**Project Title: Health Analytics  
Deliverable 4**

**Course: CSCE 5430 (Fall 2024)**

**TEAM 8  
  
  
Team Members**

* Jaideep Tripurani (Lead Developer - Authentication & Security)
* Ajay Eedara (Health Data Engineer)
* Tagore Hari Prasad Chintamaneni (Notifications Specialist)
* Satish Velaga (Dashboard & Analytics Lead)
* Sai Shruthik Errammagari (Bed Management Specialist)
* Devendra Kumar Gaddipati (System Architect & Quality Assurance)
* Sai Venkata Manish Lingamallu (Appointment Management Developer)
* Ajay Kumar Aitha (Backend Integration Specialist)
* Siddhartha Alapati (Access Control Specialist)

**Section A: Report Requirements**

**Introduction:**

The health monitoring application is a secure and user-friendly platform that allows individuals to register, log in, and manage their health metrics conveniently. Utilizing Firebase Authentication for user management and Firestore for data storage, the system enables users to input daily health data, including vital metrics such as weight, blood pressure, and heart rate. The dashboard provides visual insights into health trends, predicts hypertension risk using a Random Forest model, and offers personalized health recommendations and features like doctor appointment scheduling and hospital bed availability.

**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.
    - Password confirmation is required to verify the password.
    - User details should be stored securely in Firebase.
  + **User Login**:
    - Users must be able to log in with an email address and password.
    - Error handling for invalid credentials must be implemented.
* **User Profile Management**:
  + Users should be able to update their profile information, including age, gender, blood type, and email address.
  + 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.
  + **Health Data Submission**: The system should store the health data in Firestore in the user's profile after submission.
* **Health Data 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**: Data visualization must be provided to display trends in key health metrics.
* **Doctor Appointment System**:
  + Users should be able to book doctor appointments by selecting a specialty and available doctor.
  + Appointment availability should be based on the selected doctor's schedule.
  + Users must be able to view scheduled appointments.
* **Bed Availability System**:
  + Users should be able to check the availability of different categories of hospital beds (ICU, General, Private).
  + Users must be able to book a bed if available.
* **Notification System**:
  + Users must receive notifications after booking appointments or beds.
  + Customized health recommendations should be sent via email every hour based on the user's health data.
* **Data Update and Access Control**:
  + Logout functionality must be available for users to securely log out.

**2. Non-Functional Requirements:**

* **Security**:
  + 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 (e.g., valid blood pressure, non-negative values for weight and height).
* **Usability**:
  + 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.
* **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, pandas, plotly, schedule for scheduling email notifications.

**Update Plan:**

**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 to simplify authentication and focus on Firebase Authentication, which provides adequate security.
2. **Removal of Google Fit API Integration**: The original plan included fetching data from wearables and the Google Fit API. 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

1. **Tech Stack Simplification**: JavaScript frameworks like React, Angular, and D3.js have been replaced 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.
2. **Data Storage Simplification**: AWS Redshift/Athena has been removed, and we focus on using Firebase Firestore for data storage, as it provides everything we need for managing user data without extra overhead.

**UML Design:A diagram of a company

Description automatically generated with medium confidence**

**Classes:**

* **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()
* **DoctorAppointmentSystem**:
  + Attributes: doctors, appointments
  + Methods: schedule\_appointment(), check\_availability(), view\_appointments()
* **BedAvailabilitySystem**:
  + Attributes: beds
  + Methods: check\_bed\_availability(), book\_bed()
* **NotificationSystem**:
  + Attributes: email\_address, email\_password
  + Methods: send\_health\_report(), send\_appointment\_notification(), schedule\_hourly\_emails()
* **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.
* DoctorAppointmentSystem and BedAvailabilitySystem interact with Dashboard for booking appointments and beds.
* NotificationSystem sends notifications for various activities..

**Sequence Diagram:A diagram of a company

Description automatically generated with medium confidence**

1. **User Signup Process**:
   * User → System: Enters credentials and personal information.
   * 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 → NotificationSystem: Sends customized recommendations to user via email.
   * Dashboard → User: Displays the recommendations.
3. **Doctor Appointment Scheduling**:
   * User → Dashboard: Requests to schedule a doctor's appointment.
   * Dashboard → DoctorAppointmentSystem: Selects specialty and doctor.
   * DoctorAppointmentSystem → Firebase: Updates appointment details.
   * DoctorAppointmentSystem → NotificationSystem: Sends appointment confirmation via email.
   * Dashboard → User: Displays appointment details.
4. **Bed Booking**:
   * User → Dashboard: Requests to book a bed.
   * Dashboard → BedAvailabilitySystem: Checks bed availability.
   * BedAvailabilitySystem → Firebase: Updates bed booking details.
   * Dashboard → User: Displays bed booking confirmation.

**Use Case Diagram:**

**A diagram of a health data collection

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.
  3. **Doctor Appointment Scheduling**:
     + **Actors**: User, DoctorAppointmentSystem
     + **Description**: User schedules an appointment with a doctor based on availability.
     + **Error Case**: Doctor not available on the selected day.
  4. **Bed Availability and Booking**:
     + **Actors**: User, BedAvailabilitySystem
     + **Description**: User checks bed availability and books a bed if available.
     + **Error Case**: No beds available.

