Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«СИСТЕМА УЗГОДЖЕННЯ ВЗАЄМОДІЇ НАДАВАЧІВ ТА СПОЖИВАЧІВ ПОСЛУГ»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Гринченко Н. О.	КН-П-202
2	Клементьєв Д. С.	КН-П-202
3	Луцевич Д. А.	КН-П-202
4	Радовська Н.	КН-Д-201

Автор звіту

_____ Гринченко Нікіта Олександрович

Одеса – 2024

АНОТАЦІЯ

УДК 004.9
Г-82

Гринченко Н. О. Система узгодження взаємодії надавачів та споживачів послуг: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп’ютерні науки». - Одеса: ОТУШ, 2024 - 20 с.

Робота присвячена дослідженням, розвиненню та впровадженню інформаційної системи, головним призначенням якої є узгодження взаємодії надавачів та споживачів послуг. Додаткові функції системи включають реєстрацію, авторизацію та автентифікацію, засоби створення оголошень, персональний кабінет, пошуковий механізм.

Актуальність роботи визначається тим, що в сучасній ситуації цифрового світу необхідно оптимізувати процеси взаємодії між постачальниками послуг і споживачами. Важливість цього завдання зростає в зв'язку зі збільшенням кількості онлайн-сервісів, що пропонують різні послуги, і вимог до швидкого і зручного доступу до цих сервісів. Крім того, актуальність даного дослідження посилюється необхідністю інтеграції безпечних методів обробки і зберігання персональних даних користувачів. Це особливо важливо в контексті нових законодавчих вимог до захисту персональних даних.

Метою цього дослідження є інформаційна система, що забезпечує узгодження взаємодії надавачів та споживачів послуг. Предметом дослідження цієї системи виступає оптимізація процесів взаємодії користувачів системи, зокрема, пошукові алгоритми.

Призначення даного дослідження є розробка та впровадження інформаційних систем, що забезпечують ефективну взаємодію між постачальниками послуг та споживачами з урахуванням аспектів безпеки та зручності використання.

Задачі дослідження включають:

- Провести аналіз існуючих інформаційних систем для узгодження взаємодії надавачів та споживачів послуг, визначити їхні переваги та недоліки.
- Розробити програмний комплекс, що включає засоби реєстрації, авторизації, автентифікації, створення оголошень, персональний кабінет та пошуковий механізм.
- Провести тестування розробленої системи у різних умовах для оцінки її ефективності та надійності.
- Впровадити механізми захисту даних для забезпечення безпеки персональних даних користувачів.

Проведені випробування серверної частини були спрямовані на перевірку коректності роботи всіх ключових функцій системи, а саме: реєстрації, авторизації, редагування користувачів, створення та управління оголошеннями, а також завантаження зображень. Випробування здійснювалися за допомогою таких інструментів:

- Swagger UI: Використовувався для інтерактивного тестування кінцевих точок API, дозволяючи виконувати запити та переглядати відповіді сервера у реальному часі.
- Postman: Використовувався для детального тестування HTTP-запитів, включаючи налаштування параметрів запитів та аналіз отриманих відповідей.
- Azure Web App: Платформа, на якій було розгорнуто та протестовано серверну частину API.

Для вирішення поставленого завдання були використані методи когнітивного аналізу, композиції та програмної інженерії. Були проведені огляди та аналіз нормативних та технічних документів.

В результаті дослідження була розроблена інформаційна система, що дозволяє ефективно координувати взаємодію між постачальником послуги і споживачем. Система забезпечує надійну реєстрацію та автентифікацію користувачів, а також безпечне зберігання персональних даних. Проведені випробування показали високу ефективність і надійність розробленої системи.

Робота містить 8 посилань на друковані та електронні джерела інформації. Розроблена інформаційна система має значний потенціал для застосування у різних галузях цифрової економіки, сприяючи підвищенню ефективності та зручності надання та отримання послуг.

ЗМІСТ

ВСТУП

За результатами всеукраїнського опитування, проведеного Київським міжнародним інститутом соціології у 2023 році на замовлення Програми розвитку ООН (UNDP) в Україні за підтримки Швеції та у партнерстві з Міністерством цифрової трансформації України, кількість українців, які щоденно користуються інтернетом, протягом року зросла з 72% до 80%. З 2021 року цей показник збільшився на понад 10%. Державними електронними послугами користувалися 64% респондентів, а серед найпопулярніших ресурсів — портал та застосунок "Дія", якими користувалися 51% українців станом на початок жовтня 2023 року. Майже 80% користувачів оцінюють свій досвід користування державними електронними послугами як позитивний. Це свідчить про зростання рівня цифрової грамотності та задоволеність громадян електронними послугами [1]. Тому важливо, особливо у воєнний час, охопити всі сфери діяльності людей у цифрову екосистему, де люди зможуть віддалено отримувати та ділитися своїми послугами.

