

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: «Проектування інтерактивного застосунку на базі месенджера для
розумного розподілу витрат у групі друзів»

здобувача *Жовни Антона Олександровича*

(ПІБ)

_____ (Підпис)

Науковий керівник:

д.е.н., професор

_____  Ріппа. С.П.

**Робота допущена до захисту перед
екзаменаційною комісією з атестації
здобувачів вищої освіти**

завідувач кафедри:

к.е.н., доцент

_____ Тішков Б.О.

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:

Керівник проєктної групи(гарант)
освітньо-професійної програми

_____ Помазун О.М.
“ ____ ” _____ 2025 р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____ Тішков Б.О.
“ ____ ” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти _____

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: **«Проектування інтерактивного застосунку на базі месенджера для
розумного розподілу витрат у групі друзів»**

Тему затверджено наказом ректора Університету від « 7 » березня 2025 р. № 466- ст.

**Кваліфікаційна бакалаврська робота виконується на матеріалах
аналізу існуючих рішень, робіт вітчизняних та зарубіжних фахівців, сучасних веб-
технологій та гейміфікації.**

План кваліфікаційної бакалаврської роботи

Розділ I ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Розділ II РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розділ III ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

Об'єкт дослідження: Процеси розподілу спільних фінансових витрат та методи
гейміфікації соціальної взаємодії у неформальних групах.

Предмет дослідження: Методи та засоби розробки інформаційних систем для
гейміфікованого управління спільними фінансами на базі сучасних веб-технологій.

Мета кваліфікаційної бакалаврської роботи: Розробити та апробувати інформаційну систему у форматі інтегрованого в месенджер міні app, призначену для автоматизації розподілу спільних витрат у групі друзів з інтеграцією елементів гейміфікації для підвищення залученості та мотивації користувачів.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I: Провести аналіз предметної області, визначити актуальність проблеми розподілу витрат у неформальних групах та поточні виклики.

Вивчити існуючі інформаційні системи та програмні продукти, призначені для обліку та розподілу спільних витрат, провести їх порівняльний аналіз, виявити переваги та недоліки.

Дослідити принципи та механізми гейміфікації, а також можливості їх застосування для підвищення мотивації та залученості користувачів у фінансових застосунках.

Визначити функціональні та нефункціональні вимоги до майбутньої інформаційної системи.

У розділі II: Розробити інфологічну модель бази даних інформаційної системи, що відображає основні сутності та їхні зв'язки.

Спроекувати даталогічну модель бази даних, обґрунтувати вибір документо-орієнтованої СКБД (MongoDB) та її структуру.

Обґрунтувати вибір архітектури інформаційної системи (клієнт-сервер, хмарна) та її основних компонентів.

Визначити принципи взаємодії між клієнтською частиною та серверною частиною (Backend API).

Розробити загальну структуру комплексу технічних засобів та програмного забезпечення для реалізації системи.

У розділі III: Реалізувати ключові компоненти інформаційної системи «Pay or Play» з використанням обраних технологій (Node.js, React, TypeScript, MongoDB).

Провести пілотне впровадження та апробацію системи на контрольному прикладі, продемонструвавши функціональність розподілу витрат, ігрових механік та системи рейтингів.

Навести результативні вихідні документи та проаналізувати отримані результати впровадження з точки зору їх повноти, достовірності, адекватності та потенційної практичної цінності.

Сформулювати обґрунтовані рекомендації щодо подальших досліджень, удосконалень та перспектив впровадження інформаційної системи.

**Завдання підготував
науковий керівник**

Ріппа Сергій Петрович

« 10 » березня 2025 р.

**Завдання одержав
здобувач**

Жовна Антон Олександрович

« 10 » березня 2025 р.

Відгук
на кваліфікаційну бакалаврську роботу
здобувача *Жовни Антона Олександровича* на тему:
«ПРОЄКТУВАННЯ ІНТЕРАКТИВНОГО ЗАСТОСУНКУ НА БАЗІ
МЕСЕНДЖЕРА ДЛЯ РОЗУМНОГО РОЗПОДІЛУ ВИТРАТ У ГРУПІ ДРУЗІВ»
спеціальність 122 «Комп'ютерні науки»

Актуальність теми дослідження. Тема кваліфікаційної бакалаврської роботи є актуальною, оскільки присвячена дослідженню та практичному застосуванню сучасних технологій автоматизації повсякденних фінансових процесів у неформальних соціальних групах. У контексті активного використання месенджерів та цифрових сервісів для комунікації та організації спільних витрат, запропоноване рішення є своєчасним і затребуваним.

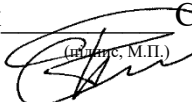
Наукова новизна. Виконане здобувачем дослідження містить елементи наукової новизни, зокрема: автор детально проаналізував сучасні підходи до розподілу витрат у групах, виявив недоліки існуючих рішень і запропонував власну концепцію інтерактивного застосунку, що інтегрується з месенджером. Запропоновано оригінальну архітектуру системи з використанням бот-платформи, бази даних та інтерфейсу для зручної взаємодії користувачів.

Якість проведеного аналізу проблеми, текстового та табличного матеріалу. Зміст представленої до захисту роботи, що містить результати опрацювання теоретичних аспектів розробки ІТ-сервісів для фінансової взаємодії в групах, а також практичні результати проєктування та реалізації архітектури застосунку, є логічно структурованими і послідовними. Основний зміст роботи вдало доповнено таблицями, діаграмами, прикладами сценаріїв використання та фрагментами коду.

Обґрунтованість пропозицій і висновків та їх практична цінність. Пропозиції щодо застосування методів інтеграції з месенджерами та автоматизації розрахунків є достатньо обґрунтованими; висновки, зроблені автором за результатами виконаної роботи, узагальнюють її основний зміст. Результати кваліфікаційної роботи можуть бути використані при створенні реального застосунку для спільного обліку витрат у побуті, подорожах, спільних заходах тощо.

Наявність недоліків. Зауважень до виконаної кваліфікаційної роботи не має.

Загальний висновок і оцінка роботи. Кваліфікаційна бакалаврська робота на тему «ПРОЄКТУВАННЯ ІНТЕРАКТИВНОГО ЗАСТОСУНКУ НА БАЗІ МЕСЕНДЖЕРА ДЛЯ РОЗУМНОГО РОЗПОДІЛУ ВИТРАТ У ГРУПІ ДРУЗІВ» у цілому відповідає встановленим вимогам та рекомендується до захисту, а її автор *Жовна Антон Олександрович* заслуговує на присвоєння освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 «Комп'ютерні науки».

Науковий керівник  С. П. Ріппа, д.е.н., професор,
(д-р екон. наук) професор кафедри інформаційних систем в економіці
Навчально-наукового інституту "Інститут
інформаційних технологій в економіці" КНЕУ

« 3 » червня 2025 р.

РЕЦЕНЗІЯ
на кваліфікаційну бакалаврську роботу
здобувача Жовни Антона Олександровича на тему
«ПРОЄКТУВАННЯ ІНТЕРАКТИВНОГО ЗАСТОСУНКУ НА БАЗІ
МЕСЕНДЖЕРА ДЛЯ РОЗУМНОГО РОЗПОДІЛУ ВИТРАТ У ГРУПІ ДРУЗІВ»
за спеціальністю «Комп'ютерні науки»

Актуальність теми дослідження. Обрана для кваліфікаційної бакалаврської роботи (КБР) тематика є актуальною, оскільки охоплює дослідження можливостей автоматизації повсякденних фінансових взаємодій між користувачами в неформальних групах. У сучасному цифровому середовищі, де популярність месенджерів та мобільних фінансових сервісів постійно зростає, створення інтерактивного застосунку для розумного розподілу витрат є затребуваним і практично значущим рішенням.

Наукова новизна. Представлене в бакалаврській роботі дослідження містить елементи наукової новизни, зокрема: автором проведено аналіз існуючих рішень у сфері спільного управління витратами, визначено їхні обмеження та запропоновано власну концепцію інтеграції функціоналу розподілу витрат у середовище популярного месенджера. Запропоновано архітектуру застосунку з використанням бот-платформ, REST API та баз даних, що дозволяє забезпечити зручність, прозорість і автоматизацію фінансових розрахунків у групах.

Якість проведеного аналізу проблеми, текстового та табличного матеріалу. Зміст кваліфікаційної роботи, аналіз і перелік літератури свідчать про проведення ґрунтовного дослідження особливостей інтеграції IT-рішень у середовище месенджерів. У роботі представлено обґрунтування вибору технологій (наприклад, Telegram Bot API, Node.js, MongoDB), побудовано діаграми взаємодії, інформаційні моделі, приклади сценаріїв використання. Основний зміст роботи доповнено таблицями, схемами та фрагментами коду, що ілюструють реалізацію ключових функцій.

Обґрунтованість пропозицій і висновків та їх практична цінність. Запропоновані в роботі підходи до організації спільного обліку витрат є обґрунтованими, а висновки — логічно випливають із проведеного аналізу та реалізованого прототипу. Результати роботи можуть бути використані для створення повноцінного застосунку, що інтегрується з популярними месенджерами, і є потенціал для подальшого розвитку як стартап-рішення або як частини фінансових сервісів.

Наявність недоліків. У проєкті, представленому на рецензування, є несуттєві недоліки: зокрема, викладення матеріалів щодо інформаційної моделі не завжди є послідовним; також доцільним було б розширити функціональність системи для обліку боргів у різних валютах або з урахуванням часткових платежів. Крім того, не повністю витримано вимоги до оформлення списку використаних джерел.

Практична значущість висновків і рекомендацій. Розроблений застосунок має прикладне значення та може бути використаний у повсякденному житті для спрощення фінансових розрахунків у групах друзів, колег або під час спільних подорожей. Його функціонал може бути адаптований до різних сценаріїв використання, а також масштабований для інтеграції з платіжними сервісами.

Рецензент: _____
(підпис)

Погореловська І.Д., к.е.н., доцент,
доцент кафедри комп'ютерних та інформаційних
технологій і систем факультету фінансів та
цифрових технологій Державного податкового
університету

« 3 » червня 2025 р.

АНОТАЦІЯ

кваліфікаційної бакалаврської роботи
здобувача першого (бакалаврського) рівня вищої освіти, 4 курсу,
виконаної на тему: «Проектування інтерактивного застосунку на базі месенджера
для розумного розподілу витрат у групі друзів»
Київ: кафедра інформаційних систем в економіці, 2025 р.

Кваліфікаційна бакалаврська робота присвячена актуальній проблемі автоматизації розподілу спільних фінансових витрат, яку пропонується вирішувати з використанням сучасних інформаційних технологій та елементів гейміфікації. Робота складається з трьох логічно пов'язаних розділів.

У першому розділі надана характеристика предметної області, проведено аналіз існуючих рішень для обліку витрат та досліджено можливості застосування гейміфікації для підвищення залученості користувачів.

Другий розділ є проектним. У ньому обґрунтовано вибір архітектури системи, розроблено інфологічну та даталогічну моделі бази даних, а також визначено вимоги до технічного та програмного забезпечення.

Третій розділ – конструктивний. Тут представлено реалізацію. Детально описано інформаційне, технічне та програмне забезпечення, використане для створення системи. Наведені результати пілотного впровадження та апробації ключового функціоналу на контрольному прикладі, підтверджуючи працездатність системи.

Висновки містять рекомендації щодо подальших удосконалень та перспектив впровадження системи, підкреслюючи її інноваційність та практичну цінність у порівнянні з існуючими альтернативами.

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 108 сторінок, 13 таблиць, 15 рисунків, список використаних джерел з 30 найменувань, 4 додатки.

«Проектування інтерактивного застосунку на базі месенджера для розумного розподілу витрат у групі друзів»

Об'єкт дослідження: Процеси розподілу спільних фінансових витрат та методи гейміфікації соціальної взаємодії у неформальних групах.

Предмет дослідження: Методи та засоби розробки інформаційних систем для гейміфікованого управління спільними фінансами на базі сучасних веб-технологій.

Мета дослідження: Проектування та розробка інформаційної системи у форматі інтегрованого в месенджер застосунку, призначену для автоматизації розподілу спільних витрат у групі друзів з інтеграцією елементів гейміфікації для підвищення залученості та мотивації користувачів.

Відповідно до поставленої мети визначені такі завдання:

1. Проаналізувати предметну область, виявити актуальність проблеми та вивчити існуючі рішення для розподілу витрат.
2. Дослідити принципи гейміфікації та можливості їх застосування у фінансових застосунках.
3. Визначити функціональні та нефункціональні вимоги до майбутньої інформаційної системи.
4. Розробити інфологічну та даталогічну моделі бази даних.
5. Обґрунтувати вибір архітектури системи, комплексу технічних та програмних засобів.
6. Реалізувати ключові компоненти інформаційної системи «Pay or Play» з використанням Node.js, React, TypeScript та MongoDB.
7. Провести пілотне впровадження та апробацію системи на контрольному прикладі, продемонструвавши її функціональність.
8. Проаналізувати отримані результати впровадження та сформулювати рекомендації щодо подальших удосконалень та перспектив.

Теоретична, методична та практична значущість отриманих результатів. Теоретична значущість полягає в узагальненні підходів до інтеграції фінансового обліку та гейміфікації у розподілених інформаційних системах. Методична значущість полягає у розробці та апробації методології проектування та реалізації міні app. Практична значущість полягає у створенні готового до пілотного використання програмного продукту, який вирішує актуальну проблему розподілу витрат, підвищує залученість користувачів та має значний потенціал для комерціалізації.

Рік виконання кваліфікаційної бакалаврської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: розподіл витрат, гейміфікація, mini app, MongoDB, Node.js, React, API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	4
ВСТУП.....	5
РОЗДІЛ 1	8
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ.....	8
1.1 Характеристика предметної галузі та об'єкта дослідження	8
1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі.....	11
РОЗДІЛ 2	15
РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	15
2.1 Аналіз і специфікація вимог до інформаційної системи.....	15
2.2 Постановка та алгоритм розв'язання задачі	22
2.2.1 Постановка задачі	22
2.2.2 Алгоритм розв'язання задачі.....	26
2.3 Моделювання інформаційної системи	31
2.3.1 Моделювання поведінки системи	31
2.3.2 Моделювання структури системи	34
РОЗДІЛ 3	39
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ	39
3.1 Інформаційне забезпечення.....	39
3.1.1 Загальна характеристика інформаційного забезпечення.....	39
3.1.2 Організація збору і передання первинної інформації.....	41
3.1.3 Побудова системи класифікації та кодування.....	44
3.1.4 Проєктування форм первинних документів та відеокадрів	49
3.1.5 Структура інформаційних масивів.....	51
3.1.6 Вибір СКБД	54
3.1.7 Інфологічна модель бази (сховища) даних	56
3.1.8 Даталогічна модель бази (сховища) даних	58
3.2 Технічне забезпечення	60
3.2.1 Загальні положення та схема автоматизації.....	60
3.2.2 Структура комплексу технічних засобів	61
3.2.3 Опис автоматизованого робочого місця	64
3.2.4 Схема мережі передачі даних	65
3.3 Програмне забезпечення	69

3.3.1 Структура програмного забезпечення	69
3.3.2 Системне програмне забезпечення	69
3.3.3 Прикладне програмне забезпечення.....	72
3.3.4 Програмна документація	74
3.4 Результати реалізації інформаційної системи	75
ВИСНОВКИ	80
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	83
ДОДАТКИ.....	85

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІЗ – інформаційне забезпечення

ІС – інформаційна система

ВСТУП

Сучасні інформаційні технології відкривають нові горизонти для створення інтерактивних рішень, які не лише спрощують повсякденні задачі, а й роблять їх захопливими. Однією з таких задач є розподіл витрат у групах друзів — процес, який часто перетворюється на рутину через брак зручних і мотивуючих інструментів. Розробка інтерактивного застосунку на базі месенджера із використанням ігрових механік пропонує свіжий підхід до цієї проблеми. Основний акцент робиться на розважальному аспекті: замість сухих фінансових розрахунків користувачі отримують ігровий досвід, який стимулює участь і додає веселощів у соціальні взаємодії. Такий підхід особливо актуальний для України, де традиції спільного дозвілля займають важливе місце в культурі.

Актуальність теми. У цифрову епоху питання розподілу витрат у групах друзів залишається актуальним, але традиційні інструменти часто ігнорують емоційну складову цього процесу. Застосунок вирішує це, інтегруючи гейміфікацію в управління витратами, що робить його не просто утилітарним, а й розважальним. В Україні, де неформальні зустрічі та спільний відпочинок є частиною соціального життя, такий інструмент може стати популярним серед молоді, яка цінує інновації та веселощі. На відміну від банківських систем, орієнтованих на точність і формальність, перетворює розрахунки на гру, підвищуючи залученість і соціальну взаємодію.

Аналіз останніх досліджень і публікацій. Сучасний цифровий простір характеризується зростаючою інтеграцією соціальних взаємодій та фінансових операцій. Аналіз останніх досліджень у галузі інтерактивних технологій та гейміфікації чітко вказує на їхній значний потенціал у підвищенні мотивації та залученості користувачів, особливо у сферах, що традиційно асоціюються з рутинними або складними завданнями, такими як управління фінансами.

Наприклад, дослідження в області *фінансових технологій (FinTech)* постійно підкреслюють важливість зручності та доступності для кінцевого користувача. Праці, зосереджені на інтеграції соціальних платформ із фінансовими

інструментами, вказують на те, що користувачі прагнуть максимально спростити та інтегрувати повсякденні операції у звичне для них середовище. Це підтверджується зростанням популярності мобільних платіжних застосунків та інтегрованих функцій розподілу рахунків у банківських програмах, що прагнуть зробити фінансові операції більш інтуїтивними та менш обтяжливими [1].

Паралельно цьому, галузь *гейміфікації* активно досліджує, як ігрові елементи та механіки можуть бути застосовані у неігрових контекстах для підвищення залученості, мотивації та продуктивності. Багато досліджень демонструють, що додавання ігрових елементів, таких як бали, рівні, бейджі, змагання та інтерактивні завдання, значно підвищує інтерес користувачів до виконання завдань, включно з тими, що мають фінансовий підтекст [2]. Це особливо помітно у застосунках для вивчення мов, спорту та освіти, де гейміфікація успішно стимулює регулярне використання та досягнення цілей.[3]

З огляду на платформу, *месенджери* еволюціонували від простих засобів комунікації до потужних екосистем для міні-застосунків. Дослідження показують, що інтеграція функціоналу безпосередньо у месенджери є надзвичайно перспективною для сценаріїв неформальної взаємодії та повсякденних потреб, оскільки це усуває необхідність завантажувати окремі застосунки та забезпечує миттєвий доступ у звичному соціальному середовищі [4]. Telegram Mini Apps, зокрема, демонструють потенціал для створення повноцінних інтерактивних досвідів, не виходячи з месенджера, що відкриває широкі можливості для застосунків, орієнтованих на групову взаємодію.

Незважаючи на ці тенденції, поєднання гейміфікованих механік з процесом розподілу витрат, особливо у розважальному та неформальному контексті, залишається відносно малодослідженим. Більшість існуючих рішень для розподілу витрат зосереджені на суто фінансових аспектах та розрахунках, не інтегруючи елементи гри для підвищення соціальної динаміки та зменшення дискомфорту, пов'язаного з обговоренням боргів. Це робить представлену бакалаврську роботу новаторською, оскільки вона прагне об'єднати функціональність розподілу витрат з інтерактивними ігровими елементами, створюючи унікальний користувацький

досвід для груп друзів.

Мета і завдання дослідження.

Метою роботи є розробка застосунку, який перетворює розподіл витрат у групах друзів на захопливу гру.

Основні завдання:

1. Визначити вимоги до функціональності з акцентом на розважальні елементи.
2. Розробити архітектуру бекенду для стабільної роботи системи.
3. Впровадити ігрові механіки (рейтинги, нагороди) для стимулювання участі.
4. Забезпечити синхронізацію даних у реальному часі для динамічного ігрового процесу.
5. Провести тестування, щоб переконатися в зручності та розважальній цінності.

Об'єктом дослідження є процес розподілу витрат у групах друзів через інтерактивні застосунки з ігровим підходом.

Предмет дослідження є застосунок на базі Telegram із функціями групового управління, ігровими елементами та рейтингом користувачів.

Методи дослідження

У роботі застосовуються: системний аналіз для створення ігрової структури, об'єктно-орієнтоване програмування для реалізації застосунку, тестування для оцінки зручності та аналіз даних для перевірки ефективності гейміфікації.

Теоретична, методична та практична значущість

Теоретична значущість полягає у розробці продукту, який об'єднує ігрові елементи з фінансовими функціями, крім цього демонструє перспективи розробки міні apps в порівнянні зі звичними для всіх рішеннями. Методична цінність — у розробці алгоритмів для інтерактивних систем із гейміфікацією. Практична значущість — у наданні українським користувачам розважального інструменту для спільного дозвілля.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об'єкта дослідження

Предметною галуззю даної бакалаврської роботи є інформаційні технології, зокрема підгалузь, пов'язана з розробкою інтерактивних застосунків для месенджерів. Ця сфера охоплює створення програмного забезпечення, яке інтегрується з платформами спілкування, для вирішення різноманітних завдань користувачів із акцентом на зручність, інтерактивність та розважальність. Основні риси цієї галузі включають:

- Розробка мобільних застосунків: Створення програмного забезпечення, адаптованого до мобільних пристроїв із врахуванням їхньої специфіки та вимог до продуктивності.
- Інтеграція з API месенджерів: Використання інтерфейсів програмування застосунків (API) для вбудовування функціоналу в екосистему месенджерів, що забезпечує легкий доступ для користувачів без додаткових установок.
- Гейміфікація: Застосування ігрових механік для підвищення залученості, мотивації та розважального досвіду користувачів. Це включає рейтинги, нагороди, випадкові події тощо.
- Синхронізація в реальному часі: Використання технологій, таких як WebSocket, для забезпечення миттєвої взаємодії між учасниками застосунку.

Ця галузь характеризується швидким розвитком завдяки зростанню популярності месенджерів як універсальних платформ. Згідно з даними Statista (2023), кількість активних користувачів месенджерів у світі перевищує 3 мільярди, що створює значний потенціал для інтерактивних рішень із розважальним ухилом [5]. Розробка таких застосунків орієнтована на користувача, де ключовими є простота інтерфейсу, доступність і додавання елементів гри для підсилення соціальної взаємодії.

Об'єктом дослідження є застосунок, розроблений як Telegram Mini App. Цей

застосунок не є аналогом банківських систем чи платіжних сервісів, а спрямований на створення веселого та інтерактивного досвіду під час вирішення фінансових питань у компанії друзів. Основні характеристики об'єкта:

- Інтеграція з Telegram: Працює як міні app у месенджері Telegram, що дозволяє запускати його без окремого встановлення, лише через посилання чи меню.
- Управління групами: Організатор (хост) створює лобі, запрошує учасників через посилання, може змінювати назву групи або видаляти її.
- Розподіл витрат: На даному етапі передбачено три режими:
 - Рівномірний розподіл: Сума чека ділиться порівну між усіма учасниками.
 - Ручний розподіл: Організатор визначає, хто скільки має заплатити.
 - Ігровий розподіл: Запускається одна з трьох ігор із ризиком для визначення платника чи групи платників.
- Ігрові механіки:
 - Рулетка: Випадково визначає результат із такими варіантами:
 - "Сьогодні за вас платить [ім'я учасника]".
 - "Ви закриваєте весь стіл, оплачуючи за всіх".
 - "Ви платите половину чека, решту ділять інші".
 - "Сьогодні ваш щасливий день — платять усі, крім вас".
 - "Подвійний ризик: ви платите за себе і ще за одного".
 - "Група дякує: ви платите лише 10% від суми".
 - "Джекпот: ви оплачуєте чек і отримуєте промокод".
 - "Випробування долі: виберіть іншого платника".
 - Щаслива карта: Кожен учасник витягує віртуальну карту (наприклад, від 1 до 10). Той, у кого найменше число, платить за всіх. Для інтерактивності додано анімацію витягування карт і можливість "переграти" один раз за рішенням організатора.
 - Платник у тіні: Гра за мотивами "Мафії". Учасники таємно отримують ролі ("мафія" або "жителі") через застосунок. Після голосування, якщо

перемагає "мафія" — платить сторона "жителів", і навпаки.
Організатор модерує гру.

- Система рейтингів і промокодів: Кожен учасник має рейтинг, який залежить від дій:

- "Сплатив" — рейтинг зростає.
- "Відмовився" або "Взяв у борг" — рейтинг знижується.

На головній сторінці відображається карусель промокодів (наприклад, для McDonald's), де якість пропозицій залежить від рейтингу: від "знижка 10%" до "безкоштовний бургер".

- Історія ігор і боргів: У профілі користувача зберігається інформація про попередні ігри, статуси платежів і борги перед друзями.

Додаток вирізняється серед інших застосунків завдяки поєднанню фінансового функціоналу з розважальними іграми, що робить процес розподілу витрат цікавим і соціально орієнтованим. Дослідження показують, що гейміфікація підвищує залученість користувачів на 30-50% [6], що підтверджує актуальність такого підходу.

Взаємозв'язок між предметною галуззю та об'єктом дослідження

Застосунок є прикладом сучасного рішення в предметній галузі інформаційних технологій, оскільки він:

- Використовує інтеграцію з Telegram через API для зручного доступу.
- Застосовує гейміфікацію як ключовий елемент, що відповідає трендам галузі.
- Вимагає синхронізації дій у реальному часі (наприклад, через WebSocket) для забезпечення інтерактивності під час ігор.

Практичне значення об'єкта дослідження

"Pay or Play" має практичну цінність завдяки:

- Спрощенню розподілу витрат у групах друзів із додаванням розважального елементу.
- Підвищенню соціальної взаємодії через ігровий формат.
- Можливості адаптації до інших месенджерів чи сценаріїв використання.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Перед початком розробки необхідно ретельно проаналізувати літературні джерела та практичний досвід у предметній галузі — інформаційних технологіях із фокусом на інтерактивні застосунки для месенджерів, гейміфікацію та системи розподілу витрат. Цей аналіз допоможе виявити ключові тенденції, технології та підходи, які можна застосувати в проєкті, а також визначити прогалини в існуючих рішеннях, щоб акцентувати актуальність і новизну пропонованого застосунку.

Літературні джерела

Сучасні месенджери, зокрема Telegram, надають унікальні можливості для створення інтерактивних застосунків через Telegram Mini Apps. Дослідження у цій сфері підкреслюють, що вбудовані API, такі як Web Apps API, дозволяють розробникам створювати легкі застосунки, які не потребують встановлення, знижуючи бар'єри для користувачів [7]. Це особливо важливо для "Pay or Play", де доступність і швидкий старт є ключовими для залучення аудиторії, оскільки Telegram Mini Apps дозволяють повноцінно функціонувати застосунку всередині месенджера без окремого завантаження.

