КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю

121 Інженерія програмного забезпечення: програмна інженерія

на тему:

РОЗРОБКА ВЕБ ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ АНОНІМНИХ ДИСКУСІЙ

Виконав студент 4-го курсу
Станіслав ДОМБРОВСЬКИЙ _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Лариса КАТЕРИНИЧ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри інтелектуальних програмних систем
«25» травня 2022 р.,
Протокол № 10
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 30 сторінок, 8 використаних джерел, 7 рисунків, 2 таблиці.

Ключові слова: веб додаток, фреймворк, сервіс, Java, Spring Boot, WebSocket, gRPC, Protobuf, PostgreSQL.

Об’єктом роботи є процес передачі інформації від клієнта – серверу, з сервера до незалежного сервісу який підтримує всі поставлені функції та зворотній процес передачі інформації.

Предметом роботи є веб додаток, який містить веб сервіс, логін сервіс та веб сервіс які написані мовою Java за допомогою фреймворків Spring Boot, gRPC та протоколів WebSocket та Protobuf.

Метою випускної кваліфікаційної роботи є створення веб додатку який забезпечував би проведення обговорення стосовно вибраного питання та який зберігав би анонімність опонентів.

Інструментом розробки обрано IntelliJ IDEA – інтегроване середовище розробки від компанії JetBrains. Дане середовище містить всі необхідні інструменти для створення проектів. Відлагоджувач, інструментарій для роботи з Git, підсвічування певного синтаксису, засоби для рефакторингу коду.

Результати роботи: під час виконання даної випускної кваліфікаційної роботи було розглянуто та досліджено засоби для розробки додатку, опрацьовано інформацію про роботу з базами даних. У ході написання випускної кваліфікаційної роботи підготовлено практичну частину: веб додаток.

Програмний продукт, основу додатку, можна використовувати в майбутньому, розширюючи та доукомплектовуючи базу даних питань та функціонал самого проекту. Додаток буде корисним для багатьох користувачів, оскільки він розкриває в повній мірі досить актуальну задачу.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ПЗ — сукупність програм системи оброблення інформації та програмних документів, необхідних для експлуатації цих програм.

HTTP – протокол передачі даних, що використовується в комп'ютерних мережах.

TCP – протокол транспортного рівня в комп'ютерних мережах.

GET-запит – запит між клієнтом та сервером який запитує дані з вказаного ресурсу.

UTF-8 – кодування яке реалізовує представлення набору Юнікод та сумісне з 8-бітовим кодуванням тексту.

ООП – об'єктно-орієнтоване програмування.

БД – певний набір даних, організованих відповідно до концепції, яка описує характеристику цих даних та взаємозв'язки між елементами.

СУБД – система управління базами даних.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Із розвитком сучасних технологій, все більше людей мають смартфон, планшет чи ноутбук з доступом до інтернету. Мало хто взагалі зараз може уявити своє життя без інтернету та без онлайн послуг, користування якими на даний момент досить полегшують, замінюючи звичні процедури як дзвінок в службу підтримки для вирішення будь-якої проблеми, чи отримання актуальної інформації про товар на проведення цих самих процедур через онлайн чати, за допомогою яких можна з легкістю описати свою проблему більш зрозуміло та отримати письмово відповідь в реальному часі що досить суттєво економить час. Через це багато компаній, випускаючи свій онлайн продукт, ставить собі в плани створення чату, щоб зробити простішим користування своїм клієнтам. Так як онлайн чати стали досить популярними, багато сьогоденних сфер занять підлаштовують під них. Підготовка до дебатів є саме тим об’єктом, який не змінить свої властивості при переході з стандартного нам обміну думками між двома сторонами використовуючи зал в реальному житті, в щось інше як обмін думками між двома сторонами використовуючи онлайн чат.

Актуальність роботи та підстави для її виконання. Сфера застосування додатку для проведення анонімних дискусій є досить корисною для різних груп людей, чи це команди які мають регулярно готуватись до дебатів, чи адвокати які мають підтримувати свою форму, чи просто користувачі які вирішили витратити вільний час та подискутувати на вибрану тему з невідомим користувачем, так як дебати передбачають чесність, рівність можливостей, повагу до опонента і толерантність.

На даний момент це є досить актуально в світі, враховуючи що з появою коронавірусу є чимало обмежень на проведення певних заходів чи перебування в людному місці, перехід дебатів в онлайн може не тільки зберегти а й прибавити інтерес людей до даної сфери.

Мета й завдання роботи. Дивлячись на сучасні тенденції та розвиток технологій, метою даної дипломної роботи є створення веб додатку який забезпечував би проведення обговорення стосовно вибраного питання та який зберігав би анонімність опонентів. Користувач сам вибирає відповідь на запропоноване питання та очікує на опонента з протилежною відповіддю, це забезпечить кожному користувачеві рівні можливості вибираючи ту сторону, яка є йому ближча. Для досягнення мети поставлено такі завдання:

- Дослідження наявних засобів та технологій для розробки веб додатку;
- Аналіз відомих проектів – аналогів;
- Вибір технологій для реалізації додатку;
- Проектування архітектури проекту;
- Реалізація сервісу для коректного підбору питань та створення чатів на основі відповідей;
- Реалізація сервісу для коректної роботи чатів;

Об’єкт, методи й засоби дослідження та розроблення. Розробці практичної частини передувало ознайомлення із створенням веб додатків та концепціями створення сервісів для підтримки чатів, його структуризації та аналізу додаткових допоміжних бібліотек та служб.

Об’єкт дослідження – процес передачі інформації від клієнта – серверу, з сервера до незалежного сервісу який підтримує всі поставлені функції та зворотній процес передачі інформації.

Методи дослідження – методи побудови клієнт-серверних додатків, методи створення сервісів, побудова UML-діаграм, методи ООП.

Під час розробки веб додатку були враховані тенденції та сучасні рекомендації, які можна знайти на веб ресурсах для розробників. Зокрема, опрацьовано статті, в яких зроблена оцінка щодо популярності та доцільності використання тої чи іншої системи для контактування веб додатку з іншим

сервісом, що забезпечував би повний функціонал, зв’язок з іншими клієнтами та зв’язок з БД.

Інструментами для створення веб додатку обрано такі засоби як мову програмування Java, фреймворк Spring Boot, протокол WebSocket для передачі інформації між браузером та сервером, фреймворк для віддаленого виклику процедур gRPC, протокол передачі даних Protobuf та базу даних PostgreSQL.