Питома вага кіберзагроз зростає і ця тенденція в міру розвитку інформаційних технологій та їх конвергенції з технологіями штучного інтелекту в найближче десятиліття посилюватиметься. Зростання такого впливу на функціонування структур управління як національних, так і транснаціональних формує нову безпекову ситуацію. Між світовими центрами сили відбувається поділ сфер впливу у кіберпросторі, посилюється їх прагнення за рахунок такого поділу забезпечити реалізацію власних геополітичних інтересів. [3]. Це вимагає подальших досліджень і розробок у цій сфері та підкреслює потребу впровадження сучасних засобів захисту інформації. Для забезпечення належного рівня кібербезпеки необхідно розробляти комплексні стратегії, що включають технічні, організаційні, правові та освітні заходи.

У цій роботі розглянуто популярну систему взаємодії користувачів, яка включає розміщення пропозицій від одних користувачів та пошук відповідних пропозицій іншими. При цьому враховуються різні фільтри, такі як геолокація обох користувачів.

Відповідно до функціональних вимог більшості цих інформаційних систем вони повинні забезпечувати можливість зберігання історії транзакцій, пошуку пропозицій, фіксації факту прийняття або відхилення. Перспективним є впровадження методів оплати. Це вимагає наявності заходів авторизації та автентифікації та захисту персональних даних від несанкціонованого доступу інших користувачів.

Детальне технічне завдання проєкту представлено далі. У рамках цього звіту розглядається клієнтська частина, що являє собою веб-додаток або мобільний додаток. Звіт складається з трьох розділів, у яких описується загальна характеристика системи, деталі створення клієнтської частини та результати її розгортання і тестування. Висновки відображають досягнуті результати в рамках цієї частини проєкту, а загальний висновок є сукупністю звітів усіх учасників проєкту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис.

Програмне забезпечення (ПЗ) розроблено з метою створення системи, яка налагоджує та спрощує взаємодію користувачів з різними ролями (клієнт і виконавець). Програмне забезпечення повинно працювати на різних пристроях (смартфони, планшети, комп'ютери), щоб користувачі мали можливість зручно взаємодіяти з системою у будь-який час та з будь-якого місця. Це забезпечує зручність та гнучкість у використанні послуг,

Призначення:

Система орієнтована на забезпечення ефективної взаємодії між користувачами різних ролей (клієнт і виконавець), створення зручного середовища для обміну інформацією між користувачами та надання можливості вибору (клієнти вибирають виконавця, а виконавці - клієнта).

Вимоги до функціоналу:

Аутентифікація користувачів.

Для використання всієї функціональності ПЗ вимагається аутентифікація користувача. Для реєстрації користувач повинен вказати свою електронну пошту, ім'я (для відображення його даних профілю у інших користувачів / звернення до нього) та придумати пароль (є можливість подальшого додавання реєстрації за допомогою сторонніх сервісів - Google, Apple тощо). Після реєстрації користувач повинен підтвердити свою електронну адресу шляхом введення коду / переходу за посиланням з листа, надісланого йому на пошту після реєстрації. Також під час реєстрації вказується роль користувача (клієнт / виконавець).

Для входу в обліковий запис використовується електронна пошта та пароль.

Особистий кабінет

Особистий кабінет користувача містить інформацію про користувача, його історію замовлень. Користувач може редагувати свої дані та змінювати аватар. Також доступна можливість вийти з облікового запису. Залежно від ролі, надаються наступні можливості:

Для клієнта:

Збереження улюбленців: клієнт може зберегти кілька своїх домашніх тварин.

Додавання оголошення - користувач розміщує пропозицію щодо послуги для пошуку виконавця.

Для виконавця:

Додавання резюме: виконавець може додати кілька резюме з різними категоріями послуг та категоріями тварин.

Додавання пропозиції в “збережені”: виконавець може додати декілька пропозицій та переглянути їх пізніше.

Ці функціональності дозволяють користувачам управляти своїми профілями відповідно до їхніх ролей.

