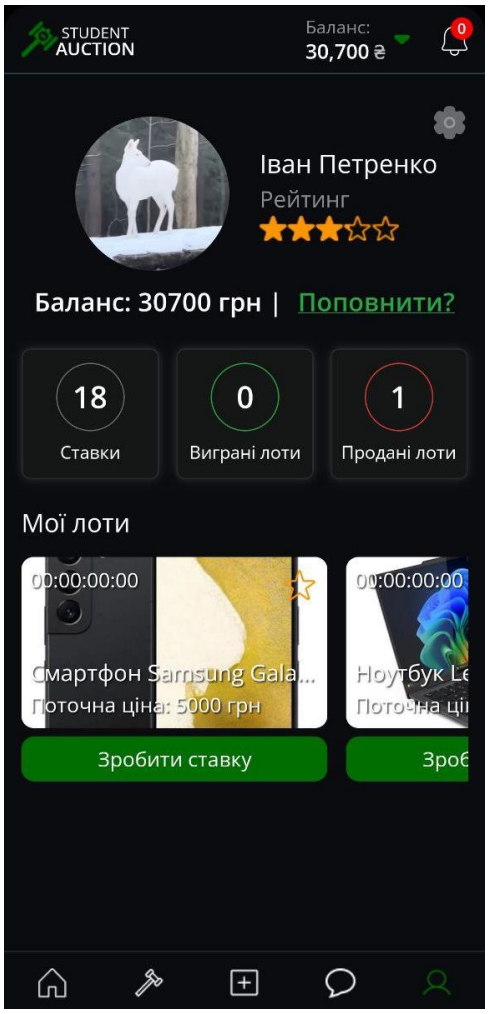

Рисунок 4.9 — Оновлений екран профілю

Авторизація користувача та перехід на головну сторінку займає приблизно 500 мс при підключенні до стабільного Wi-Fi. Після успішної авторизації переходи між сторінками застосунку відбуваються миттєво. При зміні інформації профілю данні обробляються за 300 мс, оновлення данних на сторінці «Профіль» також відбувається за короткий термін, що не перевищує 600 мс. Відсутність блокувань інтерфейсу під час цих операцій свідчить про асинхронну обробку і хорошу оптимізацію процесів.

Отже, механізми авторизації, реєстрації та редагування профілю працюють коректно, забезпечуючи користувачам можливість змінювати персональну інформацію та бачити зміни в інтерфейсі без затримок.

4.4 Тестування чату (Websocket)

Для перевірки стабільності та коректності роботи модуля чатів використовується реалізація на основі WebSocket, що забезпечує миттєвий обмін повідомленнями між користувачами. Тестування включало кілька ключових сценаріїв.

Перший сценарій — надсилання та отримання текстових повідомлень. У процесі тестування базового сценарію обміну текстовими повідомленнями потрібно звернути увагу не лише на факт доставки, а й на поведінку інтерфейсу під час відправки та прийому.

Спочатку двоє користувачів відкривають один і той же діалог на різних пристроях. Коли перший вводить «Привіт. Як справи?» і натискає кнопку відправлення, він одразу бачить власне повідомлення у вигляді бульбашки праворуч. Одночасно на іншому пристрої, під час активного перегляду тієї ж розмови, з’являється нова бульбашка з тим самим текстом ліворуч без жодної помітної затримки, зображення наведено на рисунку 4.10.

Рисунок 4.10 — Відправлення тексту в чаті

Другий сценарій — надсилання картинкок. Користувач відправляє фото, за допомогою хука useUpload миттєво витягується посилання на картинку з пристрою, створюється об’єкт для створення повідомлення та відправляється на сервер, наведено на рисунку 4.11. Було перевіріно різні формати та розміри картинок, всі вони успішно обробляються та додаються в чат.

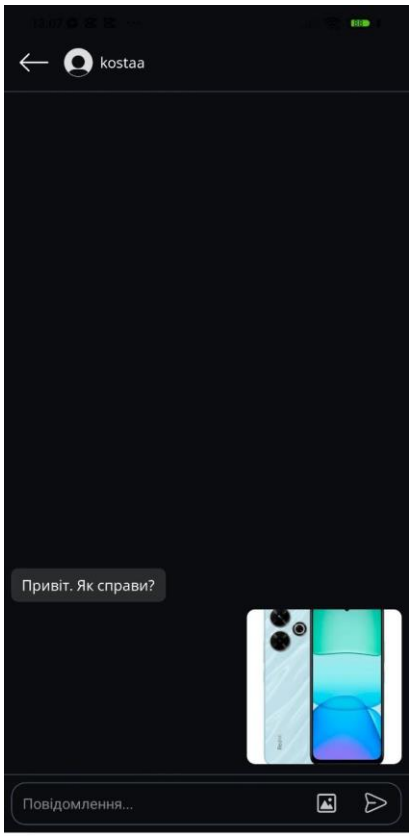

Рисунок 4.11 — Відправлення картинки в чаті

Під час тестування стійкості чату до перебоїв мережі було перевірено поведінку клієнта при відключенні й подальшому відновленні зв’язку. На час вимкненого Wi-Fi усі вхідні повідомлення залишались на сервері і після повернення мережі синхронізувалися в інтерфейсі в правильному порядку. Це гарантує, що жодне повідомлення не загубиться і користувачі побачать історію спілкування в повному обсязі.

Також перевірено базову валідацію безпеки, спроби відправити порожнє повідомлення або занадто довгий текст, коректно блокуються, що не дає змогу нагрузити сервер чи націлено спричинити помилку в додатку.

Час доставки текстового повідомлення між двома користувачами фактично миттєвий завдяки WebSocket. Надсилання зображення відбувається з незначною затримкою близько 600 мс, цей час залежить від розміру надісланого файлу. У разі обриву з'єднання клієнт автоматично перепідключається до чату, а отримання пропущених повідомлень відбувається не пізніше ніж за 2 с.

