**Deliverable 4: Project Phase 2**

**CSCE 5430 (Fall 2024)**

**Reasons for Changes from Deliverable 2:**

1. **Security Mechanism Update**: The original plan included implementing **JWT for secure access** to the platform, but this feature was removed due to simplifying authentication and focusing on Firebase Authentication, which provides adequate security.
2. **Removal of Google Fit API Integration**: The original plan included fetching data from **wearables and Google Fit API**. However, since the Google Fit API is discontinued, this feature has been removed. We will focus more on user-provided data rather than relying on third-party data integration from wearables.

A screenshot of a computer

Description automatically generated

we removed the implementation of **JWT for secure access** because using **Firebase Authentication** already provides a secure and reliable way to manage user accounts, which avoids extra complexity. We also had to remove the integration of **Google Fit API** since it is no longer available, which means we can't directly fetch data from wearables. Instead, we shifted to relying on **user-provided health metrics**. To simplify the development and integration, we replaced **JavaScript frameworks like React, Angular, and D3.js** with **Streamlit** and **Plotly**. Streamlit allows us to easily build user interfaces and dashboards, and Plotly helps with interactive data visualization—all within Python, which makes the tech stack more cohesive and easier to maintain. We also decided to remove **AWS Redshift/Athena** and focus on using **Firebase Firestore** for data storage, as it provides everything we need for managing user data without the extra overhead of other services.

**a. Requirements**

**1. Functional Requirements:**

**User Authentication System**

* **User Signup**:
  + Users must be able to register an account by providing an email address, username, password, date of birth, gender, and blood type.
  + Email validation must be implemented to ensure a correct format.
  + Password confirmation is required to verify the password.
  + System should store user details securely in Firebase.
* **User Login**:
  + Users must be able to log in with an email address and password.
  + The system should authenticate using Firebase Authentication.
  + Error handling for invalid credentials should be implemented.

**User Profile Management**

* Users should be able to update their profile information, including age, gender, and blood type.
* User profile data should be securely updated in the Firestore database.

**Data Collection System**

* **Health Metrics Input**:
  + Users must be able to input daily health metrics, including weight, height, blood pressure, heart rate, body temperature, glucose level, oxygen saturation, hours of sleep, and physical activity level.
  + The system should validate all health metrics to ensure that they fall within expected ranges (e.g., valid values for blood pressure, non-negative height and weight).
* **Health Data Submission**:
  + Users should be able to submit their health data daily.
  + The system should store the health data in Firestore in the user's profile.

**Dashboard Analytics**

* **Health Metrics Display**:
  + Users should be able to view key health metrics such as BMI, Blood Pressure, Heart Rate, Body Temperature, Glucose, and Oxygen Level.
  + Data visualization must be provided to display trends in key health metrics (e.g., line charts for heart rate trends).
* **Hypertension Risk Prediction**:
  + The system should predict the user’s hypertension risk using the trained Random Forest model.
  + Predictions must be stored in the user’s profile for reference.

**Recommendation System**

* **Hypertension Recommendations**:
  + The system should provide personalized recommendations based on hypertension risk.
  + Recommendations must include suggestions for diet, exercise, sleep, and other lifestyle factors.

**Chatbot Integration**

* **Health Chatbot**:
  + Users must be able to ask health-related questions using an chatbot.

**Data Update and Access Control**

* Logout functionality must be available for users to securely log out.

**2. Non-Functional Requirements:**

**Security:**

* **User Data Protection**:
  + Passwords must be hashed before storage using generate\_password\_hash().
  + Authentication and data updates should use secure Firebase services to prevent unauthorized access.
  + Health data should be stored in Firestore, ensuring proper access control for each user.
* **Data Validation**:
  + Health data inputs must be validated for acceptable ranges before submission (blood pressure should be in valid format, non-negative values for weight and height).

**Usability:**

* **User-Friendly Interface**:
  + The interface should be intuitive and easy to use, with appropriate prompts for the user during signup, login, and data submission.
  + Provide descriptive error messages for incorrect inputs (e.g., incorrect email format, mismatched passwords).

**Performance:**

* The system should support concurrent users without significant degradation in response time.
* The prediction model should return risk levels in real-time without delays.

**Availability:**

* The system should be accessible across devices, including desktop and mobile devices.
* The dashboard should provide real-time updates for logged-in users.

**3. Technical Requirements:**

**Technology Stack:**

* **Frontend**: Streamlit for UI
* **Backend**: Python for logic and Firebase for data storage
* **Database**: Firebase Firestore for user and health data storage
* **AI/ML Integration**:
  + Random Forest Model for hypertension risk prediction.
* **Third-party Libraries**:
  + streamlit, firebase-admin, joblib, openai, pandas, and plotly for data visualization.