Пошук пропозиції

Ця функціональність забезпечує пошук актуальних пропозицій для виконавця і включає наступні можливості:

Перегляд стрічки пропозицій. Виконавець може переглядати список актуальних пропозицій у вигляді карток з основною інформацією: назва, короткий опис, тип та інше.

Можливість фільтрації і сортування. Виконавець може фільтрувати пропозиції за категорією послуги, категорією тварини та іншими критеріями. Також є можливість сортування за датою, популярністю тощо.

Пошук пропозицій. Виконавець може знайти конкретну пропозицію за ключовими словами, назвою тощо.

Детальний перегляд пропозиції. При натисканні на картку пропозиції з ленти відкривається її детальний огляд з усією інформацією, наданою замовником.

Взаємодія з пропозицією. Виконавець може відгукнутися на пропозицію, зв'язатися з клієнтом для уточнення деталей, а також додати її до збережених.

Пошук виконавця за їхніми резюме

Ця функціональність забезпечує зручний і ефективний процес пошуку виконавців відповідно до вимог клієнта і включає наступні можливості:

Перегляд ленти резюме виконавців: Клієнт може переглянути список резюме, що були опубліковані виконавцями, у вигляді карток.

Можливість фільтрації і сортування: У клієнта є можливість фільтрувати резюме за категорією послуги, категорією тварини та іншими критеріями. Також доступна можливість сортування за рейтингом, популярністю тощо.

Детальний перегляд резюме: При натисканні на картку резюме відкривається їхнє детальне відображення з усією інформацією, яку вказав виконавець.

Взаємодія з виконавцем: Клієнт може відгукнутися на резюме, зв'язатися з користувачем для обговорення деталей і умов співпраці.

Архітектура та технології:

Програмне забезпечення розроблено за клієнт-серверною архітектурою. Завдяки цьому забезпечується горизонтальне і вертикальне масштабування, розподіл обчислювальних ресурсів (зниження навантаження на клієнтські пристрої, ефективне використання ресурсів сервера), централізоване управління даними (всі дані зберігаються на сервері), гнучкість при розробці і підтримці ПЗ (клієнт і сервер можуть бути розроблені і оновлені незалежно один від одного). Основні компоненти системи включають в себе backend, frontend (веб-додаток, мобільний додаток).

Backend

Серверна частина проекту побудована на основі Azure Web App та ASP.NET Core. Дані зберігаються в CosmosDB, що забезпечує високу продуктивність та масштабованість бази даних. Для зручного тестування та документування API використовується Swagger UI.

Основні технічні деталі:

- Azure Web App: Використовується для хостингу серверної частини додатку, забезпечуючи безперебійну роботу і масштабованість.
- ASP.NET Core: Основний фреймворк для розробки бекенду, що забезпечує високу продуктивність та підтримку сучасних стандартів.
- CosmosDB: Документо-орієнтована база даних, яка забезпечує масштабованість та високу доступність даних.
- Swagger UI: Інструмент для документування API, що дозволяє легко тестувати та взаємодіяти з різними кінцевими точками API.
- API: Розроблено набір RESTful API для взаємодії з фронтендом та мобільним додатком. Всі дані передаються у форматі JSON.

Frontend

Включає в себе веб сайт і мобільний додаток, які призначені для взаємодії користувачів із системою.

Веб Сайт

Веб інтерфейс замовника та виконавця який доступен з будь якого браузеру. Надає користувачам зручний інтерфейс для створення резюме та пропозицій, редагування особистих даних та відгуків на пропозиції.
Веб сайт розроблений на основі NextJs та компонентів shadcn/ui, tailwind використаний для стилізації.

Мобільний додаток

Це інтерфейс користувача, завдяки якому можна взаємодіяти із системою (переглядати і управляти пропозиціями, шукати виконавців та працювати з профілем користувача). Мобільний додаток взаємодіє з бекендом через API. Для розробки використовується кросплатформне рішення (фреймворк React Native з використанням Expo).