Таким чином, через детальне тестування всіх сценаріїв роботи WebSocket-чату було доведено його надійність, стійкість до мережевих відмов та здатності обробляти великі навантаження, а також у коректній валідації вхідних даних.

4.5 Тестування адміністративної панелі

Адміністративна панель є важливим інструментом для модерації та управління контентом платформи. У цьому підрозділі перевіряється коректність роботи функціоналу адміністративної панелі, можливості видалення користувача та оновлення даної інформації в мобільному додатку.

Для початку відкриваємо розділ «Користувачі» в адміністративній панелі та очікуємо завантаження таблиці з усіма профілями. На екрані відображаються стовпці з ім'ям користувача, email, роллю та кнопкою для можливості видалення, наведено на рисунку 4.12.

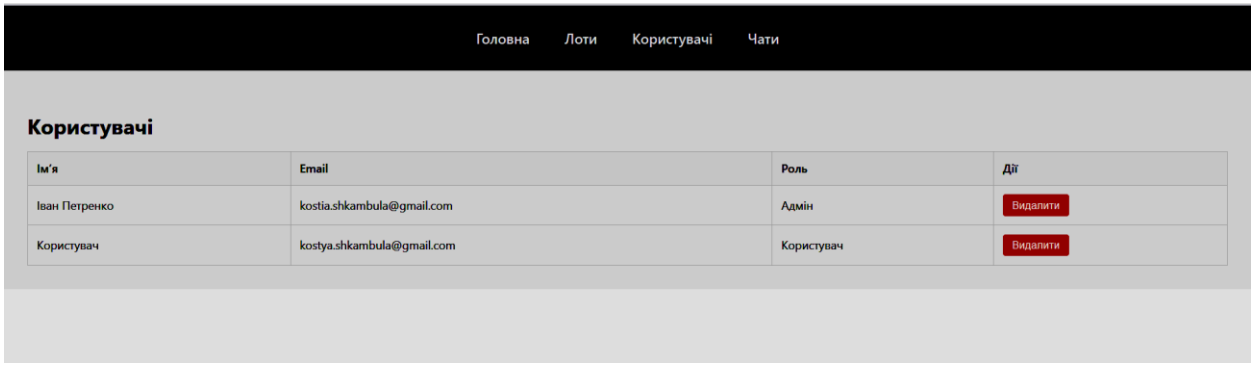

Рисунок 4.12 — Список користувачів у адміністративній панелі

Для того, щоб видалити при потребі користувача потрібно обрати даного користувача в таблиці та натиснути кнопку «Видалити». Після цього, обліковий

запис обраного користувача зникає зі списку в адміністративній панелі, а також буде недоступний в застосунку. Вигляд сторінки в адміністративній панелі після видалення користувача з ім’ям «Користувач» наведено на рисунку 4.13.

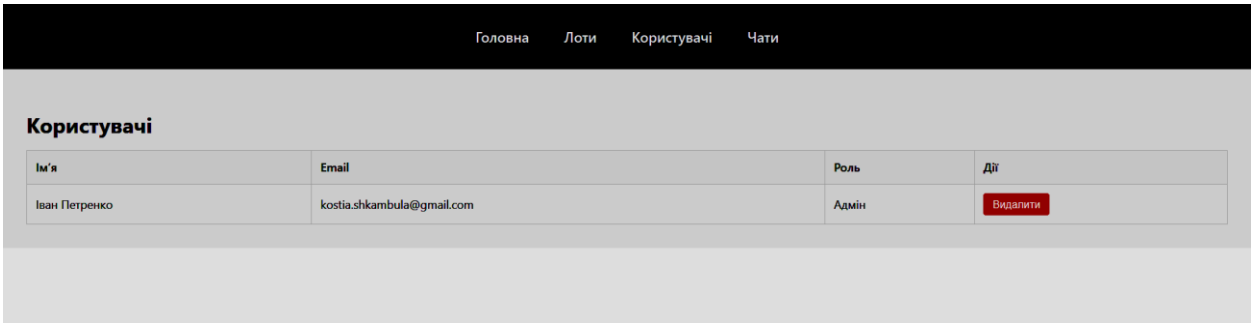

Рисунок 4.13 — Вигляд сторінки після видалення користувача

Перевіримо коректність видалення в застосунку, на головній сторінці відображаються наявні користувачі, тепер можна побачити, що в застосунку лише один зареєстрований користувач. Вигляд головної сторінки моільного застосунку після видалення користувача наведено на рисунку 4.14.

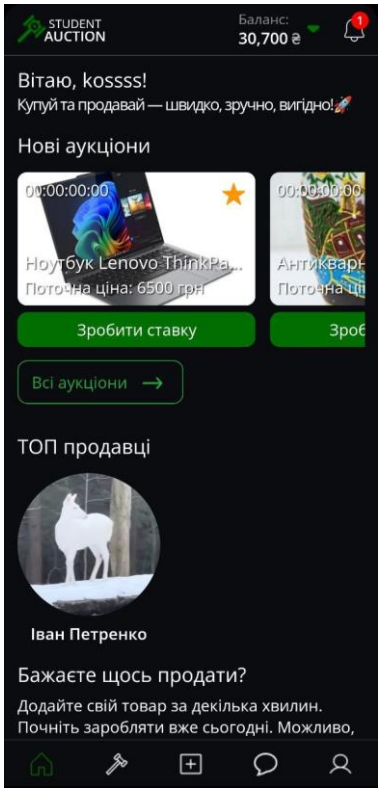

Рисунок 4.14 — Відсутність профілю видаленого користувача

Час завантаження розділу «Користувачі» з відображенням таблиці з кількома записами становить приблизно 200 мс (цей показник зростатиме разом із кількістю записів). Видалення користувача та оновлення даних у застосунку займає від 400 мс до 600 мс. Навіть під час навантаження кількох послідовних операцій вмдалення користувача та оновлення цієї інформації в застосунку, інтерфейс залишається стійким і не демонструє помітних затримок в часі, що свідчить про високу швидкодію сервера та ефективне кешування на стороні клієнта.