Гейміфікація відіграє центральну роль у концепції, перетворюючи розподіл витрат на розважальний процес. Багато досліджень демонструють, що ігрові елементи можуть значно підвищити залученість користувачів, стимулюючи їхню мотивацію через змагання, винагороди та досягнення [2]. Зокрема, мета-аналізи емпіричних досліджень з гейміфікації показують, що її ефективність у підвищенні залученості є значною та може бути виміряна зростанням активності користувачів на 30-50% [3]. Це підтверджує доцільність інтеграції ігор, таких як рулетка, "Щаслива карта" та "Платник у тіні", у фінансовий застосунок, перетворюючи рутинний процес на захопливий досвід.

Для реалізації ігрових механік у реальному часі планується використання технології WebSocket. WebSocket забезпечує постійний двосторонній зв'язок між клієнтом і сервером, що є ідеальним для інтерактивних застосунків, таких як

багатокористувацькі ігри або групові чати [5]. Ця технологія дозволяє миттєво оновлювати дані на пристроях усіх учасників, забезпечуючи плавну синхронізацію дій під час гри чи оновлення статусів оплати, що є критично важливим для інтерактивних ігрових сценаріїв.

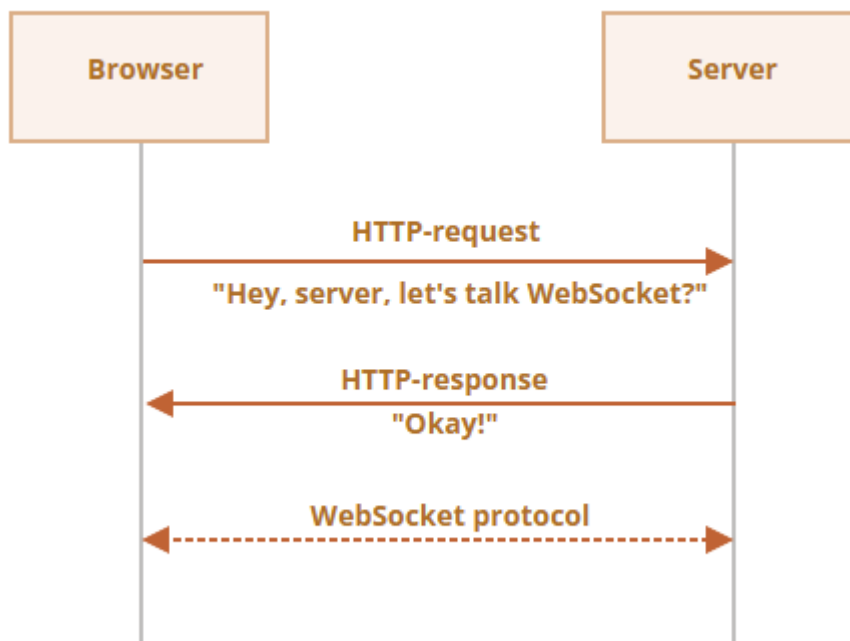


Рисунок 1.1 – Діаграма роботи системи websocket

Джерело: [1]

Автентифікація та безпека

Безпека користувацьких даних та транзакцій є критично важливою для будь-якого застосунку, навіть якщо його основний фокус лежить на розвагах. Використання JSON Web Tokens (JWT) для автентифікації є широко визнаним стандартом у сучасній веб-розробці. JWT забезпечує ефективне управління сесансами та захист даних, дозволяючи надійно перевіряти автентифікацію користувачів без необхідності постійного звернення до бази даних для кожної авторизованої дії (рис.1.2.1). Це ідеально підходить для ІС, забезпечуючи довіру користувачів та захист їхніх даних [6].

Існуючі застосунки для розподілу витрат

Аналіз існуючих застосунків для розподілу витрат, таких як Splitwise,

Tricount або Venmo, показує, що вони є функціональними та зручними інструментами для управління спільними витратами. Наприклад, Splitwise дозволяє ділити витрати пропорційно чи порівну, відстежувати борги та надсилати нагадування [7]. Однак, ці платформи зосереджені виключно на фінансовій функціональності та не включають ігрових елементів, що обмежує їхню привабливість для аудиторії, яка шукає розваги поряд із практичністю. Це підкреслює їхню орієнтованість на суто раціональний підхід до фінансового менеджменту.

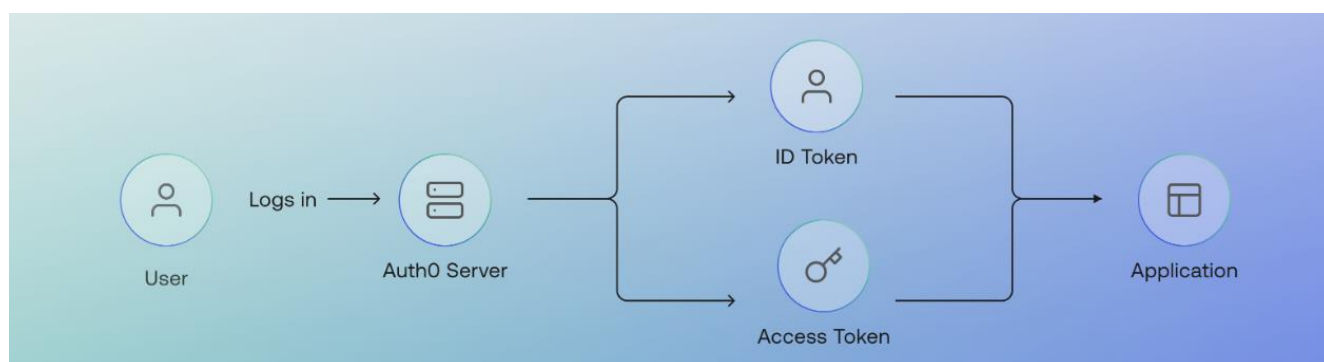


Рисунок 1.2. 1 – Діаграма роботи JWT

Джерело: [1]

Застосунки з гейміфікацією

Практичний досвід показує значний успіх гейміфікації в застосунках, які перетворюють рутинні або складні завдання на захопливий досвід. Приклади, такі як Duolingo, що використовує ігрові механіки для мотивації до навчання мов через очки, рівні та змагання, або Habitica, яка перетворює щоденні завдання на рольову гру з винагородами та штрафами, доводять ефективність цього підходу [8]. Ці приклади підтверджують, що ігрові елементи можуть зробити взаємодію з застосунком цікавішою та мотивуючою, що є наріжним каменем концепції ІС.

Деякі соціальні мережі та месенджери вже інтегрують ігри для підвищення взаємодії та залученості користувачів. Наприклад, у Facebook Messenger свого часу були вбудовані ігри, що дозволяли користувачам змагатися з друзями безпосередньо в чаті, а Telegram також має власну платформу для ігор через ботів [10]. Це підтверджує потенціал ігор у месенджерах та свідчить про те, що користувачі готові взаємодіяти з ігровим контентом у своєму звичному середовищі

спілкування, що "Pay or Play" розширює, додаючи до цього фінансовий контекст та елементи ризику. (таблиця 1.1)

Таблиця 1.1 – Аналіз методів оптимізації

<i>Критерій порівняння</i>	<i>Існуючі застосунки для розподілу витрат (наприклад, Splitwise, Tricount, Venmo)</i>	<i>"Pay or Play" (пропонований застосунок)</i>
Основна функція	Суто фінансовий розподіл спільних витрат, облік боргів та їх відстеження.	Розважальний розподіл витрат з елементами ризику, випадковості та соціальної взаємодії.
Цільова аудиторія	Люди, що потребують зручного та точного обліку спільних фінансів; орієнтація на прагматичність.	Групи друзів, які шукають цікавий та веселий спосіб ділити рахунки, готові до експериментів та розваг.
Гейміфікація	Відсутня. Фокус на раціональності, точності розрахунків та фінансовій дисципліні.	Високий рівень гейміфікації, інтегровані ігри ("Рулетка", "Щаслива карта", "Платник у тіні") з фінансовими наслідками.
Соціальна взаємодія	Обмін фінансовими даними, нагадування про борги. Зазвичай менш емоційна, більш транзакційна.	Високий рівень соціальної взаємодії через спільне проходження ігор, прийняття рішень та обговорення результатів.
Фінансовий контекст	Прямий та основний. Управління реальними грошовими операціями, чітке визначення "хто кому винен".	Прямий фінансовий контекст, але опосередкований ігровими механіками. Мета — зробити процес ділення легшим та цікавішим.
Платформа	Переважно окремі мобільні застосунки та веб-версії.	Інтегрований у Telegram Mini App (мінімізує бар'єри входу, не потребує окремого встановлення).
Інтерактивність у реальному часі	Низька/Середня (синхронізація даних, чати для обговорень).	Висока, за допомогою WebSocket для миттєвої синхронізації ігрових дій, результатів та статусів оплати.
Ключова відмінність / Новизна	Сухий, функціональний підхід до фінансового менеджменту.	Поєднання гейміфікованих ігор із ризиком для розподілу реальних витрат у розважальній формі, акцент на емоціях та веселоощах.

Джерело: сформовано автором на основі виконаного дослідження

РОЗДІЛ 2

РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної системи

Аналіз і специфікація вимог є критично важливим етапом у розробці будь-якої інформаційної системи, особливо для інтерактивного застосунку на базі месенджера, такого як "Pay or Play". Цей етап дозволяє чітко визначити цілі системи, її функціональність та характеристики, які забезпечать її успішну роботу. Для "Pay or Play" ключовим завданням є не лише забезпечення коректного розподілу витрат у групі друзів, але й створення захопливого соціального досвіду через інтеграцію ігрових механік. Правильно проведений аналіз вимог формує міцну основу для проєктування та реалізації системи, дозволяючи врахувати очікування користувачів і технічні особливості платформи Telegram, на якій базується застосунок.

Предметна область застосунку охоплює розробку інтерактивних рішень для месенджерів із фокусом на соціальну взаємодію та розваги. Цей застосунок, реалізований як Telegram Mini App, дозволяє користувачам створювати групи, запрошувати друзів і розподіляти витрати у групі з використанням ігрових елементів. Користувачі можуть обирати між традиційними методами розподілу витрат, такими як рівномірний чи ручний, або ж запускати одну з розважальних ігор, яка випадковим чином визначає, хто і скільки платить. Такий підхід перетворює буденний процес розрахунків на веселий і захопливий досвід, що є основною відмінністю "Pay or Play" від аналогів, орієнтованих суто на фінансові операції.

Для успішної реалізації "Pay or Play" необхідно детально розглянути як функціональні, так і нефункціональні вимоги до системи. Функціональні вимоги визначають ключові можливості застосунку, які забезпечують його основну логіку та ігровий характер, тоді як нефункціональні вимоги гарантують зручність, безпеку та продуктивність системи для користувачів.

Функціональні вимоги до застосунку визначають конкретні функції та дії, які система повинна виконувати для задоволення потреб користувачів. Вони охоплюють процеси управління групами, розподілу витрат через ігрові механіки, управління платежами та системою рейтингу.

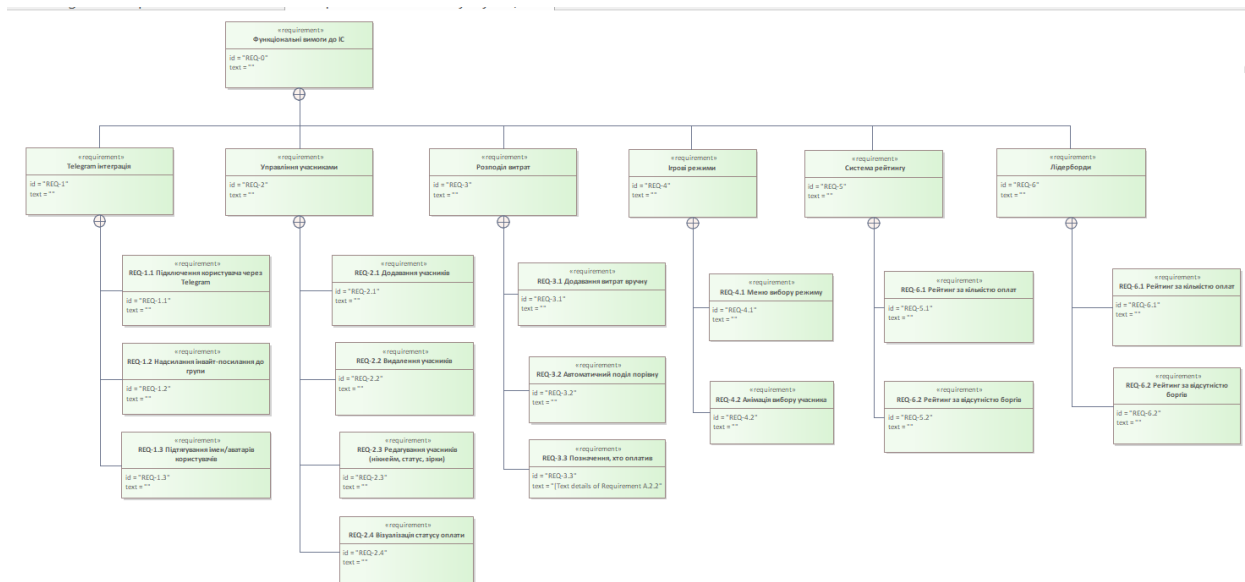


Рисунок 2.1 Діаграма функціональних вимог

Джерело: сформовано автором на основі виконаного дослідження

Управління групами:

Застосунок повинен дозволяти організатору створювати нові групи (лобі) для спільного розподілу витрат. При створенні групи, за замовчуванням її назва встановлюється як "[Ім'я організатора] Group", з можливістю подальшого редагування.

Організатор повинен мати можливість генерувати унікальне посилання-запрошення. Учасники приєднуються до лобі, активуючи це посилання через Telegram, що спрощує процес залучення друзів.

Організатор лобі повинен мати функціонал для зміни назви групи та для її повного видалення.

Після створення лобі та додавання учасників, на головній сторінці застосунку повинна відображатися назва активної групи замість кнопки "Створити групу", забезпечуючи швидку орієнтацію користувачів у поточному стані застосунку.

Управління учасниками:

Застосунок повинен дозволяти додавати користувачів до лобі після того, як вони перейшли за унікальним посиланням-запрошенням та підтвердили приєднання.

Організатор лобі повинен мати функціонал для видалення будь-якого учасника зі своєї групи.

В інтерфейсі лобі повинна відображатися інформація про всіх учасників, включаючи їхні імена, поточний статус (наприклад, "очікує оплати", "сплатив", "борг") та суми, які вони мають сплатити або які їм винні.

Взаємодія зі статусами оплати: Біля кожного учасника повинні бути інтерактивні кнопки ("сплатив", "відмовився", "взяв у борг у"), що дозволяють організатору оновлювати статус платежу.

Розподіл витрат:

Організатор повинен мати можливість ввести загальну суму чека для розподілу між учасниками лобі.

Система повинна автоматично розраховувати частку кожного учасника шляхом рівномірного розподілу загальної суми чека між усіма активними учасниками лобі.

Організатор повинен мати можливість вручну вказати, яку суму має сплатити кожен учасник, відхиляючись від рівномірного розподілу.

Ігровий режим розподілу витрат: Застосунок повинен пропонувати інтерактивний ігровий режим розподілу витрат, що включає три різні міні-ігри. Запуск гри відбувається за рішенням організатора, після чого учасникам на їхніх пристроях запускається обрана гра.

Рулетка:

Логіка: Віртуальна рулетка, що обертається, із секторами, кожен з яких представляє унікальний фінансовий "приз" або "наслідок" для гравця, який її запускає.

Варіації "призів":

1. "Сьогодні за вас платить [ім'я іншого учасника]": Борг поточного гравця покривається випадковим іншим учасником.
2. "Ви закриваєте весь стіл": Поточний гравець сплачує весь рахунок за всіх учасників.
3. "Ви платите половину чека, решту ділять інші": Частка гравця зменшується вдвічі, залишок розподіляється між іншими.
4. "Сьогодні ваш щасливий день — платять усі, крім вас": Поточний гравець повністю звільняється від оплати.
5. "Подвійний ризик: ви платите за себе і ще за одного": Частка гравця подвоюється, і він бере на себе додатковий борг одного з учасників.
6. "Група дякує: ви платите лише 10% від суми": Гравець сплачує лише невелику частину свого боргу.
7. "Джекпот: ви оплачуєте чек і отримуєте промокод": Гравець сплачує чек, але отримує цінний купон.
8. "Випробування долі: виберіть іншого платника": Гравець може вказати іншого учасника, який візьме на себе його борг.

Щаслива карта:

Логіка: Кожен учасник по черзі витягує віртуальну карту з числом від 1 до 10.

Правило: Учасник, у кого випаде найменше число, сплачує весь чек за всіх.

Інтерактивність: Гра супроводжується інтерактивною анімацією витягування карти. За рішенням організатора, може бути надана можливість переграти раунд (наприклад, якщо хтось не згоден з результатом).

Платник у тіні

Логіка: Гра за мотивами "Мафії", де учасники отримують приховані ролі, що визначають їхню фінансову відповідальність.

Ролі: Призначаються ролі "Боржників" (аналог мафії) та "Чесних платників" (аналог мирних жителів).

Процес: Проходить декілька раундів, що включають обговорення, голосування та дії "Боржників" у "нічній фазі".

Результат: Залежно від того, яка сторона (Боржники чи Чесні платники) виграє гру, визначається, хто несе фінансову відповідальність за спільний рахунок. Перемога "Боржників" може призвести до того, що "Чесні платники" або всі учасники платять більше, тоді як перемога "Чесних платників" означає, що "Боржники" несуть основний фінансовий тягар.

Система рейтингу та купонів:

Рейтинг кожного учасника повинен динамічно змінюватися залежно від його платіжної дисципліни. Своєчасні оплати підвищують рейтинг.

Відмова від оплати або накопичення боргу (статус "відмовився", "взяв у борг") призводить до зниження рейтингу учасника.

Історія ігор та боргів: Кожен користувач повинен мати доступ до своєї персональної історії всіх ігор, в яких він брав участь, та детального обліку його боргів.

Відображення промокодів (купонів): На головній сторінці застосунку повинна бути реалізована карусель промокодів та купонів (наприклад, для McDonald's). Якість та привабливість пропозицій у каруселі повинні прямо залежати від поточного рейтингу користувача – чим вищий рейтинг, тим цікавіші та цінніші купони доступні.

Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики програмної системи, її обмеження та критерії успішності, що доповнюють функціональні можливості. Вони є ключовими для забезпечення належного досвіду користувача та стійкості застосунку, були визначені наступні критично важливі нефункціональні вимоги:

Продуктивність:

Повинен забезпечувати швидкий відгук на дії користувачів, особливо при ініціалізації міні-застосунку з Telegram. Час завантаження та початкового відображення інтерфейсу має бути мінімальним (наприклад, не більше 1-2 секунд) для створення позитивного першого враження та уникнення фрустрації користувачів [1].

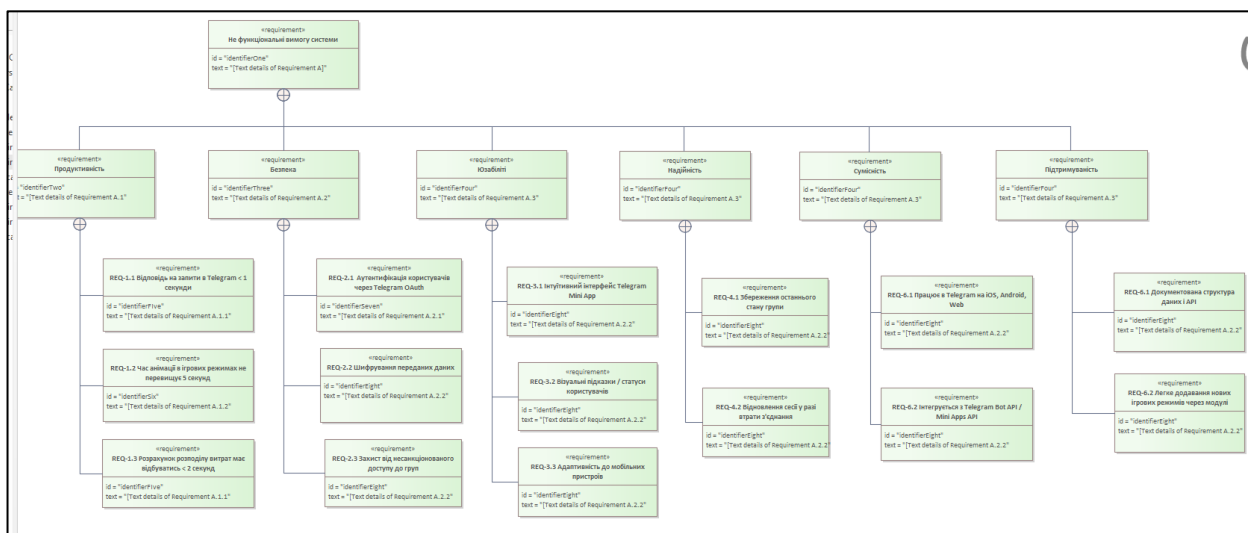


Рисунок 2.2 Діаграма не функціональних вимог

Джерело: сформовано автором на основі виконаного дослідження

Для забезпечення плавного ігрового процесу та інтерактивності в реальному часі, система повинна миттєво обробляти ігрові події (наприклад, результат рулетки, вибір карти, голосування в "Платнику в тіні") та оновлення статусів. Затримка між діями гравців та відображенням їхніх наслідків для всіх учасників лобі не повинна перевищувати 200-300 мс [2].

Застосунок має бути здатним ефективно обробляти одночасні запити та синхронізувати дії для множини активних лобі та їхніх учасників, забезпечуючи стабільну роботу навіть при пікових навантаженнях. Це передбачає оптимізоване використання ресурсів бекенду.

Безпека:

Користувачі повинні автентифікуватися за допомогою даних, що надаються Telegram Web Apps. Бекенд повинен надійно верифікувати ці дані, щоб запобігти несанкціонованому доступу до облікових записів та ігрових лобі [3].

Уся комунікація між клієнтом (Telegram Mini App) і бекендом (REST API, WebSocket) повинна відбуватися через захищені канали (HTTPS, WSS), щоб унеможливити перехоплення та спотворення чутливих даних, таких як ігрові результати, суми витрат або персональна інформація [4].

Система повинна чітко розмежовувати доступ до інформації та функціоналу лобі. Лише авторизовані учасники групи повинні мати доступ до її даних, а організатор лобі має мати ексклюзивні права на управління складом групи та статусами платежів.

Зручність використання:

Інтерфейс застосунку має бути інтуїтивно зрозумілим, чистим та привабливим, розробленим з урахуванням гайдлайнів Telegram Mini Apps, щоб забезпечити легку навігацію та взаємодію для користувачів будь-якого рівня підготовки.

Система повинна надавати чіткий і своєчасний зворотний зв'язок на дії користувача (наприклад, успішне створення лобі, оновлення статусу оплати, результати гри) та відображати поточний статус системи (наприклад, завантаження даних, очікування інших гравців) [5].

Оскільки Telegram Mini App в основному використовується на мобільних пристроях, інтерфейс повинен бути повністю адаптивним, забезпечуючи коректне відображення та зручну взаємодію на екранах різного розміру та орієнтації.

Надійність:

Система повинна забезпечувати стійке зберігання всіх критично важливих даних (інформація про користувачів, лобі, історія витрат, результати ігор), щоб уникнути втрати даних у разі збоїв або відключень. Використання MongoDB з належним резервним копіюванням та реплікацією сприятиме цьому [6].

У разі розриву з'єднання або тимчасового виходу гравця з лобі, система повинна дозволяти йому безперешкодно відновити сесію та повернутися до поточного стану гри.

Сумісність:

Застосунок повинен коректно функціонувати у всіх основних платформах iOS, Android та веб-версії, забезпечуючи однакові функціональні можливості та користувацький досвід незалежно від використовуваного пристрою [1].

Застосунок повинен повністю відповідати специфікаціям Telegram Bot API та Telegram Web App API для гарантування стабільної та коректної взаємодії з месенджером [1].

Підтримка:

Код бекенду та фронтенду повинен бути добре структурований, модульний та документований, що полегшить його подальшу підтримку, виправлення помилок та розвиток нових функцій.

Архітектура застосунку має бути спроектована таким чином, щоб дозволяти відносно легке додавання нових ігрових механік та купонів без суттєвих змін у базову структуру системи.

Реалізація вимог передбачає використання MongoDB для зберігання даних про групи, учасників, статуси та історію. Створення груп відбувається через генерацію унікального посилання, яке учасники активують для приєднання. Розподіл витрат у рівномірному режимі розраховується автоматично, у ручному — через введення організатором сум, а в ігровому — через запуск відповідної гри на бекенді з використанням логіки випадкового вибору. Ігри, такі як рулетка чи щаслива карта, реалізуються з анімаціями на фронтенді та обробкою результатів через WebSocket. Управління платежами та рейтингами зберігається в базі даних, а промокоди відображаються залежно від рейтингу користувача.

2.2 Постановка та алгоритм розв’язання задачі

2.2.1 Постановка задачі

Основними завданнями системи є реалізація функціоналу створення та управління групами, включаючи створення групи, запрошення учасників через унікальне посилання, редагування назви групи та видалення групи. Також необхідно розробити три режими розподілу витрат: рівномірний, ручний та ігровий. У ігровому режимі передбачено три типи ігор: рулетка з різними

варіантами результатів, щаслива карта з інтерактивною логікою та платник у тіні, натхненний грою "Мафія". Крім того, система має забезпечувати управління платежами, де організатор може встановлювати статуси "сплатив", "відмовився" або "взяв у борг", що впливає на рейтинг користувача. Важливою частиною є інтеграція каруселі промокодів на головній сторінці, де якість пропозицій залежить від рейтингу користувача, а також розробка сторінки рейтингу, що відображає список учасників за кількістю оплат та боргів.

Система має працювати в рамках Telegram Mini App, не інтегруючись з платіжними системами, а покладаючись на ручне відстеження платежів організатором. Користувачі повинні мати активні акаунти в Telegram, а учасники групи — довіряти один одному щодо чесності в платежах.

Вхідними даними для системи є інформація про користувачів з Telegram, дані про групи та учасників, сума чека для розподілу та результати ігор. Вихідними даними є розподілені суми для кожного учасника, статуси платежів, рейтинги користувачів та промокоди.