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Відповідно до технічного завдання (ТЗ), система була спроектована з використанням клієнт-серверної архітектури. Система включає наступні архітектурні компоненти:

- «Mockup»
- «Backend»
- «Frontend»

Частина «Mockup» являє собою набір моделей та описів поведінки користувачів. Модель відображає інтерфейси взаємодії користувачів різного типу з інформаційною системою, траєкторії їхніх рухів та переходів між різними інтерфейсами. При створенні інтерфейсів застосовано принципи розумної навігації, включно з підходами UI/UX, висновками колористики та ергономіки. Частина розміщена на платформі Figma:
https://www.figma.com/design/KRhpRsAPICoDGu8lJHkUjy/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC?node-id=0-1&t=iQUc0cKH063YVKH3-1

Частина «Backend» реалізована як централізований вузол збереження та обробки даних, які створюються та модифікуються протягом життєвого циклу інформаційної системи. Ця частина включає базу даних проєкту та програмний інтерфейс (API) доступу до основних її функцій, таких як керування контентом, авторизація, узгодження профілів користувачів та керування оголошеннями. Сервер API доступний за посиланням: https://shelterbackapi.azurewebsites.net/swagger/.

Частина «Backend» виконана на основі програмного забезпечення .NET C#, з використанням баз даних CosmosDB та AzureDB. Додатково використано бібліотеку Newtonsoft JSON для роботи з JSON-даними.

Клієнтська частина є інтерфейсом взаємодії користувача з системою. Включає в себе веб сайт і мобільний додаток, які мають зручний і зрозумілий користувацький інтерфейс. Таке рішення забезпечує високу ефективність і доступність обміну інформацією між клієнтами та виконавцями незалежно від пристрою та платформи, які вони використовують. Усіма функціональностями системи можна скористатися як на веб сайті, так і в мобільному додатку (IOS і Android).

Для покращення безпеки, з урахуванням частого підключення мобільних пристроїв до незахищених мереж передачі даних, інформаційний обмін між частинами «Backend» та «Frontend» захищений шифруванням за допомогою сучасного криптоалгоритму SHA-256. Це забезпечує безпеку системи в відкритих мережах та спрощує задачі автентифікації та доступу до контенту з обмеженим доступом. Важливо зазначити, що алгоритм SHA-256 широко використовується в галузі криптовалют, де безпека даних є критично важливою. Наприклад, біткоїн та інші криптовалюти використовують SHA-256 для забезпечення цілісності та безпеки транзакцій [4]. Слід зазначити, що алгоритм SHA-256 є найпоширенішим у різних сферах цифрових технологій [5].

РОЗДІЛ 2
РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Клієнтська частина (мобільний додаток) інформаційної системи розроблена з використанням наступних технологій:

Expo (React Native) - це фреймворк (React Native) і платформа (Expo) для кросплатформної розробки мобільних додатків (iOS та Android). Expo включає в себе набір готових інструментів та сервісів, які прискорюють і спрощують процес розробки. За допомогою Expo Go можна тестувати додаток на своєму пристрої, достатньо тільки завантажити додаток Expo Go на свій пристрій та відсканувати QR-код. Для створення готових файлів APK (Android) та IPA (iOS) використовується команда '**expo build:android**' або '**expo build:ios**'. Expo забезпечує хмарну збірку, де додаток збирається на серверах Expo. Також можна отримати файл збірки локально, якщо додати флаг '**--local**'. Після завершення збірки готові файли APK (Android) та IPA (iOS) можна завантажити на пристрої для тестування або в App Store та Google Play для публікації.

Reanimated - бібліотека для створення анімацій у React Native додатках. Вона забезпечує високу продуктивність завдяки виконанню анімацій у UI потоці замість JS потоку, як у стандартному Animated.

Zustand - легка та гнучка бібліотека, призначена для управління станом у додатку. Вона забезпечує ефективний спосіб збереження даних додатка. Використання 'Persist middleware' дає можливість зберігати дані стану в сховищі пристрою (Async Storage).
Основні функції клієнтської частини:

- Реєстрація та авторизація користувача
- Редагування даних користувача
- Пошук та перегляд оголошень
- Пошук та перегляд резюме
- Перегляд профілю виконавця

Для доступу до системи користувач повинен бути авторизований. На початку йому пропонується пройти реєстрацію або увійти в уже існуючий профіль. Для реєстрації потрібно вказати електронну пошту, ім'я та придумати пароль. Всі поля перед відправкою на сервер проходять валідацію за допомогою регулярних виразів.

```
export const emailRegex = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$";
export const nameRegex = "^[a-zA-Za-яA-Я]+([-' ][a-zA-Za-яA-Я]+)*$";
export const passwordRegex = "^.{6,}$";
```

