**Project Title: Health Analytics  
Deliverable 5**

**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 for Phase - 3:**

**1. Functional Requirements:**

**Health Analytics Dashboard**

1. **User Profile Management**:
   * **Fetch User-Specific Health Data**:
     + Retrieve user profile and daily health data ( weight, height, blood pressure) from Firebase Firestore based on a unique identifier (uid).
     + Data should include personal details and time-stamped health metrics for dashboard visualization.
   * **Update Personal Information**:
     + Allow users to update their name, age, gender, blood type, and email address via an intuitive sidebar form.
     + Validate input fields ( age must be numeric and within a valid range).
2. **Health Metrics Visualization**:
   * **Interactive Cards for Metrics**:
     + Display the latest values of key metrics like BMI, blood pressure, heart rate, and oxygen levels.
     + Use visually appealing and color-coded cards for easy differentiation.
   * **Historical Data Visualization**:
     + Provide time-series plots for metrics like blood pressure, heart rate, glucose levels, and sleep duration.
     + Ensure graphs are interactive ( zoom, hover for details) and support dynamic time ranges (daily, weekly, monthly).
   * **Activity Level Display**:
     + Bar charts to represent activity levels over time, aiding users in tracking their physical exertion trends.
3. **Data Integration**:
   * **Daily Health Metrics**:
     + Automatically organize and clean daily health data retrieved from Firestore.
     + Handle missing or invalid data gracefully, skipping entries where height or weight is zero.
   * **BMI Calculation**:
     + Use the formula BMI=weight (kg)height (m)2BMI = \frac{\text{weight (kg)}}{\text{height (m)}^2}BMI=height (m)2weight (kg)​ for each user entry.
     + Display BMI on the dashboard and update it whenever new data is added.
4. **Navigation**:
   * **Page Switching**:
     + Ensure smooth transitions between the health dashboard and the doctor's appointment system using st.session\_state.
   * **Intuitive Layout**:
     + Organize tabs (Vitals, Activity & Sleep, Lab Results) logically to simplify user navigation.

**2. Hypertension Risk Prediction**

1. **Prediction**:
   * **Input Features**:
     + Use the latest user data: BMI, heart rate, systolic and diastolic blood pressure, and age.
   * **Model Usage**:
     + Load the pre-trained hypertension\_risk\_model.pkl using **joblib**.
     + Pass user inputs to the model and receive a binary output:
       - 0: Low-risk.
       - 1: High-risk.
   * **Result Display**:
     + Show risk level on the dashboard as "Low" or "High" with accompanying visual cues ( color-coded labels).
2. **Recommendations**:
   * **Tailored Guidance**:
     + - **Low-Risk**: Focus on general health maintenance.
       - **High-Risk**: Focus on risk mitigation strategies like low-sodium diets or stress reduction.
   * **Categories**:
     + Divide recommendations into actionable areas such as:
       - **Dietary Proteins** (lean chicken, tofu).
       - **Food Choices** (leafy greens, low-sodium options).
       - **Exercise Routine** ( daily moderate workouts).
       - **Mental Wellness** ( meditation, relaxation techniques).
       - **Hydration Needs** ( 8-10 glasses of water).
       - **Sleep Hygiene** ( maintaining 7-8 hours of quality sleep).
3. **Integration**:
   * **Data Retrieval**:
     + Fetch the latest health metrics from Firestore and pass them to the prediction system.
   * **Dynamic Recommendations**:
     + Update and display new recommendations whenever new data is available or predictions change.

**3. Chatbot Integration**

1. **Search Capability**:
   * **Query Processing**:
     + Accept natural language queries from users ( "What should I eat for high blood pressure?").
     + Validate and process queries before submitting to the Google Custom Search API.
   * **API Integration**:
     + Send queries to the Google Custom Search API using stored credentials (GOOGLE\_API\_KEY, SEARCH\_ENGINE\_ID).
     + Parse API responses to extract:
       - **Title**: The headline of the search result.
       - **Snippet**: A brief description of the content.
       - **Link**: A clickable URL to the resource.
   * **Display Results**:
     + Show the top 3 results in a readable format, with clickable links and descriptive snippets.
2. **User Interaction**:
   * **Input Field**:
     + Provide a text input box for users to type their health-related questions.
   * **Response Management**:
     + Display appropriate feedback:
       - Search results if found.
       - Warnings for no results or invalid queries.
       - Error messages for API failures or connectivity issues.
3. **UI Integration**:
   * **Embedded Design**:
     + Position the chatbot seamlessly within the dashboard, styled to match the overall theme.
   * **Submit Button**:
     + Trigger the search operation and display results when the user clicks the "Send Question" button.