Вихідні повідомлення системи інформують користувачів про стан їхніх фінансових зобов'язань, результати ігор, а також для надання додаткових переваг у вигляді промокодів. Всі вихідні повідомлення інтегровані безпосередньо в інтерфейс Telegram Mini App, що забезпечує зручність та миттєвий доступ для користувачів. (Таблиця 2.1)

Таблиця 2.1 — Опис вихідних повідомлень ІС

<i>з/п</i>	<i>Назва вихідного повідомлення</i>	<i>Ідентифікатор</i>	<i>Форма подання і вимоги до неї</i>	<i>Періодичність видання</i>	<i>Термін видання і допустимий час затримки</i>	<i>Користувач і інформації</i>
	Стан лобі та розподіл витрат	MSG-LOBBY-STATUS	Електронна форма в інтерфейсі Telegram Mini App. Вимоги: чітке відображення назви групи, списку учасників, суми чека, індивідуальних	Після створення лобі, додавання витрат, зміни статусу платежу,	Миттєво, затримка до 500 мс.	Усі учасники лобі

			боргів/виплат, статусу платежів кожного учасника. Адаптивний дизайн.	завершення гри.		
Результати ігор	MSG-GAME-RESULT	Електронна форма в інтерфейсі Telegram Mini App. Вимоги: візуально привабливе відображення результату гри (анімація рулетки, витягування карти, оголошення ролей/переможців у "Платнику в тіні"), чітке вказання платника/розподілу.	Після завершення кожного раунду гри.	Миттєво, затримка до 300 мс.	Усі учасники лобі	
Історія ігор та боргів	MSG-HISTORY	Електронна форма в інтерфейсі Telegram Mini App. Вимоги: хронологічний список усіх ігор (дата, тип гри, результат) та фінансових операцій (дата, сума, статус боргу, контрагент). Можливість фільтрації.	На вимогу користувача.	До 1 секунди.	Кожен користувач особисто	
Рейтинг користувача	MSG-RATING	Електронна форма в інтерфейсі Telegram Mini App. Вимоги: відображення поточного рейтингу користувача, його місця у списку лідерів (якщо реалізовано), а також факторів, що вплинули на рейтинг (кількість оплат, боргів).	Миттєво після зміни статусу платежу або на вимогу користувача.	До 500 мс.	Кожен користувач особисто	
Доступні промокоди	MSG-PROMO	Електронна форма (карусель/список) на головній сторінці інтерфейсу Telegram Mini App. Вимоги: візуально привабливе відображення карток промокодів з назвою, описом та можливістю активації/копіювання. Залежність якості пропозицій від рейтингу.	Постійно на головній сторінці, оновлення при зміні рейтингу.	Миттєво (оновлення), завантаження сторінки до 1 секунди.	Кожен користувач особисто	

Джерело: сформовано автором на основі виконаного дослідження

Вхідна інформація для системи "Pay or Play" – це дані, необхідні для коректної роботи застосунку, забезпечення його функціональності та підтримки взаємодії між користувачами. Вона надходить як безпосередньо від користувачів через інтерфейс Telegram Mini App, так і опосередковано від платформи Telegram. Призначення вхідної інформації полягає у забезпеченні створення та управління групами, фіксації та розподілу витрат, реалізації інтерактивних ігрових режимів, відстеження статусів платежів та підтримки системи рейтингу та промокодів.

Способи одержання вхідної інформації:

- *Введення користувачем:* Основна частина даних (назви груп, суми чеків, ігрові рішення, статуси платежів) вводиться користувачами через спеціалізовані форми та елементи управління в інтерфейсі застосунку.
- *Отримання від Telegram API:* Дані про користувачів (Telegram ID, ім'я) автоматично надходять від Telegram при авторизації користувача в міні-застосунку.
- *Внутрішні розрахунки та оновлення:* Результати ігор, оновлення рейтингів та статуси боргів генеруються системою на основі введених користувачем даних та внутрішньої логіки. Хоча ці дані є "вихідними" для певних процесів, вони часто стають "вхідними" для подальших обчислень або оновлень стану бази даних. (Таблиця 2.2)

Таблиця 2.3 — Опис вихідних повідомлень

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Дані авторизації користувача	INPUT-USER-AUTH	Автоматичне отримання від Telegram API.	При першому запуску застосунку та при кожній сесії.	Telegram API
2	Створення/редагування групи	INPUT-GROUP-CREATE-EDIT	Електронна форма в інтерфейсі Telegram Mini App.	За потреби користувача.	Користувач - організатор

3	Запрошення учасника до групи	INPUT-INVITE-MEMBER	Унікальне посилання-запрошення.	За потреби користувача-організатора.	Користувач - організатор / Користувач -учасник
4	Додавання витрати (чека)	INPUT-ADD-EXPENSE	Електронна форма в інтерфейсі Telegram Mini App.	За потреби користувача-організатора.	Користувач - організатор
5	Вибір режиму розподілу витрат	INPUT-DISTRIBUTION-MODE	Елемент вибору (перемикач, список) в інтерфейсі.	При додаванні/редагуванні витрати.	Користувач - організатор
6	Параметри ручного розподілу	INPUT-MANUAL-DISTRIBUTION	Електронна форма в інтерфейсі Telegram Mini App.	При виборі ручного режиму розподілу.	Користувач - організатор
7	Дії в ігрових режимах	INPUT-GAME-ACTION	Інтерактивні елементи в інтерфейсі гри.	Під час активної гри.	Користувач -учасник
8	Встановлення статусу платежу	INPUT-PAYMENT-STATUS	Елемент вибору (кнопки, випадаючий список) в інтерфейсі лобі.	За потреби користувача-організатора.	Користувач - організатор
9	Запит історії	INPUT-REQUEST-HISTORY	Кнопка "Історія" в інтерфейсі.	За потреби користувача.	Користувач
10	Перегляд промокодів	INPUT-VIEW-PROMO	Навігація на головну сторінку/карусель промокодів.	За потреби користувача.	Користувач

Джерело: сформовано автором на основі виконаного дослідження

2.2.2 Алгоритм розв’язання задачі

Алгоритм розв’язання задачі охоплює створення групового лобі, управління витратами через автоматичний або ручний розподіл, організацію інтерактивних ігор для визначення платіжних обов’язків, а також обробку рейтингів і промокодів. Система базується на клієнт-серверній архітектурі, де серверна частина реалізована з використанням TypeScript, MongoDB для зберігання даних, Express для обробки

HTTP-запитів і WebSocket для синхронізації дій у реальному часі. Безпека забезпечується через JWT-автентифікацію, а ігровий функціонал робить процес розподілу витрат веселим і соціально привабливим. Основна ідея полягає в тому, щоб уникнути прямого інтегрування з платіжними системами, покладаючи управління статусами платежів на організатора.

Використовувана інформація – це сукупність даних, що необхідні для коректного функціонування всіх підсистем застосунку. Вона включає як дані, що надходять від користувачів та платформи Telegram, так і інформацію, яка зберігається та обробляється всередині системи. Призначення цієї інформації полягає у забезпеченні логіки створення та управління групами, розрахунків витрат, проведення ігор, ведення обліку платежів та рейтингів, а також формування списків доступних промокодів.

Таблиця 2.4 — Перелік масивів використовуваної інформації

№ з/п	Масив	Ідентифікатор	Максимальна кількість записів
1	Користувачі	USERS_DB	Необмежено (залежить від Telegram API)
2	Групи (Лобі)	LOBBIES_DB	Необмежено
3	Учасники груп	PARTICIPANTS_DB	Необмежено
4	Витрати (Чеки)	EXPENSES_DB	Необмежено
5	Борги	DEBTS_DB	Необмежено
6	Ігри	GAMES_DB	Необмежено
7	Учасники ігор	GAME PARTICIPANTS_DB	Необмежено
8	Промокоди	COUPONS_DB	Залежить від кількості пропозицій
9	Призначені промокоди	USER_COUPONS_DB	Необмежено
10	Журнал подій	EVENT LOGS_DB	Необмежено

Джерело: сформовано автором на основі виконаного дослідження

Результати розв'язання задачі – це інформація, яка формується в процесі роботи алгоритму та надається користувачам через вихідні повідомлення, а також зберігається для подальшого використання системою. Призначення цих результатів – забезпечити прозорість фінансових взаєморозрахунків, інформувати про хід та підсумки ігор, відображати статус користувача в системі (рейтинг) та надавати йому додаткові переваги (промокоди).

Таблиця 2.5 — Перелік масивів результатної інформації

<i>№ з/п</i>	<i>Масив</i>	<i>Ідентифікатор</i>	<i>Максимальна кількість записів</i>
1	Розподілені суми	DISTRIBUTED_AMOUNTS_RES	Необмежено
2	Оновлені статуси платежів	UPDATED_PAYMENT_STATUSES_RES	Необмежено
3	Результати ігор	GAME_RESULTS_RES	Необмежено
4	Оновлені рейтинги	UPDATED_RATINGS_RES	Необмежено
5	Доступні промокоди	AVAILABLE_PROMOCODES_RES	Необмежено

Джерело: сформовано автором на основі виконаного дослідження

2.2.4 Алгоритм розв’язання задачі

Алгоритм розв’язання задачі охоплює створення групового лобі, управління витратами через автоматичний або ручний розподіл, організацію інтерактивних ігор для визначення платіжних обов’язків, а також обробку рейтингів і промокодів. Система базується на клієнт-серверній архітектурі, де серверна частина реалізована з використанням TypeScript, MongoDB для зберігання даних, Express для обробки HTTP-запитів і WebSocket для синхронізації дій у реальному часі. Безпека забезпечується через JWT-автентифікацію, а ігровий функціонал робить процес розподілу витрат веселим і соціально привабливим. Основна ідея полягає в тому, щоб уникнути прямого інтегрування з платіжними системами, покладаючи управління статусами платежів на організатора.

Використовувана інформація – це сукупність даних, що необхідні для коректного функціонування всіх підсистем застосунку "Pay or Play". Вона включає як дані, що надходять від користувачів та платформи Telegram, так і інформацію, яка зберігається та обробляється всередині системи. Призначення цієї інформації полягає у забезпеченні логіки створення та управління групами, розрахунків витрат, проведення ігор, ведення обліку платежів та рейтингів, а також формування списків доступних промокодів.

Таблиця 2.6 — Перелік масивів використовуваної інформації

№ з/п	Масив	Ідентифікатор	Максимальна кількість записів
1	Користувачі	USERS_DB	Необмежено (залежить від Telegram API)
2	Групи (Лобі)	LOBBIES_DB	Необмежено
3	Учасники груп	PARTICIPANTS_DB	Необмежено
4	Витрати (Чеки)	EXPENSES_DB	Необмежено
5	Борги	DEBTS_DB	Необмежено
6	Ігри	GAMES_DB	Необмежено
7	Учасники ігор	GAME PARTICIPANTS_DB	Необмежено
8	Промокоди	COUPONS_DB	Залежить від кількості пропозицій
9	Призначені промокоди	USER_COUPONS_DB	Необмежено
10	Журнал подій	EVENT LOGS_DB	Необмежено

Джерело: сформовано автором на основі виконаного дослідження

Результати розв'язання задачі – це інформація, яка формується в процесі роботи алгоритму та надається користувачам через вихідні повідомлення, а також зберігається для подальшого використання системою. Призначення цих результатів – забезпечити прозорість фінансових взаєморозрахунків, інформувати про хід та підсумки ігор, відображати статус користувача в системі (рейтинг) та надавати йому додаткові переваги (промокоди).

Таблиця 2.7 — Перелік масивів результатної інформації

№ з/п	Масив	Ідентифікатор	Максимальна кількість записів
1	Розподілені суми	DISTRIBUTED AMOUNTS_RES	Необмежено
2	Оновлені статуси платежів	UPDATED_PAYMENT_STATUSES_RES	Необмежено
3	Результати ігор	GAME_RESULTS_RES	Необмежено
4	Оновлені рейтинги	UPDATED RATINGS_RES	Необмежено
5	Доступні промокоди	AVAILABLE_PROMOCODES_RES	Необмежено

Джерело: сформовано автором на основі виконаного дослідження

Математичний опис переважно стосується алгоритмів розподілу витрат та обчислення рейтингу користувачів.

Якщо обрано рівномірний розподіл, загальна сума чека ділиться порівну між усіма активними учасниками групи .

Формула(1.2):

$$Di = \frac{S_{total}}{N_{active}} \quad (1.1)$$

де:

- Di – частка боргу для кожного i -го учасника.
- S_{total} – загальна сума чека.
- N_{active} – кількість активних учасників у групі.

Обчислення рейтингу користувача

Рейтинг користувача R є динамічним показником, який залежить від його платіжної дисципліни та дій у системі. Рейтинг додає бали за успішні оплати та віднімає за борги або відмови.

Формула(1.2):

$$R = (N_{paid} * K_{paid}) - (N_{refused} * K_{refused}) - (N_{debt} * K_{debt}) \quad (1.2)$$

де:

- R – поточний рейтинг користувача (CURRENT_RATING_SCORE).
- N_{paid} – кількість успішних оплат (кількість статусів "сплатив" для користувача).
- K_{paid} – коефіцієнт впливу успішної оплати на рейтинг (наприклад, 10 балів).
- $N_{refused}$ – кількість відмов від оплати (кількість статусів "відмовився" для користувача).
- $K_{refused}$ – коефіцієнт впливу відмови на рейтинг (наприклад, 20 балів штрафу).
- N_{debt} – кількість разів, коли користувач "взяв у борг" (кількість статусів "взяв у борг" для користувача).
- K_{debt} – коефіцієнт впливу "боргу" на рейтинг (наприклад, 5 балів штрафу).

Випадковий вибір у грі "Рулетка"

Результат гри "Рулетка" визначається за допомогою генератора

псевдовипадкових чисел.

Формула(1.3):

$$\text{Result} = \text{Random}(1, M) \quad (1.3)$$

де:

- Result – обраний варіант результату рулетки.
- Random(min,max) – функція генерації псевдовипадкового цілого числа у діапазоні від min до max включно.
- M – загальна кількість можливих варіантів результатів рулетки.

Прийняті допущення та оцінки відповідності:

- Система базується на довірі між учасниками щодо фактичних платежів, оскільки пряма інтеграція з платіжними системами відсутня.
- Точність фінансових розрахунків забезпечується використанням чисел з плаваючою комою та округленням до двох знаків після коми для грошових сум.
- Надійність обчислення рейтингів залежить від коректного введення статусів платежів організатором.
- Випадковість у ігрових режимах реалізується через псевдовипадкові алгоритми, що є стандартною практикою для розважальних застосунків.

2.3 Моделювання інформаційної системи

2.3.1 Моделювання поведінки системи

Поведінка системи моделюється через послідовність дій користувачів, обробку запитів сервером і синхронізацію даних між учасниками.

Поведінка системи починається з автентифікації користувача через Telegram, що автоматично генерує JWT-токен для подальшої взаємодії з сервером [1]. Користувач, який виступає організатором, створює групу (лобі) через головну

сторінку застосунку. При створенні групи система генерує унікальний ідентифікатор і записує в базу даних дані про групу, включаючи назву (за замовчуванням — "Група [Ім'я організатора]"), організатора та список учасників. Організатор отримує посилання для запрошення, яке інші користувачі активують, щоб приєднатися до лобі. Приєднання нового учасника викликає подію WebSocket, яка сповіщає всіх членів групи про оновлення, забезпечуючи синхронне відображення списку учасників на всіх пристроях. Організатор може змінювати назву групи або видалити її, що також синхронізується через WebSocket. У разі видалення групи сервер надсилає сповіщення учасникам, і запис про групу видаляється з бази даних.

Після створення групи система дозволяє організатору вибрати один із трьох режимів розподілу витрат: автоматичний (порівну), ручний або ігровий. У режимі автоматичного розподілу організатор вводить загальну суму чека, яку сервер ділить порівну між учасниками, зберігаючи результат. WebSocket передає оновлені дані про частки кожному учаснику, відображаючи їх у клієнтському інтерфейсі в реальному часі. У ручному режимі організатор самостійно вказує суму для кожного учасника через інтерфейс, після чого сервер оновлює записи в базі даних і синхронізує дані через WebSocket. Ігровий режим активує одну з трьох ігор — "Рулетка", "Щаслива карта" або "Платник у тіні", — які додають елемент випадковості та розваги до розподілу витрат.

У грі "Рулетка" сервер генерує випадковий результат із набору варіантів, таких як: "Сьогодні за вас платить [Ім'я іншого учасника]", "Ви закриваєте весь стіл, оплачуючи за всіх", "Ви пропускаєте оплату", "Ви платите половину своєї частки", "Ви платите подвійну частку", "Випадковий учасник оплачує вашу частку", "Ви отримуєте промокод на знижку", "Ви оплачуєте лише напої". Результат генерується за допомогою псевдовипадкового алгоритму в TypeScript [2] і передається через WebSocket усім учасникам, забезпечуючи синхронне відображення. Організатор підтверджує результат, після чого сервер оновлює статуси платежів у базі даних.

Гра "Щаслива карта" моделює інтерактивну поведінку, де сервер випадково

розподіляє учасникам віртуальні карти з унікальними значеннями (наприклад, числові бали, бонуси чи штрафи). Кожен учасник може вибрати дію: обміняти карту з іншим гравцем, залишити свою або відкрити її. Обмін картами реалізується через WebSocket, який передає запити між клієнтами та сервером, синхронізуючи дії в реальному часі. Після відкриття карт сервер визначає результат (наприклад, звільнення від оплати або додатковий платіж) і оновлює дані в MongoDB. Ця гра стимулює соціальну взаємодію, оскільки учасники можуть стратегічно обмінюватися картами, що підвищує залученість.

Гра "Платник у тіні" імітує механіку "Мафії". Сервер випадково призначає учасникам ролі "Мешканців" або "Платників", приховуючи їх від інших. У кожному раунді учасники голосують через клієнтський інтерфейс, щоб визначити "Платника". Голоси обробляються сервером, а результати передаються через WebSocket. Якщо "Мешканці" виграють, "Платники" беруть на себе оплату, інакше рахунок ділиться між усіма. Організатор встановлює статуси платежів ("Сплатив", "Відмовився", "Взяв у борг"), які впливають на рейтинг учасників. Ця гра додає елемент інтриги та змагання, що відповідає розважальній спрямованості системи.

Рейтинг учасників формується на основі історії ігор і платежів, яка зберігається в MongoDB. Кожна дія, позначена організатором (сплата, відмова або борг), впливає на рейтинг: сплата додає бали, відмова чи борг їх віднімає. Система періодично оновлює рейтинг, використовуючи агрегацію даних із бази. На головній сторінці відображається карусель промокодів (наприклад, знижки в McDonald's), які сервер видає залежно від рейтингу учасника: вищий рейтинг відкриває доступ до кращих пропозицій. Промокоди зберігаються в MongoDB і оновлюються через серверну логіку.

Сторінка рейтингу відображає список учасників, відсортованих за кількістю сплат і боргів. Сервер виконує запит до MongoDB, агрегує дані та повертає результат клієнту. Зміни статусів, зроблені організатором, синхронізуються через WebSocket, забезпечуючи актуальність інформації. JWT-автентифікація захищає доступ до даних групи та рейтингу, дозволяючи лише авторизованим користувачам переглядати або змінювати інформацію [1]. Таким чином, поведінка системи

моделюється як послідовність взаємодій між клієнтом, сервером і базою даних, з акцентом на ігрову механіку та синхронізацію в реальному часі.

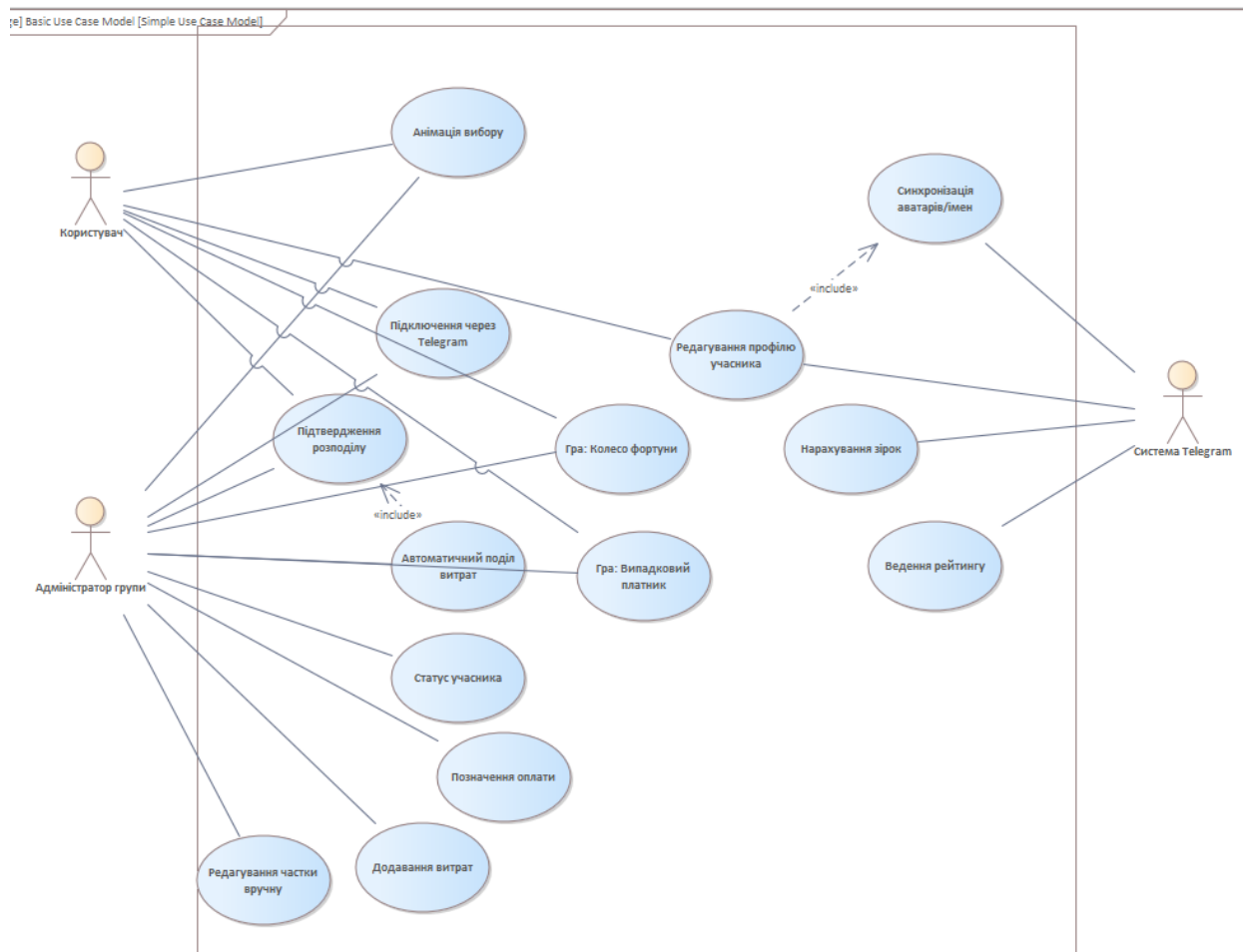


Рисунок 2.3 — Діаграма прецедентів

Джерело: сформовано автором на основі виконаного дослідження

2.3.2 Моделювання структури системи

Діаграми послідовності відображають порядок взаємодій між об'єктами в системі. Вони показують, як об'єкти обмінюються повідомленнями для виконання певної функції або процесу. Діаграми послідовності допомагають зрозуміти динаміку системи та взаємодію між її компонентами.

Діаграма послідовності відображає процес авторизації користувача через месенджер. Вона ілюструє взаємодію між користувачем, інтерфейсом Telegram, API Telegram, менеджером користувачів та базою даних. Спочатку користувач натискає кнопку "Увійти" в інтерфейсі програми. Це ініціює запит до

TelegramInterface, який відповідає за взаємодію з Telegram API. TelegramInterface відправляє запит на авторизацію до TelegramAPI, щоб отримати токен користувача. TelegramAPI обробляє запит і повертає токен користувача. Цей токен передається до UserManager, який відповідає за управління обліковими записами користувачів. UserManager перевіряє наявність запису в базі даних. Якщо запис існує, він оновлюється; якщо ні, створюється новий запис. Після успішної обробки запиту UserManager відправляє дані користувача назад до TelegramInterface, яке відображає інтерфейс користувача.

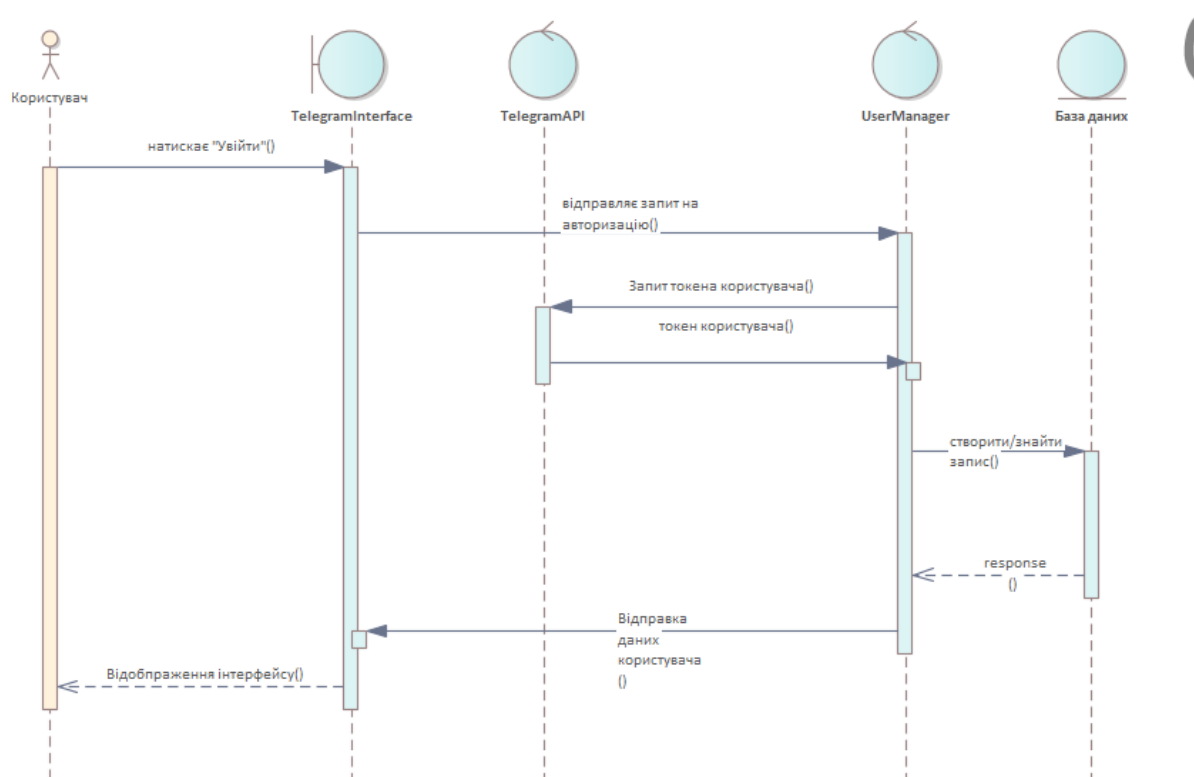


Рисунок 2.4 — Діаграма Авторизації

Джерело: сформовано автором на основі виконаного дослідження

Наступна діаграма ілюструє процес редагування профілю учасника в системі. Вона показує взаємодію між учасником, інтерфейсом Telegram, менеджером користувачів та базою даних. Процес починається, коли учасник ініціює редагування свого профілю, наприклад, змінює нікнейм. Ця дія передається до TelegramInterface, яке відповідає за обробку запитів користувачів. TelegramInterface

передає оновлені дані до UserManager для валідації. UserManager перевіряє коректність даних і передає їх до бази даних для збереження. База даних записує оновлені дані і повертає відповідь про успішність операції. UserManager отримує цю відповідь і передає результат операції назад до TelegramInterface. Нарешті, TelegramInterface надсилає учаснику повідомлення про статус операції, інформуючи його про успішне оновлення профілю.