Якщо користувач вже має акаунт, йому потрібно вказати електронну пошту та пароль для авторизації. Далі, якщо всі поля пройшли валідацію, надсилається запит на сервер. Якщо статус запиту дорівнює 200, то надсилається наступний запит для авторизації, результатом якого є токен авторизації та дані користувача. Токен зберігається в сховищі пристрою, а дані користувача - у userStore (глобальне сховище, яке доступне з будь-якого місця додатку) з додатковим збереженням у сховищі пристрою за допомогою Persist middleware. Обмін даними з серверною частиною реалізовано за допомогою бібліотеки axios. Нижче представлений код описаної вище логіки.

```
// authTypes.ts
type RegisterUserType = {
  email: string;
  password: string;
  name: string;
  role: "customer" | "client";
};

type UserType = {
  id: string;
  email: string;
  name: string;
  role: "customer" | "client";
  phone: string | null;
  avatar: string | null;
};

type AuthUserResponseType = {
  token: string;
  user: UserType;
};


// authApi.ts
const REGISTER_USER_URL =
  "https://shelterbackapi.azurewebsites.net/Account/RegisterUser";

const LOGIN_USER_URL = (email: string, password: string) =>

`https://shelterbackapi.azurewebsites.net/Account/AuthenticateUser?userEmail=${email}&us
erPassword=${password}`;

export const loginUser = async (
  email: string,
  password: string
): Promise<AuthUserResponseType | null> => {
  try {
    const response = await axios.post(LOGIN_USER_URL(email, password));
    const token = response.data.token;
    const user: UserType = {
      id: response.data.user.id,
      email: response.data.user.email,
      name: response.data.user.name,
```

```
      role: response.data.user.role,
      phone: response.data.user.phone,
      avatar: response.data.user.avatar,
    };
    return { token, user };
  } catch {}
  return null;
};

export const registerUser = async (
  user: RegisterUserType
): Promise<AuthUserResponseType | null> => {
  try {
    const registerRes = await axios.post(REGISTER_USER_URL, user);
    if (registerRes.status === 200) {
      const loginRes = await loginUser(user.email, user.password);
      if (loginRes) {
        return loginRes;
      }
    }
  } catch {}
  return null;
};


// userStore.ts
type State = {
  user: UserType | null;
};

type Action = {
  setUser: (user: UserType) => void;
  updateUser: (updatedInfo: Partial<UserType>) => void;
  clear: () => void;
};

const useUserStore = create<State & Action>()(
  persist(
    (set, get) => ({
      user: null,
      setUser: (user) => set({ user }),
      updateUser: (updatedInfo) =>
        set((state) => ({
          user: state.user ? { ...state.user!, ...updatedInfo } : null,
        })),
      clear: () => set({ user: null }),
    }),
    {
      name: "user-storage",
      storage: createJSONStorage(() => AsyncStorage),
    }
  )
);
```

Також для реалізації управління станом авторизації користувача використовується:

- authContext - контекст, який створюється за допомогою React API. Він дозволяє створити глобальний об'єкт.
- authProvider - компонент, який огортає усі компоненти застосунку, забезпечуючи доступ до контексту з будь-якої його частини.

Нижче представлений код authContext, authProvider, приклад його використання та створення хуку для спрощеного взаємодії з контекстом.

```
// AuthContext.tsx
const AuthContext = createContext<AuthContextData | undefined>(undefined);

interface AuthProviderProps {
  children: ReactNode;
}

export const AuthProvider = ({ children }: AuthProviderProps) => {
  const [isAuthenticated, setIsAuthenticated] = useState<boolean>(false);
  const [error, setError] = useState<string | null>(null);
  const userStore = useUserStore();

  const signIn = async (email: string, password: string) => {
    const res = await loginUser(email.toLowerCase(), password);
    if (res) {
      await AsyncStorage.setItem("userToken", res.token);
      userStore.setUser(res.user);
      setIsAuthenticated(true);
      router.replace("/index");
      return true;
    } else {
      setError("Невірний email або пароль");
      setTimeout(() => {
        setError(null);
      }, 3000);
      return false;
    }
  };


  useEffect(() => {
    const checkAuth = async () => {
      const token = await AsyncStorage.getItem("userToken");
      setIsAuthenticated(!!token);
      router.replace("/");
    };

    checkAuth();
  }, []);

  const signUp = async (user: RegisterUserType): Promise<boolean> => {
    const res = await registerUser(user);
    if (res) {
      await AsyncStorage.setItem("userToken", res.token);
```

```
      userStore.setUser(res.user);
      setIsAuthenticated(true);
      router.replace("/index");
      return true;
    } else {
      setError("Помилка реєстрації");
      setTimeout(() => {
        setError(null);
      }, 3000);
      return false;
    }
  };

  const signOut = async () => {
    await AsyncStorage.removeItem("userToken");
    setIsAuthenticated(false);
  };

  return (
    <AuthContext.Provider
      value={{
        isAuthenticated,
        error,
        signIn,
        signOut,
        resetPassword,
        signUp,
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};
```

```
export const useAuth = () => {
  const context = useContext(AuthContext);
  if (context === undefined) {
    throw new Error("useAuth must be used within an AuthProvider");
  }
  return context;
};

// app -> _layout.tsx
export default function RootLayout() {
  return (
    <AuthProvider>
      <Slot />
    </AuthProvider>
  );
}
```

