TASK MANAGEMENT SYSTEM

MEDICATION REMINDER

(PROJECT FINAL REPORT)

Team ID: 12

Team Members and ID:

Dharani Satwika Komaravolu - 1002148504

Sai Puneeth Thummaluru - 1002158041

Rohith Reddy Papareddy - 1002120248

Sai Prasanna Rachakonda – 1002090739

Table of Contents

Part – 1 - Project Initiation

Project Proposal

It is a form of Medication Reminder Task Management System that helps its users keep track of their medication schedule in medication schedules. It will provide capabilities for users to enter their medication information, establish relative dosage alerts, check medication adherence, and be reminded of impending doses. The scheme's goal is to increase medication adherence and users' self-care by reducing the possibility of missed doses. The project's goal is to guarantee that the best technologies are implemented in the medication management process in order to boost general comfort in using technology such as AI, augmented reality, and cloud services while also improving the patient's health.

Client Requirements

Deliver admissible messages that may be personalized or given via alerts. Medication history can be tracked, and information on failed doses and refills may be offered. It also does the following. This will also enable pill identification utilizing augmented reality technology. Reminder Task Management System is intended to assist users monitor and manage their prescription regimens. It will provide a framework for users to register their pharmaceutical information, establish up dosage reminders, track medication adherence, and get notifications about impending doses. The initiative's goal is to improve medication adherence and encourage better health management among users by lowering the likelihood of missed doses. System Objectives.

Objectives

The Medication Reminder Task Management System objectives are the utilization of medication adherence, the improvement of the users' experience, and the development of a safe mass application that can undergo necessary future modifications according to the user needs. The precise goals are:

- **Enhancing compliance to medication**.
  Objective: Ensure consumers are reminded in a special way to ensure they follow their schedule of taking medicine.
  Details: The apps built in notification system will enable the users to set notifications of several medicines on different times of the day. The reminders are intended to be fully portable and fully customizable and this includes the customizable notification sound, customizable frequency and a fully customizable delivery mode according to the requirement of the user. This feature will be vital in responding to user preference, user

needs and general increase in the use of the application in a very effective, user focused manner.
Expected Outcome: Better and easier to handle medication, less stress on the possibility of something wrong happening.

- **Ensuring Security and Scalability:**
  Objective: To design an application that is safe, efficient, and easy to use, that will grow as the need increases yet still safeguard the data and stay compliant.
  Details: The system is innovative so far as scalability is concerned and it is cloud based as well as modular. Moreover, levels of security for instance the HIPAA compliant encryption of personal health information and secure authentication processes without compromising the users' health information integrity. HIPAA and GDPR rules will be taken to consideration because these acts cover data management in healthcare organizations.
  Expected Outcome: The described application will be able to integrate into the capacity for an increasing number of users effectively and also serve as a reliable tool for managing personal health data.

Functional Overview

Scope:

The Medication Reminder Task Management System integrates many functionalities to provide a comprehensive solution for medication adherence. Key features include:

- **Manage medication input and profiles:**
- **Description:** It includes managing medication input to the system and medication profiles. Users may share a lot of different pharmaceutical details like pharmaceutical names, dosages, schedules and instructions. They can also modify basic personal details, reminders, and particular medical notes like:
  - medication alerts or instructions, for example. This tool focuses on ten medications, and through it users can see all their prescription information in one place and easily track them.
  - The profile management system will be of special help for careers who have different dependents and need to follow the prescribed dosage in their patients.

- **Notification and Reminder System:**
  Description: Maximum, personal, and periodic notifications help the user to take each pill at the right time. Multi-dose medications and weekly regimens are recognizable in the reminder system, and the users can set an individual frequency and further a choice to delay the alarm.

Purpose: Due to these features, the system reminds users that the frequency of taking specific medications is different and thus helps to ensure that users do not forget about taking their medications. The workers can choose when and how they should be reminded, which decreases the number of situations where a reminder interferes with the work of the worker.

- **Health Data Integration with AI and AR:**
  Description: AI-based algorithms look into the adherence patterns and summarizes the findings along with recommendations whereas AR-based pill identification offers users the opportunity to confirm medication using their eyes.
  Purpose: The AI feature directly improves the way users develop good habits through personalized recommendations on their adherence behavior, the AR ability makes medication errors less likely since users are capable of identifying the pills. This integration adds further value into the using of the app, as the user is not only reminded of something, but also learns something new about their health.
- **Security and Compliance:**
  Description: The system has features of data encryption, safe sign on and meet the legal frameworks of healthcare for instance HIPAA and GDPR.
  Purpose: In general, security is very important in any health-system related application. To ensure the app users provide legit information, the app has measures of guarding such information as well as adhering to the healthcare privacy rules and regulations so as to gain trust.

Unique Value Proposition

Compared to the existing systems, the Medication Reminder Task Management System is defined by the integration of basic functions related to medication and innovative technologies. Although many reminder systems are merely reliant on notifications, this system also leverages AI for more personalization and AR for medication confirmation as key customer complaint areas are tackled in this space. They mean that it is highly useful for patients who take multiple Prescription Medications and patients who need extra help, which creates a culture of health self-organization.

Cost and Time Estimation

Total Project Duration and Budget

The Medication Reminder Task Management System is effectively designed to implement over a period of two months with sprint system that comprises of four sprints of two weeks each. It also give a systematic pattern from requirement's identification, to its implementation, right up to

deployment; it will also aid to design, test and make sure that the best optimized version of each segment of the system is implemented.

- **Total Duration:** 2 months (8 weeks)
- **Project Team Composition:** 4 core team members:
    - **Project Manager:** Responsible for the project schedules, the project milestones and how they are going to manage the team.
    - **Frontend Developer:** Responsible for the design of interface components of a program and addresses the frontend operations.
    - **Backend Developer:** Covers the development of the server-side parts of the system, the matronizing of the database, and the API incorporation.
    - **Quality Assurance (QA) Tester:** Performs testing of all the projects' phases to determine bugs and take a suitable action.

Monthly Cost per Team Member: $5500

Total Budget: of about $40000 in biochemical for the personnel cost, software cost and testing expenses.

1. COCOMO Model Calculation

Given the complexity of integrating AI and AR into a healthcare system, we classify this project as Semi-Detached. We'll estimate 20,000 lines of code (LOC) for a robust application that includes a user interface, backend logic, data handling, AI-based recommendations, and AR-based pill identification.

1. **Effort Calculation**:
    - Using the formula: Effort (E)=a×(KLOC)b
    - Effort (E)=a×(KLOC)b
    - Estimated **KLOC (Kilo Lines of Code)**: 20000/1000=20 KLOC
    - Plugging in values: E=3.0×(20)^1.12≈3.0×29.59=88.77 person-months

COCOMO RESULTS for Task Management system								
MODE	**"A" variable**	**"B" variable**	**"C" variable**	**"D" variable**	**KLOC**	**EFFORT, (in person-months)**	**DURATION, (in months)**	**STAFFING, (recommended)**
organic	2.4	1.05	2.5	0.38	20.000	55.756	11.522	4.839

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm). Note: the decimal separator is a period.

The final estimates are determined in the following manner:

effort = a*KLOC^b, in person-months, with KLOC = lines of code, (in thousands), and:

staffing = effort/duration

where a has been adjusted by the factors:

Product Attributes	
Required Reliability	1.00 (N)
Database Size	1.00 (N)
Product Complexity	1.00 (N)
Computer Attributes	
Execution Time Constraint	1.00 (N)
Main Storage Constraint	1.00 (N)
Platform Volatility	1.00 (N)
Computer Turnaround Time	1.00 (N)
Personnel Attributes	
Analyst Capability	1.00 (N)
Applications Experience	1.00 (N)
Programmer Capability	1.00 (N)
Platform Experience	1.00 (N)
Programming Language and Tool Experience	1.00 (N)
Project Attributes	
Modern Programming Practices	1.00 (N)
Use of Software Tools	1.00 (N)
Required Development Schedule	1.00 (N)
New (Values are probably wrong)	
Required reusability	1.00 (N)
Documentation match to life-cycle needs	1.00 (N)
Personnel continuity	1.00 (N)
Multisite development	1.00 (N)

2. Function Point Analysis (FPA) Calculation

For FPA, we'll assess the function points based on estimated inputs, outputs, user interactions, internal logical files, and external interface files. The complexity of each function is rated (low, average, or high), and these ratings are multiplied by standard weights.

Component Complexity and Weight Assignments

Component	Count	Complexity	Weight	Function Points (FP)
External Inputs (EI)	10	Average	4	10×4= 40
External Outputs (EO)	8	Average	5	8×5=40
User Inquiries (EQ)	5	High	6	5×6=30
Internal Logical Files (ILF)	4	High	7	4×7=28
External Interface Files (EIF)	3	Low	5	3×5=15

Total Unadjusted Function Points (UFP)

UFP=40+40+30+28+15=153

Complexity Adjustment Factor (CAF)

The Complexity Adjustment Factor (CAF) is based on 14 general system characteristics (such as performance, security, reliability). For a healthcare-related application with enhanced security and reliability requirements, we'll assume a CAF of 1.15.

Adjusted Function Points (AFP)

AFP=UFP×CAF=153×1.15=176

Effort Calculation Based on AFP

Assuming **20 person-hours per function point**:

Effort (Person-Hours)=176×20=3520

Converting person-hours to person-months (160 hours per month):

Effort (Person-Months)=3520/160=22

FPA Summary for Medication Reminder Task Management System

- **Adjusted Function Points (AFP)**: 176
- **Effort**: ~22 person-months

The FPA model suggests a development effort of about 22 person-months. This estimate may lean more towards frontend and user interaction components, as FPA is primarily focused on functional aspects.

3. Lines of Code (LOC) Calculation

Using Lines of Code (LOC) to estimate effort, based on a conservative estimate of 20,000 LOC for the complete application.

Industry Standard Productivity

For complex systems, an industry standard productivity rate of **10 LOC per hour** is often used for accurate estimation.

Effort in Hours

Effort (Hours)=LOC Productivity Rate=20000/10=2000

Effort in Person-Months

Converting hours to person-months (assuming 160 hours per month):

Effort (Person-Months)=2000/160=12.5 person-months

LOC Summary for Medication Reminder Task Management System

- **Effort**: ~12.5 person-months

This LOC-based estimate indicates about 12.5 person-months. Since LOC estimation doesn't account for AI/AR complexities, this estimate might be on the conservative side.

