National College of Ireland

A Mobile Cloud Computing Framework for Real-Time Big Data Analytics in Healthcare

Research Project
MSc Cloud Computing

Sai Shruthi Yadavalli
Student ID: x23242281

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sai Shruthi Yadavalli
Student ID:	X23242281
Programme:	MSc Cloud Computing **Year:** 2024
Module:	Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	2024/12/12
Project Title:	A Mobile Cloud Computing Framework for Real-Time Big Data Analytics in Healthcare
Word Count:	1732
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sai Shruthi Yadavalli

Date: 2024/12/12

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Health Prediction Application

Sai Shruthi Yadavalli
x23242281

The Health Prediction Application is an all-in-one solution that needs inputs from the users in order to come up with predictions. This document will describe how to set up, deploy, and use the application which will consist of a Flask based back-end and a mobile application front-end implemented under React Native via Expo. In the back end it has user control, data control and Machine learning predictions, In front end it has mobile user interface.

The backend itself is implemented using Flask, with SQLAlchemy being used to handle the data storage, Flask-Bcrypt for handling of passwords and Flask-JWT-Extended for authentication. The backend also consists of an input from a pre-trained machine learning model to interpret input data and make corresponding predictions. The Flask API has end points of registration, login, prediction and logout. People can sign up with a user name and password, launch the system and get a token to enter, upload features for a prediction, and finally exit the site.

Utilizing React Native with Expo as the technology, the mobile frontend is created for enabling users to touch base with the backend. It offers the user to sing up, sign in, as well as to enter data for health forecasts. Nevertheless, the results are presented transparently, in the form of the press prediction, as well as the probability of achieving that press. Some of the features that the frontend entails include form validation and even easy navigation for improved user interfaces.

To start project the backend dependencies are mentioned in the requirements.txt file so that all necessary libraries of python are installed. The backend can be deployed with Google Cloud Platform (other than GCP, it is hard to think of another platform that can be used), and app.yaml specific for this process prevents issues with this. For the mobile frontend the package.json file defines the necessary build and running dependencies of the application. The frontend could be run with Expo Go or built under the APK which will be installed on the Android devices.

The Health Prediction Application has a strong back end, easy to use mobile front end, and prediction functionalities make this application useful in generating business intelligence. Everything, from setting up the app to its primary implementations are contained in this manual.

1 System Configuration

1.1 Backend:

1. **Python**: Version 3.8.16.

2. **Flask**: Version 3.0.3. Framework for building RESTful APIs.
3. **SQLite**: Version 3.43.2. Integrated database for user authentication and management.
4. **Scikit-learn**: For machine learning model predictions.
5. **Gunicorn**: Web server for deploying the application.

1.2 Frontend:

1. **Node.js**: Required for running Expo and React Native.
2. **Expo**: For managing React Native projects.
3. **Android Emulator** (or physical device): To test the APK on Android.

2 Backend Setup and Usage

2.1 Installing Dependencies

Ensure Python 3.10 is installed. Use the following command to install dependencies from requirements.txt:

```
pip install -r requirements.txt
```

2.2 Database Initialization

The application uses SQLite for user management. When the backend starts, the database initializes automatically if it doesn't already exist. Ensure the directory /tmp/ exists on your system for the database.db file.

2.3 Flask Application

The backend is implemented in app.py and serves multiple purposes:

1. **User Registration and Login**: Manages user authentication with JWT tokens.
2. **Prediction API**: Accepts user input, processes it using a pre-trained machine learning model, and returns predictions.

To start the Flask application:

```
python app.py
```

This will run the backend at http://localhost:5000 by default.

2.4 Key Backend Endpoints

1. **Register** (POST /register): Registers a new user. Requires username and password in the request body.
2. **Login** (POST /login): Authenticates the user and returns a JWT token.
3. **Predict** (POST /predict): Accepts input features for prediction. Requires authentication with a valid JWT token.
4. **Logout** (POST /logout): Invalidates the current session (client-side token removal).