Можливі сфери застосування. На основі реалізованої практичної частини випускної кваліфікаційної роботи клієнт має можливість дати відповідь на запропоновані питання, у разі знаходження опонента з протилежною відповіддю, для клієнтів формується чат в якому клієнти дискутують на відповідну тему, притримуючись своєї відповіді на питання, при цьому не знаючи інформації про опонента, додаток забезпечує анонімність користувачам. Програмний продукт, основу додатку для проведення анонімних дискусій, можна використовувати в майбутньому, розширюючи та доукомплектовуючи функціонал. Додаток буде корисний багатьом категоріям користувачів, чи для команд які мають регулярно готуватись до дебатів, чи адвокатам які мають підтримувати свою форму, чи просто користувачам які вирішили витратити вільний час та подискутувати на вибрану тему з невідомим користувачем, адже додаток допомагає працювати над своїми слабкими місцями, навчаючись вмінню дискутувати та притримуватись своєї позиції.

Додаток написаний для коректної роботи в браузері на пристроях які мають можливість під’єднання до інтернету, тому використовувати його можна не тільки на смартфонах, планшетах, але і на пристроях.

РОЗДІЛ 1. ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ ВЕБ ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ АНОНІМНИХ ДИСКУСІЙ

1.1 Аналіз тематики веб додатку

Дискусії як публічний спір має за мету з’ясувати і зіставити різні точки зору, виявити спільну думку, знайти правильне рішення в спільному питанні. І це є надзвичайно актуально на сьогоднішній день, враховуючи що часто більшість проблем появляються через те що сторони переслідують далеко не однакові цілі, керуються різними мотивами, вирішення проблем в сьогоденні можливе тільки при відкритому обміні думками, громадської згоди.

Часто дискусії мають різне за мету. Це може бути дискусія заради отримання істини, під час якої учасники дискусії можуть отримати насолоду а при кінці людина почуває себе піднесено і краще. Чи це дискусія заради переконання кого-небуть, одні можуть вважати що відстоюють праве діло, захищають суспільні інтереси, а інші роблять це заради перемоги – самоствердження. Чи можливо для когось це дискусія заради дискусії, для таких людей байдуже про що сперечатись, якщо заперечувати їхню позицію, вони точно будуть її захищати. Більше того, всі ці типи дискусій завжди були і будуть в нашому житті. Хто б що не говорив, світ без суперечок не може існувати, побажання до уникнення всіх суперечок просто нездійсненне. Суперечки просто необхідні, тому що вони є одною з невід’ємних особливостей спілкування між людьми і досягнення взаєморозуміння.

Тому потрібно правильно вміти дискутувати, потрібно пробувати зрозуміти мотиви дій опонента та його висловлювання, вміти визначати манеру сперечатися, та правильно порівнювати свої здібності й можливості з силами суперника.

Саме тому було вибрано як мету моєї бакалаврської кваліфікаційної роботи створити веб додаток який би надавав можливість користувачам приймати участь в дискусіях, при цьому навчаючись відстоювати свою думку та

слухати опонента. Для дискусій було вибрано розробити додаток так, щоб користувачі були рівноправні, щоб не було сценарію де потрібно було б відстоювати ту думку, якої користувач не притримується, оскільки дискусія може закінчитись результативно, якщо розбиратись в предметній області і в самій позиції.

А в якості інструмента веб додатка для звичного ведення дискусій було обрано використовувати чат. І це не дивно, оскільки чати мають досить великий вплив на користування інтернетом користувачами та на розробку веб додатків. В сьогоденні чати зустрічаються на кожному кроці, чи це месенджер де ми спілкуємось з близькими та друзями, чи чати на онлайн трансляціях де не дві а безліч людей можуть обговорити певне питання, чи чат-боти які зустрічаються на всіх популярних сайтах та мобільних додатках.

Беручи до уваги інтернет спілкування, в нього є достатньо і плюсів і мінусів, та можливість швидкого інформування важливих людей, можливість встановлювати нові знайомства, обмін знаннями та діалог з людьми які перебувають в іншій країні чи на іншому континенті перекривають всі мінуси.

1.2 Огляд додатків аналогів

Як уже було сказано, за мету було поставлено розробити такий сервіс, який приносив би користь його користувачам придумуючи дієві аргументи, але також додаток дозволяв би користувачам поспілкуватись з опонентами, та який не загружав би повністю людину, при цьому можна було б провести деякі вільні проміжки часу в додатку.

Якщо говорити про веб сервіси які надають схожий функціонал поодинці, їх є достатньо багато. Це є й різні навчальні платформи які запропонують дуже багато тем з якими можна ознайомитись і навчитись чого нового, і форуми на яких можна знайти потрібні відповіді на проблеми чи на турбуюче питання. І навіть месенджери та соціальні мережі які стали надзвичайно популярними через свої можливості як засіб для комунікацій та слідкуванням за знайомими,

засіб для отримання актуальних новин.

При дослідженні інтернет ресурсів, не було знайдено прямих аналогів даного веб додатку, та присутні проекти, які мають схожий функціонал, або схожу мету.

Kialo Edu – проект, основна мета якого навчатись критичного мислення, та якщо поглянути на процес навчання, він полягає в тому, що клієнту надається певне питання, на яке він має вибрати зі списку найбільш підходяще твердження яке виступає за одну з сторін: заперечення або захист. При цьому користувачі можуть обговорювати уже запропоновані твердження, дискутувати про них, та оцінювати інших користувачів.

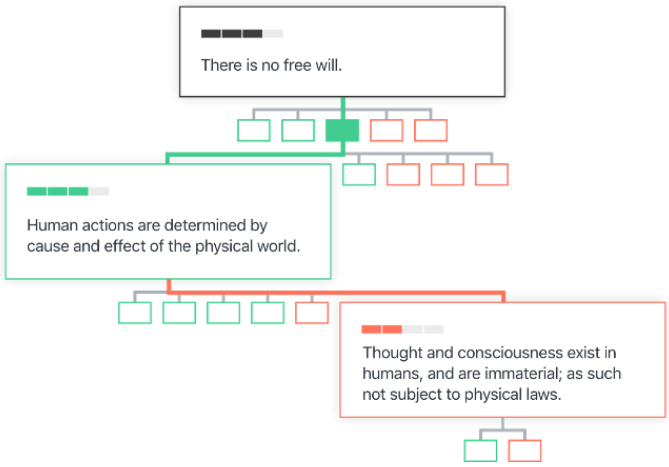

Рис. 1.2.1 – Приклад інтерфейсу проекту Kialo Edu

DebateIsland Education – проект, який спрямований на те, щоб допомогти студентам отримати більше інформації про різні теми, одночасно розвиваючи критичне мислення та навички дискусії. Викладачі створюють нові теми, в яких студенти, після проходження цього на уроці, обговорюють тему яка висвітлена.