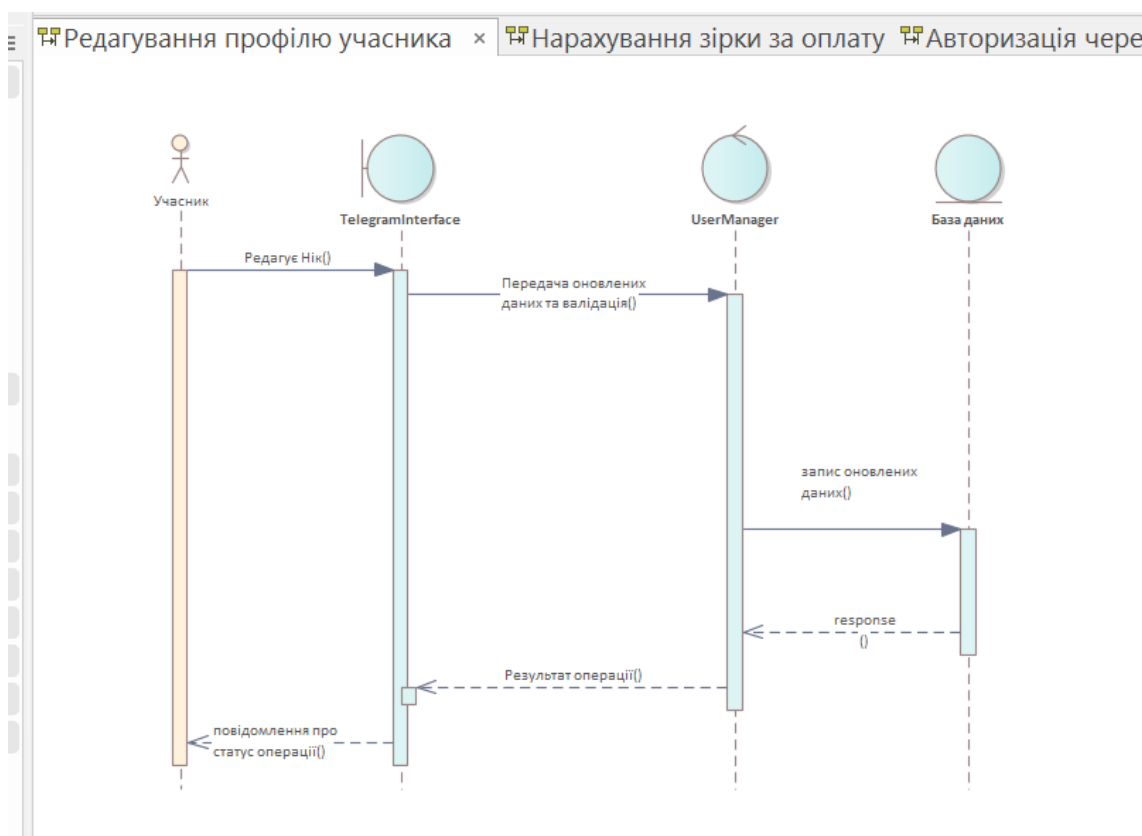


Рисунок 2.5 — Діаграма Редагування профілю

Джерело: сформовано автором на основі виконаного дослідження

Наступна діаграма ілюструє процес нарахування зірок користувачу за виконання сплати в системі. Вона показує взаємодію між адміністратором групи, інтерфейсом Telegram, менеджером платежів, менеджером зірок та базою даних. Процес починається, коли адміністратор групи натискає кнопку "Користувач сплатив" в інтерфейсі програми. Це ініціює запит до TelegramInterface, яке відповідає за обробку дій користувачів. TelegramInterface передає запит на реєстрацію оплати до PaymentManager, який оновлює статус витрати в системі. Після цього PaymentManager ініціює процес нарахування зірок, передаючи запит до StarManager. StarManager обробляє запит і додає зірку користувачу в базі даних.

Після успішного оновлення даних база даних повертає підтвердження. Нарешті, TelegramInterface отримує статус операції і повідомляє адміністратора про успішне нарахування зірок. Цей процес стимулює користувачів до своєчасних платежів, надаючи їм винагороду у вигляді зірок.

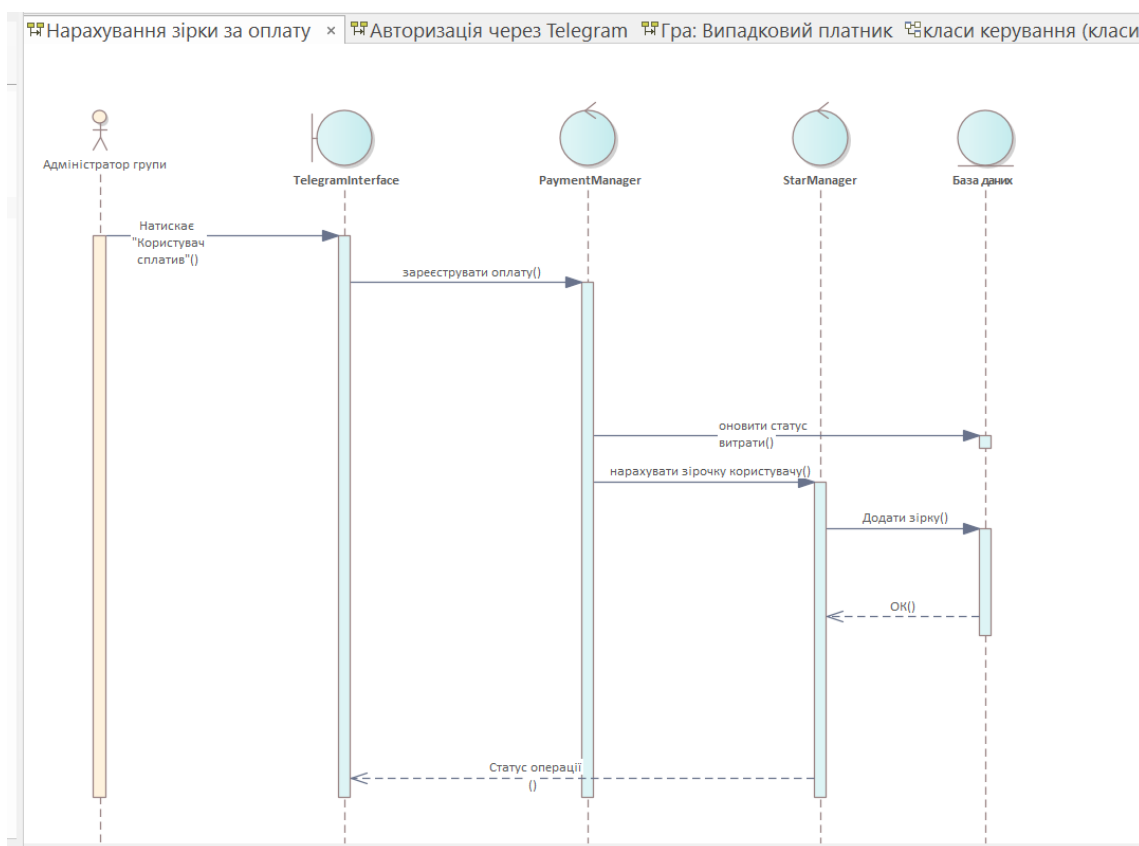


Рисунок 2.6 — Діаграма нарахування зірок

Джерело: сформовано автором на основі виконаного дослідження

Наступна діаграма ілюструє процес запуску гри "Випадковий платник" у застосунку. Процес починається, коли адміністратор групи ініціює запуск гри через TelegramInterface. TelegramInterface передає запит до GameManager, який відповідає за управління ігровими процесами. GameManager запитує список учасників у бази даних. Після отримання списку учасників, GameManager обирає випадкового платника. GameManager оновлює статус сесії учасників і визначає переможця. Дані про переможця передаються назад до TelegramInterface. TelegramInterface відображає результат гри адміністратору, інформуючи про вибраного платника. Цей процес додає елемент розваги до управління витратами, роблячи його більш інтерактивним і цікавим для учасників.

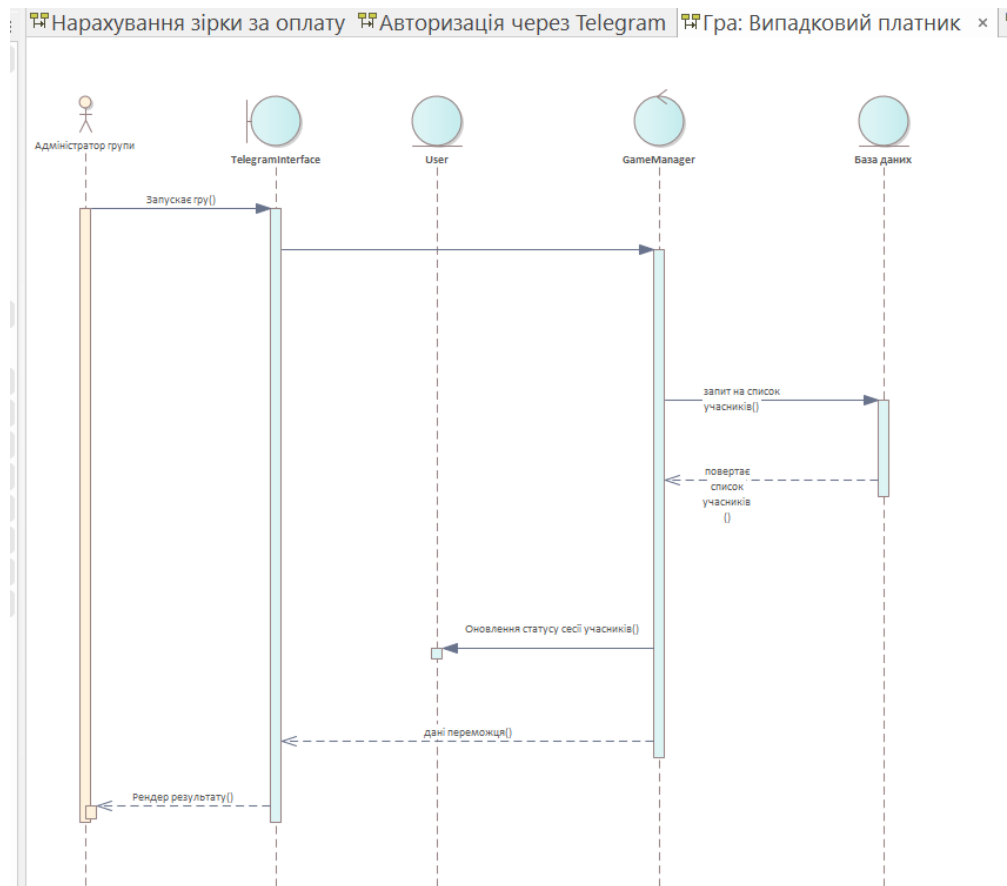


Рисунок 2.7 Діаграма запуску ігрової механіки для учасників сесії

Джерело: сформовано автором на основі виконаного дослідження

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

3.1.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення застосунку «Pay or Play», інтегрованого в Telegram як mini app, є основою для реалізації функціоналу, спрямованого на розумний розподіл витрат у групі друзів із сильним акцентом на розважальну та ігрову складову. Система розроблена для створення захоплюючого досвіду, де користувачі не лише ділять витрати, але й беруть участь у соціально-інтерактивних іграх, таких як «Рулетка», «Щаслива карта» та «Платник у тіні», які додають елемент азарту та змагальності. Інформаційне забезпечення охоплює організацію даних, їх зберігання, обробку та передачу між клієнтською частиною (побудованою на React) та серверною частиною (реалізованою на TypeScript із використанням Express і MongoDB). Воно також включає механізми синхронізації через WebSocket для забезпечення реального часу та автентифікацію через JSON Web Tokens (JWT) для безпеки. Основна мета інформаційного забезпечення — підтримка динамічного, безпечного та розважального середовища, де всі дії, від створення груп до результатів ігор, обробляються швидко й надійно.

Інформаційне забезпечення базується на використанні MongoDB як основної бази даних, яка обрана через її здатність ефективно працювати з неструктурованими даними, що відповідає потребам системи з різноманітними типами ігрових сценаріїв і транзакцій. Дані в MongoDB організовані в колекції, які включають профілі користувачів, інформацію про групи, історію транзакцій, результати ігор та промокоди. Наприклад, профіль користувача містить унікальний ідентифікатор (отриманий через Telegram), ім'я, рейтинг, історію боргів і сплат, що дозволяє відстежувати активність учасника та його внесок у групові витрати. Групи зберігаються як окремі документи, що включають ідентифікатор організатора,

назву (за замовчуванням «[Ім'я організатора] Group»), список учасників і унікальне посилання для запрошення. Транзакції фіксують розподіл витрат, де організатор вручну встановлює статуси («сплатив», «відмовився», «взяв у борг»), а також результати ігор, такі як «Сьогодні за вас платить [ім'я учасника]», «Ви закриваєте весь стіл», «Скиньтеся порівну», «Отримайте знижку 20%», «Заплатіть половину», «Ваш сусід праворуч платить за вас», «Один учасник звільнений від оплати» або «Сьогодні ваш щасливий день – платите лише 5%» у грі «Рулетка». Промокоди, прив'язані до рейтингу користувача, зберігаються в окремій колекції, що дозволяє динамічно видавати пропозиції, наприклад, знижки в McDonald's, залежно від активності учасника [1].

Для забезпечення інтерактивності та синхронізації в реальному часі інформаційне забезпечення використовує WebSocket, який дозволяє миттєво передавати оновлення між сервером і клієнтами. Це критично для ігрових сценаріїв, де, наприклад, у «Щасливій карті» учасники по черзі відкривають карти з результатами, такими як «Звільнення від оплати» чи «Перекласти борг», а сервер синхронізує ці дії для всіх учасників. У грі «Платник у тіні», що базується на механіці «Мафії», WebSocket забезпечує передачу ролей (жителі чи платники) та результатів голосування, щоб усі учасники бачили, хто оплачує рахунок залежно від перемоги однієї зі сторін. WebSocket також використовується для оновлення статусів учасників, наприклад, коли організатор позначає «взяв у борг», що впливає на рейтинг і відображається на всіх пристроях у реальному часі. Це створює динамічний ігровий досвід, де соціальна взаємодія та азарт є ключовими елементами [2].

Безпека інформаційного забезпечення реалізується через JWT, який генерується після авторизації користувача через Telegram. Кожен запит до сервера, наприклад, створення групи, зміна її назви чи встановлення статусу платежу, супроводжується перевіркою токена, що захищає дані від несанкціонованого доступу. Особливо це важливо для дій організатора, який має привілеї, такі як видалення групи чи управління транзакціями. JWT інтегрується з MongoDB для зберігання та валідації токенів, забезпечуючи надійний захист профілів

користувачів, історії боргів і рейтингів [3].

Рейтингова система та промокоди є важливими елементами інформаційного забезпечення, що підсилюють розважальну спрямованість. Рейтинг обчислюється на основі дій учасника: оплата підвищує його, а борг чи відмова знижують. Ці дані зберігаються в профілі користувача та використовуються для видачі промокодів, які відображаються в каруселі на головній сторінці. Наприклад, користувач із високим рейтингом може отримати знижку 50% у закладі, тоді як із низьким — лише 10%. Сторінка рейтингу відображає статистику всіх учасників, включаючи кількість сплат і боргів, що додає змагальний елемент і мотивує активну участь [1].

3.1.2 Організація збору і передання первинної інформації

Організація збору і передання первинної інформації в системі «Pay or Play», інтегрований у Telegram як міні app, є ключовим елементом для забезпечення функціональності застосунку, спрямованого на розподіл витрат у групі друзів із акцентом на розважальну та ігрову складову. Первинна інформація охоплює дані про користувачів, групи, транзакції, ігрові сесії, рейтинги та промокоди, які збираються через клієнтську частину (React), передаються на сервер (TypeScript, Express, MongoDB), синхронізуються в реальному часі через WebSocket і захищаються за допомогою JWT. Основна мета організації збору і передання інформації — створення динамічного та інтерактивного середовища, де користувачі беруть участь у іграх, таких як «Рулетка», «Щаслива карта» та «Платник у тіні», що додають азарту та соціальної взаємодії, забезпечуючи при цьому точність і актуальність даних.

Загальні вимоги до організації збору і передання інформації:

- Дані повинні передаватися та оброблятися в реальному часі або з мінімальною затримкою, особливо для ігрових сесій та оновлення статусу платежів.
- Інформація, що вводиться користувачами або генерується системою, має бути максимально точною для коректного розрахунку боргів, рейтингів та

відображення даних.

- Система повинна забезпечувати цілісність даних під час передачі та зберігання, запобігаючи втратам або пошкодженням.
- Захист інформації від несанкціонованого доступу, зміни або витоку за допомогою механізмів автентифікації (JWT) та авторизації.
- Процес введення інформації для користувача має бути інтуїтивно зрозумілим та мінімально трудомістким.
- Архітектура повинна дозволяти обробляти зростаючий обсяг даних та кількість користувачів без значної деградації продуктивності.

Збір первинної інформації починається з автентифікації користувача через Telegram, який надає унікальний ідентифікатор і базові дані, такі як ім'я. Ця інформація автоматично передається на сервер через API-запит, де створюється або оновлюється профіль користувача в базі даних MongoDB. Профіль включає унікальний ідентифікатор, ім'я, рейтинг, історію транзакцій (сплати, борги, відмови) та ігрову активність [1].

Наприклад, коли користувач створює групу, клієнтська частина через форму на сторінці створення групи збирає назву (за замовчування «[Ім'я організатора] Group») і генерує унікальне посилання для запрошення, яке сервер зберігає в колекції груп. Учасники, які активують посилання, додаються до групи через API-запит, а їхні профілі пов'язуються з групою в базі даних. Цей процес забезпечує швидке формування групового лобі, де організатор може керувати учасниками, змінювати назву або видаляти групу, а інформація про ці дії миттєво передається на сервер для збереження.

Передача інформації для ігрового модуля є критично важливим для підтримки розважальної спрямованості. У грі «Рулетка» організатор запускає гру через клієнтський інтерфейс, а сервер генерує випадковий результат. Ці результати передаються через WebSocket усім учасникам, щоб їхні інтерфейси синхронно відображали анімоване колесо та підсумок гри. У «Щасливій карті» клієнтська частина збирає дії учасників (відкриття карт), які сервер обробляє, генеруючи комбінації. WebSocket забезпечує синхронне відображення цих дій, що додає

інтерактивності та азарту. У грі «Платник у тіні», натхненній «Мафією», сервер розподіляє ролі (жителі чи платники) і збирає голоси учасників через клієнтські запити, а результати голосування передаються через WebSocket, щоб усі бачили, хто оплачує рахунок залежно від перемоги однієї зі сторін. Такий підхід забезпечує динамічну соціальну взаємодію, де учасники активно залучені до гри [2].

Для розподілу витрат організатор через клієнтський інтерфейс вводить загальну суму чека, обираючи автоматичний розподіл порівну або ручне визначення сум для кожного учасника. Ці дані передаються на сервер через REST API, де фіксуються в колекції транзакцій. Організатор також встановлює статуси («сплатив», «відмовився», «взяв у борг»), які впливають на рейтинг учасників. Наприклад, відмова чи борг знижує рейтинг, а оплата його підвищує. Ці статуси передаються через WebSocket, щоб усі учасники бачили актуальну інформацію на своїх пристроях. Рейтингова система тісно пов'язана з промокодами, які сервер видає на основі рейтингу користувача. Карусель промокодів на головній сторінці отримує дані через API-запит, де сервер повертає пропозиції, такі як знижки в McDonald's, залежно від рівня рейтингу [1, 2].

Безпека збору та передання інформації забезпечується через JWT. Кожен запит до сервера, наприклад, створення групи, запуск гри чи оновлення статусу, супроводжується токеном, який сервер перевіряє перед обробкою. Це захищає дані, такі як історія боргів чи рейтинг, від несанкціонованого доступу. JWT генерується після авторизації через Telegram і зберігається в клієнтській частині для подальших запитів. Сервер також періодично перевіряє валідність токенів, щоб запобігти їх компрометації [3].

Таблиця 3.1. Перелік джерел та способів надходження вхідної інформації

<i>№ з/п</i>	<i>Назва вхідного повідомлення</i>	<i>Джерело інформації</i>	<i>Носій інформації</i>	<i>Періодичність надходження</i>	<i>Спосіб надання інформації</i>
1	Дані авторизації користувача	Telegram API	Автоматичний запит (JSON)	При кожній сесії	Автоматичне отримання
2	Створення/редагування групи	Користувач	Електронна форма (JSON)	За потреби	Введення даних через UI

		організатор			
3	Запрошення учасника до групи	Користувач-організатор	Унікальне посилання (URL)	За потреби	Копіювання/надсилення посилання
4	Додавання витрати (чека)	Користувач-організатор	Електронна форма (JSON)	За потреби	Введення даних через UI
5	Вибір режиму розподілу витрат	Користувач-організатор	Елемент вибору (JSON)	При додаванні витрати	Взаємодія з UI
6	Параметри ручного розподілу	Користувач-організатор	Електронна форма (JSON)	При виборі ручного розподілу	Введення даних через UI
7	Дії в ігрових режимах	Користувач-учасник	Інтерактивні елементи (JSON)	Під час активної гри	Взаємодія з UI
8	Встановлення статусу платежу	Користувач-організатор	Елемент вибору (JSON)	За потреби	Взаємодія з UI
9	Запит історії	Користувач	Кнопка (HTTP-запит)	За потреби	Взаємодія з UI
10	Перегляд промокодів	Користувач	Навігація (HTTP-запит)	За потреби	Взаємодія з UI

Джерело: сформовано автором на основі виконаного дослідження

3.1.3 Побудова системи класифікації та кодування

Побудова системи класифікації та кодування в інформаційному забезпеченні застосунку «Pay or Play», створеного як Telegram mini app, відіграє важливу роль у структуруванні даних для ефективного управління групами, розподілом витрат і розважальними ігровими сценаріями. Система класифікації та кодування розроблена для організації даних у базі MongoDB, що дозволяє чітко ідентифікувати користувачів, групи, транзакції, ігрові сесії, рейтинги та промокоди. Вона забезпечує швидкий доступ до інформації, необхідної для інтерактивних функцій, таких як ігри «Рулетка», «Щаслива карта» та «Платник у тіні», які є центральними для розважальної спрямованості застосунку. Кодування

даних підтримує обробку інформації на сервері (TypeScript, Express), синхронізацію через WebSocket і безпечний доступ через JWT, забезпечуючи структурованість і безперебійну роботу системи.

Основою системи класифікації є розподіл даних на ключові категорії, які відповідають основним сутностям застосунку. Кожен класифікатор має ієрархічну або лінійну структуру, що забезпечує унікальність і легкість ідентифікації об'єктів.

Використовувані класифікатори та коди:

1. Користувачі

- *Найменування класифікатора:* Користувачі системи.
- *Метод кодування:* Послідовний (для внутрішнього ID), прямий (для Telegram ID).
- *Структура коду:*
 - *Telegram User ID:* Числовий або рядковий, присвоюється Telegram (наприклад, 123456789). Довжина до 10-12 символів.
 - *Внутрішній User ID:* Унікальний ідентифікатор MongoDB ObjectId (наприклад, 60c72b2f9c8f2b001c8e2d3e). Довжина 24 символи (шістнадцятковий).
- *Приклад:* Користувач з Telegram ID 123456789 має внутрішній USER_ID: 60c72b2f9c8f2b001c8e2d3e.

2. Групи

- *Найменування класифікатора:* Групи користувачів.
- *Метод кодування:* Послідовний (для внутрішнього ID), прямий (для назви).
- *Структура коду:*
 - *Group ID:* Унікальний ідентифікатор MongoDB ObjectId (наприклад, 60c72b2f9c8f2b001c8e2d4f). Довжина 24 символи.
 - *Назва групи:* Текстовий рядок (наприклад, "Друзі на обід"). Довжина до 50 символів.
- *Приклад:* Група "Поїздка в Карпати" має GROUP_ID: 60c72b2f9c8f2b001c8e2d4f.

3. Транзакції / Витрати

- *Найменування класифікатора:* Фінансові операції.
- *Метод кодування:* Послідовний (для ID), прямий (для типу).
- *Структура коду:*
 - *Transaction ID:* Унікальний ідентифікатор MongoDB ObjectId (наприклад, 60c72b2f9c8f2b001c8e2d5a). Довжина 24 символи.
 - *Тип транзакції:* Символьний код, що визначає статус платежу. Довжина 3 символи.
 - PAY – сплатив
 - REF – відмовився
 - DEBT – взяв у борг
- *Приклад:* Транзакція з ID 60c72b2f9c8f2b001c8e2d5a має тип PAY.

4. Ігрові сесії

- *Найменування класифікатора:* Типи міні-ігор.
- *Метод кодування:* Послідовний (для ID), прямий (для типу гри).
- *Структура коду:*
 - *Game Session ID:* Унікальний ідентифікатор MongoDB ObjectId (наприклад, 60c72b2f9c8f2b001c8e2d6b). Довжина 24 символи.
 - *Тип гри:* Символьний код, що ідентифікує міні-гру. Довжина 3 символи.
 - RLT – Рулетка
 - LCK – Щаслива карта
 - SHD – Платник у тіні
- *Приклад:* Сесія гри "Платник у тіні" має GAME_SESSION_ID: 60c72b2f9c8f2b001c8e2d6b та GAME_TYPE: SHD.

5. Результати ігор (на прикладі Рулетки)

- *Найменування класифікатора:* Можливі результати Рулетки.
- *Метод кодування:* Символьний (мнемонічний) або числовий.
- *Структура коду:*
 - OUT_PAY_OTHER – "Сьогодні за вас платить [ім'я учасника]"
 - OUT_FULL_TABLE – "Ви закриваєте весь стіл"

- OUT_SPLIT_EQUAL – "Скиньтеся порівну"
- OUT_DISCOUNT_30 – "Отримайте знижку 30%"
- OUT_PAY_HALF – "Заплатіть половину"
- OUT_PAY_NEIGHBOR – "Ваш сусід ліворуч платить за вас"
- OUT_FREE_ONE – "Один учасник звільнений від оплати"
- OUT_PAY_5_PERCENT – "Сьогодні ваш щасливий день – платите лише 5%"
- *Приклад:* Якщо результат рулетки - "Сьогодні за вас платить інший", він кодується як OUT_PAY_OTHER.