**b). UML design**:

* **Classes**:

**A screenshot of a computer

Description automatically generated**

* + User:
    - Attributes: email, password, username, age, gender, blood\_type, created\_at
    - Methods: signup(), login(), update\_profile()
  + HealthData:
    - Attributes: weight, height, blood\_pressure, heart\_rate, body\_temp, glucose, oxygen, sleep, activity
    - Methods: submit\_data(), get\_daily\_data()
  + HypertensionRiskModel:
    - Attributes: model (Random Forest model)
    - Methods: predict\_risk(), get\_recommendations()
  + Chatbot:
    - Attributes: api\_key (OpenAI)
    - Methods: ask\_question(), generate\_response()
  + Dashboard:
    - Attributes: user\_profile, daily\_data
    - Methods: display\_metrics(), display\_recommendations(), update\_user\_data()
  + Firebase:
    - Attributes: db\_reference
    - Methods: signup\_user(), login\_user(), update\_data(), fetch\_user\_data()
* **Relationships**:
  + User has multiple HealthData.
  + HypertensionRiskModel is associated with Dashboard for risk prediction and recommendations.
  + Chatbot interacts with the Dashboard to answer user questions.

**Sequence Diagram:**

A diagram of a user signup and health data subdumbination process

Description automatically generated

1. **User Signup Process**:
   * User → System: Enters credentials and personal information (email, password, DOB, etc.).
   * System → Firebase: Calls signup\_user().
   * Firebase → System: Registers user and stores data.
   * System → User: Displays success message.
2. **User Data Submission and Prediction**:
   * User → Dashboard: Requests to submit health data.
   * Dashboard → HealthData: Collects and validates health data.
   * Dashboard → Firebase: Updates daily data.
   * Dashboard → HypertensionRiskModel: Calls predict\_risk() with user data.
   * HypertensionRiskModel → Dashboard: Returns prediction and recommendation.
   * Dashboard → User: Displays the recommendations.

**Use Case Diagram:**

A diagram of a diagram

Description automatically generated

* **Use Cases**:
  1. **User Signup/Login**:
     + Actors: User
     + Description: User creates an account or logs in using email and password.
     + Error Case: Invalid email format, password mismatch.
  2. **Health Data Collection**:
     + Actors: User
     + Description: Users enter their health metrics like weight, heart rate, and blood pressure.
     + Error Case: Missing data fields or invalid data formats (e.g., invalid blood pressure).
  3. **Hypertension Risk Prediction**:
     + Actors: User, HypertensionRiskModel
     + Description: System predicts hypertension risk and provides recommendations based on health metrics.
     + Error Case: Prediction model fails due to invalid or incomplete data.

**c. Test Cases (Unit Tests)**

1. **Test Signup Functionality**:
   * **Input**: User provides a valid email, password, confirm password, and other details.
   * **Expected Output**: User account created successfully.
   * **Error Test**: Passwords do not match → Error message.
2. **Test Login Functionality**:
   * **Input**: User enters valid email and password.
   * **Expected Output**: User logged in successfully.
   * **Error Test**: Incorrect email or password → Authentication failed.
3. **Health Data Submission**:
   * **Input**: Valid weight, height, blood pressure, etc.
   * **Expected Output**: Data submitted and saved successfully.
   * **Error Test**: Blood pressure in wrong format → Error message.
4. **Hypertension Prediction Test**:
   * **Input**: BMI, heart rate, systolic\_bp, diastolic\_bp, age.
   * **Expected Output**: Returns risk level (0 or 1) and recommendations.
5. **Chatbot Question Answering**:
   * **Input**: User asks a health-related question.
   * **Expected Output**: Returns a related response.

**d. User Manual**

The user manual should guide users step by step on how to use the application.

**User Registration and Login:**

1. **Sign Up**:
   * Click the "Sign Up" tab.
   * Enter email, username, date of birth, gender, blood type, password, and confirm password.
   * Click "Sign Up".
2. **Log In**:
   * Click the "Login" tab.
   * Enter email and password.
   * Click "Login".

**Entering Health Data:**

1. **Navigate to Data Collection**:
   * After logging in, the data collection form will appear.
   * Enter daily health metrics: weight, height, blood pressure, heart rate, etc.
   * Click "Submit Data".

**Viewing Dashboard:**

* **Health Metrics**:
  + The dashboard displays key metrics like BMI, blood pressure, and others.
  + Recommendations are provided based on the analysis.
* **Chatbot**:
  + Ask questions in the Chatbot section to get answers related to health metrics.