У викладачів є можливість викласти новий матеріал для проходження, створення нової теми для обговорення чи створення завдань з можливістю вибору правильної відповіді.

Також проект містить інструменти штучного інтелекту які аналізують аргументи користувачів та прогнозують переможців дебатів.

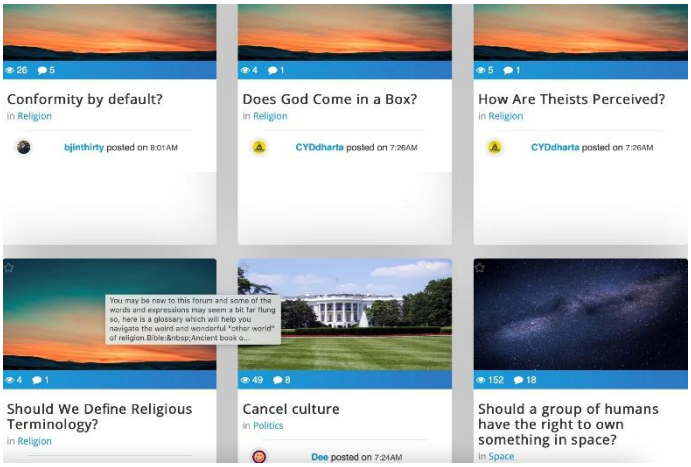

Рис. 1.2.2 – Приклад інтерфейсу проекту DebateIsland Education

Таблиця 1.2.1 – переваги та недоліки існуючих додатків для проведення дискусій

Додаток	Переваги	Недоліки
Kialo Edu	Незвичний - зручний інтерфейс для дискусій. Можливість оцінювати користувачів. Присутні уже існуючі твердження.	Дискусії повинні створювати самі користувачі. Користувачі мають самі долучатись до дискусій.
DebateIsland Education	Багатий функціонал. Присутність штучного інтелекту.	Для роботи потрібен викладач. Обмежена сфера застосування

1.3 Вимоги та задачі веб додатку

Після огляду додатків аналогів та аналізу їхнього функціоналу, було поставлено за мету створити веб додаток який би виправляв всі мінуси які були знайдені, мав би особливості які не є притаманними розгнянутим додаткам, та виконував би основні задачі:

- Додаток має бути простим для користувача.

- Інтерфейс має бути зручним та зрозумілим.
- Застосунок має сам пропонувати користувачам питання для дискусій без швидкого повторення.
- Додаток має забезпечувати рівність користувачів, клієнти самі вибирають сторону яку будуть захищати.
- Сервіс має забезпечувати щоб в чат був створений для користувачів з протилежними відповідями на питання.
- Веб додаток має забезпечувати анонімність клієнтів.
- Сервіс має бути безкоштовним для користувачів.
- Додаток має зберігати історію існуючих чатів.
- Додаток має приносити тільки користь та нові знання для користувачів.
- Додаток має мати високу швидкодію.

РОЗДІЛ 2. ЗАСОБИ ДЛЯ СТВОРЕННЯ ВЕБ ДОДАТКУ

2.1 Особливості розробки використовуючи Spring Boot

На даний момент, фреймворк Spring Boot є одним з найбільш популярним серед розробників програм які використовують мову програмування Java. Він складається з багатьох модулів, що забезпечує потреби різних сфер.

Spring Boot – фреймворк на основі мови програмування Java з відкритим кодом що забезпечує автоматизацію конфігурації додатку, який пришвидшує процес створення та запуску програми.

Spring Boot контролює управління життєвого циклу об’єктів Java та дозволяє спростити процес створення веб додатків на основі мови програмування Java.

Функціонал фреймворку достатньо великий, та найбільш корисними особливостями які використовуються є процес контролю залежностей в програмі, автоматизація конфігурації.

Для пришвидшення процесу контролю залежностей, фреймворк неявно упаковує необхідні сторонні залежності для різних типів програм на основі Spring і надає розробникам їх в зручному вигляді, надаючи starter-пакети які уже містять потрібні залежності. Для прикладу starter-пакет spring-boot-starter-web містить всі потрібні бібліотеки для створення, та швидкого запуску веб додатку, такі як spring-webvc, Jackson-json, Tomcat та інші.

Рис. 2.1.1 – Логотип фреймворку Spring Boot

В жовтні 2012 році було створено запит на додаткову функціональність у

Spring Марком Янгстремом який просив додати підтримку створення без контейнерних веб додатків в Spring. Саме цей запит дав старт розробці проекту Spring Boot яка почалась на початку 2013 року. Скоро було випущено першу версію фреймворку.

Простота Spring Boot дозволяє розробникам Java з легкістю розробляти веб додатки на основі REST архітектури, розгортати їх в докер контейнерах та швидкого прототипування проектів.

2.2 Огляд протоколу WebSocket

Дивлячись на задачі, які були поставлені для виконання роблячи даний застосунок можна зрозуміти, що головним інструментом який буде запропонований клієнту для дискусій є чат. В інтернеті можна знайти достатньо багато варіантів для реалізації робочого чату з отриманням повідомлень в реальному часі двом клієнтам зберігаючи всі атрибути повідомлення, збереження історії повідомлень та відображення її при кожному відкритті чату, та мною був вибраний саме протокол WebSocket який використовується як засіб для передачі інформації від клієнта на сервер та передачі інформації від сервера на браузер клієнту.

WebSocket був розроблений як протокол що задовільняв повну асинхронність в стандартному всім синхронному протоколі HTTP. Та що ж такого кардинально змінила поява WebSocket? Це питання можливо появляється у багатьох, хто перший раз чує даний протокол. Розбираючи протокол HTTP, можна сказати що це найбільш вчасно вживаний протокол передачі даних, який використовується та буде використовуватись в комп'ютерних мережах. Це протокол який працює для передачі гіпертекстових документів, та основна проблема в тому, що цей протокол працює як модель запит – відповідь, для кожного запиту, який був відправлений чи то від сервера, чи то від клієнта, має прийти відповідь, позитивна або негативна. Така модель використовується досить часто в протоколах, оскільки вона є зрозумілою, буденною, та беручи до

уваги чат – сервіси, потрібно щось інше, оскільки нам потрібно отримати повідомлення від співрозмовника як тільки він його відправив. Тут потрібна саме та модель, якою є WebSocket.

Як було сказано раніше, за допомогою WebSocket, протокол HTTP стає асинхронним, це виконується через те що WebSocket працює як потік повідомлень. Між клієнтом та сервером відкривається стрім який є рівноправним, через нього може проходити повідомлення чи з сервера до клієнта чи від клієнта до сервера не отримуючи при цьому зворотню відповідь. Не потрібно нічого чекати і слухати, друга сторона може відправити відповідь зразу а може й взагалі нічого не прислати, та при цьому, сторона отримувач буде оповіщена як тільки їй прийде новий потік даних. І саме це те, що потрібно для реалізації чат – сервісів, саме цього потребували розробники.