**Updated Plan**

**Phase 1: Scope Unchanged**

The scope for Phase 1 is consistent with what was defined in **Deliverable-2**. No modifications or adjustments have been made to the objectives of this phase. All key functionalities originally outlined remain part of Phase 1, ensuring that the foundational features of the system are delivered. These include:

* **Health Analytics Dashboard**:
  + Displays interactive metrics (e.g., BMI, blood pressure, heart rate, oxygen levels).
  + Enables user profile management and seamless navigation between components.
  + Integrates health metrics visualization with time-series trends.
* **Hypertension Risk Prediction**:
  + Implements ML-based risk prediction using the provided data set.
  + Offers personalized recommendations for maintaining or improving health.
* **Chatbot Integration**:
  + Leverages the Google Custom Search API for responding to user queries.
  + Provides an intuitive and responsive interface for real-time interactions.

**UML Design:**

**Class Diagram:**

**A diagram of a health data

Description automatically generated**

 **Classes**:

* User: Represents users with attributes like email, password, and profile details.
* HealthData: Contains health metrics like weight, height, blood pressure, etc.
* Dashboard: Acts as the central hub, linking all components.
* Other classes (DoctorAppointmentSystem, BedAvailabilitySystem, NotificationSystem, Firebase, Chatbot, HypertensionRiskPrediction) provide specific functionalities.

 **Relationships**:

* User interacts with Dashboard directly.
* Dashboard connects to all other components, making it the system’s hub.
* HealthData provides metrics to HypertensionRiskPrediction for risk analysis.

**Sequence Diagram:**A diagram of a health data flow

Description automatically generated

 **Actors & Participants**:

* User: Initiates the interaction by accessing the dashboard.
* Dashboard: Central component orchestrating data flow.
* HealthData: Supplies health metrics for analysis.
* HypertensionRiskPrediction: Processes data to predict risk.

 **Flow**:

* User triggers the interaction.
* Dashboard retrieves data from Firebase and HealthData.
* Data flows to HypertensionRiskPrediction for analysis.
* Results and recommendations are displayed to the user.

Use-case:  
A diagram with text and words

Description automatically generated

**Error Use Cases:**

1. **Login Failed**:
   * Triggered if invalid credentials are provided during the Signup/Login process.
   * Relationship: Signup/Login depends on Login Failed.
2. **Invalid Health Data**:
   * Triggered when the user submits incorrect or incomplete health metrics (e.g., missing weight or invalid blood pressure format).
   * Relationship: Submit Health Data depends on Invalid Health Data.
3. **Appointment Booking Failed**:
   * Triggered when no appointment slots or hospital beds are available during the booking process.
   * Relationship: Manage Appointments depends on Appointment Booking Failed.

**Relationships:**

1. **User**:
   * Interacts only with normal use cases (e.g., Signup/Login, Submit Health Data, Manage Appointments).
   * Does **not** directly interact with error cases.
2. **Internal Error Triggers**:
   * Signup/Login triggers Login Failed internally if credentials are invalid.
   * Submit Health Data triggers Invalid Health Data for incomplete or incorrect input.
   * Manage Appointments triggers Appointment Booking Failed when no slots are available.
3. **Normal Use Case Dependencies**:
   * Submit Health Data connects to Get Hypertension Risk Prediction.
   * Manage Appointments connects to Check Bed Availability and further to Send Notifications.
   * View Dashboard Metrics integrates with Ask Chatbot Questions.

**Test Case 1: User Signup (Normal Case)**

**Description**: Verify that a new user can successfully sign up with valid credentials.  
**Steps**:

1. Navigate to the signup page.
2. Enter a valid email, password, and username.
3. Submit the form.  
   **Expected Result**:

* User is registered successfully.
* A success message is displayed.
* User data is saved in the database (Firebase).