Combined Estimation Summary

The results of each model applied to the Medication Reminder Task Management System are as follows:

Estimation Model	Effort (Person-Months)	Estimated Duration (Months)	Staffing Requirement
COCOMO	~88.77	~14.35	~4 people
Function Point Analysis (FPA)	~22	~5	~4 people
Lines of Code (LOC)	~12.5	~4.5	~3 people

Budget Breakdown

The project's budget is allocated across essential cost categories to ensure efficient use of resources:

Expense Category	Estimated Cost
Developer Salaries	$30,000
Software Licenses	$2,000
Testing Tools	$1,500
Miscellaneous/Contingency	$1,000

Part – 2 - Project Planning

Work Breakdown Structure (WBS)

1: Project Initiation

- 1.1: Define Project Scope
- 1.2: Feasibility Study
- 1.3: Project Approval and Team Formation
- 1.4: Initial Risk Assessment

2: Planning

- 2.1: Requirements Gathering
- 2.2: Resource Planning
- 2.3: Project Schedule & Gantt Chart
- 2.4: Risk Management Plan
- 2.5: Cost Estimation

3: Execution

- 3.1: Prototyping
  - 3.1.1: UI/UX Design
  - 3.1.2: Prototype Development
- 3.2: Core Development
  - 3.2.1: Frontend Development
  - 3.2.2: Backend Development
- 3.3: Advanced Features Development
  - 3.3.1: AI Recommendations
  - 3.3.2: AR Integration
  - 3.3.3: Voice Command Integration

4: Monitoring and Control

4.1: Progress Reviews

4.2: Testing and Quality Assurance

4.2.1: Unit Testing

4.2.2: Integration Testing

4.2.3: User Acceptance Testing

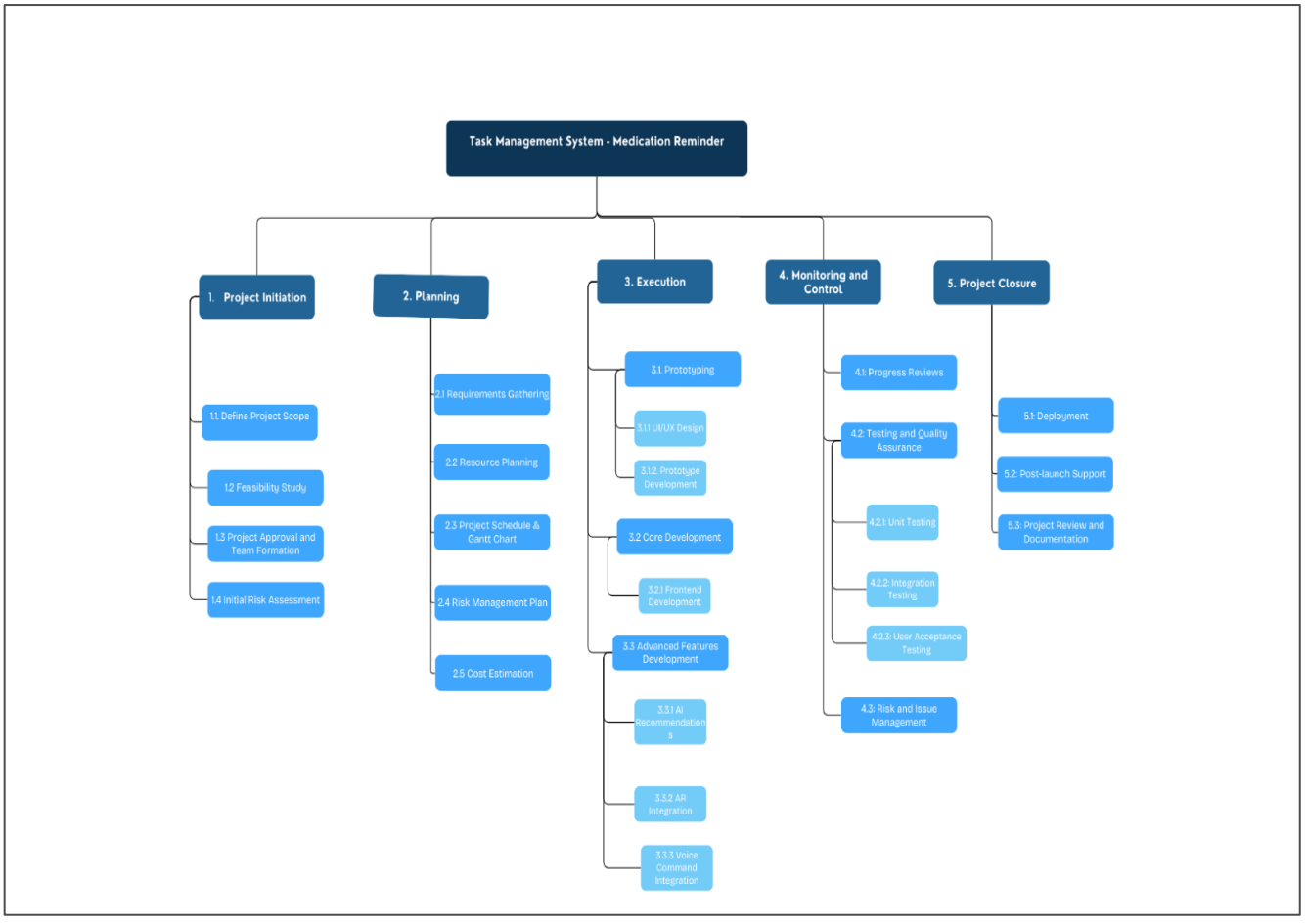
4.3: Risk and Issue Management

5: Project Closure

5.1: Deployment

5.2: Post-launch Support

5.3: Project Review and Documentation

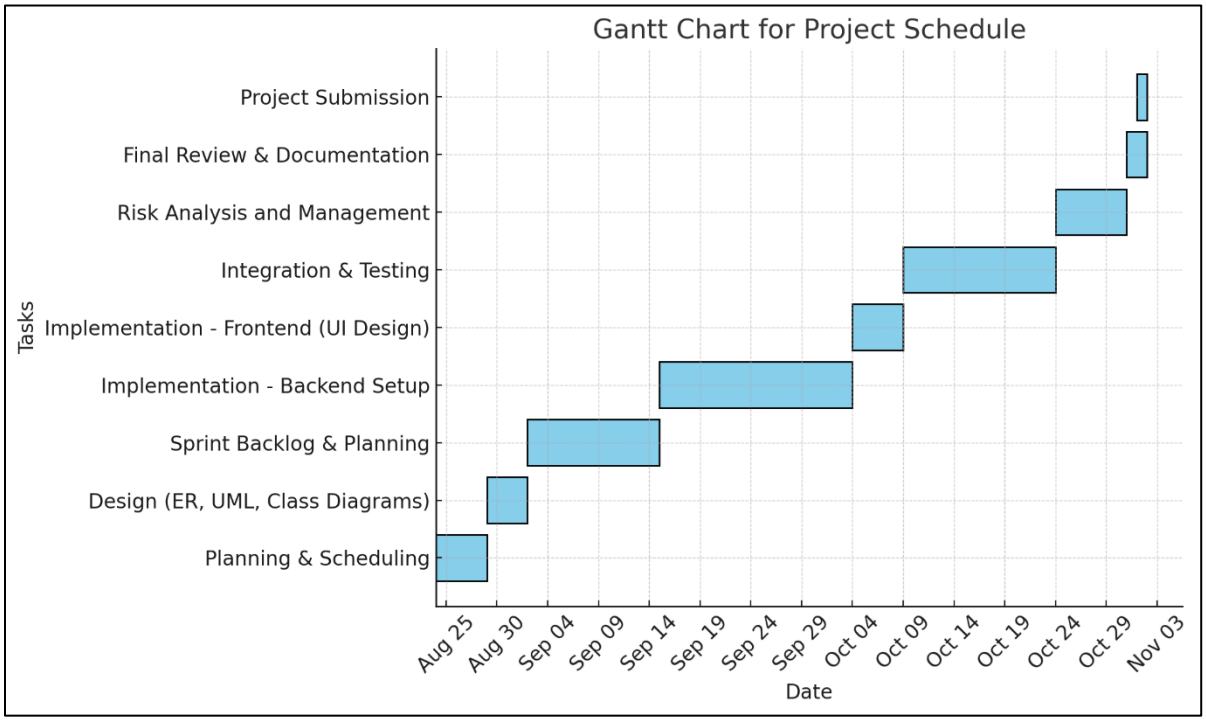


Project Schedule

The project is organized into four 2-week sprints, each focused on a specific aspect of development. The project schedule below outlines the primary objective and activities for each sprint, along with the timeline.

Sprint	Objective	Details	Timeline
Sprint 1	Requirement Gathering & Setup	Requirements documentation, wireframe design, environment setup	Weeks 1–2
Sprint 2	Core Functionality Development	User registration, medication input, notification setup	Weeks 3–4
Sprint 3	UI & Notification Development	UI refinement, usability testing, reminder integration	Weeks 5–6
Sprint 4	Final Testing & Deployment	System testing, documentation, deployment	Weeks 7–8

Gantt chart:



Sprint Planning

- **Sprint Duration -** The sprint will last 2 weeks each**.**
- **Total Sprints** - In the 2-month period the project will be divided into 4 sprints**.**

Sprint Backlog Overview

Sprint 1: Requirements Gathering and Initial Setup

Duration: 2 weeks

Goals:

- Gather the detailed user requirements.
- Set up the project environment and all the repositories.

Backlog Items:

- Requirement gathering done through stakeholder interviews.
- Users personal and user stories creation.
- Establish project management tool (Jira) and version control (GitHub).
- The initial wire frames for UI design.
- A team meeting is performed to review gathered requirements.

Team Responsibilities:

- Product Owner: Lead requirements gathering.
- Developers: Help to setup the environment and also help in creating wireframes.
- Scrum Master: Help to establish a meeting, gain first progress results.

Sprint 2: **Core Functionality Development**

Duration: 2 weeks

Goals:

- Develop core functionalities for medication input and reminder settings.

Backlog Items:

- Register and login the users.
- Add medication input facility for updating medication information.
- Set up for reminder alerts.
- Create a rudimentary database to record user and medication information.
- Test unit basic functionalities.

Team Responsibilities:

- Developers: They primarily focus on developing, testing, and reviewing fundamental functionality.
- Product Owner: Review and gather feedback.
- Scrum Master: Facilitate daily stand-ups and sprint reviews..

Sprint 3: User Interface and Notification System

Duration: 2 weeks

Goals:

- Enhance the overall user interface and implement the notification system.

Backlog Items:

- Develop a user interface for medication history and reminders.
- For reminders, consider using push notifications (such as Firebase).
- Allow for customizable reminders, including sound and time options.
- How to do usability testing with a small number of users.
- You will respond and adjust properly.

Team Responsibilities:

- Developer: Provide a user interface that includes notification functions.
- Product Owner: Test users and get feedback
- Scrum Master: As a scrum master you must ensure the team is on time.

Sprint 4: Final Touches and Deployment

Duration: 2 weeks

Goals:

- Come finalizing the application and preparing to deploy.

Backlog Items:

- First, integrate and test the system. Then, do final testing.
- Deploy the app to app stores in a 'in progress' condition.