Таким чином, функціонал адміністративної панелі для перегляду та видалення потрібних даних працює коректно. Видалення миттєво відображаються як в адміністративній панелі, так і в мобільному застосунку, що гарантує надійне адміністрування платформи.

ВИСНОВКИ

У даній бакалаврській роботі розроблено мобільний застосунок для проведення студентського аукціону. В ході виконання роботи було виконано ряд завдань, які забезпечили створення ефективної та надійної платформи.

Проведено детальний аналіз існуючих платформ для проведення онлайн-аукціонів, від найпопулярніших застосунків, як eBay, до вузькоспеціалізованих рішень таких як Catawiki і LiveAuctioneers, який дозволив виокремити ті аспекти інтерфейсу та механіки, що є критично важливими для швидкого й інтуїтивного користування, водночас відзначити недоліки у навігації, високих комісіях чи складних параметрах реєстрації, які слід уникнути.

Визначення потреб саме студентської аудиторії сприяло тому, аби створити систему із мінімальними комісіями, можливістю локальної верифікації і рейтинговим механізмом, що формує довіру сучасного покоління студентів.

Особливу увагу було приділено UX/UI-дизайну. Розроблена темна кольорова палітра з яскравими акцентами дозволяє виділити ключові елементи управління, а набір базових компонентів використовується на всіх екранах. Прототипи в Figma та їхнє втілення у коді гарантують зручність користування на різних пристроях і розмірах екранів.

На основі отриманих даних була спроєктована клієнт-серверна архітектура шляхом побудови REST API на NestJS із жорсткою валідацією даних за допомогою DTO-класів та декоратора ValidationPipe. Доступ до інформації захищено JWT-автентифікацією та HTTPS-протоколом. Інтеграція Websocket та cron-задач забезпечило обробку подій та обмін інформацією у реальному часі, зокрема для реалізації real-time чатів.

Для реалізації ключових функцій застосунку були виділені окремі модулі: авторизація та управління обліковими записами, створення й перегляд лотів, подання ставок, робота з повідомленнями та відгуками. На стороні клієнта ці модулі реалізовані як набір повторно використовуваних компонентів і кастомних хуків, що спрощує подальшу підтримку та розширення функціоналу.

Після технічної розробки мобільний застосунок було протестовано. Перевірено коректність створення лотів, механізми валідації ставок, роботу форм авторизації та редагування профілю, стабільність роботи слайдерів і списків під навантаженням, а також функціонал адміністративної панелі. Результати тестування підтвердили високу продуктивність і надійність системи, а також відповідність вимогам безпеки завдяки використанню JWT-токенів, HTTPS та коректної валідації.

Для подальшого розвитку розробленого застосунку доцільно інтегрувати платіжні системи, що автоматизують фінансові операції. На рівні інтерфейсу бажано впровадити персоналізовані теми оформлення, адаптивні анімації та розширені фільтри в каталозі, ґрунтуючись на зворотному зв’язку користувачів. Крім того, рекомендаційна система лотів, динамічне ціноутворення й алгоритмів виявлення шахраїв дозволить підвищити залученість користувачів і безпеку торгів. Ці напрями розвитку роблять мобільний застосунок «Студентський аукціон» не просто корисним інструментом для студентів сьогодні, а й перспективною платформою, здатною адаптуватися та масштабуватися під потреби майбутніх поколінь.

Таким чином, дана бакалаврська робота повністю виконує поставлені завдання та підтверджує актуальність розробки мобільного аукціонного сервісу. Мобільний застосунок «Студентський аукціон» забезпечує інтуїтивний інтерфейс для створення лотів, участі в торгах та обміну повідомленнями, демонструючи стабільність, безпеку та зручність у користуванні. Розроблене рішення закладає міцний фундамент для подальших досліджень і вдосконалення функціоналу платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп’ютерна онлайн-система проведення студентських аукціонів у мобільному додатку // Костянтин Миколайович Шкамбула, Олександр Вікторович Дудник : вебсайт. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23859.

2. EBay – Official Website: вебсайт. URL: https://www.ebay.com/ (дата звернення: 22.03.2025).

3. Catawiki – Online Auction for Collectibles: вебсайт. URL: https://www.catawiki.com/en (дата звернення: 22.03.2025).

4. LiveAuctioneers – Live Online Auctions: вебсайт. URL: https://www.liveauctioneers.com/ (дата звернення: 24.03.2025).

5. React Native Documentation: вебсайт. URL: https://reactnative.dev/ (дата звернення: 30.03.2025).

6. Вступ до JSX: вебсайт. URL: https://uk.legacy.reactjs.org/docs/introducing-jsx.html (дата звернення: 01.04.2025).

7. JavaScript | MDN Web Docs: вебсайт. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 02.04.2025).

8. React Documentation: вебсайт. URL: https://react.dev/ (дата звернення: 02.04.2025).

9. Expo Development Tools: вебсайт. URL: https://expo.dev/ (дата звернення: 03.04.2025).

10. REST: вебсайт. URL: https://uk.wikipedia.org/wiki/REST (дата звернення: 04.04.2025).

11. HTTP: вебсайт. URL: https://uk.wikipedia.org/wiki/HTTP (дата звернення: 04.04.2025).

12. NestJS – Progressive Node.js Framework: вебсайт. URL: https://nestjs.com/ (дата звернення: 05.04.2025).

13. Що таке API: простими словами про складне: вебсайт. URL: https://hostiq.ua/blog/ukr/what-is-api/ (дата звернення: 05.04.2025).

14. MongoDB – NoSQL Database: вебсайт. URL: https://www.mongodb.com/ (дата звернення: 05.04.2025).

15. Figma: The Collaborative Interface Design Tool: вебсайт. URL: https://www.figma.com/ (дата звернення: 08.04.2025).

16. Web Content Accessibility Guidelines (WCAG 2.1): вебсайт. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 09.04.2025).