Роутинг у додатку здійснюється за допомогою expo-router - це інструмент для навігації в додатках на React Native. Для визначення маршрутів використовується файлова система. Кожен екран представлений окремим

файлом в окремій директорії. Таким чином, є дві головні групи - (app) - основні екрани додатку та (auth) - все, що стосується авторизації. Нижче представлений приклад структури файлів.

```
app
 ├── (app)
 │    ├── home
 │    │    ├── advts
 │    │    │    ├── _layout.tsx
 │    │    │    ├── [id].tsx
 │    │    │    └── index.tsx
 │    │    ├── _layout.tsx
 │    │    └── index.tsx
 │    ├── profile
 │    │    ├── advts
 │    │    │    ├── _layout.tsx
 │    │    │    ├── add-advt.tsx
 │    │    │    └── index.tsx
 │    │    ├── resumes
 │    │    │    ├── _layout.tsx
 │    │    │    └── index.tsx
 │    │    ├── user-data
 │    │    │    ├── _layout.tsx
 │    │    │    ├── edit-user.tsx
 │    │    │    └── index.tsx
 │    │    ├── _layout.tsx
 │    │    └── index.tsx
 ├── (auth)
 │    ├── _layout.tsx
 │    ├── sign-in.tsx
 │    ├── sign-up.tsx
 └── _layout.tsx
```

Файли _layout.tsx описують загальну структуру групи файлів (екранів). Я в цьому файлі налаштовую стиль переходів між екранами. Також можна вказати редирект для користувачів конкретної ролі (заборонити їм доступ до певних екранів) та налаштувати додаткові стилі.
Файли index.tsx - це головний файл (екран) конкретної групи. З нього вже користувач може перейти на інші екрани.

Для переходу на інший екран потрібно імпортувати 'router' з 'expo-router' і викликати метод push("pageName") для додавання нового шляху та збереження історії переходів або replace("path") для заміни шляху без збереження історії переходів. Нижче представлений приклад використання для переходу на екран реєстрації з екрану входу користувача і заміни шляху при успішній авторизації.

```tsx
// sign-in.tsx
import { router } from "expo-router";
const onSignUpPress = () => {
  router.push("sign-up");
```

```
};
const onSignInPress = async () => {
...
  router.replace("/");
...
}
```

Для відображення певних компонентів та різної функціональності для різних ролей я реалізував компонент <WithRole>, до якого потрібно передати роль, і в залежності від ролі користувача компонент буде відображатися або ні. Код компонента представлений нижче.

```
// WithRole.tsx
const WithRole = ({ children, role }: { children: ReactNode; role: Role }) => {
  const { user } = useUserStore();
  if (user?.role !== role) {
    return <></>;
  }
  return children;
};
```

```
<WithRole role={Role.client}>
  <TouchableOpacity
    style={styles.menuItem}
    activeOpacity={0.7}
    onPress={() => router.navigate("profile/advts")}
  >
    <Ionicons
      name="documents"
      color={Colors.purple}
      size={moderateScale(20)}
    />
    <Text style={styles.menuItemText}>Мої пропозиції</Text>
  </TouchableOpacity>
</WithRole>
```

Таким чином, доступ до екрану "Мої пропозиції" буде лише у користувачів з роллю "client".

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом розробки є мобільний додаток, який був створений на фреймворку React Native за допомогою платформи Expo. Результатом компіляції коду клієнтської частини є файли збірки - APK (для Android) та IPA (для iOS). Тестування проводилося на симуляторах (iOS та Android) та на реальному пристрої "Samsung Galaxy S20+" на операційній системі Android 12.

Спочатку користувачу потрібно пройти реєстрацію або увійти в свій профіль, якщо користувач вже зареєстрований. Зовнішній вигляд екранів вказаний на рис. 1.