2.5 Deployment

To deploy the application, use the provided app.yaml configuration for Google App Engine:

```
gcloud app deploy
```

This sets up the application to run on a GCP instance. Modify the JWT_SECRET_KEY in the Flask app for secure deployments.

3 Frontend Setup and Usage

The frontend is built using **React Native** with Expo. It provides an intuitive interface for users to interact with the backend.

3.1 Setting Up Expo

Install the Expo CLI if it's not already installed:

```
npm install -g expo-cli
```

Clone the repository, navigate to the frontend directory, and install dependencies:

```
npm install
```

3.2 Running the Application

To start the development server, use:

```
npm start
```

This command provides options to run the app on:

1. **Android Emulator**
2. **iOS Emulator** (on macOS with Xcode)
3. **Physical Device** (via Expo Go app).

To run directly on an Android emulator, use:

```
npm run android
```

3.3 Building the APK

To generate a standalone APK for distribution:

1. Use the Expo EAS (Expo Application Services) CLI to build:

```
eas build --platform android
```

2. Download the APK from the Expo dashboard or locally.

3.4 Key Features of the Frontend

1. **User Authentication**: A login and registration screen allows users to access the app securely.
2. **Health Prediction Input**: A form captures input features, such as medical test results, for predictions.
3. **Results Display**: The app displays the model's prediction and associated probability.