Протокол WebSocket можна використати в процесі користування веб додатком, для початку, браузер підключається до сервера на 80 порт використовуючи протокол TCP та по ньому відправляється стандартний GET-запит, правда з проханням використати WebSocket, у відповідь отримує відповідь в якій сервер або підтверджує використання даного протоколу або відмовляє. Перевага даного протоколу в тому, що в подальшому клієнт чи сервер відправляє використовуючи дане TCP з'єднання певну послідовність байт, в якій міститься тільки повідомлення, не включаючи заголовки та метадані.

Таблиця 2.2.1 – структура повідомлення використовуючи протокол WebSocket

0x00	повідомлення	0xFF

Тобто передається проста послідовність байт, де першим елементом іде нульовий байт – 0x00, останнім елементом іде байт – 0xFF, а поміж ними саме повідомлення закодоване використовуючи UTF-8.

Чудовою відмінністю WebSocket від протоколу HTTP є те, що веб-сокети не мають часу життя, вони можуть нескінченно висіти в очікуванні на

повідомлення від клієнта чи сервера, при цьому не споживаючи ресурси.

2.3 Особливості розробки використовуючи gRPC

Беручи до уваги представлені раніше фреймворки та протоколи, може скластись бачення як правильно побудувати клієнт – серверний додаток, додатково використовуючи протокол WebSocket для отримання актуальної інформації для чатів, та постає питання як ще зробити сервер так, щоб він отримував актуальну інформацію від інших клієнтів, підбирав опонента та формував правильно новий чат. В сьогоденні досить часто використовується мікросервісна архітектура для побудови додатків, це забезпечує і незалежне виконання поставлених задач окремими сервісами, і швидкий старт сервісів, і можливість додавання змін в окремий сервіс не оновлюючи функціонал інших. Дивлячись на широкий список плюсів, було вирішено створити окремий сервіс для вирішення багатьох проблем, та протоколом для з'єднання зовнішнього сервісу та клієнтського сервера було вибрано gRPC.

Фреймворк gRPC – система віддаленого виклику процедур яка була розроблена компанією Google у 2015 році як наступне покоління інфраструктури RPC Stubby. gRPC використовує транспортний протокол HTTP/2 для передачі даних та мову опису інтерфейсів Protobuf для опису всіх повідомлень які будуть передаватись від клієнта серверу чи від сервера клієнту використовуючи gRPC через методи які будуть викликані клієнтом.

gRPC працює з багатьма мовами програмування, що дає змогу використовувати його для зв'язку між клієнтом та сервером які написані на різних мовах. C#, C++, Dart, Go, Java, Kotlin, Node, Objective-C, PHP, Python, Ruby, Rust – список мов які підтримуються, та його можна легко доповнити тими мовами, для яких написані неофіційні бібліотеки для роботи з gRPC.

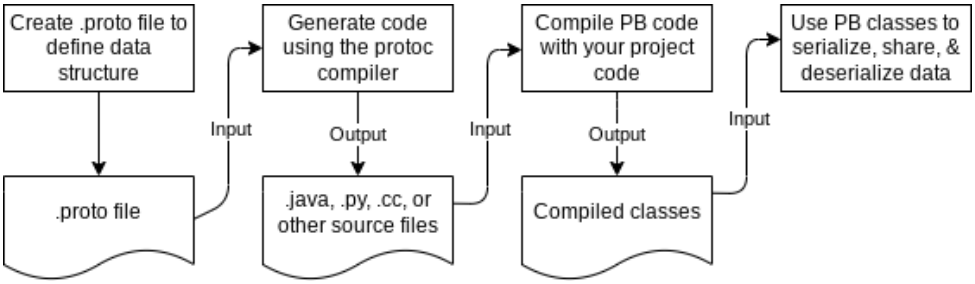

Рис. 2.3.1 – Схема роботи з Protobuf

На рисунку 2.3.1 зображено схему роботи з протоколом Protobuf. Для початку потрібно створити потрібні файли формату .proto в яких буде описаний сервіс з методами які будуть запропоновані клієнту та повідомлення які будуть використані в даних методах. Наступним етапом є генерація класів для заданої мови програмування використовуючи компілятор protoc. Далі протобафний код компілюється разом із кодом програми. Та останнім етапом є використання класів згенерованих на основі Protobuf для серіалізації та десеріалізації даних які будуть передані через протокол який підтримує мову Protobuf. Для мого прикладу даним протоколом є gRPC.

Для з’єднання з сервером використовується канали (channel) через які клієнт відкриває синхронні чи асинхронні стаби (stub), через які буде викликати потрібні методи.

Для реалізації веб додатку досліджено та використано офіційну бібліотеку для використання gRPC в мові програмування Java. Надано саме ті можливості, щоб реалізувати всі поставлені цілі, для цього присутні і класи які будуть перехоплювати повідомлення перед відправкою клієнту та серверу для додавання потрібних заголовків, і класи які будуть реалізовувати ті методи, які описані за допомогою Protobuf, і додаткові опції, для прикладу keep-alive, які покращують роботу додатку.

Для опису методів, які будуть викликані клієнтом, та повідомлень які будуть пересилатися використовується Protocol Buffers – формат серіалізації який gRPC використовує по замовчуванню. Protobuf задає строгу типізацію всіх полів, типів даних, та повідомлень які будуть передані. Основною специфікою

формату Protobuf є те, що всі дані які планує пересилати клієнт чи сервер, Protobuf зберігає та передає в бінарному вигляді, де дані передаються в бінарному вигляді та ключі полів це числові індекси а не імена.

Початковою точкою в Protobuf є сервіси які описують всі методи які будуть запропоновані клієнту. Сервіс має своє ім’я та список методів, які записуються з ключовим словом rpc, що ідентифікує їх як метод та вхідне і вихідне повідомлення. Методами можуть виступати як прості синхронні методи з запитом та відповіддю а також стріми. Стріми бувають трьох типів: клієнтські стріми, серверні стріми та двонаправлені стріми.

Клієнтські стріми мають такий самий вигляд як простий виклик функції та має одну особливість, клієнт може надсилати потік повідомлень на сервер замість одного, сервер відповідає одним повідомленням. Клієнт відкриває стрім та одразу отримує у відповідь повідомлення, після цього клієнт може надсилати скільки хоче повідомлень до тих пір поки з’єднання не закриється. Його може закрити клієнт, сервер, чи через обрив з’єднання.