6. Рейтинг користувачів

- *Найменування класифікатора:* Рівні рейтингу.
- *Метод кодування:* Числовий, діапазонний.
- *Структура коду:*
 - *Рейтинг:* Ціле число в діапазоні, наприклад, від 0 до 1000.
- *Приклад:* Рейтинг 850.

7. Промокоди

- *Найменування класифікатора:* Типи та рівні промокодів.
- *Метод кодування:* Символьний (мнемонічний), ієрархічний за рівнем.
- *Структура коду:*
 - *Promo Code ID:* Унікальний ідентифікатор MongoDB ObjectId (наприклад, 60c72b2f9c8f2b001c8e2d7c). Довжина 24 символи.
 - *Рівень промокоду:*
 - LOW (наприклад, 0-300 балів рейтингу)
 - MID (наприклад, 301-700 балів рейтингу)
 - HIGH (наприклад, 701-1000 балів рейтингу)
 - *Код активації:* Унікальний буквено-цифровий рядок (наприклад, XYZ789ABC). Довжина до 10 символів.
- *Приклад:* Промокод для користувача з високим рейтингом має PROMO_LEVEL: HIGH та ACTIVATION_CODE: XYZ789ABC.

8. Ролі користувачів

- *Найменування класифікатора:* Ролі користувачів у системі/групі.
- *Метод кодування:* Символьний (мнемонічний).
- *Структура коду:*
 - ADMIN – Організатор групи.
 - MEMBER – Учасник групи.
- *Приклад:* role_admin для організатора.

Кодування ігрових сценаріїв відіграє ключову роль у підтримці розважального аспекту. У грі «Рулетка» результати кодуються як об'єкти з ідентифікатором гри та типом результату (наприклад, outcome_pay_other, outcome_full_table), що дозволяє серверу швидко обробляти і передавати їх через WebSocket для синхронного відображення на клієнтських інтерфейсах. У «Щасливій карті» кожна карта має унікальний код (наприклад, card_free, card_debt_shift), а комбінації результатів кодуються як масив дій, що визначають, хто платить або отримує пільгу. Наприклад, карта «Перекласти борг» кодується як card_debt_shift з параметром цільового user_id. У грі «Платник у тіні», натхненній «Мафією», ролі кодуються як role_citizen або role_payer, а результати голосування зберігаються як vote_xxx з прив'язкою до GROUP_ID і GAME_SESSION_ID. Така система кодування забезпечує чіткість і швидкість обробки ігрових даних, що додає інтерактивності та азарту, оскільки учасники бачать результати в реальному часі [2].

Для розподілу витрат система класифікації використовує категорії транзакцій, які кодуються залежно від методу: автоматичний розподіл порівну (split_equal) або ручний розподіл (split_manual). У разі автоматичного розподілу сума чека ділиться на кількість учасників, а кожна транзакція кодується з ідентифікатором TX_ID, сумою та статусом (PAY, REF, DEBT). Ручний розподіл дозволяє організатору вказати суму для кожного учасника, що також фіксується з відповідним кодом. Ці статуси впливають на рейтинг, який кодується як числове значення (RATING_SCORE) і оновлюється після кожної транзакції. Наприклад, дія «взяв у борг» (DEBT) знижує рейтинг на певний відсоток, тоді як «сплатив» (PAY) його підвищує. Рейтинг використовується для класифікації промокодів, де

пропозиції кодуються за рівнями (наприклад, PROMO_LEVEL: HIGH для знижки 50%), що забезпечує мотивацію для активної участі [1].

Безпека системи класифікації та кодування підтримується через JWT, де кожен запит до сервера включає токен (JWT_TOKEN), який перевіряється перед обробкою даних. Користувацькі дії, такі як створення групи чи запуск гри, класифікуються за роллю (організатор – ADMIN, учасник – MEMBER), що дозволяє обмежити доступ до привілейованих функцій. Наприклад, лише організатор (ADMIN) може змінювати назву групи чи встановлювати статуси транзакцій, що кодується в запитах як ROLE_ADMIN [3].

3.1.4 Проєктування форм первинних документів та відеокадрів

У контексті застосунку "первинні документи" слід розуміти як структуровані дані, що генеруються внаслідок дій користувача або системи, і які є основою для подальшої обробки та відображення. Вони мають електронну форму подання.

Таблиця 3.1 — Форма первинного документа: Створення/Редагування групи

Найменування: Створення/Редагування групи	Ідентифікатор: DOC-GROUP-SETUP
Користувачі: Організатор групи	Періодичність: За потребою
Термін видачі: Миттєво	
Поле	Зміст
GROUP_NAME	Назва групи (лобі)
ORGANIZER_ID	Унікальний ID організатора групи (Telegram ID)
PARTICIPANT_LIST	Масив унікальних ID учасників групи
INVITE_LINK	Унікальне посилання для запрошення до групи

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 3.2 — Форма первинного документа: Фіксація витрати/транзакції

Найменування: Фіксація витрати/транзакції	Ідентифікатор: DOC-EXPENSE-RECORD
Користувачі: Організатор групи	Періодичність: За потребою
Термін видачі: Миттєво	
Поле	Зміст
CHECK_AMOUNT	Загальна сума чека (витрати)
CHECK_DESCRIPTION	Короткий опис витрати
TARGET_GROUP_ID	ID групи, до якої належить витрата

DISTRIBUTION TYPE	Тип розподілу витрат
PARTICIPANT_SHARE_LIST	Список індивідуальних часток (для ручного розподілу)

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 3.3 — Форма первинного документа: Оновлення статусу платежу

Найменування: Оновлення статусу платежу	Ідентифікатор: DOC-PAYMENT-STATUS-UPDATE
Користувачі: Організатор групи	Періодичність: За потребою
Термін видачі: Миттєво	
Поле	Зміст
AFFECTED_USER_ID	ID учасника, чий статус змінюється
NEW_PAYMENT_STATUS	Новий статус платежу
DEBT_ID	ID конкретного боргу, що оновлюється
OWED_TO_USER_ID	ID користувача, якому взято в борг (якщо NEW_PAYMENT_STATUS="взяв у борг")

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 3.4 — Форма первинного документа: Результат ігрової сесії

Найменування: Результат ігрової сесії	Ідентифікатор: DOC-GAME-RESULT-LOG
Користувачі: Усі учасники групи	Періодичність: Після завершення гри
Термін видачі: Миттєво	
Поле	Зміст
GAME_SESSION_ID	Унікальний ID ігрової сесії
GAME_TYPE	Тип гри (RLT, LCK, SHD)
WINNER_LOSER_ID	ID учасника, визначеного платником/переможцем
FINANCIAL_IMPACT	Сума, пов'язана з результатом гри
GAME_OUTCOME_CODE	Код результату гри (наприклад, OUT_PAY_OTHER)

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 3.5. Форма первинного документа: Профіль користувача (оновлений)

Найменування: Профіль користувача (оновлений)	Ідентифікатор: DOC-USER-PROFILE-UPDATE
Користувачі: Система (для оновлення), користувач (для перегляду)	Періодичність: При зміні статусу платежу / Запиті користувача
Термін видачі: Миттєво	
Поле	Зміст
USER_ID	Унікальний ID користувача
CURRENT_RATING_SCORE	Поточний бал рейтингу користувача
N_PAID	Кількість успішних оплат
N_REFUSED	Кількість відмов від оплати
N_DEBT	Кількість випадків, коли користувач "взяв у борг"

Джерело: сформовано автором на основі виконаного дослідження

3.1.5 Структура інформаційних масивів

Інформаційні масиви спроектовані для зберігання та обробки даних у базі MongoDB. Масиви передаються між клієнтською частиною (React) і серверною (TypeScript, Express) через REST API, синхронізуються в реальному часі за допомогою WebSocket і захищаються через JWT, забезпечуючи структурованість і безпеку даних.

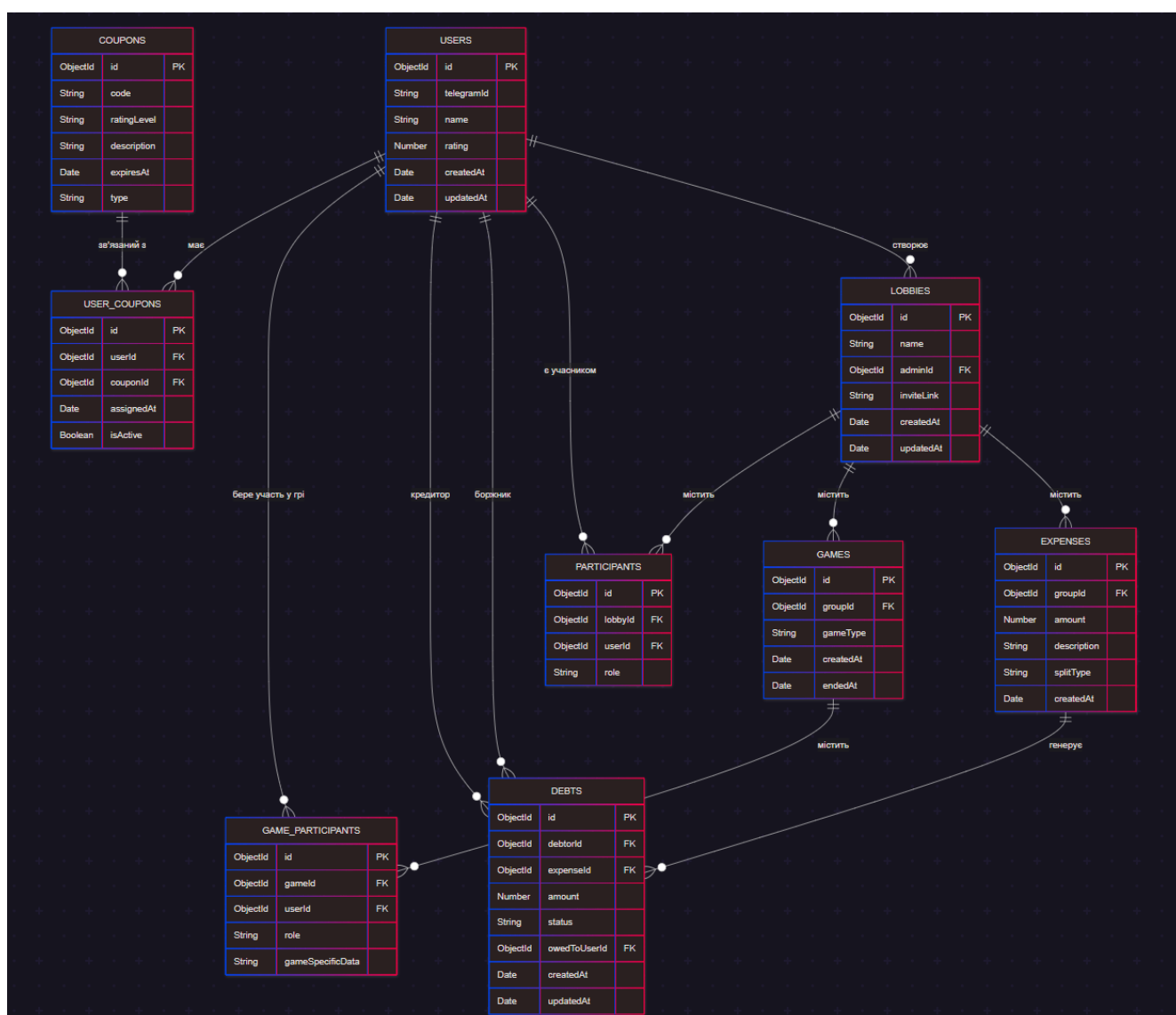


Рисунок 3.1 — Загальний вигляд моделі бази даних

Документ *USERS* (Користувачі). Цей масив є основним сховищем інформації про всіх користувачів системи. Кожен запис (документ) у цьому масиві містить унікальний ідентифікатор `_id`, наданий MongoDB, а також `telegramId` – унікальний ідентифікатор користувача від Telegram. Зберігається також `name` (ім'я

користувача) та rating (числовий показник рейтингу). Поля createdAt та updatedAt фіксують час створення та останнього оновлення профілю. Масив USERS має зв'язок "один до багатьох" з масивами LOBBIES (користувач може створити багато лобі), PARTICIPANTS (користувач може бути учасником багатьох груп), DEBTS (користувач може бути боржником у багатьох боргах) та GAME_PARTICIPANTS (користувач може брати участь у багатьох іграх).

Документ LOBBIES (Групи). Ця колекція зберігає інформацію про створені групи або "лобі", де користувачі можуть розподіляти витрати та грати в ігри. Основними полями є _id (унікальний ідентифікатор групи) та name (назва групи). Поле adminId є зовнішнім ключем, що посилається на _id користувача в масиві USERS, вказуючи на організатора групи. Кожна група має унікальне inviteLink для запрошення нових учасників. Також фіксуються createdAt та updatedAt. Масив LOBBIES пов'язаний з EXPENSES (група містить багато витрат), GAMES (група містить багато ігор) та PARTICIPANTS (група містить багато учасників).

Документ PARTICIPANTS (Учасники груп). Цей масив виступає як допоміжна таблиця для реалізації зв'язку "багато до багатьох" між користувачами та групами. Кожен запис містить _id (унікальний ідентифікатор запису), lobbyId (зовнішній ключ до LOBBIES) та userId (зовнішній ключ до USERS). Додатково може містити поле role, що визначає роль користувача в конкретній групі (наприклад, "admin", "member").

Документ EXPENSES (Витрати). У цьому масиві зберігаються деталі про кожен зафіксовану витрату (чек). Кожен документ має _id, groupId (зовнішній ключ до LOBBIES, що вказує, до якої групи належить витрата). Інші важливі поля включають amount (загальна сума витрати), description (опис чека) та splitType (тип розподілу: "equal", "manual", "game"). Поле createdAt фіксує час додавання витрати. Масив EXPENSES має зв'язок "один до багатьох" з масивом DEBTS, оскільки одна витрата може генерувати кілька боргів.

Документ DEBTS (Борги). Цей масив деталізує індивідуальні боргові зобов'язання між учасниками. Кожен запис містить _id, debtorId (зовнішній ключ до USERS, що вказує, хто є боржником), expenseId (зовнішній ключ до EXPENSES,

що вказує, з якої витрати виник борг). Також вказується amount (сума боргу), status (поточний статус боргу: "pending", "paid", "refused", "borrowed") та owedToUserId (зовнішній ключ до USERS, що вказує, кому заборгували, якщо борг "взято в борг"). Поля createdAt та updatedAt відстежують зміни статусу боргу.

Документ GAMES (Ігри). Ця колекція зберігає інформацію про кожну ігрову сесію. Кожен документ має _id, groupId (зовнішній ключ до LOBBIES), gameType (тип гри: "Рулетка", "Щаслива карта", "Платник у тіні"). Поля createdAt та endedAt фіксують тривалість сесії. Масив GAMES пов'язаний з GAME_PARTICIPANTS (гра містить багато учасників).

Документ GAME_PARTICIPANTS (Учасники ігор). Цей масив слугує для зв'язку "багато до багатьох" між іграми та користувачами. Записи містять _id, gameId (зовнішній ключ до GAMES) та userId (зовнішній ключ до USERS). Додаткові поля, такі як role (роль у грі, наприклад, "жителі", "платники" для "Платника у тіні") та gameSpecificData (інформація, специфічна для конкретної гри, наприклад, "карта" для "Щасливої карти"), дозволяють зберігати деталі участі кожного користувача в ігровій сесії.

Документ COUPONS (Промокоди). У цьому масиві зберігається інформація про доступні промокоди. Кожен документ має _id, унікальний code, ratingLevel (рівень рейтингу, для якого доступний промокод). Також включаються description (опис пропозиції), expiresAt (термін дії) та type (тип промокоду, наприклад, "discount"). Масив COUPONS має зв'язок "один до багатьох" з USER_COUPONS, оскільки один промокод може бути призначений багатьом користувачам.

Документ USER_COUPONS (Промокоди користувачів). Цей масив фіксує, які промокоди були призначені конкретним користувачам. Кожен запис містить _id, userId (зовнішній ключ до USERS) та couponId (зовнішній ключ до COUPONS). Поля assignedAt (дата призначення) та isActive (чи активний промокод) дозволяють відстежувати використання промокодів.

3.1.6 Вибір СКБД

СКБД має підтримувати гнучку роботу з даними про користувачів, групи, транзакції, ігрові сесії («Рулетка», «Щаслива карта», «Платник у тіні»), рейтинги та промокоди, забезпечуючи швидку обробку запитів, синхронізацію через WebSocket і безпеку через JWT. Після аналізу вимог застосунку, таких як гнучкість структури даних, масштабованість і підтримка реального часу, було обрано MongoDB як основну СКБД, оскільки вона найкраще відповідає потребам системи, орієнтованої на інтерактивний і розважальний досвід.

MongoDB була обрана через її NoSQL-архітектуру, яка дозволяє гнучко працювати з неструктурованими даними, що є критично важливим для «Pay or Play». Застосунок обробляє різноманітні типи даних: профілі користувачів (з ідентифікатором, ім'ям, рейтингом, історією транзакцій і ігор), групи (з назвою, списком учасників і посиланням для запрошення), транзакції (з сумами, статусами «сплатив», «відмовився», «взяв у борг») і результати ігор, такі як «Сьогодні за вас платить [user_id]», «Ви закриваєте весь стіл», «Скиньтеся порівну», «Отримайте знижку 30%», «Заплатіть половину», «Ваш сусід ліворуч платить за вас», «Один учасник звільнений від оплати» або «Сьогодні ваш щасливий день – платите лише 5%». MongoDB використовує документо-орієнтовану модель, де дані зберігаються у вигляді JSON-подібних документів, що дозволяє легко адаптувати структуру до нових ігрових сценаріїв чи функцій, таких як додавання нових типів промокодів чи ігрових результатів. Наприклад, документ групи може містити масив учасників і транзакцій, а ігрова сесія — масив результатів, що спрощує обробку складних ігрових логік, таких як комбінації карт у «Щасливій карті» чи голосування в «Платнику у тіні» [1].

Ще однією причиною вибору MongoDB є її висока продуктивність при обробці запитів і підтримка масштабованих систем. Застосунок «Pay or Play» потребує швидкого доступу до даних для відображення рейтингів, історії боргів і промокодів на клієнтській стороні (React), а також для синхронізації через WebSocket, коли, наприклад, організатор змінює статус учасника чи запускає гру.

MongoDB забезпечує швидке виконання операцій читання та запису завдяки індексації та гнучкій структурі, що дозволяє ефективно обробляти запити, такі як вибірка всіх транзакцій групи чи оновлення рейтингу користувача після дії «взяв у борг». Крім того, MongoDB підтримує горизонтальне масштабування через шардування, що є перевагою для потенційного зростання кількості користувачів і груп, особливо з огляду на інтеграцію з Telegram, яка має велику аудиторію [1].

MongoDB також добре інтегрується з серверною частиною, побудованою на TypeScript і Express, завдяки бібліотекам, таким як Mongoose, які спрощують моделювання даних і валідацію. Наприклад, схема користувача може визначати обов'язкові поля (`user_id`, `name`) і необов'язкові (`rating`, `transactions`), що забезпечує цілісність даних. Для ігрових сесій MongoDB дозволяє зберігати складні об'єкти, такі як результати «Платника у тіні», де ролі (`role_citizen`, `role_payer`) і голосування формують вкладені масиви. Це спрощує обробку соціально-інтерактивних механік, де учасники голосують, щоб визначити, хто оплачує рахунок. WebSocket-запити, які синхронізують результати ігор чи статуси транзакцій, отримують дані з MongoDB у реальному часі, що забезпечує динамічний ігровий досвід [2].

Безпека даних у MongoDB підтримується через інтеграцію з JWT, де кожен запит до бази супроводжується перевіркою токена. Наприклад, лише організатор (`role_admin`) може змінювати статуси транзакцій чи видаляти групу, що забезпечується валідацією ролі в запитах до бази. MongoDB також дозволяє налаштувати доступ на рівні колекцій, що захищає дані, такі як історія боргів чи промокоди, від несанкціонованого доступу [3].

Альтернативи, такі як реляційні СКБД (наприклад, PostgreSQL), були розглянуті, але відхилені через їхню меншу гнучкість у роботі з неструктурованими даними та складність адаптації до динамічних ігрових сценаріїв. MongoDB краще підходить для швидкої ітерації та підтримки розважальних функцій, таких як карусель промокодів, яка залежить від рейтингу користувача, чи ігрові механіки, що потребують швидкої обробки результатів.

3.1.7 Інфологічна модель бази (сховища) даних

Модель розроблена для організації зберігання даних у MongoDB, що дозволяє ефективно управляти інформацією про користувачів, групи, транзакції, ігрові сесії, рейтинги та промокоди. Вона відображає логічні зв'язки між сутностями, які підтримують ключові функції: створення груп, розподіл витрат, ігри («Рулетка», «Щаслива карта», «Платник у тіні»), а також мотивацію через рейтинги та промокоди. Інфологічна модель враховує потреби синхронізації через WebSocket, обробки запитів через REST API (TypeScript, Express) і безпеки через JWT, забезпечуючи гнучкість і підтримку розважального досвіду.

Основною сутністю моделі є *Користувач*, який представляє учасника системи. Атрибути користувача включають унікальний ідентифікатор (`user_id`, отриманий від Telegram, наприклад, «tg_123456789»), ім'я (`name`, рядок), рейтинг (`rating`, числове значення від 0 до 1000), історію транзакцій (`transactions`, масив посилань на ідентифікатори транзакцій, наприклад, [tx_abc123, tx_def456]) та історію ігрових сесій (`game_history`, масив посилань на ідентифікатори ігор, наприклад, [game_rlt_001, game_lck_002]). Користувач пов'язаний із сутністю «Група» через участь у ній (як організатор або учасник) і з сутністю «Промокод» через рейтинг, який визначає доступні пропозиції. Ця сутність дозволяє відстежувати активність користувача, його борги та сплати, що відображаються на сторінці рейтингу та впливають на видачу промокодів, наприклад, знижок у McDonald's для користувачів із високим рейтингом [1].

Сутність *Група* моделює лобі, створене організатором. Її атрибути включають унікальний ідентифікатор (`group_id`, наприклад, grp_987654321), назву (`name`, за замовчуванням «[Ім'я організатора] Group»), ідентифікатор організатора (`admin_id`, посилання на `user_id`), список учасників (`members`, масив `user_id`), унікальне посилання для запрошення (`invite_link`, наприклад, link_xyz789) і список транзакцій (`group_transactions`, масив посилань на `tx_id`). Група пов'язана з сутностями «Користувач» (через учасників і організатора) та «Транзакція» (через розподіл витрат). Ця сутність забезпечує управління групами, дозволяючи

організатору змінювати назву, додавати учасників через посилання чи видаляти групу, з синхронізацією змін через WebSocket для відображення на головній сторінці всіх учасників [1, 2].

Сутність *Транзакція* описує розподіл витрат у групі. Вона включає ідентифікатор (tx_id, наприклад, tx_abc123), групу (group_id, посилання), загальну суму чека (amount, число), метод розподілу (split_type, «equal» або «manual»), і статуси учасників (participants, масив об'єктів із полями user_id, status [PAY, REF, DEBT] і sum для ручного розподілу). Транзакція пов'язана з «Користувачем» (через статуси та рейтинги) і «Групою» (через group_id). Наприклад, при автоматичному розподілі сума ділиться порівну, а при ручному організатор вказує суму для кожного учасника. Статуси («сплатив», «відмовився», «взяв у борг») впливають на рейтинг і синхронізуються через WebSocket для відображення в реальному часі [1, 2].

Сутність *Ігрова сесія* моделює розважальні ігри. Її атрибути включають ідентифікатор (game_id, наприклад, game_rlt_001), тип гри (game_type: RLT для «Рулетки», LCK для «Щасливої карти», SHD для «Платника у тіні»), групу (group_id, посилання), учасників (players, масив user_id) і результати (results, об'єкт або масив). Для «Рулетки» результати включають, наприклад, «pay_other:[user_id]», «full_table:[user_id]», «split_equal», «discount_30», «pay_half», «pay_neighbor_left», «one_free» або «pay_10_percent». Для «Щасливої карти» results містить комбінації карт (наприклад, card_free, card_debt_shift), а для «Платника у тіні» — ролі (role_citizen, role_payer) і підсумки голосування. Ця сутність пов'язана з «Користувачем» (через players) і «Групою» (через group_id), а результати передаються через WebSocket для синхронного відображення анімацій і підсумків гри [2].

Сутність *Промокод* описує мотиваційні пропозиції. Атрибути включають ідентифікатор (promo_id, наприклад, promo_xxx123), рівень рейтингу (rating_level: low, mid, high), опис (description, наприклад, «Знижка 50% у McDonald's») і термін дії (expires_at, дата). Промокод пов'язаний із «Користувачем» через рейтинг, який визначає доступні пропозиції. Ця сутність підтримує карусель промокодів на

головній сторінці, де цінніші пропозиції надаються користувачам із вищим рейтингом [1].

Безпека моделі забезпечується через JWT, де кожен доступ до даних (наприклад, зміна статусів транзакцій чи запуск гри) супроводжується перевіркою токена з роллю (admin для організатора, member для учасника). Інфологічна модель підтримує гнучкість MongoDB, дозволяючи легко додавати нові атрибути чи сутності, наприклад, нові типи ігор, що відповідає розважальній спрямованості системи [3].

3.1.8 Даталогічна модель бази (сховища)даних

Модель розроблена для підтримки ключових функцій: управління групами, транзакціями, ігровими сесіями («Рулетка», «Щаслива карта», «Платник у тіні»), рейтингами та промокодами. Даталогічна модель деталізує колекції, їхні поля, типи даних і зв'язки, що відповідають розважальній спрямованості системи.

Основною колекцією є **Users**, яка зберігає профілі користувачів. Кожен документ у цій колекції містить поля: `user_id` (рядок, унікальний ідентифікатор від Telegram, наприклад, «tg_123456789»), `name` (рядок, ім'я користувача), `rating` (число, значення від 0 до 1000, за замовчуванням 500), `transactions` (масив рядків, посилання на ідентифікатори транзакцій, наприклад, [«tx_abc123», «tx_def456»]), `game_history` (масив рядків, посилання на ідентифікатори ігрових сесій, наприклад, [«game_rlt_001», «game_lck_002»]). Індокси створюються на `user_id` (унікальний) і `rating` для швидкого пошуку та сортування на сторінці рейтингу. Ця колекція пов'язана з іншими через `user_id`, що дозволяє відстежувати активність користувача, його борги та участь у іграх, таких як «Щаслива карта», де результати впливають на рейтинг і доступ до промокодів.