17. REST API Architecture – Medium Article: вебсайт. URL: https://medium.com/@shikha.ritu17/rest-api-architecture-6f1c3c99f0d3 (дата звернення: 15.04.2025).

18. DTO: вебсайт. URL: https://uk.wikipedia.org/wiki/DTO (дата звернення: 15.04.2025).

19. MDN Web Docs: HTTP Methods Reference: вебсайт. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods (дата звернення: 16.04.2025).

20. Introduction to JSON Web Tokens: вебсайт. URL: https://jwt.io/introduction (дата звернення: 17.04.2025).

21. Що таке JSON. Усе про цей формат передачі даних в інтернеті: вебсайт. URL: https://apix-drive.com/ua/blog/useful/scho-take-json (дата звернення: 20.04.2025).

22. Управління розподіленими cron-завданнями в NestJS: вебсайт. URL: https://javascript.org.ua/upravlinnya-rozpodilenimi-cron-zavdannyami-v-nestjs-vid-bazovih-do-rishen-gotovih-do-virobnicztva/ (дата звернення: 22.04.2025).

23. Mongoose – MongoDB ODM: вебсайт. URL: https://mongoosejs.com/ (дата звернення: 23.04.2025).

24. Typegoose – TypeScript Wrapper for Mongoose: вебсайт. URL: https://typegoose.github.io/typegoose/ (дата звернення: 23.04.2025).

25. React Hook Form – Form Validation Library: вебсайт. URL: https://react-hook-form.com/ (дата звернення: 25.04.2025).

26. TanStack Query (React Query) Documentation: вебсайт. URL: https://tanstack.com/query/latest (дата звернення: 25.04.2025).

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., проф. Азаров О. Д.

«21» березня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

«Мобільний застосунок "Студентський аукціон"»

08-54.БКР.020.00.000.ПЗ

Науковий керівник: доцент к.т.н.

_____ Дудник О. В.

Студент групи 1КІ-23мс

_____ Шкамбула К. М.

м. Вінниця — 2025

1 Підстава для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 В умовах активного зростання попиту студентів на безпечні методи купівлі та продажу товарів по вигідних цінах виникає потреба в мобільному застосунку «Студентський аукціон», що поєднує функціонал аукціону та адаптований під мобільні пристрої. Це рішення дозволяє оптимізувати та зробити безпечним процес торгів, а також спростити комунікацію між продавцем та покупцем в межах додатку.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — створення мобільного застосунку «Студентський аукціон», який забезпечує інтуїтивно зрозумілий інтерфейс для участі в аукціоні між студентами, миттєвий обмін між продавцем та покупцем в реальному часі, надійний механізм реєстрації та авторизації, а також стабільну та зручну роботу в різних умовах перебування.

2.2 Призначення розробки — створити доступний і зручний інструмент для студентів, який спростить процес купівлі та продажу товарів в форматі аукціону, забезпечить надійний механізм торгів, швидкий пошук, а також комфортну комунікацію між учасниками в мобільному застосунку.

3 Вихідні дані для виконання БКР

3.1 Аналітичний огляд існуючих мобільних аукціонних платформ.

3.2 Опис потреб студентської аудиторії.

3.3 Дизайн інтерфейсу додатку.

3.4 Технічна реалізація мобільного застосунку «Студентський аукціон».

3.5 Тестування функціональності платформи.

4 Вимоги до виконання БКР

Головна вимога — забезпечення надійного та зручного процесу торгів в режимі реального часу з комфортною комунікацією між користувачами мобільного застосунку.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати наведено у Таблиці А.1.

Таблиця А.1 — Етапи БКР

№	Назва та зміст етапу	Термін виконання	Примітка
1.	Постановка задачі	21.03.25	
2.	Аналіз предметної області та вимог до мобільного застосунку «Студентський аукціон»	22.03.25 — 29.03.25	Розділ 1
3.	Аналіз та обґрунтування вибору технологій	30.03.25 — 06.04.25	Розділ 1
5.	Розробка дизайн системи додатку	07.04.25 — 14.04.25	Розділ 2
5.	Технічна реалізація мобільного застосунку	15.04.25 — 30.04.25	Розділ 3
6.	Тестування функціональності платформи	01.05.25 — 03.05.25	Розділ 4
7.	Оформлення пояснювальної записки	04.05.25 — 12.05.25	
8.	Перевірка якості виконання бакалаврської роботи	13.05.25	

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 – «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Мобільний застосунок «Студентський аукціон»

Тип роботи: бакалаврська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97% Схожість 3%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення 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 _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

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

Автор роботи _____ Шкамбула К. М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Дудник О. В.
(підпис) (прізвище, ініціали)

ДОДАТОК Г

Ілюстративна частина

МОБІЛЬНИЙ ЗАСТОСУНОК «СТУДЕНТСЬКИЙ АУКЦІОН»

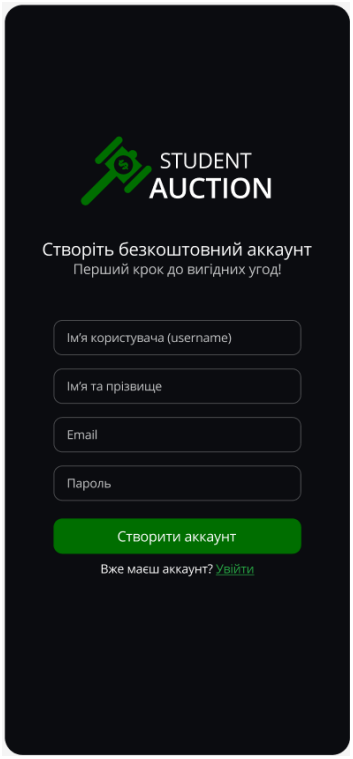

Рисунок Г.1 — Графічний інтерфейс сторінки «Авторизація»