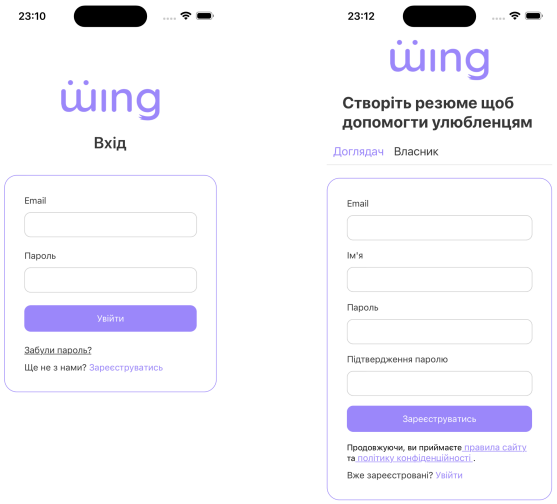

Рисунок 1 - Вхід та реєстрація користувача

Після авторизації користувач переходить на головний екран застосунку. Два основні екрани застосунку - це "Головна" і "Профіль". На головному екрані можна вибрати тип послуги та перейти на екран "Список пропозицій". На рис. 2 показано інтерфейс екранів "Головний" і "Список пропозицій".

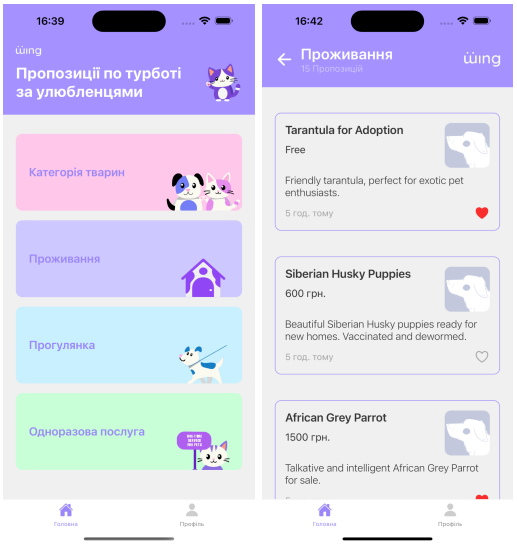

Рисунок 2 - Головний екран та список пропозицій

Також можна перейти на екран "Профіль" за допомогою таббара внизу. У профілі користувачу доступні наступні функціональні можливості (залежно від його ролі):

Клієнт:

- Особисті дані - тут користувач може переглянути свої особисті дані та, за бажанням, змінити їх.
- Мої пропозиції - на цьому екрані користувач може переглянути створені ним пропозиції та додати нову.
- Збережені резюме - тут відображається список резюме, які користувач додав "збережене".
- Історія - тут відображаються всі виконані виконавцями пропозиції.
- Вийти - вихід з облікового запису користувача. Повернення на екран авторизації.

Виконавець:

- Особисті дані - все точно так само, як у клієнта.
- Мої резюме - на цьому екрані розташований список резюме користувача. Тут же можна створити нове резюме для пошуку роботи.
- Історія - тут користувач може переглянути список виконаних пропозицій і відгуки.
- Збережені пропозиції - на цьому екрані відображається список пропозицій, які користувач додав у "збережене".
- Вийти - вихід з облікового запису користувача. Повернення на екран авторизації.

На рис. 3 показано інтерфейс та основні функціональності групи екранів “Профіль”