Колекція *Groups* відповідає за зберігання інформації про групові лобі. Документ включає поля: `group_id` (рядок, унікальний, наприклад, «grp_987654321»), `name` (рядок, за замовчуванням «[Ім'я організатора] Group»), `admin_id` (рядок, посилання на `user_id` організатора), `members` (масив рядків, список

user_id учасників), invite_link (рядок, унікальне посилання, наприклад, «link_xyz789»), group_transactions (масив рядків, посилання на tx_id, наприклад, [«tx_abc123»]). Індекс на group_id забезпечує швидкий доступ, а зв'язок із Users через admin_id і members дозволяє керувати групами та синхронізувати зміни (наприклад, нову назву чи видалення групи) через WebSocket для відображення на головній сторінці.

Колекція *Transactions* фіксує розподіл витрат. Кожен документ містить: tx_id (рядок, унікальний, наприклад, «tx_abc123»), group_id (рядок, посилання на group_id), amount (число, загальна сума чека), split_type (рядок, «equal» або «manual»), participants (масив об'єктів із полями: user_id (рядок), status (рядок, «PAY», «REF», «DEBT»), sum (число, для ручного розподілу)). Індеси на tx_id і group_id забезпечують швидкий пошук транзакцій групи. Ця колекція пов'язана з Users через user_id у participants, а статуси («сплатив», «відмовився», «взяв у борг») впливають на rating і синхронізуються через WebSocket для оновлення в реальному часі.

Колекція *GameSessions* підтримує ігрові сесії, які є основою розважального аспекту. Документ включає: game_id (рядок, унікальний, наприклад, «game_rlt_001»), game_type (рядок, «RLT», «LCK», «SHD»), group_id (рядок, посилання), players (масив user_id), results (об'єкт або масив, залежно від гри). Для «Рулетки» results може містити {outcome: «pay_other», user_id: «tg_123456789»} або {outcome: «discount_30»} (для варіантів типу «Скиньтеся порівну», «Отримайте знижку 30%», «Заплатіть половину», «Ваш сусід ліворуч платить за вас», «Один учасник звільнений від оплати», «Сьогодні ваш щасливий день – платите лише 5%»). Для «Щасливої карти» results містить масив {user_id: «tg_123456789», card: «card_free»}. Для «Платника у тіні» — {roles: [{user_id: «tg_123456789», role: «role_payer»}], vote_result: «payers_pay»}. Індеси на game_id і group_id забезпечують швидкий доступ, а WebSocket передає результати для анімацій на клієнтській стороні.

Колекція *Promocodes* зберігає мотиваційні пропозиції. Документ містить: promo_id (рядок, унікальний, наприклад, «promo_xxx123»), rating_level (рядок,

«low», «mid», «high»), description (рядок, наприклад, «Знижка 50% у McDonald's»), expires_at (дата). Індекс на rating_level прискорює вибірку промокодів для каруселі на головній сторінці, пов'язаної з rating із Users [1].

Безпека забезпечується через JWT, де кожен запит до бази перевіряє токен із роллю (admin або member). Наприклад, лише admin може змінювати group_transactions чи запускати гру. MongoDB підтримує гнучкість моделі, дозволяючи додавати нові поля чи колекції для нових ігрових сценаріїв, що відповідає розважальній спрямованості [3].

3.2 Технічне забезпечення

3.2.1 Загальні положення та схема автоматизації

Інформаційна система «Pay or Play» розроблена як Telegram mini app, що передбачає хмарну архітектуру та функціонування в розподіленому середовищі. Ядро системи, база даних MongoDB, серверна частина (TypeScript, Express) та необхідні статичні файли розташовані на віддалених серверах хостинг-провайдера. Доступ клієнтів (кінцевих користувачів) до системи здійснюється через додаток Telegram на їхніх персональних пристроях (смартфони, планшети) з обов'язковим доступом до мережі Інтернет. Це дозволяє забезпечити високу доступність, масштабованість та мінімізувати вимоги до апаратного забезпечення кінцевих користувачів.

Пояснення до схеми:

- *Кінцевий користувач (Telegram App)*: Це клієнтська сторона системи, що працює в середовищі Telegram mini app на пристроях користувачів (смартфони, планшети, десктопи).
- *Інтернет*: Забезпечує зв'язок між клієнтською частиною та хмарною інфраструктурою.
- *Хмарний хостинг*: Узагальнене представлення хмарної інфраструктури, де

розміщені всі компоненти системи.

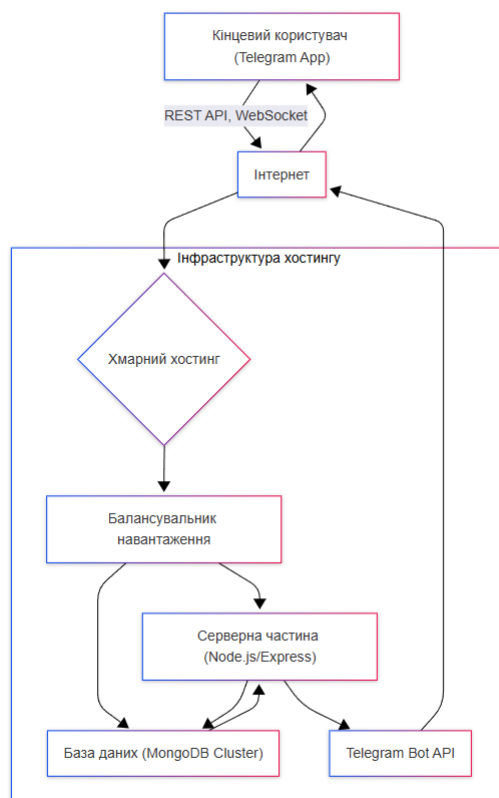


Рисунок 3.2 – Загальна схема автоматизації системи

- *Балансувальник навантаження*: Розподіляє вхідні запити між екземплярами серверної частини для забезпечення стабільності та масштабованості.
- *Серверна частина (Node.js/Express)*: Ядро системи, що обробляє бізнес-логіку, взаємодіє з базою даних та Telegram Bot API, а також надає REST API.
- *База даних (MongoDB Cluster)*: Розподілена база даних для зберігання всіх даних системи.
- *Telegram Bot API*: Інтерфейс для взаємодії системи з платформою Telegram (надсилання повідомлень, обробка команд тощо).

3.2.2 Структура комплексу технічних засобів

Комплекс технічних засобів (КТЗ) системи «Pay or Play» має розподілену структуру, що базується на моделі "клієнт-сервер" з використанням хмарної інстиляції.

- *Обґрунтування вибору структури комплексу технічних засобів:* Вибір хмарної архітектури обумовлений потребою в високій доступності, відмовостійкості, масштабованості та мінімізації витрат на адміністрування власного обладнання. Система розрахована на широке коло користувачів, які взаємодіють з нею через Telegram.
 - *Серверна частина:* Забезпечується хостинг-провайдером (наприклад, AWS, Google Cloud, Azure або DigitalOcean). Технічні відомості про сервер (CPU, RAM, Storage, Network Bandwidth) визначаються за потребами навантаження, але для мінімального запуску можна орієнтуватись на віртуальну машину з 2-4 vCPU, 4-8 GB RAM, SSD сховищем 100-200 GB та стабільним мережевим каналом (мінімум 100 Mbps). Хостинг-провайдер забезпечує фізичне розміщення, електроживлення, охолодження та базовий рівень мережевої безпеки.
 - *База даних:* Використовується MongoDB Atlas або інший хмарний сервіс MongoDB, що надає функціональність кластера для забезпечення високої доступності та автоматичного шардингу для масштабованості. Це дозволяє уникнути ручного адміністрування бази даних.
- *Опис розміщення КТЗ на об'єктах управління:* Основна частина КТЗ (сервери застосунків, сервери баз даних) фізично розміщується у дата-центрах хостинг-провайдера. Точне географічне розташування дата-центрів вибирається з урахуванням мінімізації затримок для основної аудиторії користувачів. Ці дата-центри відповідають міжнародним стандартам безпеки (Tier III/IV), що гарантує відповідні умови експлуатації (температура, вологість, безперебійне живлення) та фізичну безпеку.
- *Обґрунтування методів захисту технічних засобів:* Захист технічних засобів від механічних, теплових, електромагнітних та інших дій, а також від несанкціонованого доступу, забезпечується на кількох рівнях:
 - *На рівні хостинг-провайдера:* Дата-центри провайдера обладнані системами контролю доступу (ключ-картки, біометричні сканери),

системами відеоспостереження, кліматичними системами, системами пожежогасіння та резервними джерелами живлення. Мережеве обладнання розміщене у захищених стійках.

- *На рівні програмного забезпечення:* Використовуються брандмауери, системи виявлення вторгнень (IDS) та запобігання вторгненням (IPS) для захисту мережевого трафіку. Для обміну даними між клієнтською та серверною частинами, а також між сервером та базою даних, використовується протокол HTTPS (TLS/SSL шифрування), що забезпечує конфіденційність та цілісність даних у процесі передачі. Доступ до серверів та бази даних з боку розробників та адміністраторів захищений SSH-ключами та двофакторною автентифікацією. JWT-токени використовуються для автентифікації та авторизації користувачів у системі, обмежуючи доступ до функцій відповідно до їхніх ролей.
- *Обґрунтування та опис основних рішень по вибору типу ЕОМ і серверів:* Вибір типу ЕОМ та серверів базується на нефункціональних вимогах системи: масштабованість, надійність, продуктивність та економічна ефективність. Для серверної частини (Node.js/Express) та бази даних (MongoDB) використовуються віртуальні машини або контейнери, що надаються хмарним провайдером. Це дозволяє легко масштабувати ресурси (CPU, RAM) вгору (vertical scaling) або збільшувати кількість екземплярів (horizontal scaling) відповідно до зростання навантаження, оптимізуючи вартість транзакції. MongoDB Cluster забезпечує відмовостійкість та розподілене зберігання даних.
- *Опис технічних засобів, розміщених в обчислювальній мережі об'єкта:* У контексті системи «Pay or Play», яка є Telegram mini app, пряме використання фізичних датчиків, маніпуляторів або пускових реле не передбачається. Всі взаємодії відбуваються через програмні інтерфейси та API Telegram.
- *Опис основних рішень по вибору типів периферійних технічних засобів:* Для кінцевих користувачів, периферійні технічні засоби – це їхні власні мобільні

пристрої (смартфони, планшети) або персональні комп'ютери. Система «Pay or Play» не висуває специфічних вимог до периферійних пристроїв, оскільки взаємодія відбувається через стандартний інтерфейс Telegram. Візуалізація інформації здійснюється на екрані пристрою користувача, а введення даних – через сенсорний екран або клавіатуру.

- *Обґрунтування чисельності персоналу, що забезпечує функціонування технічних засобів:* Оскільки система "Pay or Play" розрахована на використання персональної техніки користувача (смартфони, планшети, ноутбуки) та повністю розміщена на хмарній інфраструктурі (хостинг-провайдер), власний персонал для фізичного обслуговування технічних засобів не передбачається. Обслуговування та підтримку апаратного забезпечення, мережевої інфраструктури та базового програмного забезпечення серверів забезпечує хостинг-провайдер. Для підтримки та розвитку програмного забезпечення системи потрібна команда розробників та адміністраторів бази даних, що працюють віддалено.

3.2.3 Опис автоматизованого робочого місця

- *Обґрунтування та опис рішень з вибору технічних засобів АРМ:* Система «Pay or Play» є Telegram mini app, що означає її функціонування в середовищі Telegram. Це дозволяє системі бути кросплатформною та не висувати високих вимог до конкретної апаратної конфігурації АРМ, оскільки основні обчислювальні навантаження виконуються на серверній частині. Мінімальні системні вимоги визначаються вимогами самого додатка Telegram:
 - *Процесор:* Будь-який сучасний мобільний (ARM) або десктопний (x86) процесор, здатний запускати додаток Telegram.
 - *Оперативна пам'ять:* Мінімум 2 ГБ RAM для комфортної роботи Telegram і mini app.
 - *Об'єм постійної пам'яті:* Мінімум 200 МБ вільного місця на накопичувачі для встановлення Telegram та кешування даних mini app.
 - *Операційна система:* iOS (версія 10.0+), Android (версія 5.0+),

Windows (7+), macOS (10.10+), Linux. Користувачі використовують власні пристрої, що знижує витрати на технічне забезпечення.

- *Обґрунтування та опис рішень вибору периферійних технічних засобів АРМ:* Система не потребує спеціалізованих периферійних технічних засобів. Вся взаємодія здійснюється через стандартні компоненти мобільних пристроїв або ПК:
 - *Монітор/Дисплей:* Для відображення інтерфейсу Telegram mini app. Якість зображення залежить від пристрою користувача.
 - *Клавіатура/Сенсорний екран:* Для введення текстової інформації та взаємодії з елементами інтерфейсу.
 - *Мережевий адаптер:* Вбудований модуль Wi-Fi або мобільного зв'язку (3G/4G/5G) для доступу до Інтернету.
- *Результати розрахунку (або розрахунок) числа технічних засобів кожного АРМ і потреби в мережному устаткуванні:* Оскільки АРМ користувачів – це їхні персональні пристрої, кількість АРМ потенційно дорівнює кількості активних користувачів системи, яка може сягати сотень тисяч. Для кожного АРМ передбачається використання *мобільного Інтернету (3G/4G/5G)* або *Wi-Fi* з доступом до мережі Інтернет. Спеціального мережевого устаткування на боці АРМ, окрім вбудованих модулів зв'язку, не вимагається. Використання апаратного шифрування або VPN на стороні кінцевого користувача є рекомендованою, але не обов'язковою вимогою для коректної роботи системи, оскільки сам протокол HTTPS та JWT забезпечують достатній рівень безпеки для передачі даних. Проте, користувачі можуть використовувати VPN для підвищення власної конфіденційності в мережі.

3.2.4 Схема мережі передачі даних

Система «Pay or Play» функціонує в розподіленому мережевому середовищі, де ключову роль відіграє взаємодія між клієнтською частиною (Telegram mini app на пристроях користувачів) та серверною інфраструктурою через Інтернет.

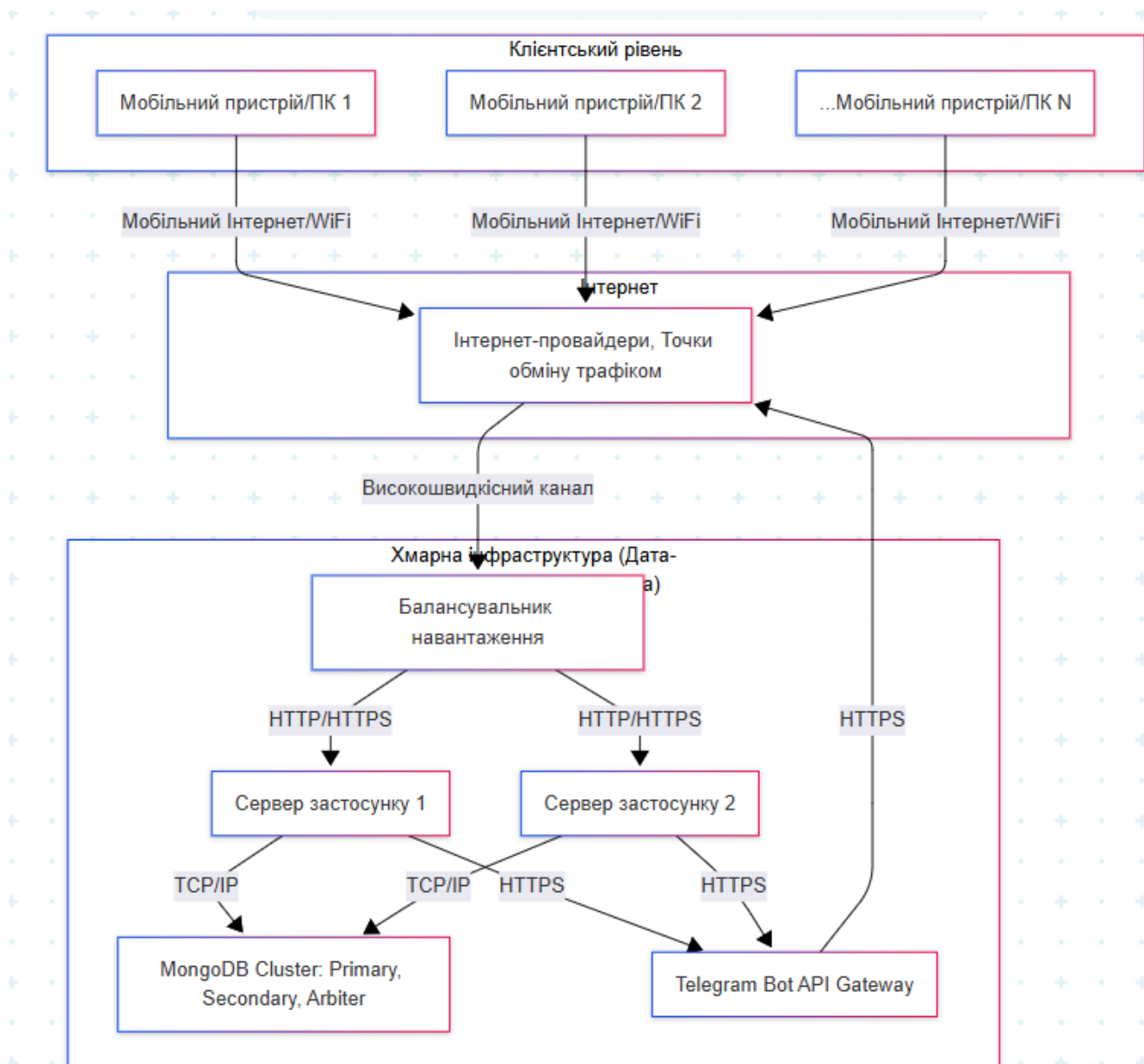


Рисунок 3.3. – Загальна схема мережі передачі даних системи

Обґрунтування та опис рішень по вибору засобів телеобробки і передачі даних: Вибір засобів телеобробки та передачі даних базується на принципах високої доступності, масштабованості та безпеки, характерних для хмарних сервісів. Вся інфраструктура системи знаходиться в хмарному середовищі, а кінцеві користувачі отримують доступ через глобальну мережу Інтернет.

- *Канали зв'язку:* Використовуються стандартні Інтернет-канали зв'язку. На клієнтському рівні – це мобільний Інтернет (3G/4G/5G) або Wi-Fi з доступом до зовнішньої мережі. На рівні хмарної інфраструктури – це високошвидкісні волоконно-оптичні канали, що надаються хостинг-провайдером, які є

частиною їхньої магістральної мережі. Кількість каналів зв'язку на стороні хостингу є надлишковою та автоматично масштабується.

□ *Вибір мережевого апаратного забезпечення:*

- *На клієнтському рівні:* Мережеве апаратне забезпечення інтегровано в мобільні пристрої користувачів (модеми 3G/4G/5G, Wi-Fi адаптери) або ПК (Ethernet-адаптери, Wi-Fi адаптери). Тип кабельного підключення (якщо це ПК) зазвичай "Вита пара: UTP" до домашнього роутера.
- *На рівні хмарної інфраструктури:* Забезпечується хостинг-провайдером і включає високопродуктивні маршрутизатори, комутатори, балансувальники навантаження (як апаратні, так і програмні), міжмережеві екрани. Всі з'єднання між компонентами в дата-центрі здійснюються через високошвидкісні оптоволоконні канали.

□ *Вимоги до каналів зв'язку, що орендуються:* Для серверної інфраструктури, що орендується у хостинг-провайдера, вимоги до каналів зв'язку є наступними:

- *Швидкість каналу:* Мінімум 1 Гбіт/с для кожного віртуального сервера, з можливістю розширення до 10 Гбіт/с і вище в міру зростання навантаження.
- *Тип IP:* Статичний публічний IP-адрес для балансувальника навантаження/точки входу API. Внутрішні IP-адреси для серверів застосунків та бази даних.

Відомості про розміщення абонентів і об'ємно-тимчасові характеристики переданих даних:

- *Розміщення абонентів:* Абоненти (кінцеві користувачі) географічно розподілені. Це можуть бути користувачі з будь-якої точки світу, де є доступ до Інтернету та Telegram.
- *Об'ємно-тимчасові характеристики переданих даних:*
 - *Користувачі Telegram mini app:* Передача текстових даних (назви груп, транзакцій, описів ігор), невеликих об'ємів JSON-даних для REST API запитів (до 50 КБ на запит) та WebSocket-повідомлень (кілька КБ на повідомлення) для синхронізації в реальному часі. Активність може

бути інтенсивною під час ігрових сесій або розподілу витрат. Середня тривалість сесії може становити 15-30 хвилин, кілька разів на день.

- *Сервер застосунку <-> База даних:* Обмін JSON-документами, що відповідають розміру даних колекцій. Запити та відповіді можуть варіюватися від кількох КБ до кількох МБ (наприклад, при вибірці великої історії транзакцій або ігор). Частота взаємодії висока, особливо під час активності користувачів.
- *Сервер застосунку <-> Telegram Bot API:* Передача JSON-структур з командами та відповідями (до 100 КБ). Частота залежить від активності користувачів та інтеграції з Telegram.

Основні показники надійності, відмовостійкості, ремонтпридатності, довговічності, достовірності засобів телеобробки і передачі даних: Ці показники забезпечуються на рівні хмарного хостинг-провайдера:

- *Надійність та відмовостійкість:* Провайдер гарантує SLA (Service Level Agreement) на рівні 99.9% - 99.99% доступності сервісів, використовуючи резервування компонентів, дублювання даних (MongoDB Cluster з реплікацією), балансування навантаження та автоматичне відновлення після збоїв.
- *Ремонтпридатність:* Обладнання провайдера має високий рівень ремонтпридатності за рахунок модульної архітектури та наявності запасних частин. Програмні компоненти та системи управління дозволяють швидке розгортання та заміну несправних вузлів.
- *Довговічність:* Технічні засоби у дата-центрах регулярно оновлюються провайдером. Для програмних компонентів системи передбачається регулярна підтримка, оновлення та доопрацювання.
- *Достовірність:* Засоби телеобробки забезпечують достовірність передачі даних за рахунок використання протоколів із контролем цілісності (TLS/SSL) та механізмів виявлення та корекції помилок на мережевому рівні. Захист від електромагнітних завад забезпечується інфраструктурою дата-центрів. Гарантійні терміни та періодичність планового ремонту обладнання

встановлюються хостинг-провайдером відповідно до їхніх внутрішніх стандартів та SLA.

3.3 Програмне забезпечення

3.3.1 Структура програмного забезпечення

Структура програмного забезпечення системи «Pay or Play» відображає архітектуру розподіленого застосунку та розподіляється на три основні категорії: системне ПЗ, прикладне ПЗ та програмна документація. Ця структура дозволяє забезпечити ефективну розробку, розгортання та підтримку системи, що функціонує як Telegram mini app.

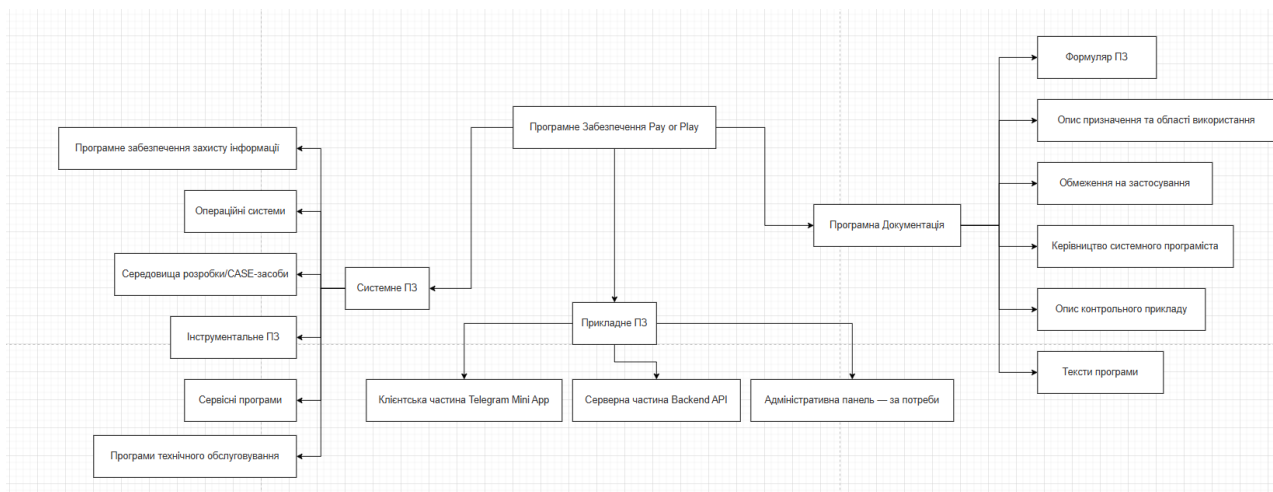


Рисунок 3.4 — Загальна схема структури програмного забезпечення системи

3.3.2 Системне програмне забезпечення

Системне програмне забезпечення є фундаментом для функціонування системи забезпечуючи базове середовище для виконання прикладних програм та взаємодії з апаратними засобами. Вибір компонентів системного ПЗ обґрунтований вимогами до масштабованості, продуктивності, безпеки та кросплатформності, що є критично важливим для хмарного веб-додатку.

До складу системного ПЗ входять:

- Операційні системи. Для серверної частини системи (backend API та база даних) використовуються дистрибутиви Linux (наприклад, Ubuntu Server або CentOS). Це рішення обрано завдяки високій стабільності, надійності, безпеці та ефективності використання ресурсів, що є перевагою для серверних застосунків у хмарних середовищах. Для розробки та тестування можуть використовуватися macOS або MS Windows на робочих станціях розробників. Кінцеві користувачі взаємодіють із системою через додаток Telegram, який функціонує на їхніх мобільних пристроях з Android або iOS, а також на ПК з MS Windows, macOS або Linux.
- RAD-засоби розробки проекту. Розробка та оформлення графічних схем, діаграм та специфікацій моделей (інфологічної та даталогічної) бази даних виконувались за допомогою Mermaid.js (для Markdown-документації) та інших онлайн-інструментів для ER-діаграм, що дозволяють візуалізувати структуру MongoDB. Для більш детального проектування та документування архітектури системи може бути використаний Enterprise Architect або Figma для проектування інтерфейсів.
- Основа для сайту. Оскільки система «Pay or Play» є Telegram mini app, вона не використовує традиційну CMS. Клієнтська частина розроблена за допомогою React (JavaScript-бібліотека) з використанням TypeScript для підвищення надійності коду. Це забезпечує інтерактивність та динамічне відображення даних без необхідності у традиційному сайті або CMS.
- Засоби для аналізу неструктурованих даних та гіпертекстової інформації. У поточному проєкті акцент робиться на структурованих даних та їх оперативній обробці. Аналіз неструктурованих даних (таких як текст з описів транзакцій або ігрових сесій) може бути реалізований за допомогою бібліотек обробки природної мови (NLP) на базі JavaScript або Python на серверній стороні (наприклад, для виявлення ключових слів або настрою, що може бути використано для подальшого вдосконалення функціоналу), але це не є основною вимогою на поточному етапі.
- Сервісні програми. До них належать утиліти та оболонки для

адміністрування серверів та бази даних (наприклад, SSH-клієнти для доступу до серверів Linux, MongoDB Compass або mongosh для роботи з MongoDB, Docker та Docker Compose для контейнеризації компонентів). Антивірусні засоби та засоби моніторингу (наприклад, Prometheus, Grafana) використовуються на серверній інфраструктурі, що надається хостинг-провайдером.