**e. Instructions to Compile/Run the Program**

**1. System Setup**

* **Python Version**: 3.8 or higher.
* **Install Dependencies**: Run in terminal:

pip install streamlit firebase-admin joblib pandas openai plotly scikit-learn

* **Firebase Setup**: Place the Firebase **Service Account Key** JSON file (e.g., ht.json) in project directory.
* **OpenAI API Key**: Replace " \_OPENAI\_API\_KEY" in the code with actual key.

**2. Running the Application**

* **Start the App**: Run the command:
* streamlit run main.py

Open the URL provided (e.g., http://localhost:8501) in browser.

* **Use the App**:
  + **Sign Up/Login**: Register or log in using the provided forms.
  + **Submit Health Data**: Fill out health metrics and view results on the dashboard.
  + **Chatbot**: Interact with the chatbot by typing health-related questions.

**3. Running Unit Tests**

* **Run Tests**:

python -m unittest test\_app.py

**f. Feedback from Code Inspection and Actions Taken**

During the code inspection session, we received feedback that we needed to clearly include **UML diagrams** for better understanding of the system design. Specifically, the feedback highlighted the need for detailed **Class, Sequence, and Use Case diagrams** to illustrate relationships, interactions, and functionality more effectively.

**Actions Taken**:

* We added detailed **UML diagrams** as part of the project documentation:
  + A **Class Diagram** was created to show the different classes (e.g., User, HealthData, Dashboard) and their relationships.
  + A **Sequence Diagram** was added to describe the interactions between users and the system for signup, login, data submission, and hypertension prediction.
  + A **Use Case Diagram** was provided to demonstrate normal and error scenarios for user registration, health data collection, and risk prediction.

**g. Reflection on Accomplishments**

**What Has Been Accomplished**: We successfully developed a health monitoring application that enables users to securely register, input daily health metrics, view detailed analytics, and receive personalized recommendations. We integrated a machine learning model to predict hypertension risk and provided a chatbot for user queries using OpenAI's API. We also transitioned to a simpler tech stack by using Streamlit and Plotly, making development and UI integration more efficient.

* **Firebase Integration**: Using Firebase for authentication and Firestore for data management worked seamlessly, providing a secure and scalable way to manage user data.
* **Machine Learning Model**: The Random Forest model was effectively integrated, delivering real-time predictions with satisfactory accuracy.
* **User Interface**: Streamlit allowed us to create an easy-to-use and interactive interface, which accelerated the development of the dashboard and reduced complexity.

**What Could Be Improved**:

* **Testing Coverage**: More comprehensive automated tests, especially for edge cases, could improve the robustness of the system.
* **Scalability of ML Model**: The current setup loads the model locally, which may limit scalability. Migrating the model to a cloud service or containerizing it could provide better support for a growing number of users.
* **User Experience Enhancements**: While the current UI is functional, future improvements could include more interactive visualizations and animations to enhance the user experience.

**h. Member Contribution Table**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Member Name** | **Contribution Description** | **Overall Contribution (%)** | **Note (if applicable)** | | **Jaideep Tripurani** | Developed the **user authentication** and **profile management** components, integrated **Firebase** for data storage, and worked on testing the functionalities. | 11% | Led integration with **Firebase Authentication**. | | **Ajay Eedara** | Implemented the **data collection system**, including health metric validation, and designed the **data entry form** using **Streamlit**. | 11% | Ensured input validation for accurate data entry. | | **Tagore Hari Prasad Chintamaneni** | Developed the **Random Forest hypertension prediction model**, integrated it into the system, and wrote unit tests for prediction accuracy. | 11% | Conducted model training and **accuracy testing**. | | **Satish Velaga** | Created the **dashboard analytics** using **Plotly** and **Streamlit**, including data visualization components and health metrics display. | 11% | Worked on enhancing **user experience** in the dashboard. | | **Siddhartha Alapati** | Integrated the **chatbot** into the application and worked on its connection to user health metrics for personalized advice. | 11% | Optimized chatbot interactions for personalized advice. | | **Sai Shruthik Errammagari** | Worked on the **recommendation system** by providing detailed dietary and exercise advice based on hypertension risk predictions. | 11% | Ensured recommendations were personalized and actionable. | | **Devendra Kumar Gaddipati** | Developed the **UML diagrams** (Class, Sequence, and Use Case diagrams) and contributed to creating **test cases** for various components. | 11% | Addressed feedback to clearly document UML diagrams. | | **Sai Venkata Manish Lingamallu** | Wrote the **user manual** and created the **instructions for compiling/running** the system, ensuring ease of use for end-users and developers. | 11% | Simplified **user instructions** for better accessibility. | | **Ajay Kumar Aitha** | Assisted in **project documentation**, including sections on **requirements, feedback** received, and **reflection** on project progress and improvements. | 11% | Coordinated the feedback incorporation process. | |