Аналогічними є серверні стріми, та клієнт надсилає запит на відкривання стріма, при цьому надсилаючи повідомлення і після цього стрім буде відкритий та сервер може надсилати повідомлення.

Двонаправлений стрім працює аналогічно як клієнтський та серверний стрім та безліч повідомлень може надсилати і клієнт і сервер.

```
service GrpcService {
   //простий реквест
   rpc request(Request) returns (Respond);
   //Клієнтський стрім
   rpc clientStream(stream Request) returns (Respond);
   //Серверний стрім
   rpc serverStream(Request) returns (stream Respond);
   //Двонаправлений стрім
   rpc bidirectionalStream(stream Request) returns (stream Respond);
}
```

Для задання повідомлень використовується тип message, в який поміщаються всі потрібні атрибути простого типу, вкладені повідомлення, комплексні типи map виду ключ – значення або ж атрибути типу one of для яких задано перелік можливих значень з яких може бути обрано максимум одне значення.

```
message Client {
   string client_name = 1;
   int32 client_id = 2;
   string client_email = 3;
   string client_password = 4;
}
```

Таким чином описуються повідомлення, для прикладу, вижче зображено приклад опису повідомлення яке буде містити інформацію про користувача. Поля “client_name”, “client_id”, “client_email”, “client_password” з типами даних та номеру поля. Номера полів використовуються при кодуванні та декодуванні повідомлень, що є важливим, та їхня зміна часто може призвести до помилок.

```
Client john = Client.newBuilder()
      .setClientId(1)
      .setClientName("Test Name")
      .setClientEmail("test.name@example.com")
      .setClientPassword(“********”)
      .build();
```

Компілятор protoc генерує на основі написаних proto файлів потрібні класи та білдери для зручного створення об’єктів чи отримання інформації з них. Вижче зображено приклад білдера який згенерований використовуючи мову програмування Java для створення об’єкта класу Person.

Protobuf має досить широкий функціонал, що дозволяє створювати потрібні об’єкти, використовуючи стандартні типи даних чи створені особисто.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ ДОДАТКУ

3.1 Основна мета проекту

Приступаючи до виконання дипломної роботи, я поставив за мету освоїти навички створення веб додатків з функціональністю чатів.

Досліджуючи дану сферу, були опрацьовані статті, дописи на тематичних платформах щодо вибору фреймворків для зручної розробки. Сервісна архітектура це саме те, що дало змогу розробити потрібний додаток. gRPC це досить популярний фреймворк, про який написано чимало статтей, що і дало мені змогу розібратися з його особливостями.

Також при виконанні проекту, я ознайомився із базою даних Postgres, яку використовував для зберігання всіх даних, які потрібні були для дослідження.

3.2 Обґрунтування вибору мови програмування

Для написання програмної реалізації веб додатку було обрано Java в якості мови програмування. Вибираючи мову програмування, було достатньо варіантів, для прикладу якщо ставити за мету швидкого написання, то Python був би зручнішим, та я ставив за мету надійність мого додатку. Саме Java ідеально з цим справилась. Ще до запуску програми, компіляція виявляє достатньо багато можливих помилок, що досить є корисним при розробці.

Веб та бекенд в Java розвинуті досить сильно і уже достатньо багато років Java використовується для вирішення подібних задач. Хоча бекенд написаний за допомогою мови програмування в рази сильніше розвинутий, є достатньо багато фреймворків які орієнтовані на веб та справляються з усіма поставленими задачами. Та враховуючи що Java є об’єктно орієнтованою мовою програмування, це дозволяє використовувати достатньо багато корисних патернів програмування.

3.3 Структура проекту

Структура проекту поділяється на декілька частин: веб фронтенд, веб сервіс, чат сервіс, логін сервіс та база даних. Веб сервіс написаний за допомогою фреймворку SpringBoot який дозволяє з легкістю контактувати з веб фронтендом за допомогою REST підходу та слідкувати за правильною роботою додатку. Веб сервіс в свою чергу контактує з допоміжними чат сервісом та логін сервісом за допомогою протоколу GRPC та отримує відповідну інформацію.