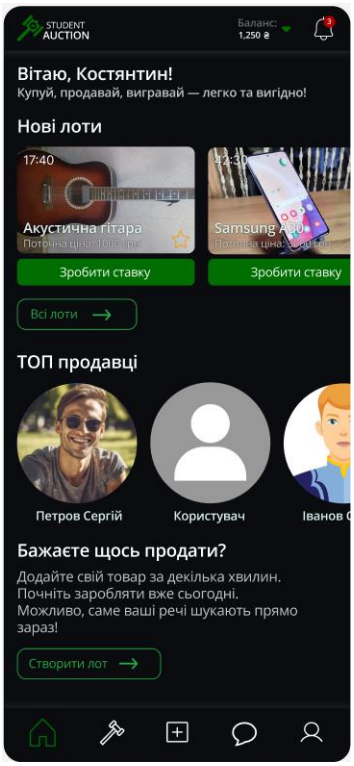

Рисунок Г.2 — Графічний інтерфейс сторінки «Головна»

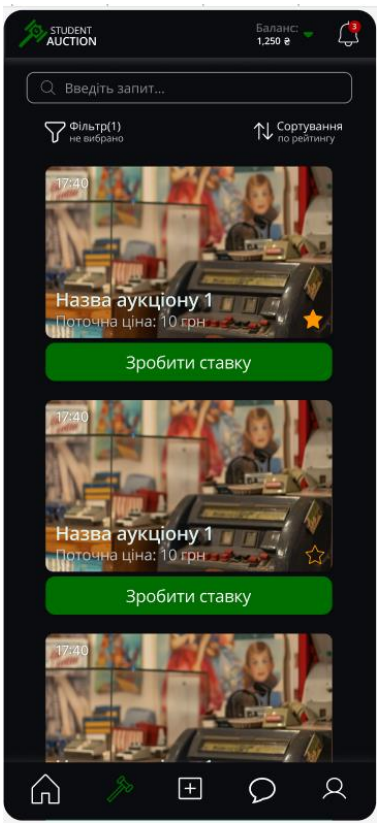

Рисунок Г.3 — Графічний інтерфейс сторінки «Каталог лотів»

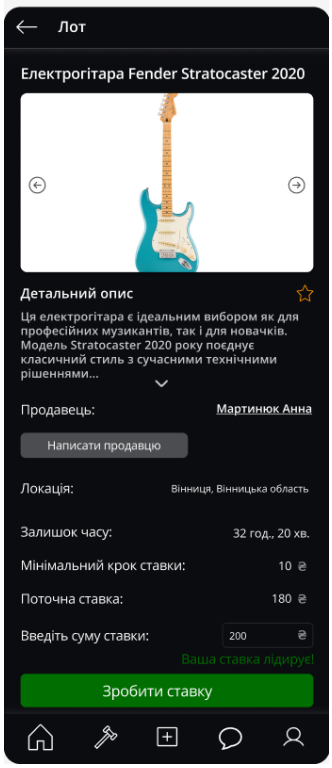

Рисунок Г.4 — Графічний інтерфейс сторінки «Деталі лоту»

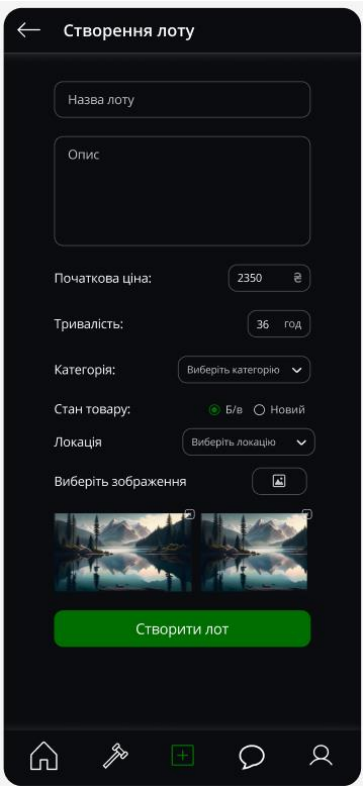

Рисунок Г.5 — Графічний інтерфейс сторінки «Створення лоту»