- Програми технічного обслуговування. Тестові програми та програми контролю входять до складу інструментарію розробки (наприклад, Jest, Mocha для unit-тестування та інтеграційних тестів, Postman для тестування API). Моніторинг працездатності серверів та баз даних здійснюється через інструменти хмарного провайдера.
- Програмне забезпечення захисту інформації. На додаток до вбудованих механізмів операційних систем, використовуються:
 - Брандмауери (на рівні ОС та мережі провайдера).
 - TLS/SSL сертифікати для шифрування трафіку між клієнтом та сервером (HTTPS, WSS).
 - JSON Web Tokens (JWT) для безпечної автентифікації та авторизації користувачів.
 - Засоби контролю доступу на рівні бази даних MongoDB.
- Програми машинної графіки. Для розробки інтерфейсу Telegram mini app використовуються інструменти для роботи з векторною та растровою графікою (наприклад, Figma для дизайну, Adobe Photoshop/Illustrator для створення елементів інтерфейсу).
- Програми обробки мовних сигналів. Пряма обробка мовних сигналів не передбачається в поточній версії, але можлива інтеграція з Telegram Bot API для голосових повідомлень у майбутньому.
- Інструментальне ПЗ.
 - Транслятори мов програмування: Node.js (середовище виконання JavaScript) та TypeScript Compiler (tsc).
 - Компільтони/Інтерпретатори: V8 JavaScript engine (вбудований в

Node.js), npm/yarn (пакетні менеджери).

- Системи контролю версій: Git та GitHub/GitLab для спільної розробки та управління вихідним кодом.
- Інтегровані середовища розробки (IDE): Visual Studio Code з відповідними розширеннями для TypeScript, React, Node.js та MongoDB.

3.3.3 Прикладне програмне забезпечення

Прикладні програмні засоби системи поділяються на:

- ПЗ загального призначення: Включає компоненти, які забезпечують базову функціональність, що є типовою для багатьох веб-застосунків. Це бібліотеки для роботи з датами, валідації даних, базові компоненти UI/UX.
- Проблемно-орієнтоване ПЗ: Це основні модулі, що реалізують унікальну бізнес-логіку системи «Pay or Play»:
 - Клієнтська частина (Telegram Mini App): Розроблена на React та TypeScript. Включає модулі для:
 - Автентифікації та авторизації користувачів через Telegram API та JWT.
 - Управління групами (лобі): Створення, приєднання, перегляд учасників, зміна назви, адміністрування.
 - Управління витратами та боргами: Додавання нових витрат, вибір типу розподілу (рівномірний, ручний, за грою), перегляд історії транзакцій, відстеження та управління індивідуальними боргами.
 - Ігрові механіки: Реалізація логіки ігор «Рулетка», «Щаслива карта», «Платник у тіні», відображення результатів та анімацій.
 - Система рейтингів: Відображення рейтингу користувача та загального лідерборду.

- Система промокодів: Відображення доступних промокодів та їх активація.
- Синхронізація даних: Взаємодія з серверною частиною через WebSocket для оновлень у реальному часі та REST API для інших запитів.
- Серверна частина (Backend API): Розроблена на Node.js з фреймворком Express.js та TypeScript. Включає модулі для:
 - REST API: Обробка HTTP-запитів від клієнтської частини.
 - WebSocket Server: Підтримка двостороннього зв'язку в реальному часі для ігор, оновлень груп та транзакцій.
 - Взаємодія з базою даних MongoDB: Використання бібліотеки Mongoose.js для роботи з колекціями, виконання CRUD-операцій та агрегацій.
 - Бізнес-логіка: Реалізація правил розподілу витрат, логіки ігор, розрахунку рейтингів, управління промокодами.
 - Інтеграція з Telegram Bot API: Відправка повідомлень користувачам, обробка базових команд.
 - Механізми безпеки: Перевірка JWT-токенів, валідація вхідних даних.
- ПЗ для роботи глобальних мереж: Крім базових мережевих протоколів, система використовує бібліотеки для роботи з WebSocket (наприклад, ws для Node.js) для реалізації обміну даними в реальному часі.
- ПЗ для організації (адміністрування) обчислювального процесу: Включає скрипти для розгортання (наприклад, CI/CD пайплайни на GitHub Actions або GitLab CI/CD), моніторингу та логування (наприклад, Winston для Node.js, інтеграція з сервісами логування хостинг-провайдера).

3.3.4 Програмна документація

Програмна документація системи «Pay or Play» розробляється для забезпечення повного та чіткого опису програмних засобів, їх можливостей, режимів роботи, порядку використання, а також вимог до інформаційного й технічного забезпечення. Вона є невід'ємною частиною проекту, що дозволяє як користувачам, так і розробникам та системним адміністраторам ефективно взаємодіяти із системою. Для підготовки програмної документації можуть використовуватися інструменти, що підтримують стандартизовані формати, такі як Markdown, а також спеціалізовані засоби, наприклад, Enterprise Architect (для UML-діаграм, специфікацій) або Swagger/OpenAPI для документування API.

До складу програмної документації входять:

- Формуляр: Документ, що описує основні характеристики програми (версія, дата розробки, розробник), її комплектність (перелік файлів, модулів, залежностей) та базові відомості про експлуатацію.
- Відомості про призначення програми, області її використання для розв'язання завдань: Детальний опис функціонального призначення системи «Pay or Play», її цілей, задач, які вона вирішує (розподіл витрат, організація ігор, мотивація користувачів) та цільової аудиторії.
- Обмеження на застосування мінімальної конфігурації технічних засобів: Опис мінімальних вимог до пристроїв кінцевих користувачів (операційна система, версія Telegram, обсяг пам'яті), а також базових вимог до серверної інфраструктури (для хостинг-провайдера).
- Керівництво системного програміста: Містить відомості, необхідні для встановлення, налаштування, перевірки та забезпечення функціонування системи в різних середовищах. Це включає:
 - Інструкції з розгортання (розгортання на сервері, налаштування середовища).
 - Опис архітектури мікросервісів (якщо застосовується) або модульної архітектури.

- Налаштування конфігураційних файлів (з'єднання з БД, порти, ключі API).
- Опис інструментів для моніторингу та логування.
- Інструкції з оновлення та обслуговування системи.
- Опис API (за допомогою Swagger/OpenAPI специфікації).
- Опис контрольного прикладу: Надається детальний опис тестових сценаріїв та прикладів даних, які використовуються для перевірки коректності функціонування системи, як це було наведено в розділі 3.1.7.3. Це дозволяє переконатися, що система працює відповідно до вимог.
- Тексти програми: Вихідний код всіх компонентів системи (клієнтська частина на React/TypeScript, серверна частина на Node.js/TypeScript) зберігається в системі контролю версій (Git) і доступний для перегляду та модифікації авторизованими розробниками. Це забезпечує можливість подальшого розвитку та підтримки системи.

3.4 Результати реалізації інформаційної системи

В основі виконаної випускної бакалаврської роботи лежить комплекс оригінальних проектних пропозицій, спрямованих на створення інтерактивної системи розподілу витрат та розваг у групі друзів на базі Telegram mini app. Суть цих пропозицій полягає у впровадженні інноваційних гейміфікованих механік у повсякденний процес управління спільними фінансами, що дозволяє не лише ефективно вирішувати проблему розподілу боргів, але й перетворити його на захоплюючу соціальну взаємодію. Це досягається шляхом розробки гнучкої та масштабованої архітектури на основі MongoDB та Node.js/React, що забезпечує швидкий доступ до даних та динамічне оновлення інтерфейсу через WebSocket.

Для апробації розроблених рішень та оцінки їх функціональності було проведено пілотне впровадження ключового функціоналу системи «Pay or Play» на контрольному прикладі. Враховуючи, що система є Telegram mini app, пілотне впровадження відбувалось у тестовому середовищі, імітуючи реальну взаємодію користувачів. Тестування проводилося з використанням набору вхідних даних,

деталізованих у Додатку Д «Контрольний приклад завантаження даних».

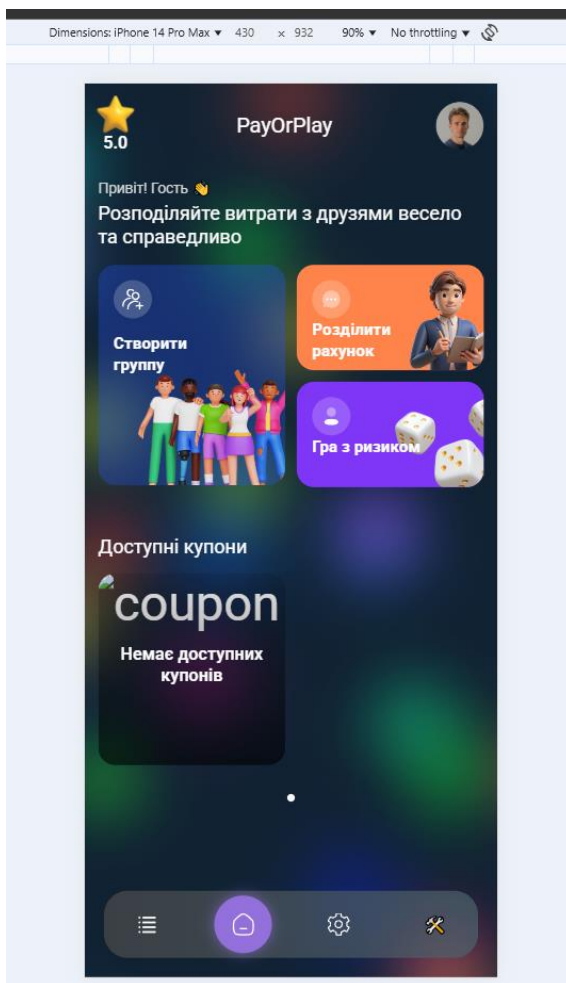


Рисунок 3.5 — Загальний вигляд в тестовому середовищі

Сценарій апробації включав взаємодію групи друзів, що складалася з трьох користувачів. Було імітовано створення групи, додавання спільної витрати з подальшим рівномірним розподілом, а також проведення ігрової сесії "Рулетка", що призвела до визначення одного з учасників "платником за всіх". Завершальним етапом сценарію було оновлення рейтингів користувачів та автоматичне призначення промокоду одному з учасників за досягнення певного рівня рейтингу.

Проведене тестування показало, що розроблене програмне забезпечення функціонує відповідно до заявлених вимог. Усі ключові функціональні блоки — створення груп, додавання транзакцій, розподіл витрат, проведення ігор, оновлення рейтингів та видача промокодів — працюють коректно. Тестування продуктивності на контрольному прикладі демонструвало миттєвий відгук системи, підтверджуючи ефективність обраної архітектури та бази даних для

сценаріїв з невеликою кількістю даних. Синхронізація через WebSocket забезпечувала оновлення даних в реальному часі без відчутних затримок. Тестування безпеки підтвердило, що використання JWT-токенів забезпечує базовий рівень захисту доступу до API, блокуючи несанкціоновані дії. Зручність використання (UX) інтерфейсу Telegram mini app виявилася інтуїтивно зрозумілою, забезпечуючи плавний та інтерактивний досвід.

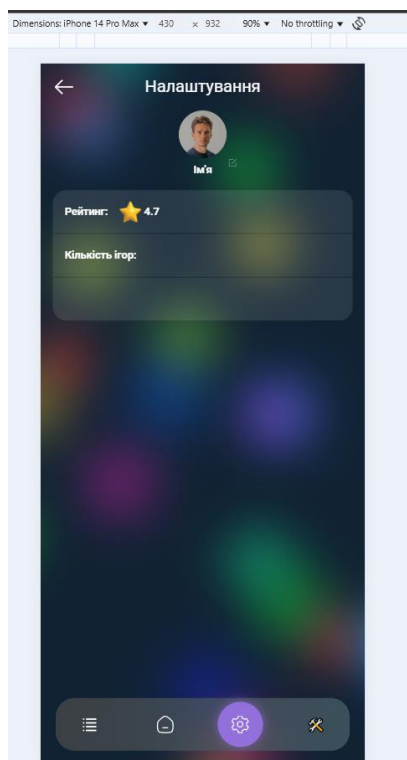


Рисунок 3.5 — Вигляд налаштувань в тестовому середовищі

Результативні вихідні документи, що відображають стан бази даних після проведення пілотного впровадження, представлені фрагментами колекцій *lobbies*, *expenses*, *debts*, *games* та *user_coupons* у Додатку Г.2 «Результативні вихідні документи», де також наведені скріншоти інтерфейсу системи на різних етапах тестування. Пілотне впровадження проводилося внутрішньою командою розробки без залучення зовнішніх організацій чи підприємств.

Порівняно з існуючими на ринку альтернативами, які були проаналізовані в підрозділі 1.2 (наприклад, Splid, Splitwise, Settle Up), система «Pay or Play» має значну потенційну практичну цінність та економічну ефективність завдяки інтеграції розважальних елементів та платформи Telegram. Це створює додаткову

мотивацію для користувачів своєчасно сплачувати борги та активно взаємодіяти з системою, вирішуючи поширену проблему затягування розрахунків у групах. Інтеграція в Telegram як міні app значно знижує бар'єр входу для нових користувачів, оскільки не потребує встановлення окремого застосунку, і дозволяє використовувати вірусні механіки розповсюдження через запрошення до груп.

З економічної точки зору, використання хмарної інфраструктури (MongoDB Atlas, хмарний хостинг) та NoSQL бази даних мінімізує витрати на адміністрування та масштабування. Застосування RAD-підходів та готових рішень для інфраструктури значно скорочує час на розробку та виведення продукту на ринок, підвищуючи його конкурентоспроможність. Система промокодів може бути монетизована через партнерство з брендами, пропонуючи цільові пропозиції. Важливою перевагою є відсутність прямих комісій за транзакції, що робить її більш привабливою для користувачів порівняно з деякими фінансовими застосунками.

Аналіз отриманих результатів впровадження підтверджує повноту реалізації основного функціоналу та життєздатність концепції. Дані, отримані в ході апробації, є достовірними в рамках тестового середовища та відповідають очікуваній логіці. Припущення щодо продуктивності та масштабованості, засновані на виборі MongoDB та хмарної архітектури, є адекватними для початкового етапу розвитку. Можливі похибки на етапі тестування були пов'язані з невеликим обсягом даних, що не дозволяє повною мірою оцінити продуктивність під високим навантаженням. Однак, обрані технології (Node.js, MongoDB) відомі своєю здатністю до масштабування.

Система «Pay or Play» демонструє високий ступінь готовності до реального використання. Основний функціонал розроблено та протестовано, що дозволяє перейти до етапу публічної бета-версії.

Рекомендації щодо подальших досліджень та удосконалень включають: проведення детального моделювання навантаження та оцінку вартості транзакції при зростанні кількості користувачів, із застосуванням методів аналізу черг для оптимізації роботи серверів. Доцільним є впровадження мікросервісної архітектури для рознесення функціоналу на окремі незалежні сервіси (наприклад,

окремий сервіс для ігор, окремий для транзакцій), що підвищить масштабованість та відмовостійкість. Для забезпечення стабільності при пікових навантаженнях та легкої інтеграції нових модулів рекомендується використати брокери повідомлень, такі як Kafka або RabbitMQ, для асинхронної обробки подій.

У частині гейміфікації варто розробити більш складні алгоритми розрахунку рейтингу, що враховуватимуть не лише своєчасність оплати, а й активність в іграх, участь у створенні груп та відгуки інших користувачів. Використання теорії ігор може допомогти у створенні нових, більш збалансованих та захоплюючих ігрових сценаріїв. Для покращення користувацького досвіду та аналітики пропонується впровадити A/B тестування для оптимізації інтерфейсу та ігрових механік, а також використовувати аналітичні платформи (наприклад, Google Analytics, Mixpanel) для збору та аналізу поведінки користувачів. Розробка OLAP-сховища даних дозволить проводити глибокий аналіз агрегованих даних та формувати стратегію розвитку продукту. Крім того, для поглибленого захисту інформації доцільно розглянути використання сервісів управління секретами та впровадження деталізованого логування подій безпеки.

Рекомендації та пропозиції, викладені в цьому підрозділі, ґрунтуються на основних теоретичних положеннях та методичних підходах, описаних у попередніх розділах, та відповідають критеріям практичної реальності, враховуючи сучасні тенденції розвитку веб-технологій та хмарних рішень.

ВИСНОВКИ

Робота присвячена розробці та апробації інформаційної системи «Pay or Play», реалізованої як Telegram mini app, що є інноваційним рішенням для гейміфікованого розподілу витрат у групі друзів. Головною метою роботи було створення функціональної, зручної та привабливої платформи, яка перетворить рутинний процес фінансових розрахунків на інтерактивну соціальну взаємодію. Поставлені завдання, що включали аналіз предметної області, проектування архітектури, розробку та пілотне впровадження системи, були успішно виконані.

У ході роботи було детально проаналізовано предметну область, виявлено ключові проблеми, пов'язані з традиційними методами розподілу витрат, та вивчено існуючі на ринку альтернативи. Це дозволило обґрунтувати актуальність розробки нової системи, яка поєднує в собі функціональність фінансового обліку з елементами гейміфікації та зручністю використання платформи Telegram.

Проектування системи здійснювалося з урахуванням сучасних підходів до розробки веб-застосунків. Інфологічна модель бази даних визначила ключові сутності та їхні зв'язки, що забезпечило логічну цілісність даних. Даталогічна модель, реалізована на базі документо-орієнтованої СКБД MongoDB, дозволила створити гнучку та масштабовану структуру даних, яка ідеально підходить для динамічної та розвиваючої схеми. Застосування MongoDB також забезпечує високу продуктивність та відмовостійкість.

Технічне забезпечення системи базується на хмарній архітектурі, що передбачає розміщення основних компонентів (серверна частина на Node.js/Express та база даних MongoDB) на віддалених серверах хостинг-провайдера. Це рішення обґрунтоване високою доступністю, масштабованістю та мінімізацією витрат на адміністрування. Кінцеві користувачі взаємодіють із системою через додаток Telegram, що є кросплатформним та не висуває високих вимог до апаратного забезпечення АРМ. Захист інформації забезпечується на рівні хмарної інфраструктури, протоколів передачі даних (HTTPS/WSS) та програмних механізмів (JWT).

Програмне забезпечення системи розроблено з використанням сучасних технологій та методологій. Для клієнтської частини (Telegram mini app) обрано React та TypeScript, що забезпечує інтерактивність та надійність. Серверна частина реалізована на Node.js з фреймворком Express.js та TypeScript, що гарантує високу продуктивність та ефективну обробку запитів. Важливим елементом є використання WebSocket для синхронізації даних у реальному часі, що критично для ігрових механік та миттєвого відображення змін.

Пілотне впровадження ключового функціоналу на контрольному прикладі підтвердило працездатність системи та коректність її функціонування відповідно до заявлених вимог. Тестування показало, що всі основні функціональні блоки (створення груп, розподіл витрат, ігрові сесії, система рейтингів та промокодів) працюють справно, а інтерфейс користувача є інтуїтивно зрозумілим. Результати апробації свідчать про значну практичну цінність системи «Pay or Play» завдяки її інноваційному підходу до гейміфікації фінансових розрахунків, що мотивує користувачів до своєчасного погашення боргів та підвищує загальну залученість. Економічна ефективність досягається за рахунок зниження операційних витрат завдяки хмарній інфраструктурі та швидкої розробки.

Особистий внесок студента у розроблення проектних рішень полягає у:

- Самостійному аналізі та обґрунтуванні вибору технологічного стеку (Node.js, React, MongoDB) для забезпечення масштабованості та продуктивності системи.
- Розробці детальної даталогічної моделі бази даних, що враховує специфіку документо-орієнтованого підходу та вимоги до гейміфікованих елементів.
- Реалізації ключових модулів серверної логіки, включаючи обробку транзакцій, механізми розподілу витрат та базові алгоритми ігрових сесій.
- Розробці та налагодженню взаємодії клієнтської та серверної частин через REST API та WebSocket.
- Проведенні пілотного тестування системи на контрольному прикладі та аналізі отриманих результатів.

Незважаючи на успішне пілотне впровадження, система має певні обмеження

та перспективи для подальшого розвитку. Наразі, глибинний аналіз неструктурованих даних не реалізований, а масштабування під екстремальні навантаження потребує додаткового тестування.

Рекомендації щодо подальших досліджень та удосконалень:

1. Масштабування архітектури: Для забезпечення високої доступності та відмовостійкості при значному зростанні кількості користувачів доцільно розглянути перехід до повноцінної мікросервісної архітектури та використання брокерів повідомлень (наприклад, Kafka) для асинхронної взаємодії між сервісами.
2. Розширення функціоналу гейміфікації: Дослідити можливість впровадження більш складних економіко-математичних моделей та алгоритмів з теорії ігор для розробки нових, динамічних ігрових сценаріїв, що поглиблюватимуть залученість користувачів.
3. Аналітична підсистема: Створення OLAP-сховища даних для агрегації та аналізу великих обсягів інформації про транзакції та поведінку користувачів. Це дозволить ідентифікувати тенденції, оптимізувати бізнес-процеси та приймати обґрунтовані рішення щодо розвитку продукту.
4. Покращення UX/UI: Проведення А/В тестування різних елементів інтерфейсу для подальшої оптимізації користувацького досвіду та підвищення конверсії.
5. Монетизація: Розробка механізмів інтеграції з партнерами для реалізації партнерських промокодів та розширення можливостей монетизації системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Telegram Bot API [Електронний ресурс]. Режим доступу: <https://core.telegram.org/bots/api>
2. Офіційна документація MongoDB [Електронний ресурс]. Режим доступу: <https://www.mongodb.com/docs/>
3. Документація Node.js [Електронний ресурс]. Режим доступу: <https://nodejs.org/en/docs/>
4. React Official Documentation [Електронний ресурс]. Режим доступу: <https://react.dev/learn>
5. TypeScript Documentation [Електронний ресурс]. Режим доступу: <https://www.typescriptlang.org/docs/>
6. Express.js Framework Documentation [Електронний ресурс]. Режим доступу: <https://expressjs.com/>
7. WebSocket API Specification [Електронний ресурс]. Режим доступу: <https://websockets.spec.whatwg.org/>
8. JSON Web Tokens (JWT) RFC 7519 [Електронний ресурс]. Режим доступу: <https://tools.ietf.org/html/rfc7519>
9. Statista: Messenger Usage Statistics 2023 [Електронний ресурс]. Режим доступу: <https://www.statista.com/topics/4488/mobile-messenger-apps/>
10. Chou Y.-K. Actionable Gamification: Beyond Points, Badges, and Leaderboards. Octalysis Media, 2019.
11. Deterding S. et al. Gamification: Using Game Design Elements in Non-Gaming Contexts // CHI '11 Proceedings, 2011
12. Hamari J., Koivisto J. Why do people use gamification services? // International Journal of Information Management, 2015.
13. Telegram Mini Apps Developer Platform [Електронний ресурс]. Режим доступу: <https://core.telegram.org/bots/webapps>
14. Fintech Trends Report 2024 by McKinsey [Електронний ресурс]. Режим доступу: <https://www.mckinsey.com/industries/financial-services/our-insights>

- 15.Zichermann G., Cunningham C. Gamification by Design. O'Reilly Media, 2011.
- 16.MongoDB Performance Best Practices [Электронный ресурс]. Режим доступа:
<https://www.mongodb.com/docs/manual/administration/optimization-advice/>
- 17.OWASP JWT Cheat Sheet [Электронный ресурс]. Режим доступа:
https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_Cheat_Sheet.html
- 18.Nielsen J. Usability Engineering. Morgan Kaufmann, 1993.
- 19.Google Material Design Guidelines [Электронный ресурс]. Режим доступа:
<https://m3.material.io/>
- 20.ISO 9241-210:2019 Ergonomics of human-system interaction.
- 21.Stripe API Documentation [Электронный ресурс]. Режим доступа:
<https://stripe.com/docs/api>
- 22.REST API Design Rulebook by M. Massé. O'Reilly, 2011.
- 23.WebSocket vs REST Performance Comparison // IEEE Conference Publication, 2022.
- 24.Nguyen T. et al. Gamification in Fintech Applications // Journal of Behavioral Finance, 2023.
- 25.Kapp K.M. The Gamification of Learning and Instruction. Pfeiffer, 2012.
- 26.Statista: Digital Payments Report 2024 [Электронный ресурс]. Режим доступа:
<https://www.statista.com/study/42340/digital-payments-report/>
- 27.React Native Documentation [Электронный ресурс]. Режим доступа:
<https://reactnative.dev/docs/getting-started>
- 28.UX Design Principles for Gamified Systems // ACM Digital Library, 2022.
- 29.Heroku Cloud Application Platform [Электронный ресурс]. Режим доступа:
<https://www.heroku.com/>
- 30.AWS Cloud Security Best Practices [Электронный ресурс]. Режим доступа:
<https://aws.amazon.com/security/>