**Test Case 2: User Login with Invalid Credentials (Error Case)**

**Description**: Verify that the system displays an appropriate error when invalid credentials are provided during login.  
**Steps**:

1. Navigate to the login page.
2. Enter an invalid email or password.
3. Attempt to log in.  
   **Expected Result**:

* An error message (e.g., "Invalid credentials") is displayed.
* The user is not logged in.

**Test Case 3: Submit Health Data (Normal Case)**

**Description**: Verify that the user can successfully submit valid health data (e.g., weight, blood pressure).  
**Steps**:

1. Navigate to the health data submission form.
2. Enter valid data for all required fields.
3. Submit the form.  
   **Expected Result**:

* Health data is stored in the Firebase database.
* A confirmation message is displayed.

**Test Case 4: Submit Incomplete Health Data (Error Case)**

**Description**: Verify that the system prevents the submission of incomplete or invalid health data.  
**Steps**:

1. Navigate to the health data submission form.
2. Leave one or more required fields blank (e.g., missing weight).
3. Attempt to submit the form.  
   **Expected Result**:

* An error message (e.g., "All fields are required") is displayed.
* Health data is not stored.

**Test Case 5: Hypertension Risk Prediction (Normal Case)**

**Description**: Verify that the system accurately calculates hypertension risk based on valid health data.  
**Steps**:

1. Submit valid health data (e.g., BMI, blood pressure).
2. Navigate to the hypertension risk prediction page.
3. Request a risk analysis.  
   **Expected Result**:

* The system displays the correct risk level (e.g., "Low Risk" or "High Risk").
* Recommendations (e.g., diet, exercise) are displayed based on the risk level.

**Test Case 6: Appointment Booking with No Available Slots (Error Case)**

**Description**: Verify that the system prevents booking an appointment when no slots are available.  
**Steps**:

1. Navigate to the appointment booking page.
2. Select a doctor and a time slot with no availability.
3. Attempt to book the appointment.  
   **Expected Result**:

* An error message (e.g., "No slots available") is displayed.
* Appointment is not booked.

**Test Case 7: Chatbot Interaction (Normal Case)**

**Description**: Verify that the chatbot provides accurate responses to user queries.  
**Steps**:

1. Navigate to the chatbot interface.
2. Enter a valid query (e.g., "What is a healthy BMI?").
3. Submit the query.  
   **Expected Result**:

* The chatbot retrieves relevant information from the Google API or database.
* A response is displayed to the user.

**Test Case 8: View Dashboard Metrics**

**Description**: Verify that the dashboard displays accurate and up-to-date metrics for the user.  
**Steps**:

1. Log in as a registered user.
2. Navigate to the dashboard.
3. View the displayed metrics (e.g., BMI, heart rate, blood pressure).  
   **Expected Result**:

* Metrics are calculated and displayed accurately.
* The data matches the user's most recent submissions.

**Test Case 9: Chatbot Response to Valid Query**

**Description**: Verify that the chatbot provides a correct response to a health-related query.  
**Steps**:

1. Navigate to the chatbot interface.
2. Enter a valid query (e.g., "How can I reduce hypertension?").
3. Submit the query.  
   **Expected Result**:

* The chatbot retrieves and displays relevant advice or links based on the query.
* The response is meaningful and contextually accurate.

**Test Case 10: Chatbot Handling of Invalid Query**

**Description**: Verify that the chatbot provides a default response for an unrecognized or vague query.  
**Steps**:

1. Navigate to the chatbot interface.
2. Enter an invalid or unclear query (e.g., "asdf1234").
3. Submit the query.  
   **Expected Result**:

* The chatbot responds with a default message (e.g., "I'm sorry, I couldn't understand that. Please try again.").

**Test Case 11: Chatbot API Error Handling**

**Description**: Verify that the chatbot gracefully handles errors when the external API fails or is unreachable.  
**Steps**:

1. Disconnect the system from the internet or simulate an API failure.
2. Enter a valid query in the chatbot.
3. Submit the query.  
   **Expected Result**:

* The chatbot displays an appropriate error message (e.g., "Unable to fetch results. Please try again later.").
* The application does not crash.