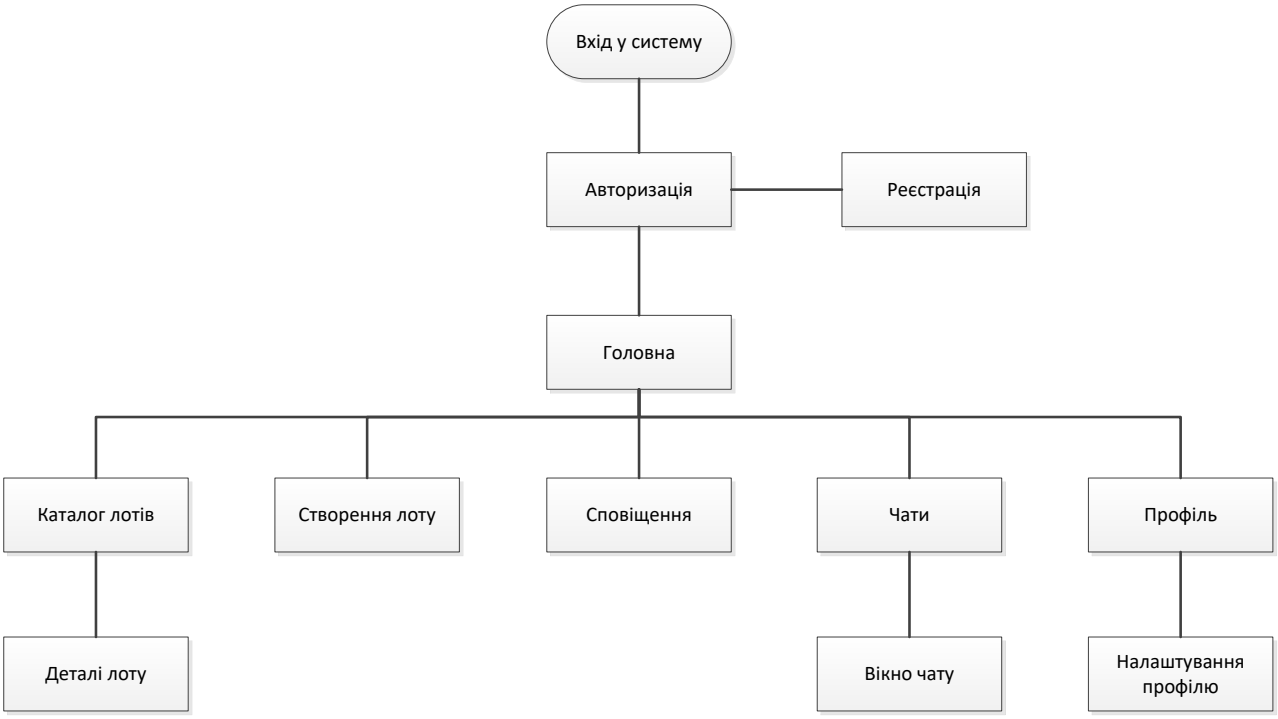

Рисунок Г.6 — Структурна схема мобільного застосунку «Студентський аукціон»

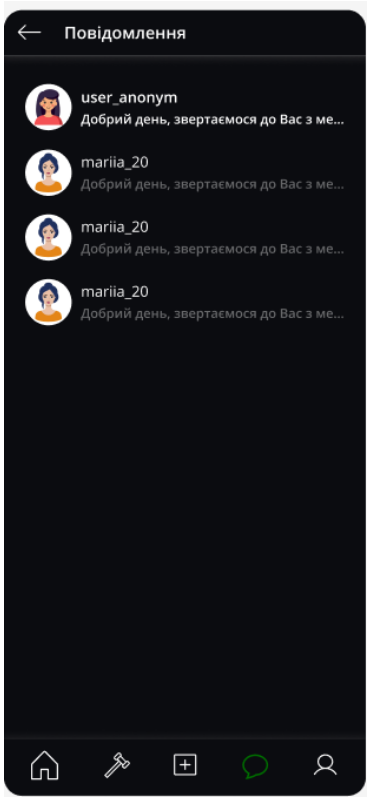

Рисунок Г.7 — Графічний інтерфейс сторінки «Чати»

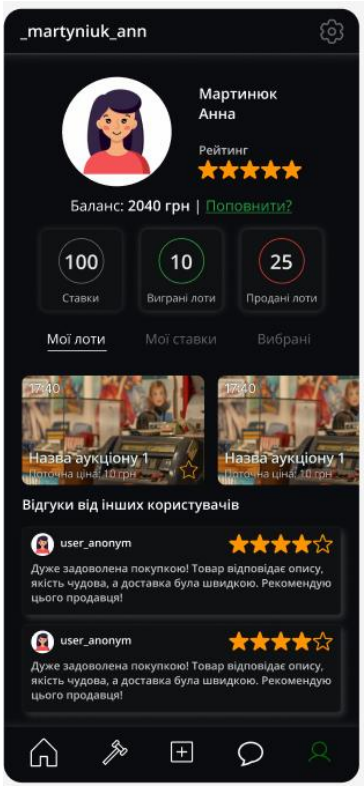

Рисунок Г.8 — Графічний інтерфейс сторінки «Профіль»

ДОДАТОК Д

Лістинг файлів застосунку

Лістинг Д.1 — Лістинг файлу SignUp.tsx

```
import { Link, router } from "expo-router";
import { Image, ScrollView, Text, View } from "react-native";
import { StatusBar } from "expo-status-bar";
import StyledText from "@/components/ui/StyledText";
import FormInput from "@/components/ui/FormInput";
import { SafeAreaView } from "react-native-safe-area-context";
import StyledButton from "@/components/ui/StyledButton";
import { useAuthStore } from "@/store/useAuthStore";
import { Controller, useForm } from "react-hook-form";
import Loader from "@/components/loaders/Loader";
interface ISignUp {
  email: string;
  password: string;
  userName: string;
  name: string;
}
const SignUp = () => {
  const { register, isLoading } = useAuthStore((state) => state);

  const {
    control,
    handleSubmit,
    formState: { errors },
  } = useForm<ISignUp>({
    defaultValues: {
      email: "",
      password: "",
      name: "",
      userName: "",
    },
  });

  const onSubmit = (data: ISignUp) => {
    register(data.email, data.password, data.userName, data.name);
  };

  return (
    <SafeAreaView className="bg-black h-full">
      <ScrollView
        keyboardShouldPersistTaps="handled"
        contentContainerStyle={{ height: "100%" }}
      >
        <View className="h-full w-full items-center justify-center px-4">
          <Image
            className="w-[220px] h-[80px]"
            resizeMode="contain"
```

```
            source={require("@/assets/images/big-logo.png")}
          />
          <View className="mt-10 mb-12">
            <StyledText className="mb-[3px] text-center text-xl">
              Створіть безкоштовний аккаунт
            </StyledText>
            <StyledText className="text-gray-50p text-center">
              Перший крок до вигідних угод!
            </StyledText>
          </View>
          <View className="w-[320px]">
            <View>
              <Controller
                control={control}
                rules={{
                  required: {
                    value: true,
                    message: "Поле обов'язкове",
                  },
                }}
                render={({ field: { onChange, onBlur, value } }) => (
                  <FormInput
                    placeholder="Ім’я користувача (username)"
                    onBlur={onBlur}
                    onChangeText={onChange}
                    value={value}
                  />
                )}
                name="userName"
              />
              {errors.userName && (
                <Text className="absolute top-[2px] right-2 text-[10px] text-red
font-openslight">
                  {errors.userName.message}
                </Text>
              )}
            </View>
            <View>
              <Controller
                control={control}
                rules={{
                  required: false,
                }}
                render={({ field: { onChange, onBlur, value } }) => (
                  <FormInput
                    placeholder="Ім’я та прізвище"
                    onBlur={onBlur}
                    onChangeText={onChange}
                    value={value}
                  />
                )}
                name="name"
```

```
                />
                {errors.name && (
                  <Text className="absolute top-[2px] right-2 text-[10px] text-red
font-openslight">
                    {errors.name.message}
                  </Text>
                )}
              </View>
              <View>
                <Controller
                  control={control}
                  rules={{
                    required: {
                      value: true,
                      message: "Поле обов'язкове",
                    },
                    pattern: {
                      value: /^[^\s@]+@[^\s@]+\.[^\s@]+$/,
                      message: "Введено некоректний email",
                    },
                  }}
                  render={({ field: { onChange, onBlur, value } }) => (
                    <FormInput
                      placeholder="Email"
                      onBlur={onBlur}
                      onChangeText={onChange}
                      value={value}
                    />
                  )}
                  name="email"
                />
                {errors.email && (
                  <Text className="absolute top-[2px] right-2 text-[10px] text-red
font-openslight">
                    {errors.email.message}
                  </Text>
                )}
              </View>
              <View>
                <Controller
                  control={control}
                  rules={{
                    required: "Поле обов'язкове",
                    minLength: {
                      value: 6,
                      message: "Пароль має бути не менше 6 символів",
                    },
                  }}
                  render={({ field: { onChange, onBlur, value } }) => (
                    <FormInput
                      placeholder="Пароль"
                      onBlur={onBlur}
```

```
                onChangeText={onChange}
                value={value}
              />
            )}
            name="password"
          />
          {errors.password && (
            <Text className="absolute top-[2px] right-2 text-[10px] text-red font-openslight">
              {errors.password.message}
            </Text>
          )}
        </View>
        <StyledButton                    className="mt-3                    mb-2" handlePress={handleSubmit(onSubmit)}>
          Створити аккаунт
        </StyledButton>
        <View
          style={{ flexDirection: "row", gap: 3 }}
          className="justify-center"
        >
          <StyledText className="text-gray-50p">
            Вже маєш аккаунт?
          </StyledText>
          <Link href="/sign-in" className="text-green font-opensregular">
            Увійти
          </Link>
        </View>
      </View>
      {isLoading && <Loader />}
    </View>
  </ScrollView>

  <StatusBar backgroundColor="#0B0C10" style="light" />
 </SafeAreaView>
 );
};
export default SignUp;
```