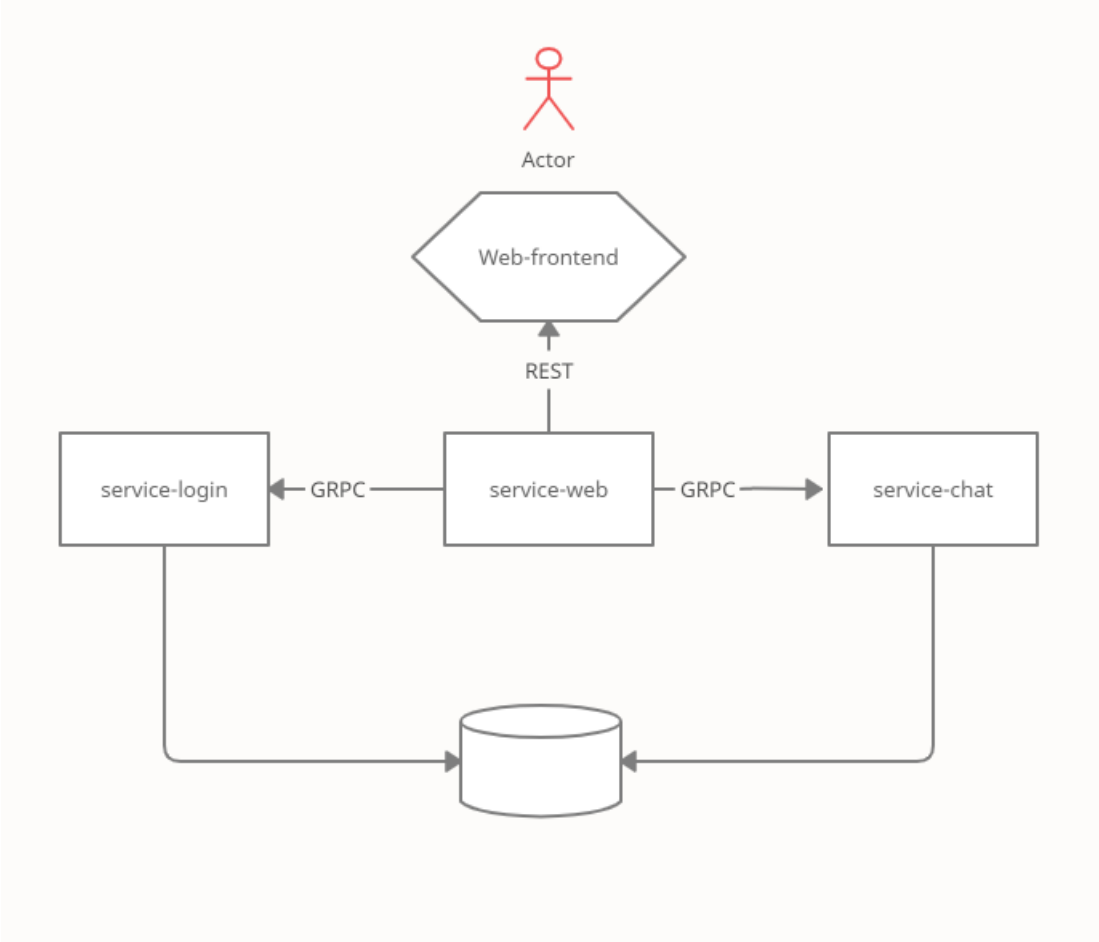

Рис. 3.3.1 – Архітектура проекту

3.3.1 Логін сервіс

Логін сервіс виконує головну роботу, перевіряючи інформацію про клієнта, якщо інформація є правильною то відбувається логін – вхід в систему, а якщо інформація є хибною то відповідно повідомляє веб сервіс про це. Також

логін сервіс дозволяє створити новий акаунт, при цьому отримуючи інформацію від веб сервісу, перевіряючи чи даного клієнта не має в базі даних, та заносячи нову інформацію в базу даних.

```
service LoginService {
   rpc signIn (login.ClientInfo) returns (login.Status);
   rpc signUp (login.ClientInfo) returns (login.Status);
   rpc changeInfo (login.ClientInfo) returns (login.Status);
}
```

Метод signIn викликається клієнтом для відправлення інформації про клієнта для входу, сервіс перевіряє чи даний користувач створений та відправляє клієнту відповідь-статус.

Метод signUp викликається клієнтом для відправлення інформації про нового клієнта для створення акаунту та входу, сервіс перевіряє чи даний користувач не створений, створює його та відправляє клієнту відповідь-статус.

Метод changeInfo викликається клієнтом для відправлення інформації про зміну певної інформаціє клієнта, сервіс перевіряє чи даний користувач створений, змінює інформацію та відправляє клієнту відповідь-статус.

```
message ClientInfo {
   string username = 1;
   string passwotd = 2;
}
```

Повідомлення ClientInfo містить тільки основну інформацію таку як юзернейм клієнта та його пароль.

```
message Status {
   string client_id = 1;
   Enum enum = 2;

   Enum Enum {
```

```
        UNKNOWN = 0;
        SUCCESS = 1;
        CLIENT_NOT_REGISTERED = 2;
        WRONG_PASSWORD = 3;
    }
}
```

Повідомлення Status містить client_id для ідентифікації клієнта та код статусу.

3.3.2 Чат сервіс

Чат сервіс працює з веб сервісом надаючи всі можливості для роботи з чатами та для отримання відповідних питань. Він виконує основну роль в даному додатку оскільки забезпечує відправку клієнтам питань та отримання відповідей на них, створення нових чатів на основі наданих відповідей та сповіщення клієнтів про них, відслідковування інформації про актуальні чати, відправку клієнтам нових повідомлення та отримання повідомлень від самих клієнтів та збереження їх до бази даних.

```
service ChatService {
    rpc login (service.ClientInfo) returns (stream service.UnsolicitedMessage);
    rpc logout (service.ClientInfo) returns (service.Status);

    rpc getQuestion (google.protobuf.Empty) returns (service.Question);
    rpc sendAnswer (service.Answer) returns (google.protobuf.Empty);

    rpc getAllChats (service.ClientInfo) returns (service.chat.ChatInfoList);
    rpc getChatMessages (service.chat.ChatInfo) returns (service.chat.ChatResponseList);
    rpc sendMessage (service.chat.ChatRequest) returns (service.Status);
}
```

Методи чат сервісу можна поділити на три різні частини:

1) Авторизація користувача у чат сервісі

Першим методом даної частини є метод login який викликається клієнтом для відправлення інформації про клієнта, сервіс перевіряє чи даний користувач створений та чи він ще не знаходиться у системі чат сервісу. Якщо ні то зберігає інформацію про нього та відкриває стрім у який буде надсилати повідомлення такі як створення нового чату або надходження нового повідомлення у чат який на даний момент не є відкритим.

Та другим методом є logout який викликається клієнтом для відправлення інформації про клієнта для якого потрібно виконати вихід з чат сервісу, сервіс перевіряє чи даний користувач знаходиться у системі чат сервісу. Якщо так то забирає інформацію про нього. Він надсилає клієнту повідомлення в якому міститься статус даної процедури.

2) Відповіді на питання

Клієнту доступні два методи getQuestion та sendAnswer. Викликаючи перший метод, сервіс підправляє клієнту питання з його ід кодом. Та другий метод sendAnswer клієнт викликає надсилаючи повідомлення в якому міститься ід питання, інформація про клієнта, відповідь так або ні, час відповіді. Враховуючи це, сервіс перебирає уже отримані раніше відповіді, якщо є аналогічне питання з протилежною відповіддю то формує чат для двох користувачів та оповіщує їх. Якщо ж ні, то запам’ятовує відповідь даного клієнта.

3) Робота з чатами

Метод getAllChats доступний клієнту для отримання інформації про усі його створені чати. Клієнт надсилає свою інформацію, сервіс перевіряє чи даний користувач знаходиться в системі та надсилає клієнту

список його чатів.

Також клієнту доступний метод getChatMessages в який клієнт надсилає свою інформацію та ід чату, у відповідь сервіс перевіряє чи клієнт знаходиться у системі та надсилає клієнту усі повідомлення які містяться у даному чаті.

За допомогою метода sendMessage клієнт надсилає інформацію про чат та повідомлення, у відповідь сервіс зберігає дане повідомлення та повертає клієнту статус про зберігання повідомлення.

3.3.3 База даних

Для отримання інформації про клієнтів, чатів а також питань, додаток використовує базу даних PostgreSQL у якій міститься дана інформація. На даний момент, PostgreSQL є однією із найбільш популярною системою управління базами даними. PostgreSQL має особливість яка відрізняє її від аналогів таких як MySQL, ManaDB чи Firebird і це є те, що дана база даних є об’єктно реляційною СУБД. Це дає певну перевагу використовувати певні користувацькі об’єкти чи складні структури даних. Це робить дану БД гнучкою та надійною.