- Create user documentation and help resources.
- Consider a marketing approach for user acquisition.
- You conduct a retrospective meeting to analyse the project process.

Team Responsibilities:

- Developers: Handle code, testing, and deployment.
- Product Owner: As Product Owner, you are responsible for documenting and developing marketing strategies.
- Scrum Master: Facilitate retrospectives and crowdsourcing. Lessons Learnt

Part – 3 - Risk Management

Risk Management

It is suggested that risk management is essential to the Medication Reminder Task Management System. This particular project, being a healthcare centered one, assumes working with personal data, integration of novel technologies into simple interface which will be comprehensible for majority of inhabitants of the tunnel, who can hardly be called information society active members. The following is a risk identification matrix according to risk type and the subsequent action that has to be taken.

Risk Identification

The following risks have been identified based on a thorough analysis of project objectives, technology dependencies, user requirements, and healthcare compliance standards:

Risk	Category	Description
API Integration Challenges	Technical	There are possible problems regarding integration process with the databases of healthcare and medications for proper data processing.
Scope Creep	Project Management	Use of further features by users could complicate work and take more time than planned.
Staff Turnover	Resource	Such change threat implies that; Different players may be, involved in carrying out projects resulting to change of key personnel which may cause project instability and also lead to project delay in terms of deliverables.

Budget Overruns	Financial	Extra charges, for instance, costs of license fees for the software, or more hours in developing the software than was planned could hamper the budget.
Low User Adoption	User Experience	Some issues may include: The system may have a complex interface and/or less user friendly, this produces a negative effect towards securing the adoption of the system by end users especially the elderly.
Data Security Breaches	Security	If unauthorized personnel get access to the user data there is potential risk looming in terms of legal implications and reputational damage by poisoning health information.
Compliance with Regulations	Legal	Noncompliance with policies as HIPAA or GDPR result in fines and legal prosecutions.
System Downtime	Tech Organizational	Frequent or extended system downtime could impact productivity and access to services, potentially affecting customer satisfaction and operations.
Market Competition	External Risks	New competitors or changing market dynamics could pose a risk to the adoption and growth of the system, leading to revenue and market share loss.

Each risk is expanded with regards to the probability, consequence and the corresponding risk control procedure to aspire for project stability, cost containment and highly satisfied users.

Risk Analysis

This risk matrix classify each identified risk according to its possibility and consequences on project performance that can help decision makers in project management.

Risk	Likelihood	Impact	Category
API Integration Issues	Medium	High	Technical
Scope Creep	High	Medium	Project Management
Staff Turnover	Medium	High	Resource
Budget Overruns	Medium	High	Financial
User Adoption	Medium	High	User Experience
Data Security Breaches	Medium	High	Security
Regulatory Compliance	Low	High	Legal

It also helps in determining those risks that should be responded to because they are likely to occur and will have a negative impact on an organization as soon as they occur The matrix also shows those areas where prevention can help to namely, decrease the probability of their occurrence.

Risk Mitigation Plan

As for each of the risks mentioned in the project, the risk management plan for the given specific risk was designed in order not to occur. The risk treatment strategies are supposed to minimize risks as far as is possible and if some risks cannot be prevented then at least they should be expected and planned for.

1. API Integration Challenges:

Risk Details: Other health care APIs or medication databases may be connected causing some compatibility issues, may be slow to response and may differ in the results achieved.

Mitigation Strategy:

- Preemptive Testing: It can be possible to make API testing from the development stage to discover format issue, connection fluctuation, or rate constraints among other vices.
- Comprehensive Documentation: Have clear, detailed needs, dependencies and informed APIs on board for best practices.
- Fallback Mechanisms: Store the cache and have backups for data that doesn't necessarily need to be highly responsive via the API.

2. Scope Creep

Risk Details: Expansion of more features beyond the necessary ones for a project might only hold back the development process or its costs may escalate.

Mitigation Strategy:

- **Clear Requirement Definition:** It is suggested that one maintains a an appropriately detailed document that captures project scope and, ideally, gain sign off from stakeholders during project initiation phase.
- **Change Control Process:** For feature request, establishing RBAC, the process matrix for Change request evaluation process, and differentiate between foundational change request and nonfoundational change request.

- **Regular Stakeholder Communication:** What to do again: In essence, one should have face-to-face contact with stakeholders at least twice two weeks with an aim of evaluating the probable scope changes as well as change requests with emphasis being made on the fact that newer additional features influence both time and money.

3. Staff Turnover

Risk Details: They recommend not having a backend developer available or a project manager as this may well throw the timescale of the project off and information gaps can be apparent.

Mitigation Strategy:

- **Cross-Training:** Overlap the skills among the team members or have every member worked or deal with the others' work often enough to avoid being crippled by the absence of a particular person.
- **Detailed Documentation:** Ensure that proper documentation about structure of codes, compilers and linkers, development and test process is done well in order to ease the handover process.
- **Backup Staffing Plan:** The internal candidates that one might consider and some external candidates via a talent acquisition agency should be found in case there is a need to replace any person within the organization.

4. Budget Overruns

Risk Details: Contingent expenses like; license fees that have been wished to be built in the current version could be expensive compared to initial estimates or more development time than anticipated.

Mitigation Strategy:

- **Contingency Budget:** Running a contingency budget (10-15%) helps to transfer the risk of unexpected costs that may appear in a project's work.
- **Monthly Budget Reviews**: Carry out regular monthly spend reviews in order to be able to immediately see the difference between the actual spend and the forecast spend.
- **Vendor Negotiations:** Licensing fees and costs of reoffer software shall be settled directly during the contracting process and at aggregate price levels with special

emphasis on volume bonuses or special discounts which may be obtained by covering more of the software.

5. Low User Adoption

Risk Details: If the interface is clumsy, or not easy to navigate, some users may not quickly take to it, especially if they are elderly people or have low IT literacy.

Mitigation Strategy:

- **User-Centered Design:** Introduce user experience design, and have the elders and the caregivers give feedback after using the design at different stages.
- **Accessibility Features:** Nevertheless, the access to additional options for enhanced usability should be provided to all the users, including the ones who need assisted browsing in terms of easy font, interfaces with limited number of links, and voice control.
- **Comprehensive Onboarding:** The development of an in-app tutorial to enable the first time user to have an ease time when using the system since it has easy controls.

6. Data Security Breaches

Risk Details: In light of this we can deduce that health data is sensitive, with its exposure or loss would mean that the privacy of users would be compromised, legal action could be taken against the project and the reputation of the project could go down.

Mitigation Strategy:

- **Data Encryption:** Secure all users data whether in storage and during transfer to another point to enhance its security**.**
- **Regular Security Audits:** Conduct security check on a regular basis to review possible breaches such as through penetration testing, code review, and access control testing.
- **Role-Based Access Control (RBAC):** RBAC must be enforced so that a certain user and personnel can only access certain data or features.

7. Compliance with Regulations

Risk Details: The project has to adhere to legal requirements like HIPAA or GDPR for instance else it would face penalties.

Mitigation Strategy:

- Legal Consultation: Consult a lawyer to understand HIPAA and GDPR laws concerning health data to find out right at the outset if the non-compliance or compliance exists at all for the app.
- Data Minimization: Keep only identity info that one is obligatory for the site functioning while focusing on user data protection.
- Compliance Training: Make sure all workers are compliant with the training required to inform them of what the law requires, how data will be processed, and roles assumed.

Part – 4 - Project Quality Assurance

Quality Assurance Overview

QA which ensures that the final product delivering to the end user complies with all stipulated user and technical requirements is a consideration in development of the Medication Reminder Task Management System. Since the system operates on health data, this has to satisfy all the security, reliability, and usability scenario. This Quality Assurance methodology for the project is founded on cycles of test, refine, assess, and test in every developmental phase. This helps in achieving a perfect user experience, reduces risks of hitches and makes certain that any feature developed will be as useful as was intended.

Quality Assurance Measures

The following elements are part of the Medication Reminder Task Management System's quality assurance measures:

Evaluation of Performance:

- Goal: To prevent a large latency or a lack of availability when demand is high, that is, during peak working hours, the number of users of the application at a given time and other usage scenarios.
- Methodology: Load and stress tests are employed to evaluate the ability of the proposed system and to ensure the provision of high speed reactions. Tools such as JMeter or LoadRunner that imitates heavy traffic is clearly measuring total impact on server response rate and affirms throughput.

- **Anticipated Result**: Even during periods of high usage, the application should react in reasonable amounts of time, and it should continue to be responsive as the user base expands.

Security Testing:

- **Goal:** Because health data is sensitive, safeguarding against illegal access, data leaks, and potential security flaws is essential.
- **Methodology**: Regular penetration testing is a method to find any weaknesses and secure authentication procedures that are examples of security measures. To prevent any unwanted access to the sensitive user data, data encryption is used both during transit and at rest.
- **Anticipated Result:** The program should show resistance to different types of assaults, guaranteeing the security of user data.

Usability Testing:

- **Goal:** Verify that the program is very easy to use, specially, for the older users or those with less technical expertise, and that all of its capabilities are accessible and easily understood.
- **Methodology:** To get input on the accessibility features, layout, and navigation, usability testing is carried out with actual users from various demographic groups. Test scenarios include using the pill identification tool, modifying notification settings, and setting up reminders.
- **Anticipated Result:** The program should be simple to use, with intuitive navigation and clear instructions. All users should find it easier to use accessibility features including voice command choices and font size adjustments.

Reliability Testing:

- **Goal:** To guarantee the system's stability and dependability, particularly for crucial functions like medication tracking and alerts.
- **Approach:** Reliability testing verifies fault tolerance, error rates, and system uptime. For essential features, like the reminder notifications, redundancy is used to make sure they continue to work even in the event that other system components have problems.
- **Anticipated Result:** The system ought to exhibit excellent dependability, with little downtime and steady operation across sessions.

Maintainability:

- **Goal:** To create a system that can be easily updated, scaled, and troubleshooted, allowing for long-term usefulness of the system with little technological debt.
- **Methodology:** Code reviews, modular coding standards, and detailed documentation promote ease of maintenance. The codebase is modularly built to allow for all the rapid upgrades or the incorporation of new features while maintaining all of the existing functionality.
- **Expected Outcome:** The application's code should be well-organized, documented, and modular to facilitate promising future updates.

Evaluation Metrics

To establish whether the system satisfies the demands of the client, many evaluation metrics will be utilized to evaluate the application's overall performance, user happiness, and security.