Лістинг Д.2 — Лістинг файлу Home.tsx

```
import Auction from "@/components/Auction";
import HomeAuctionsLoader from "@/components/loaders/HomeAuctionsLoader";
import HomeTopSellersLoader from "@/components/loaders/HomeTopSellersLoader";
import Slider from "@/components/Slider";
import DirectButton from "@/components/ui/DirectButton";
import StyledText from "@/components/ui/StyledText";
import { useAuth } from "@/hooks/useAuth";
import { router } from "expo-router";
import { useCallback, useState } from "react";
import { RefreshControl, ScrollView, View } from "react-native";
import { useAuctions } from "../auctions/useAuctions";
import SellersItem from "./SellersItem";
import { useGetTopSellers } from "./useGetTopSellers";
const Home = () => {
  const user = useAuth();
  const { auctions, isLoading, refetch: refetchAuctions } = useAuctions();
  const { topSellers, isLoading: topSellersLoading, refetch: refetchSellers } = useGetTopSellers();
  const [refreshing, setRefreshing] = useState(false);
  const onRefresh = useCallback(async () => {
    setRefreshing(true);
    await Promise.all([
      refetchAuctions(),
      refetchSellers(),
    ]);
    setRefreshing(false);
  }, [refetchAuctions, refetchSellers]);
  return (
    <ScrollView className="bg-black" refreshControl={
      <RefreshControl
        refreshing={refreshing}
        onRefresh={onRefresh}
      />
    }>
      <View className="px-4">
        <View className="mb-5">
          <StyledText className="text-xl font-opensmedium">
            Вітаю, {user?.name}!
          </StyledText>
          <StyledText className="-tracking-widest">
            Купуй та продавай — швидко, зручно, вигідно!🚀
          </StyledText>
        </View>
        <View>
          <StyledText className="text-xl font-opensmedium mb-4">
            Нові аукціони
          </StyledText>
          {isLoading ? (
```

```
            <Slider
              data={["", ""]}
              renderItem={() => <HomeAuctionsLoader />}
            ></Slider>
          ) : (
            <Slider
              data={auctions.slice(0, 8)}
              renderItem={(item, index) => <Auction {...item} key={index} />}
            />
          )}
        </View>
        <View className="mt-4">
          <DirectButton handlePress={() => router.push("/auctions")}>
            Всі аукціони
          </DirectButton>
        </View>
        <View className="mt-[30px] mb-5">
          <StyledText className="text-xl font-opensmedium mb-4">
            ТОП продавці
          </StyledText>
          {topSellersLoading ? (
            <Slider
              data={["", ""]}
              renderItem={() => <HomeTopSellersLoader />}
            ></Slider>
          ) : (
            <Slider
              data={topSellers}
              renderItem={(item, index) => <SellersItem {...item} key={index} />}
            />
          )}
        </View>
        <View className="mb-3">
          <StyledText className="text-xl font-opensmedium mb-2">
            Бажаєте щось продати?
          </StyledText>
          <StyledText className="text-gray-400">
            Додайте свій товар за декілька хвилин. Почніть заробляти вже
            сьогодні. Можливо, саме ваші речі шукають прямо зараз!
          </StyledText>
        </View>
        <View className="mb-5">
          <DirectButton handlePress={() => router.push("/create")}>
            Створити аукціон
          </DirectButton>
        </View>
      </View>
    </ScrollView>
  );
};
export default Home;
```