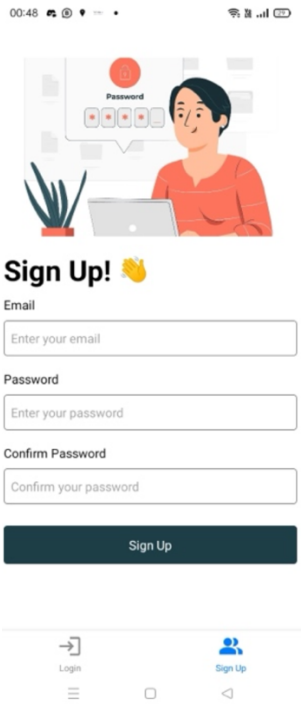

Fig 1: Sign up/Login page

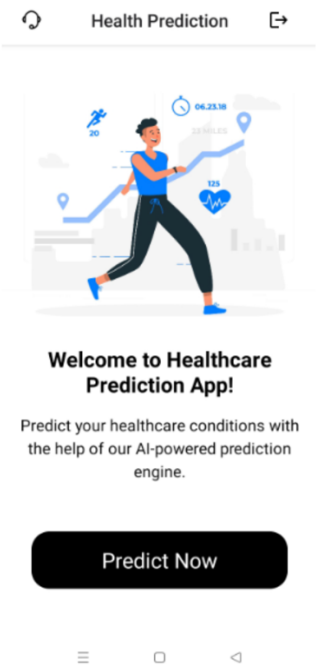

Fig 2: Home page

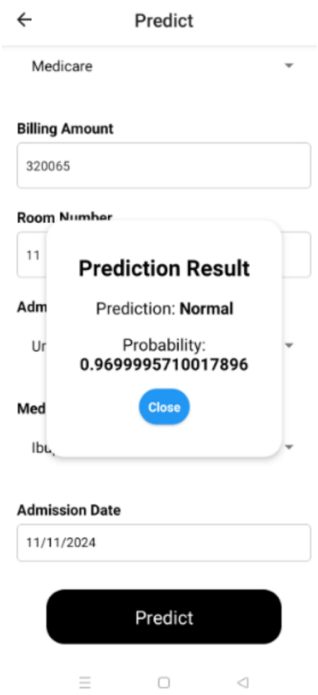

Fig 3: Input form and Predict page

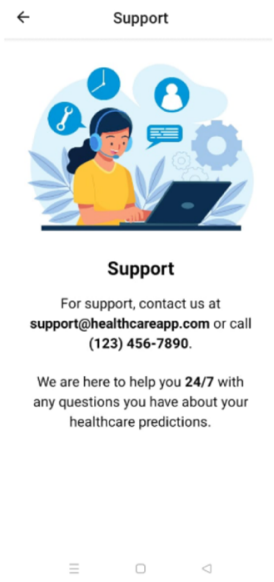

Fig 4: Support page

4 Application Features

4.1 User Authentication

The authentication process uses JWT, which guarantees secure access to the app. It means that users need to sign up with this program using their respective username and password. After successful login, a key called token is created and this can only be used in the subsequent access to the prediction API.

4.2 Machine Learning Predictions

The backend uses a pre-trained machine learning model to classify the label based on input data. The model and scaler are loaded from voting_classifier.pkl and scaler.pkl, respectively. Predictions are categorized as:

1. **Normal**
2. **Inconclusive**
3. **Abnormal**

4.3 Secure Backend Communication

Data exchanged between the frontend and backend is being encrypted by using HTTPS and secured with usage of JWT tokens.

5 Deployment Instructions

5.1 Backend Deployment on Google App Engine

1. Ensure you have installed the Google Cloud SDK and authenticated it.
2. Navigate to the project directory containing app.yaml.
3. Deploy the backend using:

```
gcloud app deploy
```

5.2 Frontend Deployment

For the React Native app, generate and distribute the APK using:

1. Expo's managed workflow with eas build.
2. Upload the APK to an app store (Google Play) for users to download.

6 Troubleshooting

6.1 Backend Issues

1. **Missing Dependencies**: Reinstall the required Python packages with:

```
pip install -r requirements.txt
```

2. **Database Not Found**: Ensure the /tmp/database.db file is created on application startup.

6.2 Frontend Issues

1. **App Not Starting**: Ensure the Expo CLI is correctly installed and dependencies are up to date:

```
npm install
```

2. **Device Connection Issues**: Verify the device is on the same network as the development server.

7 Security Recommendations

1. Use a strong JWT_SECRET_KEY in production.
2. Configure HTTPS for secure communication between the frontend and backend.
3. Store sensitive data (like database credentials) in environment variables.

8 Extending the Application

8.1 Adding Features

1. Enhanced Predictions:

Integrate additional models or update the existing model. Adjust the input form to accommodate new features.

2. **User Roles:** Introduce admin functionalities for managing users.
3. **Data Storage:** Use a more robust database like PostgreSQL for scalability.

8.2 Customization

1. Modify the frontend design using Expo's customizable components.

2. Adjust backend endpoints to fit new business logic.

9 Summary

The Health Prediction Application is an end-to-end system which includes a Flask based backend and the mobile application frontend using React Native for posting, getting, and predicting health data. The backend involves the management of basic services which include user credential services, data managing services, and prediction which is conducted by applying a pre-existing machine learning algorithm. The mobile frontend which was created with Expo and using React Native allows users to navigate the system comfortably. If you want to put your money on the health predictions then the application basically enables

the user to register, log in, enter data and view the results created by machine learning model. This was accomplished on the Google Cloud Platform (GCP) which provides flexibility and accessibility. This application demonstrates how technologies like, Flask, SQLAlchemy, JWT authentication and machine learning models can integrate with mobile apps to give health predictions.

10 References

Flask Documentation. (n.d.). Flask documentation. Retrieved from https://flask.palletsprojects.com/

React Native Documentation. (n.d.). React Native Getting Started Guide. Retrieved from https://reactnative.dev/docs/getting-started

Google Cloud Platform (GCP) Deployment Guide. (n.d.). Retrieved from https://cloud.google.com/

scikit-learn Documentation. (n.d.). Machine Learning library. Retrieved from https://scikit-learn.org/stable/

Expo Documentation. (n.d.). Expo Developer Guide. Retrieved from https://docs.expo.dev/