**Test Case 12: Chatbot Query History**

**Description**: Verify that the chatbot maintains a history of user queries and responses.  
**Steps**:

1. Enter multiple queries into the chatbot.
2. View the chatbot interaction history.
3. Scroll through past queries and responses.  
   **Expected Result**:

* The chatbot displays the complete interaction history in order.

**Hypertension Model Test Cases**

**Test Case 13: Predict Hypertension Risk with Valid Data**

**Description**: Verify that the hypertension prediction model calculates the correct risk level based on valid health data.  
**Steps**:

1. Submit valid health data (e.g., BMI, blood pressure).
2. Request a risk prediction.  
   **Expected Result**:

* The system calculates the correct risk level (e.g., "Low Risk" or "High Risk").
* Recommendations (e.g., dietary advice) are displayed alongside the risk level.

**Test Case 14: Predict Hypertension Risk with Missing Data**

**Description**: Verify that the hypertension prediction model handles missing or incomplete data gracefully.  
**Steps**:

1. Submit incomplete health data (e.g., missing systolic or diastolic blood pressure).
2. Request a risk prediction.  
   **Expected Result**:

* The system does not calculate the risk.
* An error message is displayed (e.g., "Please complete all required fields before proceeding.").

**Test Case 15: Hypertension Prediction Model Performance**

**Description**: Verify that the prediction model processes requests efficiently, even under high load.  
**Steps**:

1. Simulate multiple users submitting valid health data simultaneously.
2. Request risk predictions for each submission.  
   **Expected Result**:

* The system responds within an acceptable timeframe (e.g., < 2 seconds per request).
* No crashes or significant delays occur.

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.
* **Health Chatbot**:

A screenshot of a computer

Description automatically generated

**Risk Prediction:**

**A screenshot of a computer

Description automatically generated**

**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.
    - Risk Prediction: It predicts the the low and high rish based on health report.

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 and their relationships.
  + **Sequence Diagram**: Describes the interactions between users and the system for signup, login, data submission, hypertension prediction, appointment scheduling,chatbot, dashboard and bed booking.
  + **Use Case Diagram**: Demonstrates normal and error scenarios for user registration, chatbot, health data collection, risk prediction, appointment scheduling, and bed booking.

**Accomplishments**

**Chatbot**

1. Successfully integrated with external APIs (e.g., Google Search API) to provide accurate responses to user queries.
2. Implemented a user-friendly interface for entering health-related queries.
3. Maintains a query-response history, enhancing user engagement and usability.
4. Handles common errors (e.g., invalid queries, API failures) gracefully with meaningful error messages.
5. Supports contextual health advice related to hypertension, fitness, and general wellness.

**Dashboard**

1. Centralized platform for users to view key health metrics, including BMI, blood pressure, and heart rate.
2. Visualizes data trends over time with clear and interactive graphs.
3. Tailors recommendations based on real-time health data and risk levels.
4. Efficiently integrates with various subsystems, including the chatbot, appointment scheduler, and notifications.
5. Provides a seamless experience for managing appointments, checking bed availability, and receiving notifications.

**Hypertension Prediction Model**

1. Successfully predicts hypertension risk levels (e.g., Low, High) based on user-submitted health data (BMI, blood pressure, heart rate).
2. Provides personalized recommendations based on predicted risk levels (e.g., dietary advice, exercise routines).
3. Accurately handles a wide range of valid input data and gracefully rejects invalid or incomplete submissions.
4. Processes multiple concurrent requests efficiently, ensuring low latency even under high user load.
5. Supports evidence-based healthcare by offering data-driven insights for hypertension management.