- **Improving User Adherence**: - Compare user adherence rates before and after utilizing the system to assess increases in medication consistency and frequency.
- **Expected results:** Increased adherence rates throughout time, showing app efficacy.
- Use post-usage questionnaires to assess user satisfaction, with an emphasis on ease of use, reliability of application, and relevant reminders.
- **Expected outcome**: High satisfaction, especially among older users and those using several medications.
- Use app load times, reminder response times, and backend processing speeds to assess system performance.
- **Expected Outcome**: The app should respond within 1-2 seconds for most operations, with minimal lag during peak usage.
- Ensure compliance with HIPAA, GDPR, and data protection regulations through frequent security audits (Metric).
- **Error Rate:** The expected outcome is full compliance with healthcare data legislation and ethical data management standards.
- **Metric:** Monitor the frequency and types of mistakes reported, trying to reduce them through iterative changes.
- **Expected Outcome:** Reduced mistake rates over time when identified issues are addressed and remedied.
- Measure system uptime and dependability under standard and heavy traffic situations.
- **Expected Outcome**: The expected outcome is a dependability rate of 99.5% or better, with 0% failure rate for important operations such as reminders.

The systematic design of the drug-reminder supplemental-task management system covers its overall architecture, the data models it needs to include and interface layout in order to operate, scale, and be 'user accessible'. This explains what parts go where, the basic system layout, and how it will be implemented. It emphasizes usability, security, and adaptability as a way to reduce the amount of ongoing work for maintenance and future updates, data models, and interface layout required to provide functionality, scalability, and user accessibility. This section examines the structural components and technology decisions that serve as the system's basis, as well as the interactions between data items. The design decisions prioritize usability, security, and flexibility to allow for easy maintenance and future upgrades.

System Architecture

The layers and modules of the Medication Reminder Task Management System are convenient by separation of concerns and data processing speed. Such a mode also enables bridging of the gaps between its artificial intelligence and augmented reality operations. The layers of the architecture are including the presentation layer, the application layer, the data layer, and the advanced modules layer which includes artificial intelligence and augmented reality capability.

Presentation layer (frontend)

The frontend is the place in which all users of the system come into contact with the application. This app was developed with React Native which guarantees its compatibility with iOS and Android devices. The format of the interface developed herein is simple, straight forward as well as universally navigable for elder and/or technophobia patients. Key components include:

- **The User Registration/Login Screen** requires usernames, passwords, and optional two-factor authentication to provide safe access.
- **The dashboard** displays planned prescriptions, forthcoming reminders, and adherence data. Users may get an overview of their drugs and monitor their adherence at a glance.
- **Medication Management Interface:** Users may enter, edit, or remove medication information such as name, dosage, schedule, and special instructions.
- **Metric:** Users may adjust reminder parameters like as frequency, tone, and timing.
- **Pill Identification (AR):** Users may use their camera to scan a pill and ensure that they are taking the correct medication.

Application Layer (Backend)

The backend, which is based on Laravel (PHP)--tackles server-side activities, business rules, and data processes of all kinds. Analyzes incoming requests from the front end, gives advice using AI, takes user data and sends prompts associate this layer. Key features include:

- API management: A RESTful API allows for safe data interchange between the frontend and backend, including all CRUD activities (Create, Read, Update, Delete) for user profiles, prescription records, and reminders.
- The notification system integrates with third-party services like Firebase Cloud Messaging to give timely reminders and notifications.
- Business logic implements pharmaceutical adherence rules, including missing dose calculation, adherence history reporting, and user schedule management.
- Data Security and Compliance: Encrypts data in transit and at rest to fulfil HIPAA and GDPR regulations.

Data Layer (Database)

For the Data Layer see where user profiles are kept; consider medication records, history of adherence or anything else helpful. We use MySQL/MariaDB to run a scalable relational database. Each relationship and table has been tuned so that critical data can be accessed quickly but with reasonable cost in time to maintain data integrity. The Users Table holds user data including login credentials, personal information and role-based access.

- Medications Table: Provides full information about each medicine, including name, dose, schedule, and unique identification.
- The Reminders Table records all medication-related reminders, allowing the system to track missed doses.
- The Adherence History Table records missed or rescheduled doses for examination by AI modules.

Advanced Modules Layer (AI & AR Integration)

This layer includes sophisticated functions aimed to improve the user experience, such as personalized medicine suggestions enabled by Artificial Intelligence (AI) and Augmented Reality (AR) for pill recognition.

- The AI Recommendation Engine, powered by TensorFlow, analyses adherence trends, predicts possible concerns, and provides recommendations for better prescription scheduling.
- Vuforia SDK is used for AR-based pill identification, allowing users to visually identify pills using their camera. This function is intended for customers who take various drugs and helps to reduce the possibility of medication mistakes.

Part – 5 – Design

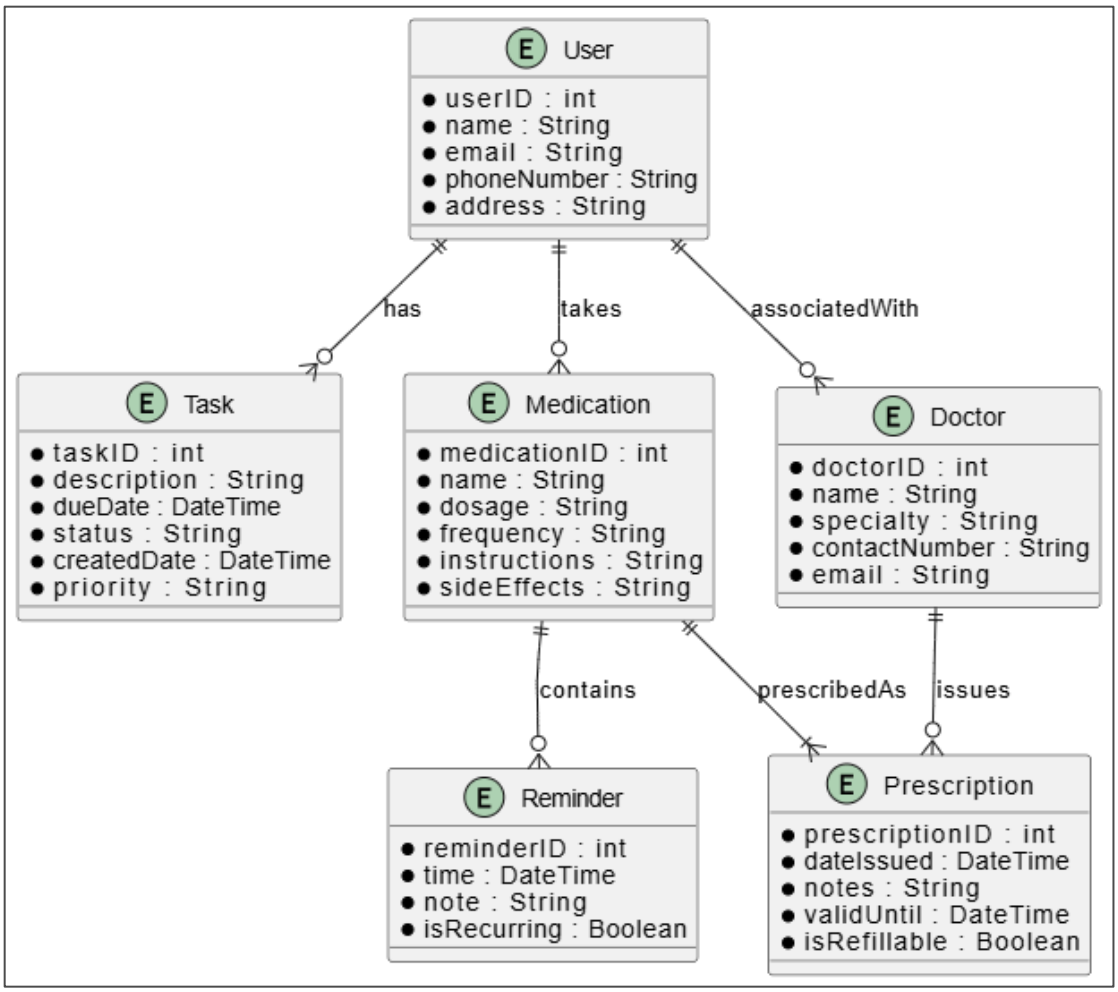
Data Modeling

Data modelling is an important aspect of system design because it describes how data entities interact with one another. The system's main data models are represented by an Entity-Relationship (ER) Diagram and a UML Class Diagram.

ER Diagram

The ER Diagram depicts the links between the system's core entities, ensuring that data is efficiently organized for quick retrieval and modification.

- **Primary Entities**:
    - User information is stored, including ID, name, contact details, login credentials, and notification choices.
    - Medication information include medication ID, name, dose, directions, and user ID.
    - Reminders are linked to medicine and user entities and provide reminder ID, planned time, frequency, and status (e.g., sent, snoozed, or missed).
    - Adherence History: Tracks adherence events, such as missing or delayed doses, and correlates them with the user and medicine.
    - Optional entity "Doctor" allows users to include their physician's information for reference or sharing adherence reports.
- **Relationships**:
    - Each user may have numerous drugs, resulting in a one-to-many link between user and medication.
    - Drug Reminder: Each drug can have several reminders, creating a one-to-many connection.
    - User-Adherence History: Users can trace their adherence over time by adding various entries.
    - Doctors can prescribe several drugs, resulting in a many-to-one connection for data linking.



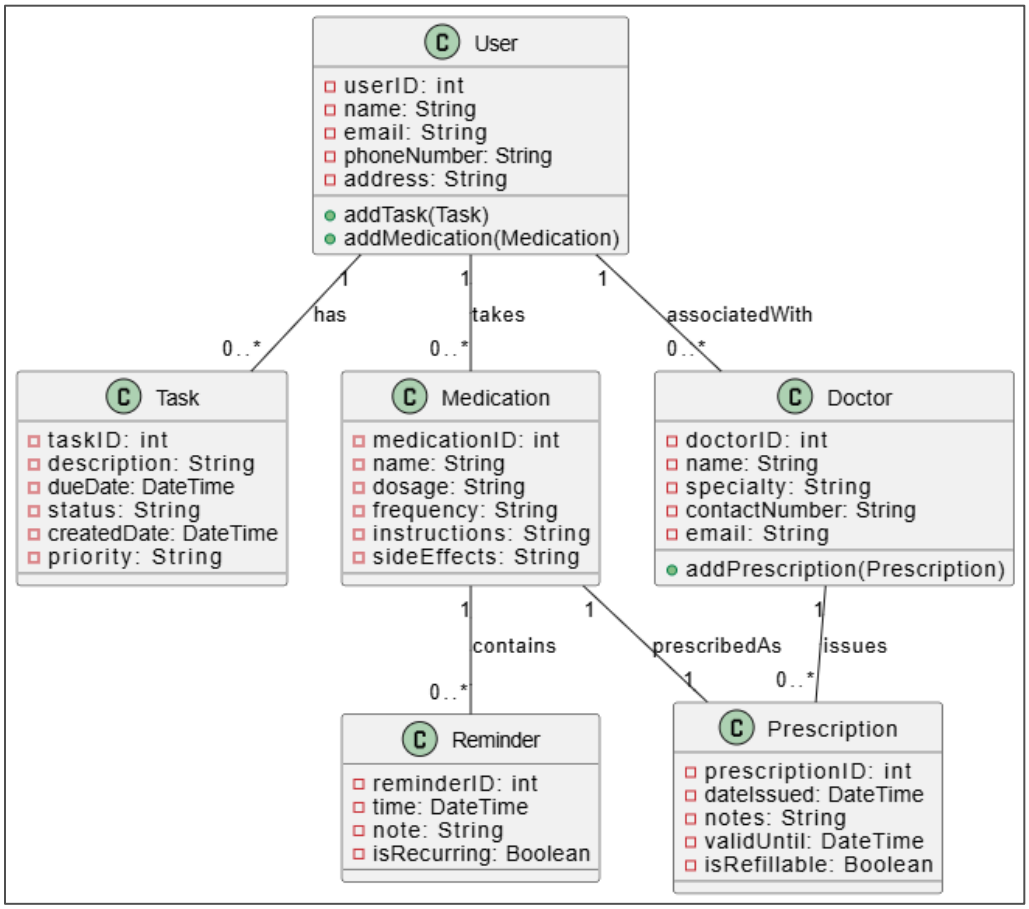
UML Class Diagram

The UML Class Diagram defines the primary classes in the application, showcasing the attributes and methods, as well as the relationships between all the attribute and entities.