Однією з переваг СУБД PostgreSQL також є її архітектура. Як і у випадку з багатьма іншими СУБД, PostgreSQL можна застосовувати в клієнт-серверному процесі - це надає безліч переваг і користувачам, і самим розробникам. В основі PostgreSQL лежить серверний процес бази даних, який виконується на одному сервері. Також доступ з програм до бази даних здійснюється за допомогою спеціального процесу. Це означає, що клієнтські програми не можуть отримувати самостійний доступ до даних навіть у тому випадку, якщо вони функціонують на тому самому ПК, на якому здійснюється серверний процес.

У сервісах було створено DBManager для з’єднання з самою базою даних який перевіряє чи PostgreSQL драйвер присутній та в подальшому відкриває

конекшн використовуючи url та дані для під’єднання. Таким чином ми отримуємо поділ клієнтів та сервера, що дає можливість створити декілька окремих систем.

3.4 Опис створеного додатку

Для коректного використання створеного додатку необхідно мати пристрій з доступом до мережі інтернет.

На початку, для користувачів є можливість авторизуватись в уже створений акаунт ввівши правильний юзернейм та пароль або зареєструвати новий акаунт.

Після реєстрації користувач попадає на стартову сторінку додатку з якої він може попасти на сторінку з питаннями або на сторінку з доступними чатами, натиснувши відповідну кнопку. Потрібно враховувати, якщо у клієнта не має створених чатів, то сторінка з чатами буде пустою. В такому випадку користувачеві потрібно попасти на сторінку з питаннями, відповісти на задане питання, та після підбору суперника відповідний чат буде створений.

Користувачеві пропонується питання, на яке доступні дві відповіді: «так» і «ні». Залежно від відповіді буде розпочато процес підбору суперника. Якщо користувач вибрав відповідь «так» то в якості суперника буде підібрано користувача який на це саме питання відповів «ні». Це забезпечить правильність створення чатів, щоб кожний користувач дискутував стосовно тієї сторони, яка йому ближча, або стосовно якої користувач має більше тверджень.

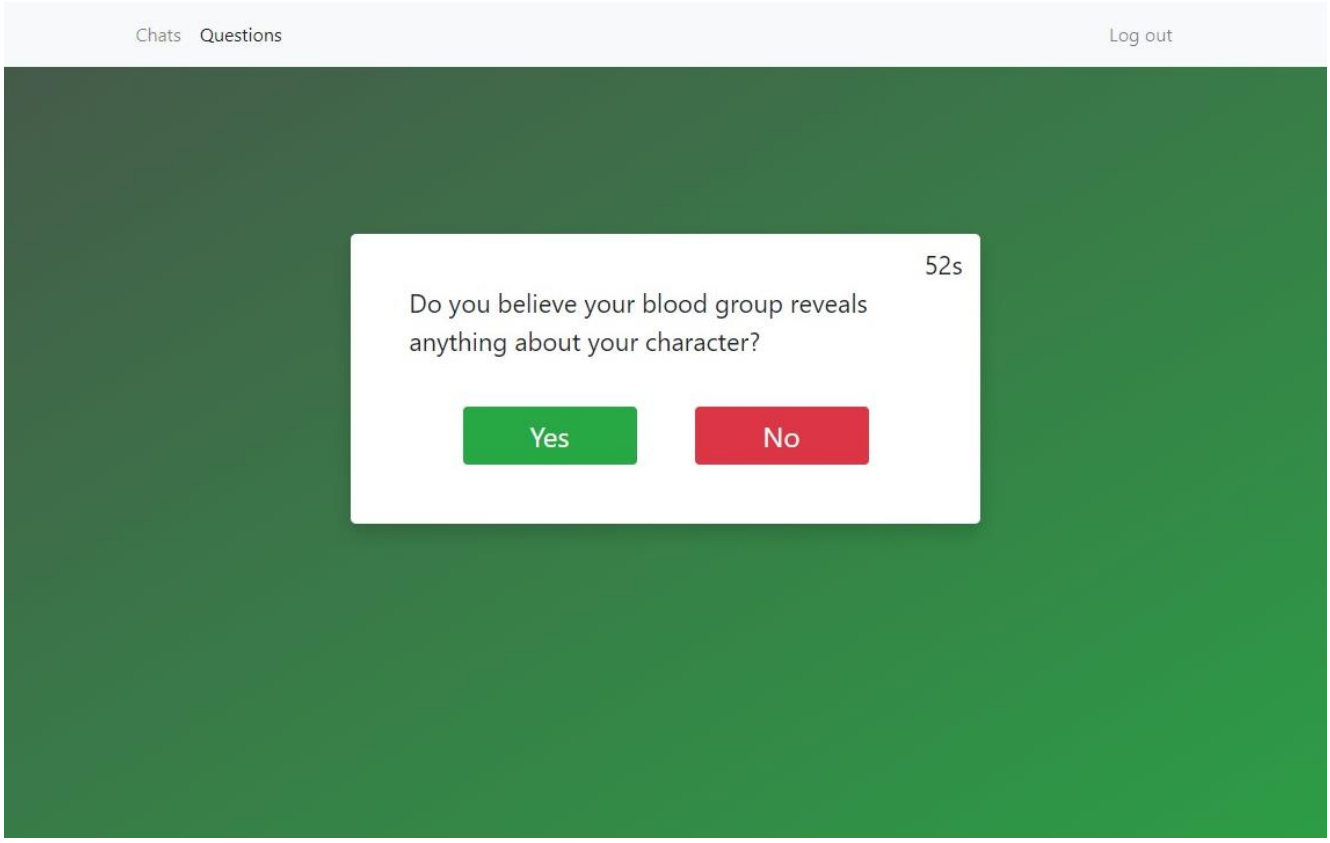

Рис. 3.4.1 – Сторінка з питанням

На Рис. 3.4.1 зображено вікно запропонованого питання. Користувачеві надано питання, можливість обрати відповідь, на відображений час, протягом якого користувач може надати відповідь на питання. Після того як час закінчиться, якщо клієнт не дасть відповідь, йому буде надано наступне питання. Користувач також має можливість відкрити сторінку з створеними чатами або ж вийти з акаунту, не відповідаючи на задане питання.

Після того як користувач надасть свою відповідь, вона буде підсвічена як вибрана з відповідним написом про те що проводиться підбір суперника. Якщо часу залишиться менше за 30 секунд то клієнту для очікування буде надано додаткові 30 секунд. Після того як суперник буде найдений, появиться відповідне вікно з даною інформацією та клієнт може або відкрити створений чат з підібраним суперником або повторити процес відповідання на питання.

Якщо клієнт відповів на питання, для нього підібрався суперник та початковий час (без додаткового) не закінчився то клієнт має змогу відповісти повторно на дане питання з можливістю підбору іншого суперника та створення

ще одного чату. Якщо ж початковий час вийшов то клієнту буде запропоновано інше питання.