ДОДАТКИ

Додаток А

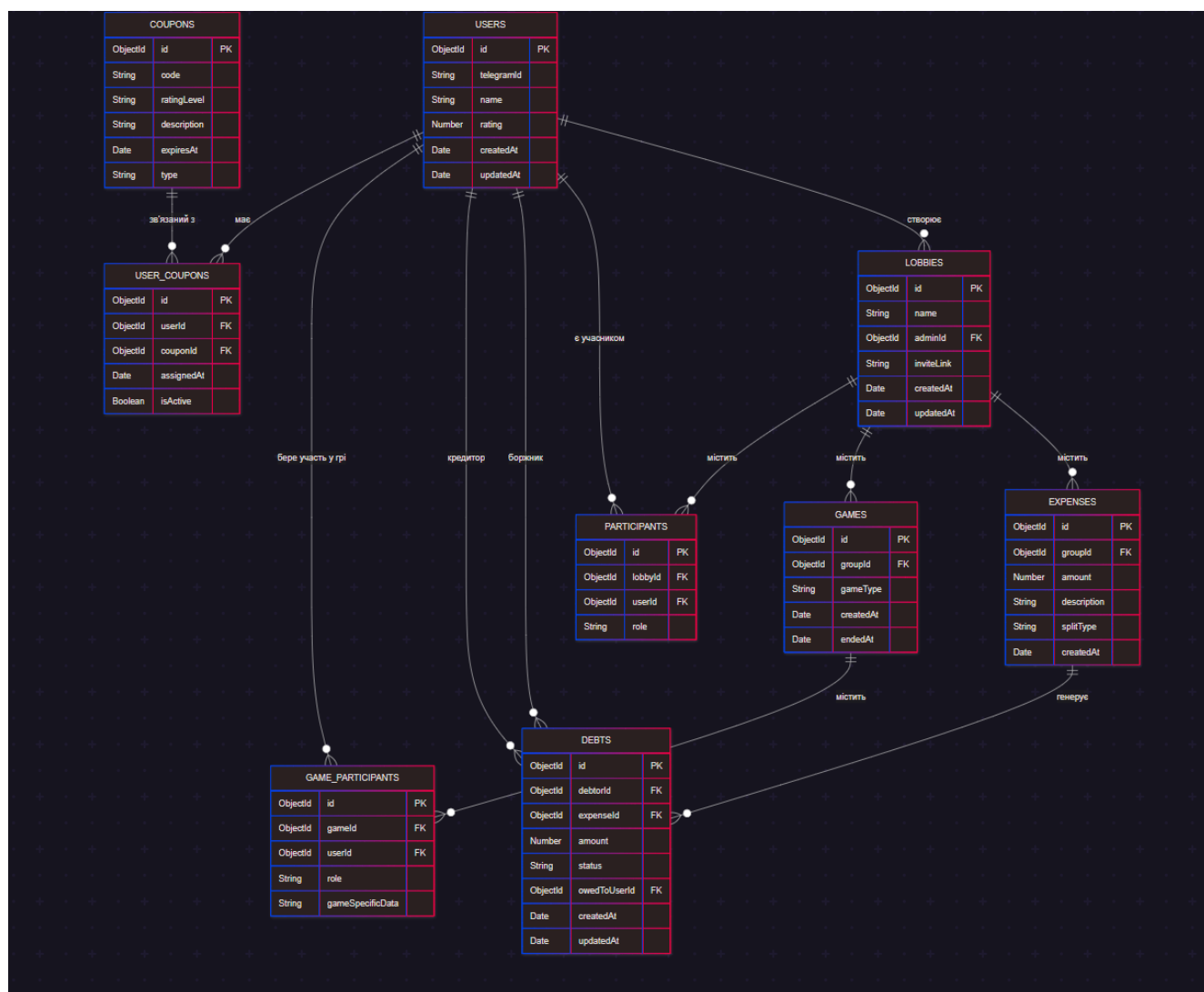


Рисунок А.2 – Схема бази даних

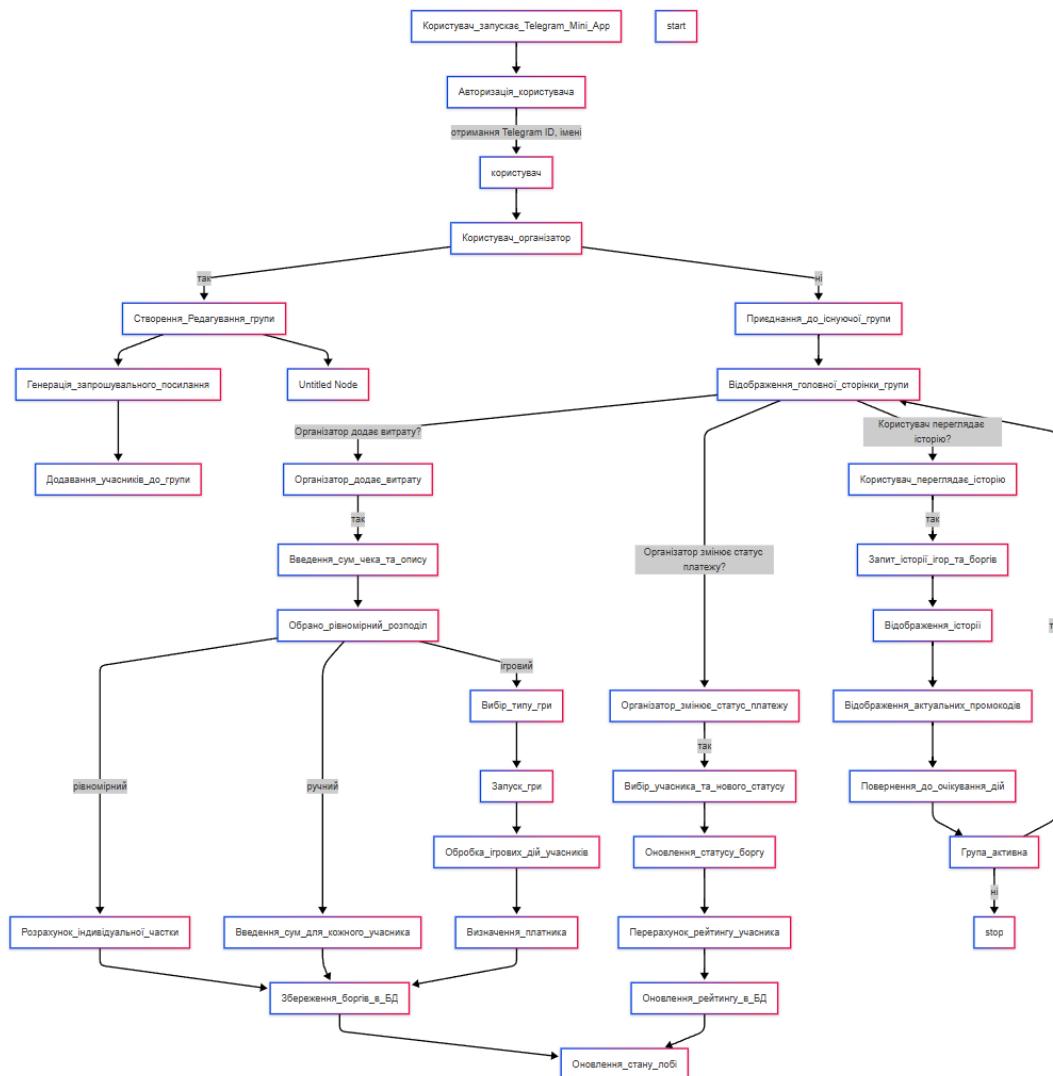


Рисунок Б.1 — Загальна flow діаграма процесів ІС

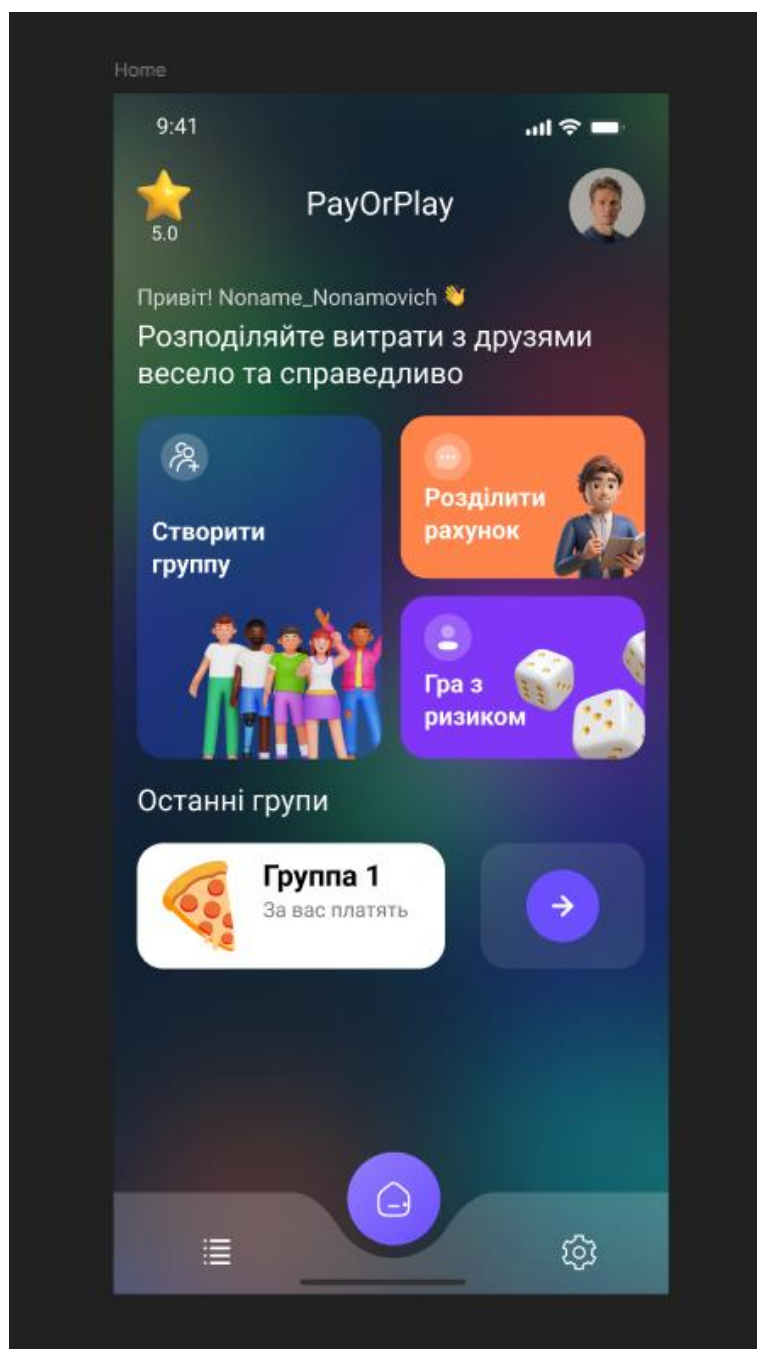


Рисунок В.1 – Прототип головного екрану

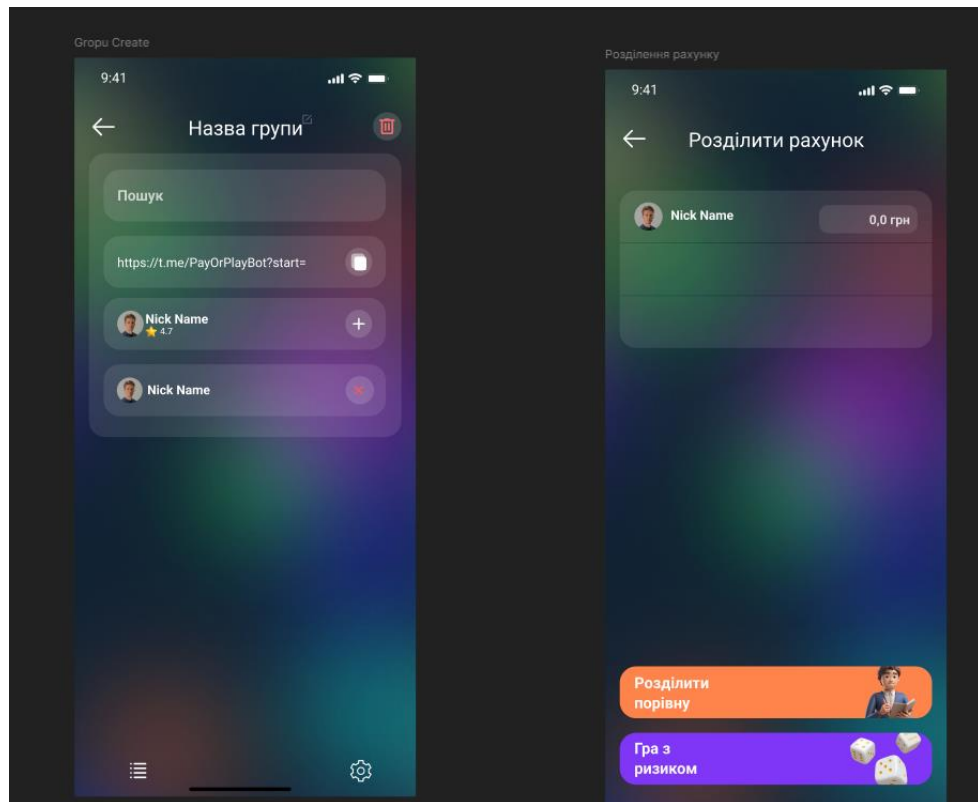


Рисунок В.2 – Прототипи створення групи та керування нею

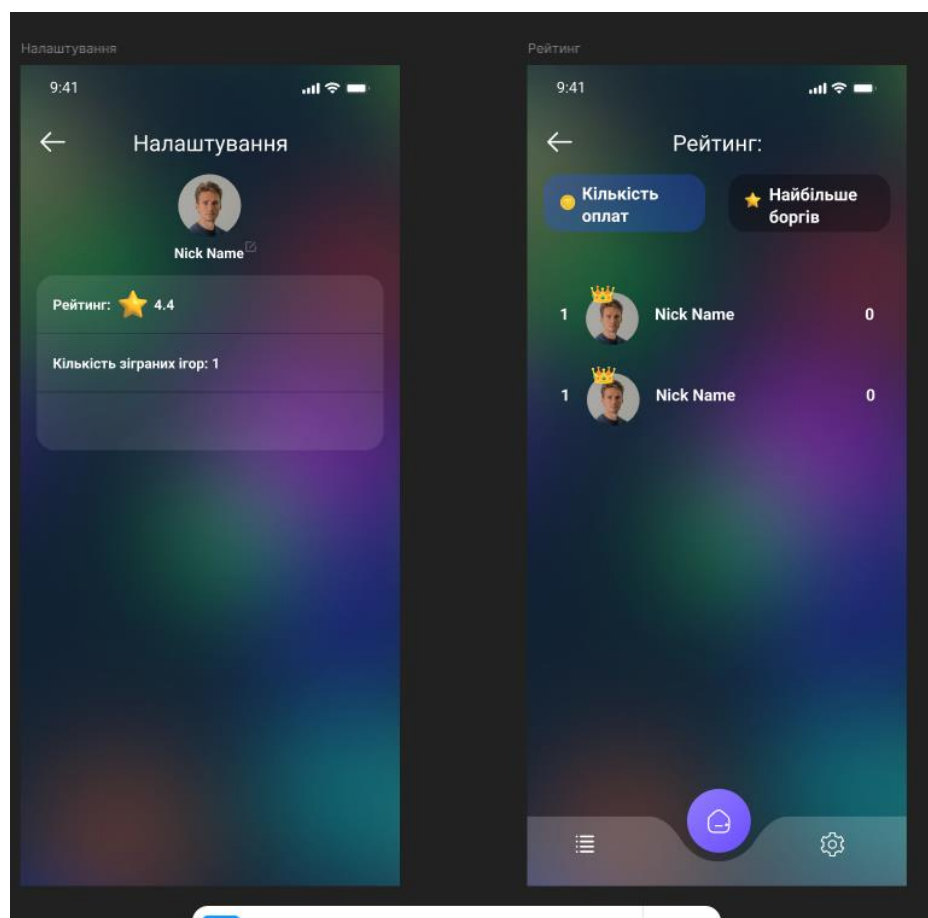


Рисунок В.3 – Прототипи налаштувань та рейтингу

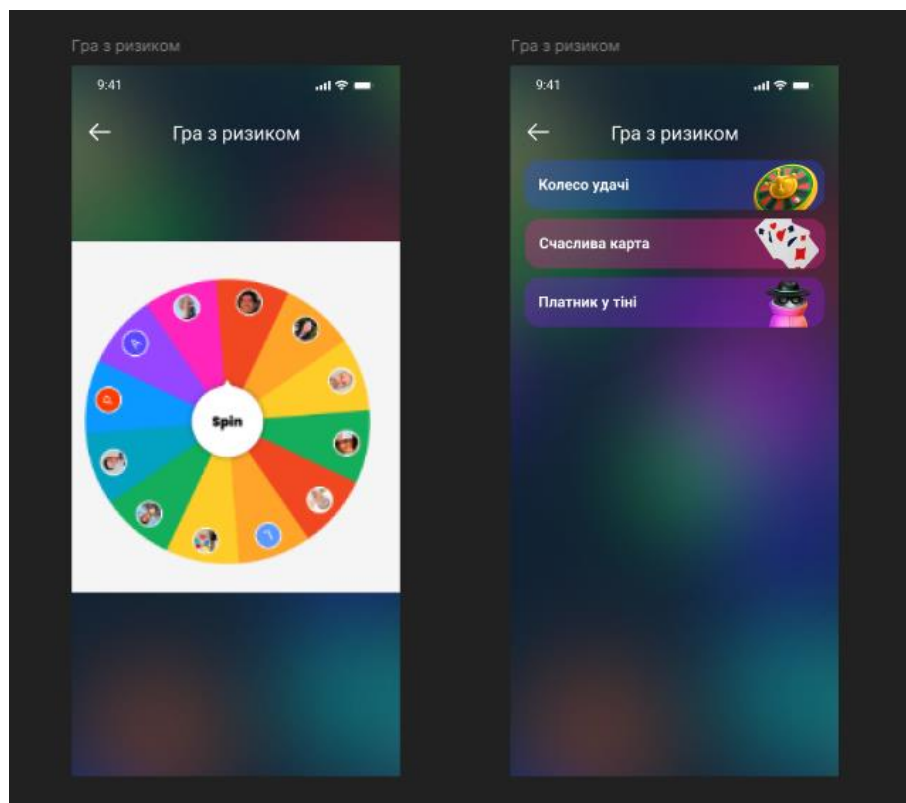


Рисунок В.4 – Прототип меню ігор

Додаток Д1. Фрагменти програмного коду «Колекція USERS»

```
// === 1. Колекція users ===
db.users.insertMany([
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"),
    "telegramId": "tg_user_1",
    "name": "Іван Петров",
    "rating": 750,
    "createdAt": ISODate("2024-05-15T10:00:00.000Z"),
    "updatedAt": ISODate("2024-06-10T11:30:00.000Z")
  },
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"),
    "telegramId": "tg_user_2",
    "name": "Олена Сидорова",
    "rating": 680,
    "createdAt": ISODate("2024-05-16T11:00:00.000Z"),
    "updatedAt": ISODate("2024-06-10T12:00:00.000Z")
  },
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7c2"),
    "telegramId": "tg_user_3",
    "name": "Микола Коваленко",
    "rating": 550,
    "createdAt": ISODate("2024-05-17T12:00:00.000Z"),
    "updatedAt": ISODate("2024-06-10T12:05:00.000Z")
  }
]);

// === 2. Колекція lobbies ===
db.lobbies.insertMany([
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
    "name": "Дружня Група",
    "adminId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"), // Іван Петров
    "inviteLink": "payorplay.app/invite/xyz789",
    "createdAt": ISODate("2024-05-20T09:00:00.000Z"),
    "updatedAt": ISODate("2024-06-09T14:15:00.000Z")
  }
]);

// === 3. Колекція participants ===
db.participants.insertMany([
  {
    "lobbyId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
    "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"),
    "role": "admin"
  },
  {
```

```

        "lobbyId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
        "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"),
        "role": "member"
    },
    {
        "lobbyId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
        "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7c2"),
        "role": "member"
    }
];

```

// === 4. Колекція expenses ===

```

db.expenses.insertMany([
    {
        "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7bb"),
        "groupId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
        "amount": 150.75,
        "description": "Вечеря в ресторані",
        "splitType": "equal",
        "createdAt": ISODate("2024-06-05T19:45:00.000Z")
    },
    {
        "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7c3"),
        "groupId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
        "amount": 75.00,
        "description": "Квитки в кіно",
        "splitType": "manual",
        "createdAt": ISODate("2024-06-06T14:00:00.000Z")
    }
]);

```

// === 5. Колекція debts ===

```

db.debts.insertMany([
    {
        "debtorId": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"), // Олена винна Івану
        "expenseId": ObjectId("6666a7b7a1b2c3d4e5f6a7bb"),
        "amount": 50.25, // 150.75 / 3 учасники = 50.25
        "status": "pending",
        "owedToUserId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"),
        "createdAt": ISODate("2024-06-05T19:45:00.000Z"),
        "updatedAt": ISODate("2024-06-05T19:45:00.000Z")
    },
    {
        "debtorId": ObjectId("6666a7b7a1b2c3d4e5f6a7c2"), // Микола винен Івану
        "expenseId": ObjectId("6666a7b7a1b2c3d4e5f6a7bb"),
        "amount": 50.25,
        "status": "pending",
        "owedToUserId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"),
        "createdAt": ISODate("2024-06-05T19:45:00.000Z"),
        "updatedAt": ISODate("2024-06-05T19:45:00.000Z")
    }
]);

```

```
"debtorId": ObjectId("6666a7b7a1b2c3d4e5f6a7c2"), // Микола винен за кіно
"expenseId": ObjectId("6666a7b7a1b2c3d4e5f6a7c3"),
"amount": 75.00,
"status": "pending",
"owedToUserId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"), // Припустимо, Іван платив за всіх
"createdAt": ISODate("2024-06-06T14:00:00.000Z"),
"updatedAt": ISODate("2024-06-06T14:00:00.000Z")
}
]);
```

```
// === 6. Колекція games ===
db.games.insertMany([
{
  "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7be"),
  "groupId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
  "gameType": "RLT", // Рулетка
  "createdAt": ISODate("2024-06-08T15:00:00.000Z"),
  "endedAt": ISODate("2024-06-08T15:05:00.000Z")
},
{
  "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7c4"),
  "groupId": ObjectId("6666a7b7a1b2c3d4e5f6a7b9"),
  "gameType": "LCK", // Щаслива карта
  "createdAt": ISODate("2024-06-09T18:00:00.000Z"),
  "endedAt": ISODate("2024-06-09T18:10:00.000Z")
}
]);
```

```
// === 7. Колекція game_participants ===
db.game_participants.insertMany([
{
  "gameId": ObjectId("6666a7b7a1b2c3d4e5f6a7be"), // Рулетка
  "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"),
  "role": "player",
  "gameSpecificData": { "outcome": "pay_all" } // Іван платить за всіх
},
{
  "gameId": ObjectId("6666a7b7a1b2c3d4e5f6a7be"), // Рулетка
  "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"),
  "role": "player",
  "gameSpecificData": { "outcome": "free" }
},
{
  "gameId": ObjectId("6666a7b7a1b2c3d4e5f6a7c4"), // Щаслива карта
  "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"),
  "role": "player",
  "gameSpecificData": { "card": "card_free" } // Олена звільнена від оплати
},
{
  "gameId": ObjectId("6666a7b7a1b2c3d4e5f6a7c4"), // Щаслива карта
  "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7c2"),
  "role": "player",

```

```
    "gameSpecificData": { "card": "card_pay_neighbor" } // Микола платить за сусіда
  }
  });
```

```
// === 8. Колекція coupons ===
```

```
db.coupons.insertMany([
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7c0"),
    "code": "MC_DISCOUNT_50",
    "ratingLevel": "high",
    "description": "Знижка 50% у McDonald's",
    "expiresAt": ISODate("2024-12-31T23:59:59.000Z"),
    "type": "discount"
  },
  {
    "_id": ObjectId("6666a7b7a1b2c3d4e5f6a7c5"),
    "code": "COFFEE_PROMO_10",
    "ratingLevel": "mid",
    "description": "Знижка 10% на каву",
    "expiresAt": ISODate("2024-09-30T23:59:59.000Z"),
    "type": "discount"
  }
]);
```

```
// === 9. Колекція user_coupons ===
```

```
db.user_coupons.insertMany([
  {
    "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7b8"), // Іван
    "couponId": ObjectId("6666a7b7a1b2c3d4e5f6a7c0"),
    "assignedAt": ISODate("2024-06-10T10:00:00.000Z"),
    "isActive": true
  },
  {
    "userId": ObjectId("6666a7b7a1b2c3d4e5f6a7bd"), // Олена
    "couponId": ObjectId("6666a7b7a1b2c3d4e5f6a7c5"),
    "assignedAt": ISODate("2024-06-10T10:05:00.000Z"),
    "isActive": true
  }
]);
```

Дата звіту: 6/13/2025

Дата редагування: ---

Звіт не був оцінений

Звіт подібності

Метадані

Назва організації:
Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок:
Проектування інтерактивного застосунку на базі месенджера для розумного розподілу витрат у групі друзів

Автор: Науковий керівник / Експерт
Жовня Антон Олександрович Ріппа С.П.

Підрозділ:
кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності означає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

2.99% 2.98% КП 1 25 Довжина фрази для коефіцієнта подібності 2	0.68% 0.63% КП 2 16984 Кількість слів	5.10% 5.10% КЦ 130687 Кількість символів
--	---	--

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		10
Білі знаки		0
Парафрази (SmartMarks)		32

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту
ПОСЛІДОВНИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ЦЕНТРИЧНИХ СЛІВ (ТРАМЕНИВ)
1	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-f2fdb8c10ff/download	43 0.25 %
2	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-f2fdb8c10ff/download	43 0.25 %
3	https://uchika.in.ua/metodichni-rekomendaciyi-shodo-vikonannya-vipusknoho-bakalavrs.html?page=2	29 0.17 %
4	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	24 0.14 %
5	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	21 0.12 %

6	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-ff2fbd8c10ff/download	20 0.12 %
7	https://ir.kneu.edu.ua/server/api/core/bitstreams/38362e99-7f5f-449c-879b-cb24b773a81d/content	16 0.09 %
8	https://ir.kneu.edu.ua/server/api/core/bitstreams/38362e99-7f5f-449c-879b-cb24b773a81d/content	16 0.09 %
9	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	16 0.09 %
10	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	16 0.09 %
з бази даних RefBooks (0.00 %)		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
з домашньої бази даних (0.13 %)		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка дистанційного курсу з іноземної мови 6/11/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	16 (2) 0.09 %
2	Автоматизована система для аналізу даних на фондовому ринку 6/26/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	6 (1) 0.04 %
з програми обміну базами даних (0.06 %)		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Марченко РПЗ-211 курсовий БД 6/18/2024 Kamianets-Podilskyi vocational college of Industry, Business and Information Technology (Kamianets-Podilskyi vocational college of Industry, Business and Information Technology)	10 (1) 0.06 %
з Інтернету (2.80 %)		
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-ff2fbd8c10ff/download	126 (6) 0.74 %
2	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	89 (5) 0.52 %
3	https://ir.kneu.edu.ua/server/api/core/bitstreams/38362e99-7f5f-449c-879b-cb24b773a81d/content	58 (4) 0.34 %
4	https://uchika.in.ua/metodichni-rekomendaciyi-shodo-vikonannya-vipusknogo-bakalavrs.html?page=2	47 (3) 0.28 %
5	https://kneu.edu.ua/userfiles/Faculty_of_Information_Systems_and_Technology/kaf%20ise/Metod_rekom_en_6_050101.doc	47 (5) 0.28 %
6	https://ir.kneu.edu.ua/bitstreams/d05c67bf-2a21-410b-9760-85452cede4b0/download	25 (3) 0.15 %
7	https://uchika.in.ua/metodichni-rekomendaciyi-shodo-vikonannya-magistersekoyi-diplo.html?page=3	23 (3) 0.14 %
8	https://ir.kneu.edu.ua/bitstreams/4559c31c-9da2-4bfa-b1bd-f336a89d8c4b/download	20 (2) 0.12 %
9	https://neoclima.ua/ua/09ahew/	10 (1) 0.06 %
10	https://ir.kneu.edu.ua/bitstreams/40f69492-bf2d-4dcb-b3e1-7da1833a5b34/download	8 (1) 0.05 %