- **User Class**:
  - **Attributes**: userID, name, email, password, contactInfo, preferences.
  - **Methods**: registerUser(), loginUser(), updateProfile(), setPreferences().
- **Medication Class**:
  - **Attributes**: medicationID, name, dosage, frequency, instructions, userID.
  - **Methods**: addMedication(), updateMedication(), deleteMedication(), getMedicationInfo().
- **Reminder Class**:
  - **Attributes**: reminderID, time, status, medicationID.
  - **Methods**: createReminder(), sendReminder(), snoozeReminder(), updateStatus().
- **AdherenceHistory Class**:
  - **Attributes**: adherenceID, date, status (taken/missed), notes, userID.

- **Methods**: logAdherence(), updateAdherence(), retrieveHistory().

- **Doctor Class** (optional):
  - **Attributes**: doctorID, name, contact, specialty.
  - **Methods**: addDoctor(), updateDoctor(), associateWithMedication().



Interface Design

The user interface is created with simplicity and convenience of use in mind, especially as many users may be technologically inexperienced or have accessibility requirements. The design includes big icons, understandable typefaces, voice command capabilities, and a simplified layout.

1. **Login and Registration Screen:**
   - **Features:** Provides secure login options with two-factor authentication and optional biometric login (for compatible devices).
   - **Purpose:** Ensuring authorized access and protecting user information.
2. **Dashboard:**
   - **Features:** Provides a central display of impending reminders, daily medication schedule, and adherence summary.

- **Purpose:** Summarizes crucial information to help users track and organize their pharmaceutical habit.

3. **Medication Management Screen**

    - **Features:** Manage drugs with areas for dose, schedule, directions, and special notes.
    - **Purpose:** Facilitates drug management for complicated regimens.

4. **Reminder Settings Screen**
    - **Features:** Customize notification tones, snooze choices, and reminder frequency. provides previews of notification forms.
    - **Purpose:** Customize reminders based on user choices, increasing system adaptability.

5. **AR-Based Pill Identification Screen**
    - **Features:** Uses the device's camera to scan and identify pills, providing visual confirmation to users.
    - **Purpose:** Helps prevent prescription mistakes by visually validating each tablet, especially for those taking various medications.

6. **Profile and Settings Screen**

    - **Features:** It includes updating personal information, managing security settings, seeing adherence history, and exporting reports.
    - **Purpose:** Provides users with control over their data and security, allowing them to share adherence records with healthcare providers as required.

Design Principles

- **Accessibility:** The app's accessibility features, such as customisable font size, high-contrast colours, and voice commands, make it easy to use for everyone, including the elderly.
- **Usability:** Minimalistic design with straightforward navigation to reduce unwanted clicks and alternatives.
- **Scalability:** Modular architecture allows for future additions without considerable reworking of old code, making the program flexible as new features are created.
- **Security:** Encrypted user interactions and healthcare-grade data security ensure user confidence.

Part – 6 – Implementation

The Medication Reminder Project Scheduling System was created specifically to satisfy routine needs. Its modern and efficient technical stack makes sure that dependability, ease of maintenance and scalability are all taken into account. This section introduces you to the system implementation methodology: each development stage, the tools and technologies used and how plans are made for deployment in order that operation should be smooth and easily available to end user community.

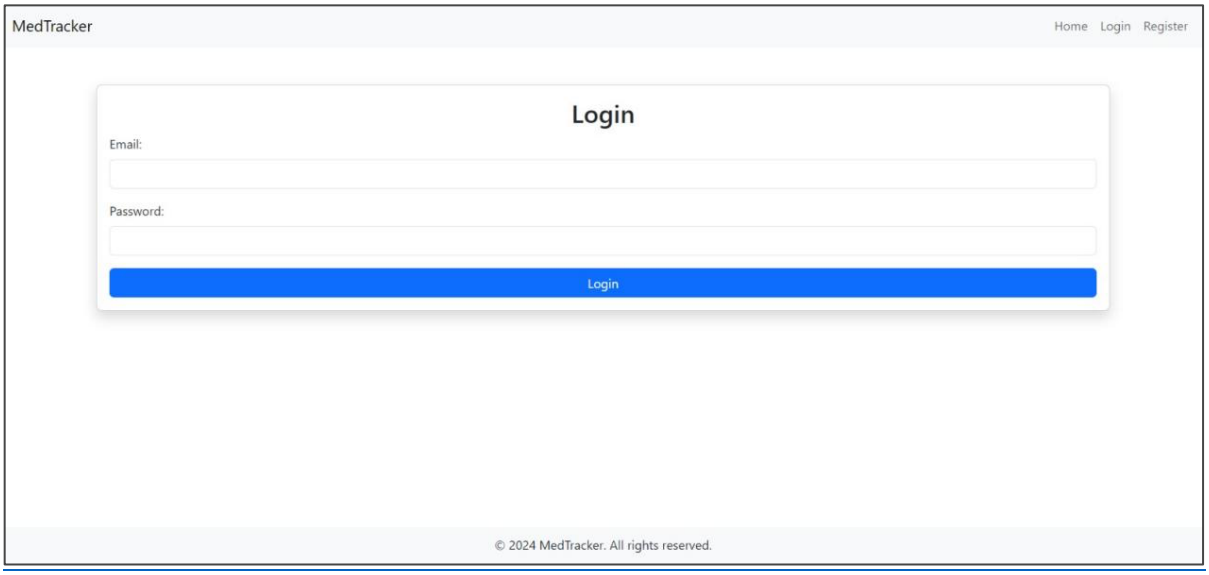
Hardware/Software and Libraries Used

- **Hardware:**
  - **Development Machine:**
    - **Processor:** Intel Core i7 or equivalent
    - **RAM:** 16 GB or higher
    - **Storage:** 256 GB SSD
    - **Smartphones:** Android and iOS devices for testing (e.g., iPhone 13, Samsung Galaxy S21)
  - **Testing Devices:**
    - **Emulators:** Android Studio Emulator, Xcode Simulator
- **Software:**
  - **Operating Systems:**
    - Windows 10 or later / macOS Monterey or later
- **Development Environment:**
  - IDE: Android Studio & Visual Studio Code.
  - Languages: Flutter Dart & Node.js in JavaScript
  - Database: Cloud Storage using MongoDB Atlas
  - Backend Framework: Node.js with Express.js
  - Version Control: Collaborated version with Git and GitHub
- **Libraries and Frameworks:**
  - **Flutter Packages:**
    - http: It is used to making Hyper Text Transfer Protocol (HTTP) requests to the backend API.
    - provider: A way to manage all the application states.
    - flutter_local_notifications: It's designed for setting up the medication reminders in the app.
- **Node.js Packages:**
  - express: Used for creating a backend server.
  - mongoose: Used for working with the MongoDB databases
  - cors: So that Cross-Origin Resource sharing is done by backend to the frontend.

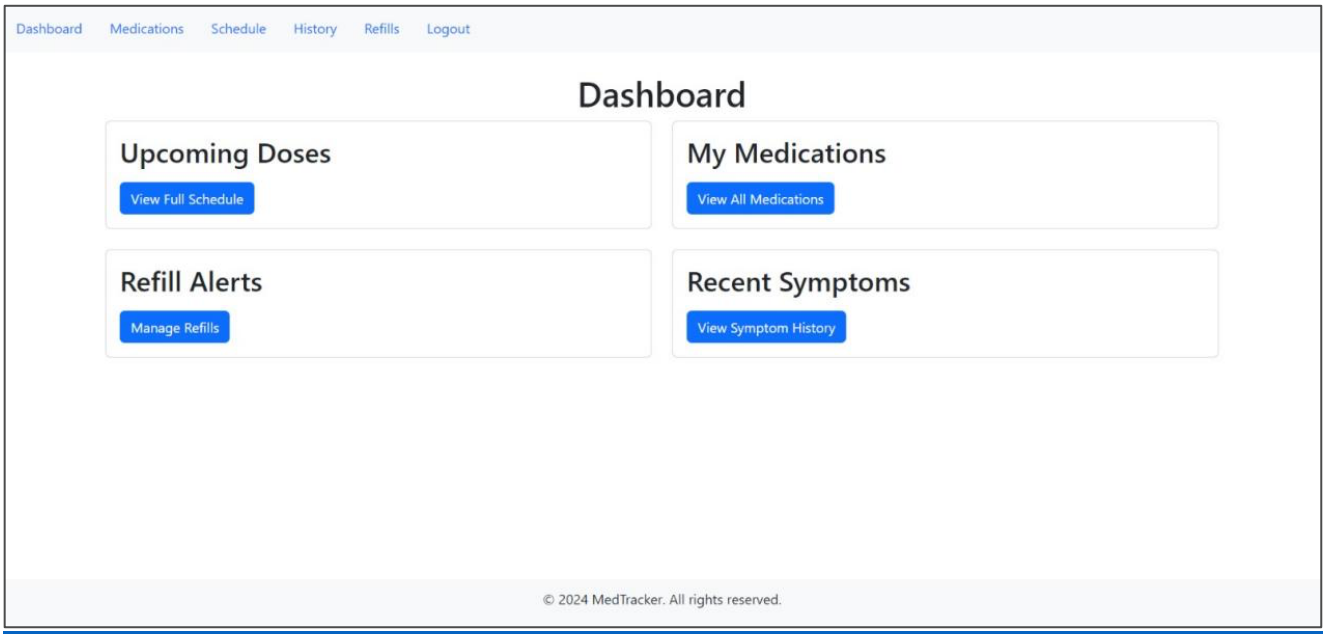
Screenshots for each step:

```
debar.js  JS App.js  JS Header.js  package.json 1  JS Register.js  JS auth.js  JS api.js
src > JS App.js > ...
1  import React from 'react';
2  import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-dom';
3  import { AuthProvider, useAuth } from './context/AuthContext';
4  import Header from './components/Common/Header';
5  import Footer from './components/Common/Footer';
6  import LandingPage from './components/LandingPage/LandingPage';
7  import Dashboard from './components/Dashboard/Dashboard';
8  import Calendar from './components/Schedule/Calendar';
9  import MedicationHistory from './components/Tracking/MedicationHistory';
10 import RefillManager from './components/Tracking/RefillManager';
11 import SymptomTracker from './components/Tracking/SymptomTracker';
12 import Login from './components/Auth/Login';
13 import Register from './components/Auth/Register';
14 import 'bootstrap/dist/css/bootstrap.min.css';
15
16 function AppRoutes() {
17   const { isAuthenticated, loading } = useAuth();
18
19   if (loading) {
20     return <div className="spinner"></div>;
21   }
22
23   return (
24     <Routes>
25       <Route path="/" element={isAuthenticated ? <Navigate to="/dashboard" /> : <LandingPage />} />
26       <Route path="/login" element={<Login />} />
27       <Route path="/register" element={<Register />} />
28       <Route path="/logout" element={<Logout />} />
29       <Route path="/dashboard" element={<Dashboard />} />
30       <Route path="/calendar" element={<Calendar />} />
31       <Route path="/history" element={<MedicationHistory />} />
32       <Route path="/refills" element={<RefillManager />} />
33       <Route path="/symptoms" element={<SymptomTracker />} />
```