Лістинг Д.3 — Лістинг файлу Auctions.tsx

```
import Auction from "@/components/Auction";
import FilterIcon from "@/components/icons/FilterIcon";
import SortIcon from "@/components/icons/SortIcon";
import AuctionLoader from "@/components/loaders/AuctionLoader";
import SearchInput from "@/components/ui/SearchInput";
import StyledText from "@/components/ui/StyledText";
import { useEffect, useState } from "react";
import { Pressable, RefreshControl, ScrollView, View } from "react-native";
import Filter from "./filter/Filter";
import Sort from "./Sort";
import { AuctionParams, useAuctions } from "./useAuctions";
export const initialParams: AuctionParams = {
  search: "",
  category: "",
  price: "",
  condition: "",
  sortBy: "popularity",
};
const Auctions = () => {
  const [isShowFilter, setIsShowFilter] = useState(false);
  const [isShowSort, setIsShowSort] = useState(false);
  const [query, setQuery] = useState<AuctionParams>(initialParams);
  const [refreshing, setRefreshing] = useState(false);
  const {auctions, isLoading, refetchWithParams} = useAuctions(query);
  const handleSubmitInput = (text: string) => {
    setQuery({ ...query, search: text });
  }
  const changeSort = (newSort: 'newest' | 'popularity' | 'priceUp' | 'priceDown')
=> {
    setQuery({ ...query, sortBy: newSort });
  }
  const changeQuery = (newQuery: AuctionParams) => {
    setQuery(newQuery);
  }
  useEffect(() => {
    refetchWithParams(query)
  }, [query])

  const openFilter = () => {
    setIsShowFilter(true);
    setIsShowSort(false);
  }
  const openSort = () => {
    setIsShowFilter(false);
    setIsShowSort(true);
  }
  const closeFilter = () => setIsShowFilter(false);
  const closeSort = () => setIsShowSort(false);
  const onRefresh = async () =>
  {
    setRefreshing( true );
```

```
        await refetchWithParams( query );
        setRefreshing( false );
      };
      return (
        <ScrollView scrollEnabled={!isShowSort && !isShowFilter}
className="bg-black" refreshControl={
          <RefreshControl
            refreshing={refreshing}
            onRefresh={onRefresh}
          />
        }>
          <View className="px-4 mt-2">
            <SearchInput handleSubmit={handleSubmitInput} />
            <View className="px-4" style={{ marginTop: 20 }}>
              <View className="flex-row justify-between mb-4">
                <Pressable onPress={openFilter} className="flex-row items-center"
style={{ gap: 5 }}>
                  <FilterIcon />
                  <View style={{ marginTop: 3 }}>
                    <StyledText className="text-base font-opensmedium leading-
none">Фільтр</StyledText>
                    <StyledText color="text-gray-70p" className="text-sm font-
opensmedium leading-none">не вибрано</StyledText>
                  </View>
                </Pressable>
                <Pressable onPress={openSort} className="flex-row items-center"
style={{ gap: 5 }}>
                  <SortIcon />
                  <View style={{ marginTop: 5 }}>
                    <StyledText className="text-base font-opensmedium leading-
none">Сортування</StyledText>
                    <StyledText style={{ marginLeft: 2 }} color="text-gray-70p"
className="text-sm font-opensmedium leading-none">по рейтингу</StyledText>
                  </View>
                </Pressable>
              </View>
              {isLoading ? <AuctionLoader count={3} /> :
                auctions.map( ( auction ) => (
                  <View key={auction._id} className="mb-6">
                    <Auction {...auction} isBig={true} />
                  </View>
                ) )
              }
            </View>
          </View>
          <Filter query={query} handleSubmit={changeQuery}
isShow={isShowFilter} close={closeFilter} />
          <Sort initialValue={query.sortBy} onChangeSort={changeSort}
isShow={isShowSort} close={closeSort} />
        </ScrollView>
      );};
    export default Auctions;
```

Лістинг Д.4 — Лістинг файлу ImageSlider.tsx

```
import { API_SERVER_URL } from "@/config/api.config";
import React, { useState } from "react";
import { View, Image, Modal, Pressable } from "react-native";

const ImageSlider: React.FC<{ imageUri: string }> = ({ imageUri }) => {
  const [visible, setVisible] = useState(false);

  return (
    <View>
      <Pressable onPress={() => setVisible(true)}>
        <Image
          source={{ uri: API_SERVER_URL + imageUri }}
          className="w-[350px] h-[230px] rounded-xl"
          resizeMode="cover"
        />
      </Pressable>
      <Modal
        visible={visible}
        transparent
        animationType="fade"
        onRequestClose={() => setVisible(false)}
      >
        <Pressable
          className="flex-1 bg-black bg-opacity-80 justify-center items-center"
          onPress={() => setVisible(false)}
        >
          <Image
            source={{ uri: imageUri }}
            className="w-full h-full"
            resizeMode="contain"
          />
        </Pressable>
      </Modal>
    </View>
  );
};

export default ImageSlider;
```

Лістинг Д.5 — Лістинг файлу Review.tsx

```
import Rating from "@/components/ui/Rating"
import StyledText from "@/components/ui/StyledText"
import { API_SERVER_URL } from "@/config/api.config"
import { IUserState } from "@/types/user.types"
import { FC, useState } from "react"
import { Image, Pressable, ScrollView, View } from "react-native"

interface ReviewProps {
  author: IUserState
  comment: string
  rating: number
}

const Review:FC<ReviewProps> = ({author, comment, rating}) => {
  const [expanded, setExpanded] = useState(false);

  return (
    <View className="px-4 py-4 mb-4" style={{backgroundColor: "rgba(21, 22, 22, 0.5)", boxShadow: "4px 4px 8px rgba(204, 204, 204, 0.2)", borderRadius: 8, borderWidth: 1, borderColor: "#191919"}}>
      <View className="flex-row items-center justify-between mb-3">
        <View className="flex-row items-center gap-2">
          <Image source={{uri: API_SERVER_URL + author.avatar}} className="w-8 h-8 rounded-full" />
          <StyledText className="font-openssemibold">{author.userName}</StyledText>
        </View>
        <Rating rating={rating} />
      </View>
      <Pressable onPress={() => setExpanded( !expanded )}>
        <StyledText numberOfLines={expanded ? undefined : 3}>
          {comment}
        </StyledText>
      </Pressable>
    </View>
  )
}
export default Review
```