**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. **Doctor Appointment Scheduling**:
   * **Input**: Select specialty, doctor, and appointment date.
   * **Expected Output**: Appointment scheduled successfully.
   * **Error Test**: Doctor not available on selected date → Error message.
5. **Bed Booking Test**:
   * **Input**: Select bed category and enter patient name.
   * **Expected Output**: Bed booked successfully.
   * **Error Test**: No beds available → Error message.

**User Manual:**

**User Registration and Login: A screenshot of a computer

Description automatically generated**

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**:

**A screenshot of a computer

Description automatically generated**

* + Click the "Login" tab.
  + Enter email and password.
  + Click "Login".

**Entering Health Data:**

**A screenshot of a computer

Description automatically generated**

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.A screenshot of a computer

  Description automatically generated
* A screenshot of a computer

  Description automatically generated
* A screenshot of a computer

  Description automatically generated
* **Doctor Appointments**: Users can schedule doctor appointments and view upcoming appointments.A screenshot of a computer

  Description automatically generated
* **Bed Availability**: Users can check bed availability and book a bed if required. A screenshot of a computer

  Description automatically generated
* **Notifications**: Users will receive email notifications for scheduled appointments, and health recommendations.

**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 plotly scikit-learn schedule

* + **Firebase Setup**: Place the Firebase Service Account Key JSON file (e.g., ht.json) in the project directory.
  + **Google Secret Key**: Replace \_GOOGLE\_API\_KEY in the code with the actual key.

1. **Running the Application**:
   * **Start the App**: Run the command:

streamlit run main.py

* + **Open the URL** provided (e.g., http://localhost:8501) in the 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.
    - Book Doctor Appointments: Schedule appointments and check availability.
    - Book Beds: Check bed availability and book as needed.

1. **Running Unit Tests**:
   * **Run Tests**:

python -m unittest test\_app.py

**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**:

* Added detailed UML diagrams as part of the project documentation:
  + **Class Diagram**: Shows the different classes (e.g., User, HealthData, Dashboard) and their relationships.
  + **Sequence Diagram**: Describes the interactions between users and the system for signup, login, data submission, hypertension prediction, appointment scheduling, and bed booking.
  + **Use Case Diagram**: Demonstrates normal and error scenarios for user registration, health data collection, risk prediction, appointment scheduling, and bed booking.

**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, receive personalized recommendations, book doctor appointments, and check hospital bed availability. We integrated a machine learning model to predict hypertension risk and provided a chatbot for user queries using Google 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.
* **Doctor Appointment and Bed Booking**: Implemented a system for scheduling doctor appointments and checking bed availability, enhancing the application's practical utility for users.
* **Notification System**: Users receive email notifications for appointments, bed bookings, and customized health recommendations.

**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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Member Name** | **Role** | **Contribution Description** | **Overall Contribution (%)** | **Note (if applicable)** | | Jaideep Tripurani | Lead Developer (Authentication & Security) | Developed the user authentication and profile management components using Firebase Authentication, integrated Firebase Firestore for secure data storage, and conducted functional testing. | 12% | Led integration with Firebase Authentication and data security features. | | Ajay Eedara | Health Data Engineer | Implemented the health data collection system using Streamlit, including developing forms for collecting health metrics and adding validation logic for data accuracy. | 12% | Ensured robust health data validation for accuracy and completeness. | | Tagore Hari Prasad Chintamaneni | Notifications Specialist | Developed and integrated the notifications feature, including Python scripts to send real-time alerts and reminders to users about appointments and health metrics updates. | 11% | Focused on implementing notification logic and integrating email notifications for users. | | Satish Velaga | Dashboard & Analytics Lead | Created the dashboard analytics using Plotly and Streamlit, developing data visualization components to present key health metrics and ensure usability. | 13% | Enhanced user experience with interactive health metrics visualization. | | Sai Shruthik Errammagari | Bed Management Specialist | Implemented the bed availability feature, developing Python functions to check, book, and manage available beds in different categories, integrating with the dashboard for real-time updates. | 12% | Created Python-based bed booking and management system integrated with the hospital dashboard. | | Devendra Kumar Gaddipati | System Architect & Quality Assurance | Created UML diagrams (Class, Sequence, Use Case) to document the system architecture, contributed to the development of unit test cases, and verified the system's behavior with comprehensive test coverage. | 11% | Addressed documentation feedback to improve UML clarity and implemented testing. | | Sai Venkata Manish Lingamallu | Appointment Management Developer | Developed the doctor appointment booking system using Python, implemented logic to handle doctor availability, and managed scheduling within the dashboard. | 12% | Worked on doctor booking system logic in Python and integrated it with the dashboard. | | Ajay Kumar Aitha | Backend Integration Specialist | Worked on integrating backend services with Firebase Firestore, ensuring efficient data flow, and supporting the seamless operation of health data features. | 11% | Contributed to backend service integration and system stability. | | Siddhartha Alapati | Access Control Specialist | Implemented the logout functionality, session management, and data access control features, ensuring secure and user-friendly handling of user sessions. | 11% | Focused on Python-based session management and access control mechanisms. | |