Login Page:



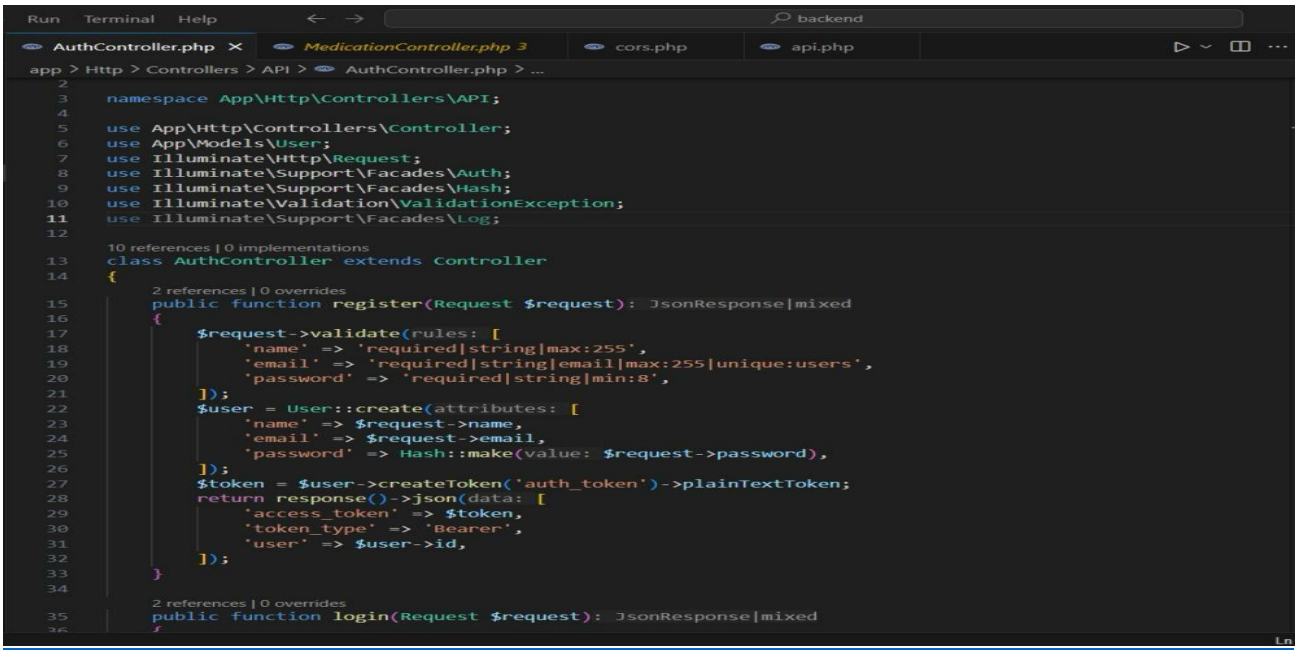
Dashboard Page:



This page is the Dashboard of the Medication Reminder Application. It provides a quick overview of important features for the user.

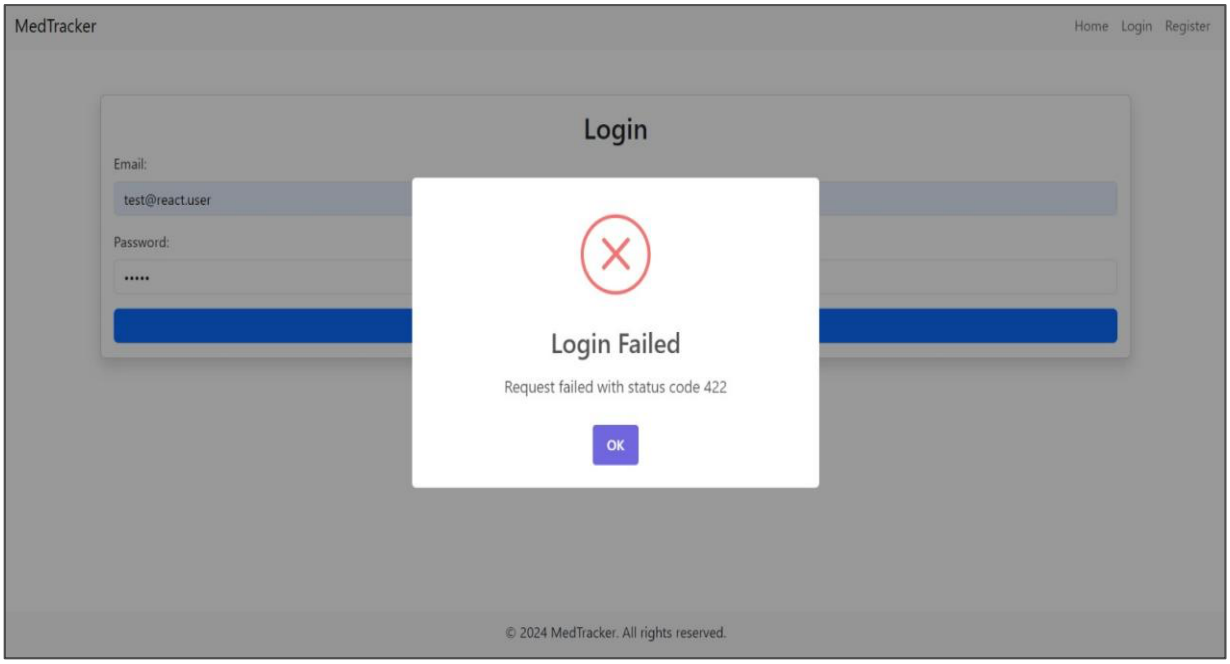
- **Upcoming Doses**: Shows the next scheduled medication doses with an option to view the full schedule.
- **My Medications**: Allows users to view all their medications by clicking "View All Medications."
- **Refill Alerts**: Displays reminders for medication refills and provides a button to manage refills.
- **Recent Symptoms**: Tracks recent symptoms and lets users view their symptom history.

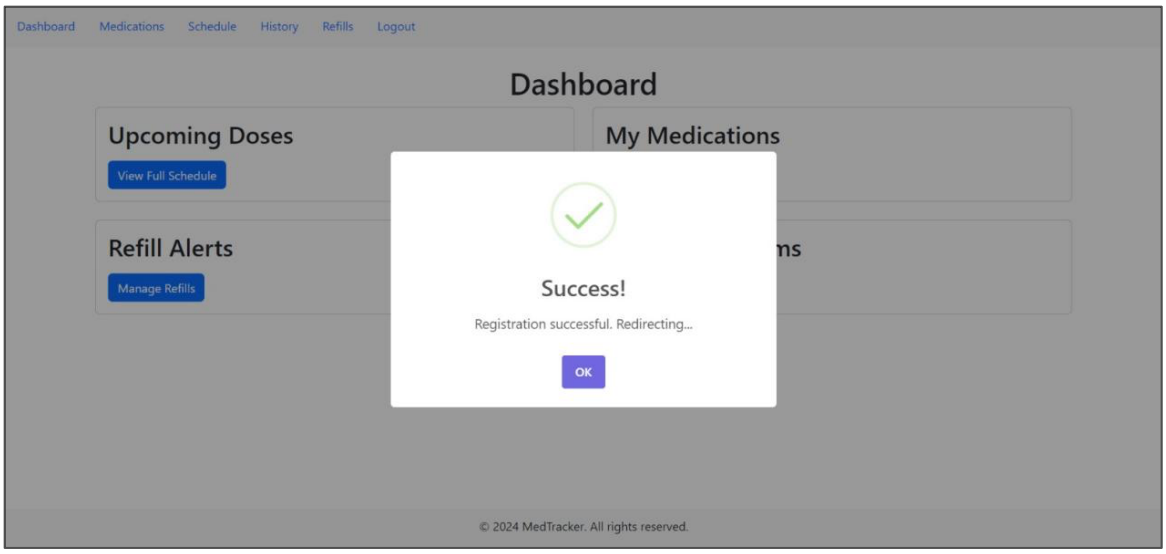
Code for dashboard page:

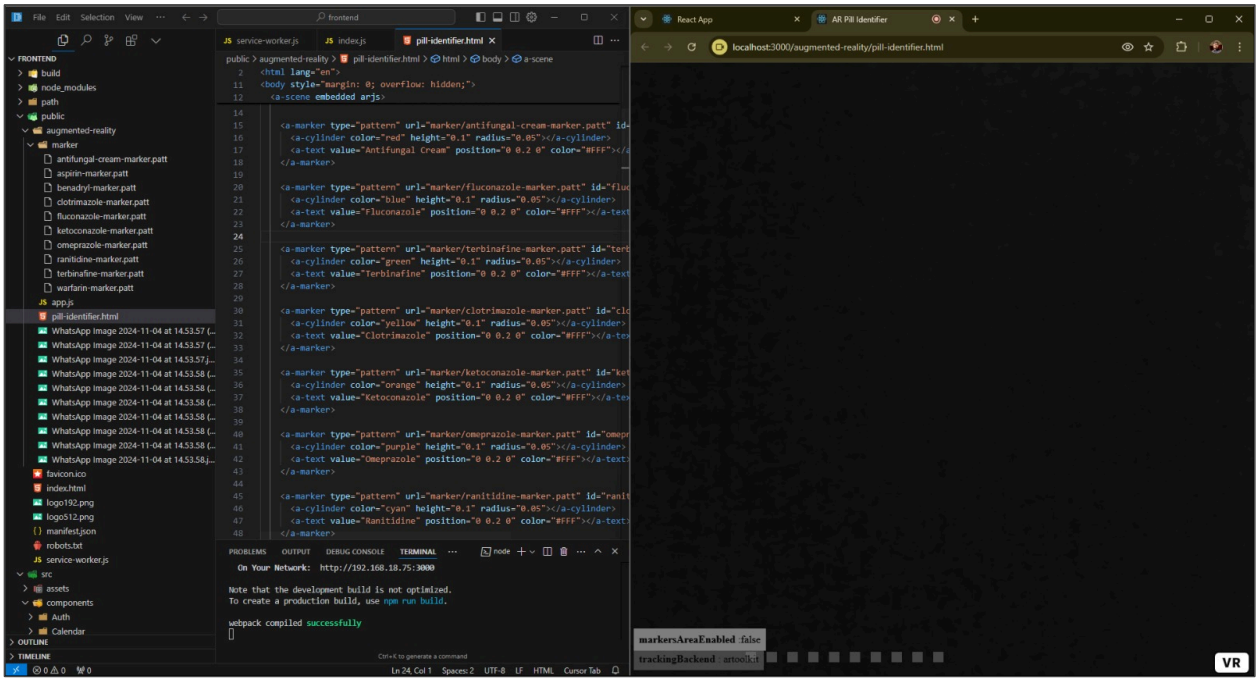


Testing the Application:

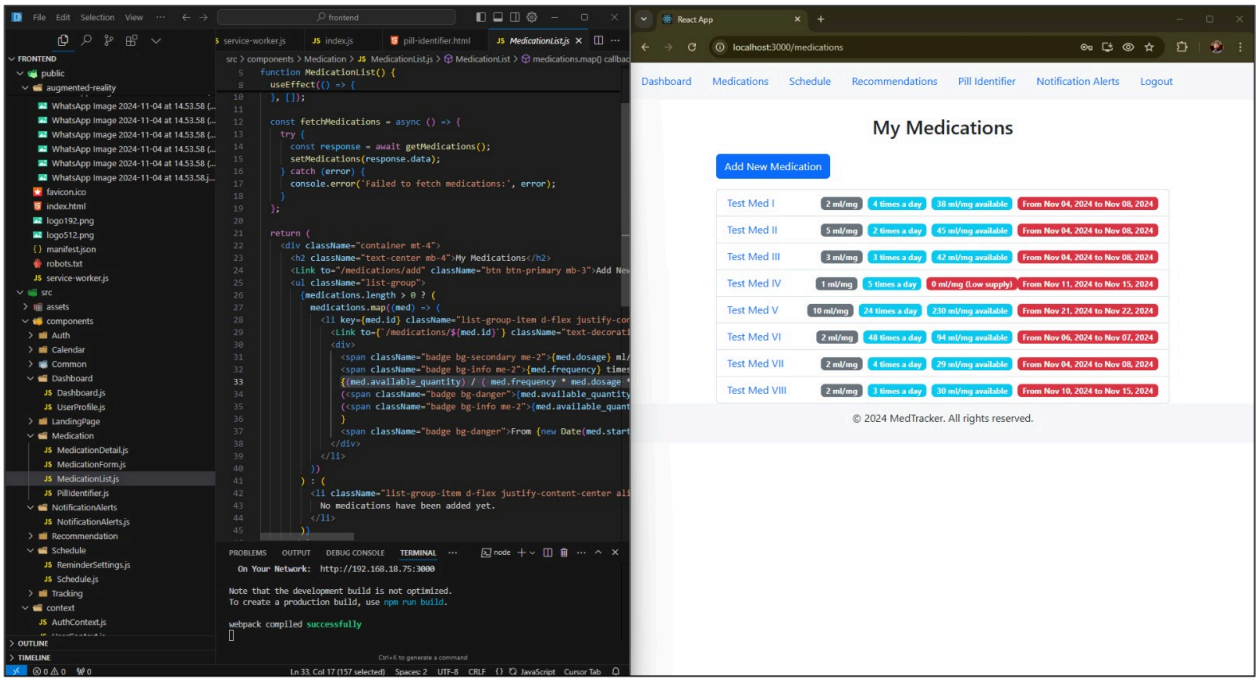
Testing login page: Failed because the password should be min 8 characters.



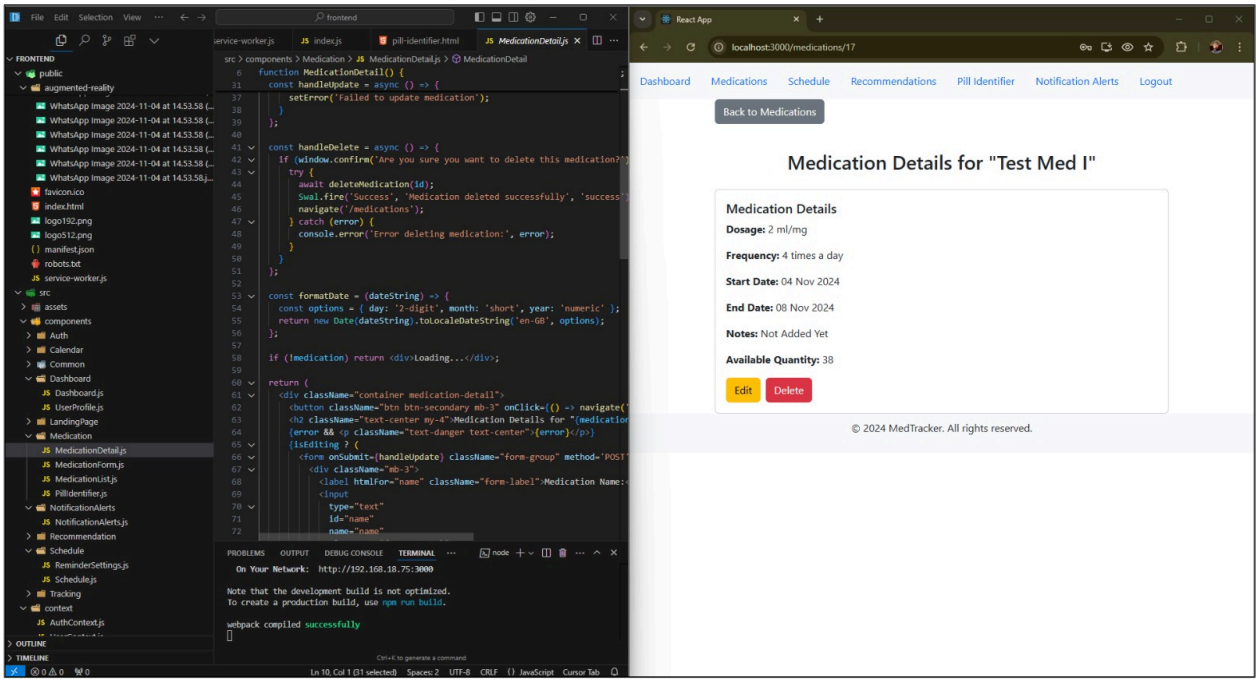

Dashboard
Medications
Schedule
History
Refills
Logout
Dashboard
Upcoming Doses
View Full Schedule
My Medications
Refill Alerts
Manage Refills
Success!
Registration successful. Redirecting...
OK
© 2024 MedTracker. All rights reserved.