**What Went Well**

1. **Seamless Integration**:
   * The **Chatbot**, **Dashboard**, and **Hypertension Model** were effectively integrated to provide a unified user experience.
   * Real-time data retrieval and processing from Firebase and external APIs worked without noticeable delays.
2. **User-Friendly Design**:
   * The chatbot interface was intuitive, encouraging user interaction.
   * The dashboard presented health metrics and trends in a visually appealing and comprehensible manner.
3. **Error Handling**:
   * The system gracefully handled common errors, such as invalid health data or unavailable appointment slots.
   * Meaningful error messages improved user trust and experience.
4. **Performance**:
   * The hypertension prediction model maintained fast response times, even with simultaneous user requests.
   * Data visualizations on the dashboard loaded quickly and accurately reflected user data.
5. **Scalability**:
   * The architecture demonstrated the potential for scaling, accommodating additional features or a larger user base.

**Areas of Improvement**

1. **Chatbot**:
   * **Improvement**: Enhance the chatbot's natural language processing (NLP) capabilities to better understand complex queries and provide more contextual responses.
   * **Current Limitation**: Responses to vague or ambiguous queries sometimes default to generic advice.
   * **Next Step**: Integrate advanced NLP frameworks like GPT-based models for improved conversational accuracy.
2. **Dashboard**:
   * **Improvement**: Enhance the accessibility features of the dashboard to ensure compliance with Web Content Accessibility Guidelines (WCAG).
   * **Current Limitation**: Some graphical elements may not be fully usable with screen readers or keyboard navigation.
   * **Next Step**: Conduct accessibility testing and make the necessary UI/UX adjustments.
3. **Hypertension Prediction Model**:
   * **Improvement**: Incorporate additional health factors (e.gcholesterol levels) to improve the model's predictive accuracy.
4. **Error Recovery**:
   * **Improvement**: Implement robust fallback mechanisms for scenarios like API downtime or database failures.
   * **Current Limitation**: While errors are handled, there is no automated recovery mechanism.
5. **User Engagement**:
   * **Improvement**: Introduce gamification or progress tracking to encourage users to submit health data consistently.
   * **Current Limitation**: Users may not interact with the system regularly without proactive engagement strategies.
6. **Performance Metrics**:
   * **Improvement**: Implement a monitoring system to track the performance of the chatbot, dashboard, and prediction model.

**Contribution Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Member Name** | **Role** | **Contribution Description** | **Overall Contribution (%)** | **Note (if applicable)** |
| Jaideep Tripurani | Lead Developer (Authentication & Security) | Developed secure authentication mechanisms and contributed to integrating chatbot APIs for enhanced user interaction. | 12% | Focused on authentication and chatbot security features. |
| Ajay Eedara | Health Data Engineer | Validated health data accuracy, enabling smooth data flow for hypertension risk prediction algorithms. | 12% | Ensured accurate health data collection for the prediction model. |
| Tagore Hari Prasad Chintamaneni | Notifications Specialist | Enabled real-time health alerts and chatbot notifications for improved user engagement. | 11% | Focused on integrating notifications with chatbot responses. |
| Satish Velaga | Dashboard & Analytics Lead | Designed dashboard visualizations, including risk prediction results and chatbot integration, enhancing the user interface. | 13% | Added interactive graphs and chatbot accessibility for seamless interaction. |
| Sai Shruthik Errammagari | Bed Management Specialist | Updated bed booking metrics for display on the dashboard, ensuring real-time updates alongside chatbot and risk prediction features. | 12% | Streamlined bed availability display for dashboard and chatbot integration. |
| Devendra Kumar Gaddipati | System Architect & Quality Assurance | Created UML diagrams and designed system workflows for chatbot, dashboard, and risk prediction model. | 11% | Improved clarity of chatbot interaction and risk prediction diagrams. |
| Sai Venkata Manish Lingamallu | Appointment Management Developer | Ensured appointment booking systems reflected on the dashboard, integrating chatbot queries for scheduling assistance. | 12% | Enhanced chatbot's ability to query appointment schedules and availability. |
| Ajay Kumar Aitha | Backend Integration Specialist | Improved backend data flows for health metrics, enabling real-time hypertension risk predictions and chatbot queries. | 11% | Integrated backend services to ensure stable predictions and chatbot response systems. |
| Siddhartha Alapati | Access Control Specialist | Secured session management for dashboard and chatbot interactions, ensuring controlled access to sensitive prediction data. | 11% | Focused on session security for chatbot and risk prediction data access. |