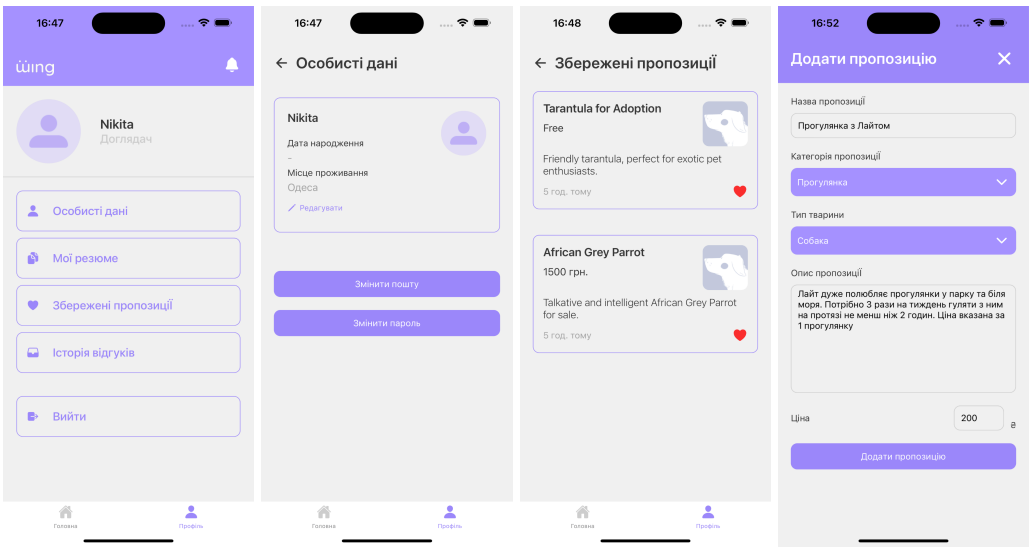

Рисунок 3 - Група екранів “Профіль” та її головні функціональності

Усі описані можливості та функціональність клієнтської частини були реалізовані та протестовані. З кінцевим результатом і повним кодом додатку можна ознайомитися в репозиторії: https://github.com/nikitaGrynch/ming

ВИСНОВКИ

Проведено аналіз існуючих інформаційних систем для узгодження взаємодії надавачів та споживачів послуг, виявлено їхні переваги та недоліки. Основним результатом цього аналізу стало визначення ключових вимог до системи, яка повинна забезпечувати ефективну взаємодію між постачальниками послуг та споживачами з урахуванням аспектів безпеки та зручності використання.

Розроблено програмний комплекс, який включає в себе засоби реєстрації, авторизації та автентифікації користувачів, створення та управління пропозиціями надавачів та споживачів послуг, персональний кабінет, а також пошуковий механізм. Особливу увагу було приділено зручному і зрозумілому інтерфейсу та продуктивності додатку.

Тестування проводилося на різних пристроях та платформах, включаючи iOS та Android симулятори і реальний пристрій на операційній системі Android 12 - Samsung Galaxy S20+. Результати тестування показали високу швидкість роботи застосунку, надійне збереження даних користувача, інтуїтивно зрозумілий інтерфейс та коректну роботу всіх модулів як окремо, так і в сукупності.

Систему було випробувано у різних умовах для оцінки її ефективності та надійності. Випробування проводилося з використанням Swagger UI для інтерактивного тестування кінцевих точок API, Postman для детального тестування HTTP-запитів, а також Azure Web App для розгортання та тестування серверної частини. Результати тестування підтвердили високу ефективність та надійність розробленої системи.

Впроваджено механізми захисту даних, які забезпечують безпечне зберігання та передачу персональної інформації користувачів. Це включає використання алгоритмів шифрування для захисту даних при їх передачі між клієнтською та серверною частинами системи.

Розроблена інформаційна система дозволяє ефективно координувати взаємодію між постачальниками послуг та споживачами, забезпечуючи надійну реєстрацію та автентифікацію користувачів, безпечне зберігання персональних даних, а також зручний та швидкий доступ до необхідних сервісів. Система має значний потенціал для застосування у різних галузях цифрової економіки, сприяючи підвищенню ефективності та зручності надання та отримання послуг.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Українці стали частіше користуватися інтернетом, 80% – онлайн щодня: соцопитування” // United Nations Development Programme URL: https://www.undp.org/uk/ukraine/press-releases/ukrayintsi-staly-chastishe-korystuvatysya-internetom-80-onlayn-shchodnya-sotsopytuvannya
2. Лященко А. А. Особливості реалізації стандартів доступу до баз геопросторових даних в середовищі універсальних СКБД. // Геоінформаційні технології у територіальному управлінні: матеріали II міжнар. наук.-практ. конф. Одеса: ОРІДУ НАДУ, 2015. с. 17-21 URL: http://www.oridu.odessa.ua/7/7/Book_new_2.pdf
3. Стратегія кібербезпеки України. Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165. URL: https://zakon.rada.gov.ua/laws/show/447/2021/print
4. SHA-256 - Bitcoin Wiki URL: https://en.bitcoin.it/wiki/SHA-256
5. Cryptographic Hash Functions: Definition URL: https://www.investopedia.com/news/cryptographic-hash-functions/
6. Expo Documentation URL: https://docs.expo.dev
7. React Native Documentation URL: https://reactnative.dev/docs/getting-started
8. Zustand Documentation URL: https://docs.pmnd.rs/zustand/getting-started/introduction
9. AI assistant - Chat GPT URL: https://chatgpt.com/