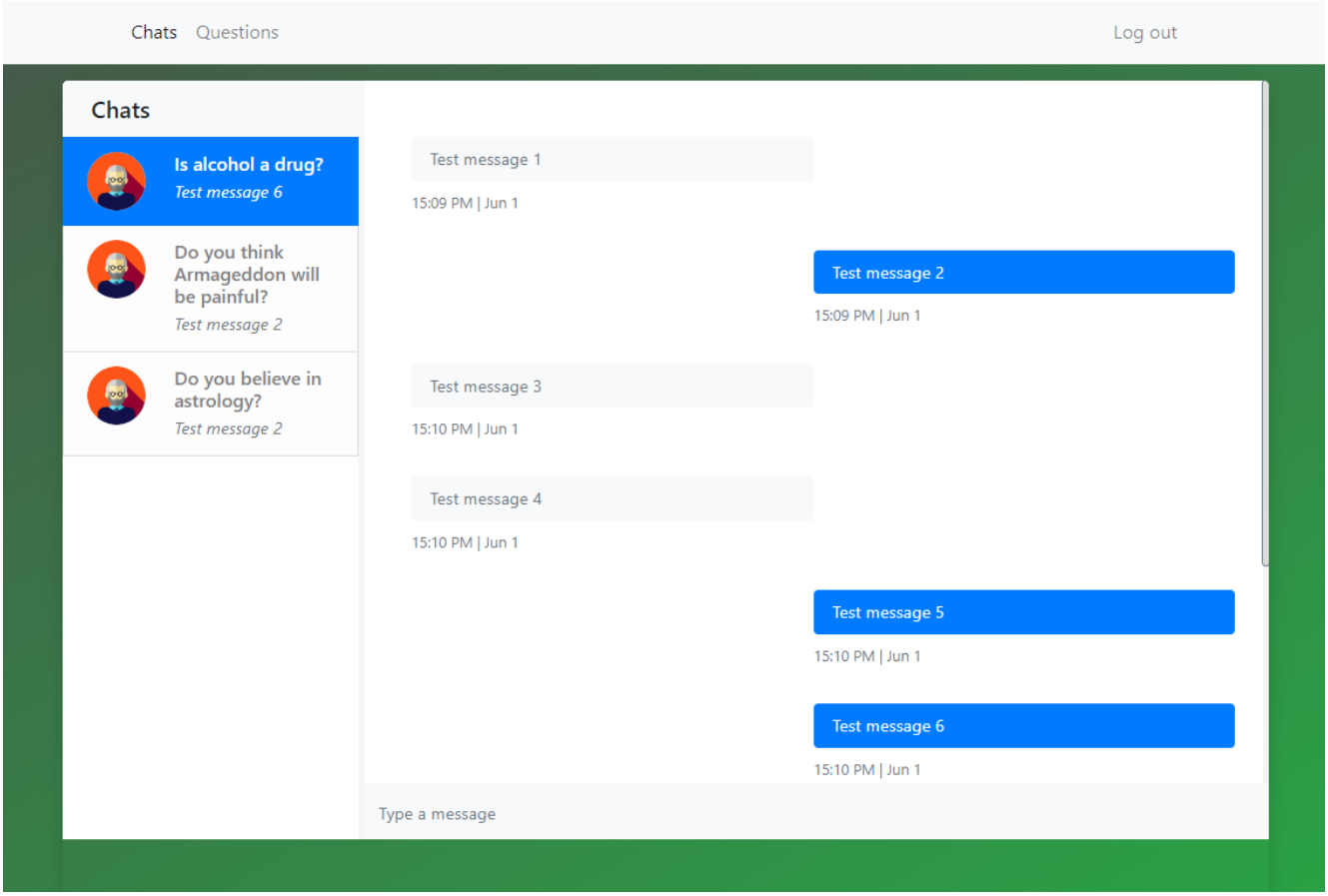

Рис. 3.4.2 – Сторінка з чатами

На Рис. 3.4.2 зображено вікно з доступними чатами. В лівій частині екрану зображені доступні чати з вказаним питанням на який користувачі давали відповіді та останнє повідомлення яке було надіслане у відповідному чаті. Користувачеві надані всі створені чати та користувач має можливість відкрити чат який його цікавить. Після цього клієнт може побачити історію всіх повідомлень які були надіслані клієнтом чи суперником а також час відправлення повідомлень. Варто зазначити що в чатах не вказані юзернейми користувачів, що забезпечує анонімність проведення дискусій. Клієнт може ввести потрібне повідомлення в відповідне поле яке розміщене в нижній частині чату та надіслати його.

ВИСНОВОК

Під час написання випускної кваліфікаційної роботи була поставлена задача створити веб додаток який би забезпечував проведення обговорення стосовно вибраного питання та який має зберігати анонімність опонентів.

При виконанні даного завдання було проаналізовано предметну область додатку, перевірено актуальність обраної теми а також було переглянуто існуючі додатки аналоги які мають схожий функціонал або частково виконують дану задачу та були визначені суттєві плюси і мінуси які були враховані при написанні даної роботи.

В процесі розробки додатку було проаналізовано відповідні засоби та фреймворки для реалізації додатку. Було розглянути мови програмування, фреймворки та протоколи для передачі даних. Для реалізації було вибрано мову програмування Java, фреймворки SpringBoot та gRPC а також протоколи WebSocket та Protobuf для коректної роботи сервісів і правильності даних які передаються.

Додаток було розроблено так, щоб він виконував задовільняв такі умови: додаток має бути простим для користувача, інтерфейс має бути зручним та зрозумілим, застосунок має сам пропонувати користувачам питання для дискусій без швидкого повторення, додаток має забезпечувати рівність користувачів, клієнти самі вибирають сторону яку будуть захищати, сервіс має забезпечувати щоб в чат був створений для користувачів з протилежними відповідями на питання, веб додаток має забезпечувати анонімність клієнтів, сервіс має бути безкоштовним для користувачів, додаток має зберігати історію існуючих чатів, додаток має приносити тільки користь та нові знання для користувачів, додаток має мати високу швидкодію.

Список посилань

[1] https://spring.io/projects/spring-boot

[2] “Spring Boot in Action”, Craig Walls

[3] https://grpc.io/docs/what-is-grpc/introduction/

[4] https://grpc.io/docs/languages/java/quickstart/

[5] “gRPC: Up and Running”, Kasun Indrasiri, Danesh Kuruppu

[6] https://en.wikipedia.org/wiki/WebSocket

[7] “Java WebSocket Programming”, Coward Dr. Danny

[8] https://developers.google.com/protocol-buffers/docs/overview