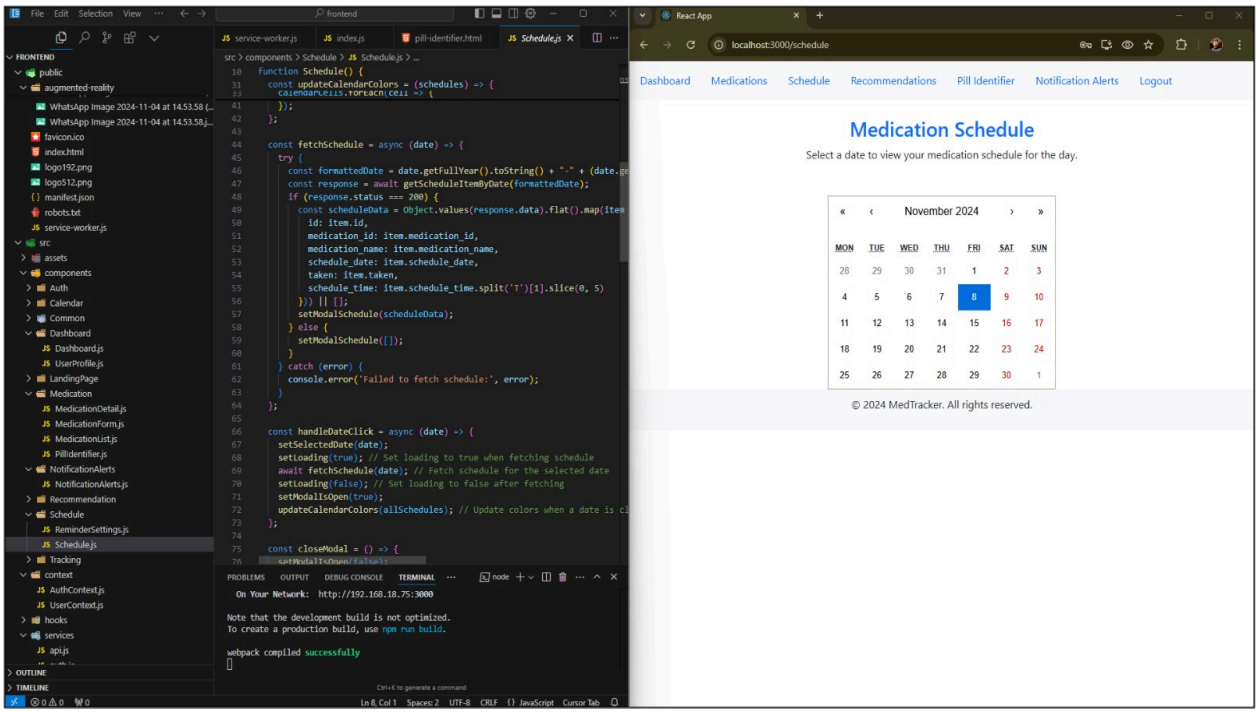

File Edit Selection View
frontend
React App
AR Pill Identifier
localhost:3000/augmented-reality/pill-identifier.html
service-worker.js
index.js
pill-identifier.html
FRONTEND
build
node_modules
path
public
augmented-reality
marker
antifungal-cream-marker.patt
aspirin-marker.patt
benadryl-marker.patt
clotrimazole-marker.patt
fluconazole-marker.patt
ketoconazole-marker.patt
omeprazole-marker.patt
ranitidine-marker.patt
terbinafine-marker.patt
warfarin-marker.patt
app.js
pill-identifier.html
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
service-worker.js
src
assets
components
Auth
Calendar
OUTLINE
TIMELINE
public > augmented-reality > pill-identifier.html > html > body > a-scene
<html lang="en">
<body style="margin: 0; overflow: hidden;">
<a-scene embedded arjs>
<a-marker type="pattern" url="marker/antifungal-cream-marker.patt" id=
<a-cylinder color="red" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Antifungal Cream" position="0 0.2 0" color="#FFF"></a
</a-marker>
<a-marker type="pattern" url="marker/fluconazole-marker.patt" id="fluc
<a-cylinder color="blue" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Fluconazole" position="0 0.2 0" color="#FFF"></a-text
</a-marker>
<a-marker type="pattern" url="marker/terbinafine-marker.patt" id="terb
<a-cylinder color="green" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Terbinafine" position="0 0.2 0" color="#FFF"></a-text
</a-marker>
<a-marker type="pattern" url="marker/clotrimazole-marker.patt" id="clo
<a-cylinder color="yellow" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Clotrimazole" position="0 0.2 0" color="#FFF"></a-tex
</a-marker>
<a-marker type="pattern" url="marker/ketoconazole-marker.patt" id="ket
<a-cylinder color="orange" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Ketoconazole" position="0 0.2 0" color="#FFF"></a-tex
</a-marker>
<a-marker type="pattern" url="marker/omeprazole-marker.patt" id="omepr
<a-cylinder color="purple" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Omeprazole" position="0 0.2 0" color="#FFF"></a-text
</a-marker>
<a-marker type="pattern" url="marker/ranitidine-marker.patt" id="ranit
<a-cylinder color="cyan" height="0.1" radius="0.05"></a-cylinder>
<a-text value="Ranitidine" position="0 0.2 0" color="#FFF"></a-text
</a-marker>
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
On Your Network: http://192.168.18.75:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
markersAreaEnabled: false
trackingBackend: artoolkit
VR

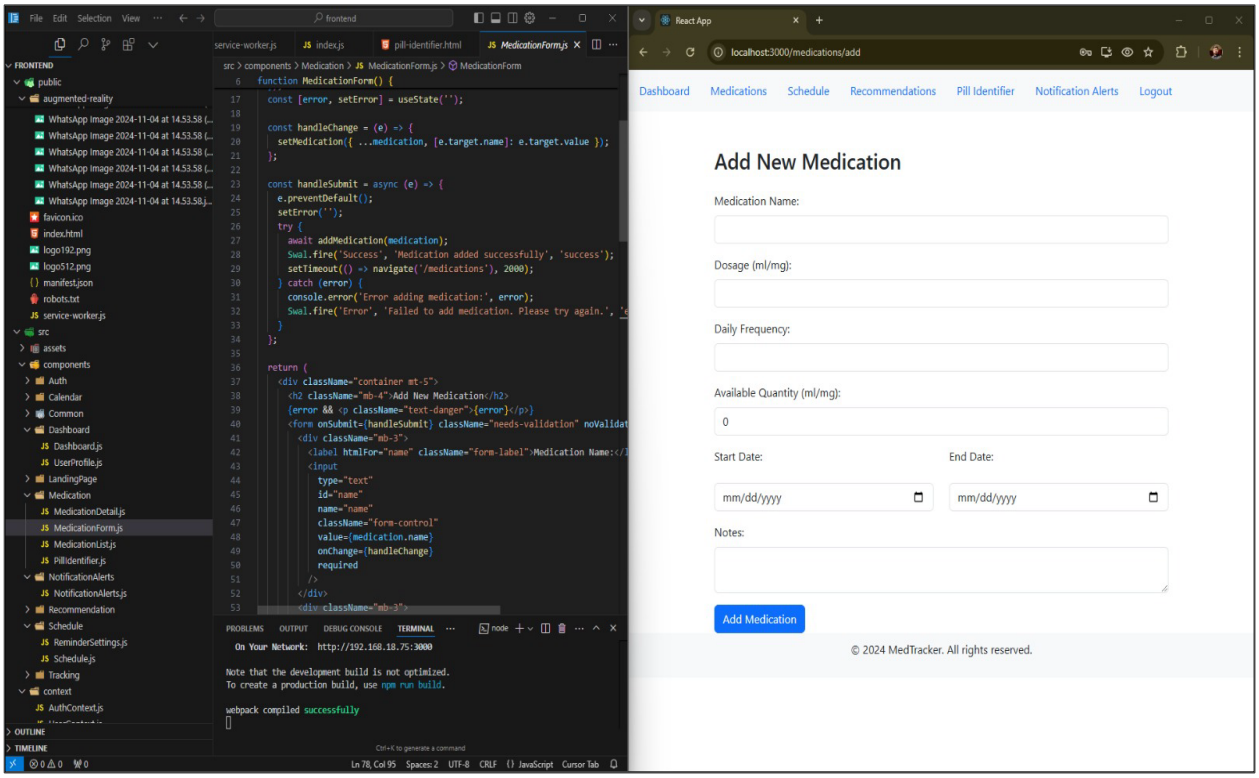
My Medications



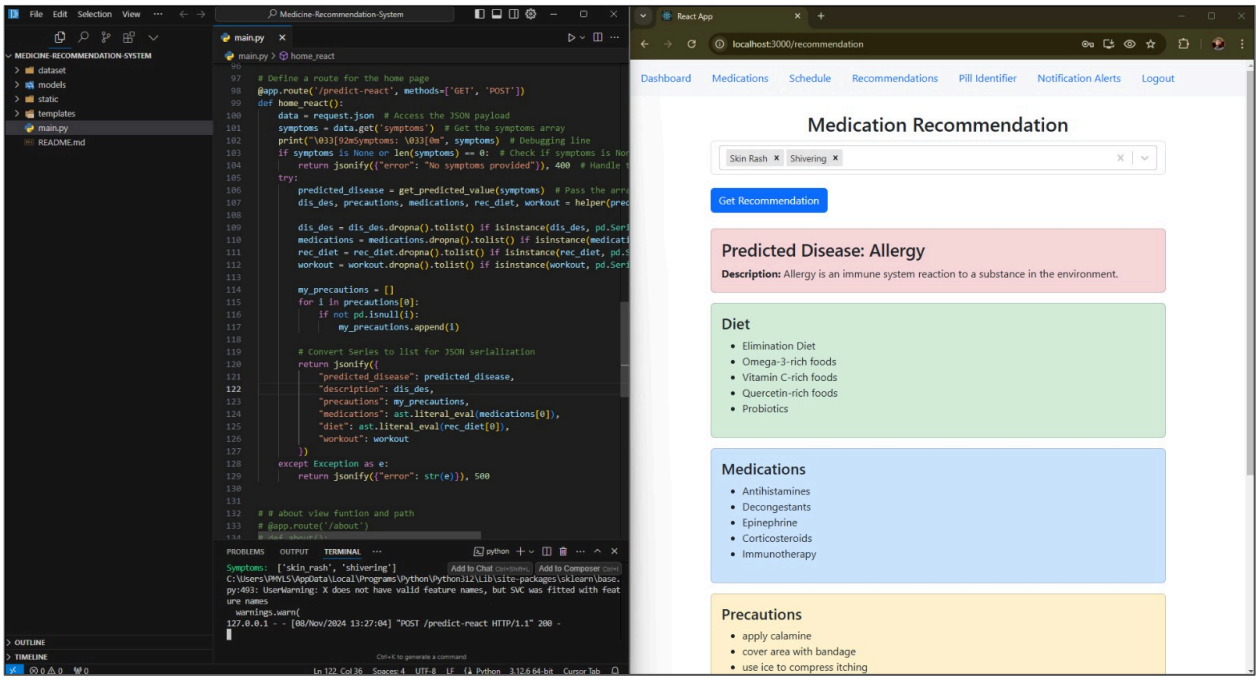
Medication Details: Contains all the details of the medications being taken.




Medication Schedule
Select a date to view your medication schedule for the day.
November 2024
Dashboard
Medications
Schedule
Recommendations
Pill Identifier
Notification Alerts
Logout
© 2024 MedTracker. All rights reserved.


Add New Medication
Medication Name:
Dosage (ml/mg):
Daily Frequency:
Available Quantity (ml/mg):
Start Date:
End Date:
Notes:
Add Medication
Dashboard
Medications
Schedule
Recommendations
Pill Identifier
Notification Alerts
Logout
© 2024 MedTracker. All rights reserved.

Medication Recommendation: Main recommendation system.



User Interface Design According to End User Needs:

The UI design's simplicity, accessibility, and usefulness are prioritized since we are targeting consumers, particularly the elderly or those with chronic health concerns. The design approach centres on:

Development Phases

The Medication Reminder Task Management System is implemented in phases to enable organized development and iterative testing. Each step is intended to build on the preceding one, progressively adding layers of functionality, improving the user experience, and strengthening security.

Phase 1: Requirements Gathering and Planning

- **Activities**:
    - Define and document user requirements, especially around adherence features, reminder customization, and medication management.
    - Conduct stakeholder interviews, including healthcare professionals and potential users, to confirm feature priorities.

    - Outline user personas and scenarios to guide design decisions and feature prioritization.
- **Outcome**: A comprehensive project plan and requirement document that aligns team members on the project's scope, objectives, and expected deliverables.

Phase 2: System Design and Prototyping

- **Activities**:
    - Create wireframes for key user interfaces (UI) such as medication input screens, reminder settings, and AR-based pill identification.
    - Develop a prototype for user testing, focusing on navigation and accessibility.
    - Finalize data models and relationships within the database, including tables for users, medications, reminders, and adherence history.
- **Outcome**: A functional prototype with UI mockups and a solid database schema ready for initial development.

Phase 3: Core Development

- **Activities**:
    - **Frontend**: Implement the core UI elements in React Native, focusing on registration, medication management, and reminder customization screens.
    - **Backend**: Develop RESTful APIs in Laravel to support CRUD operations for user data, medication records, and reminders. Implement business logic for adherence tracking.
    - **Notification System**: Integrate Firebase Cloud Messaging (FCM) for push notifications, ensuring that reminders are timely and reliable.
    - **Database Integration**: Connect the frontend and backend to the MySQL database, ensuring data consistency and security across transactions.
- **Outcome**: A working version of the application with core functionalities, enabling basic medication management and notification features.

Phase 4: AI and AR Integration

- **Activities**:
    - **AI Module:** Create models using TensorFlow to predict appropriate medication regimens and enhance adherence based on user data.
    - **AR Module:** Use Vuforia SDK to create pill identification functionality, enabling users to scan pills for visual confirmation.
    - **Testing:** Conduct performance testing to assess AI and AR feature responsiveness and accuracy, particularly with diverse user inputs.

- **Outcome**: Advanced functionality that enhances the application's value, with AI-driven recommendations and AR-based pill identification fully operational.

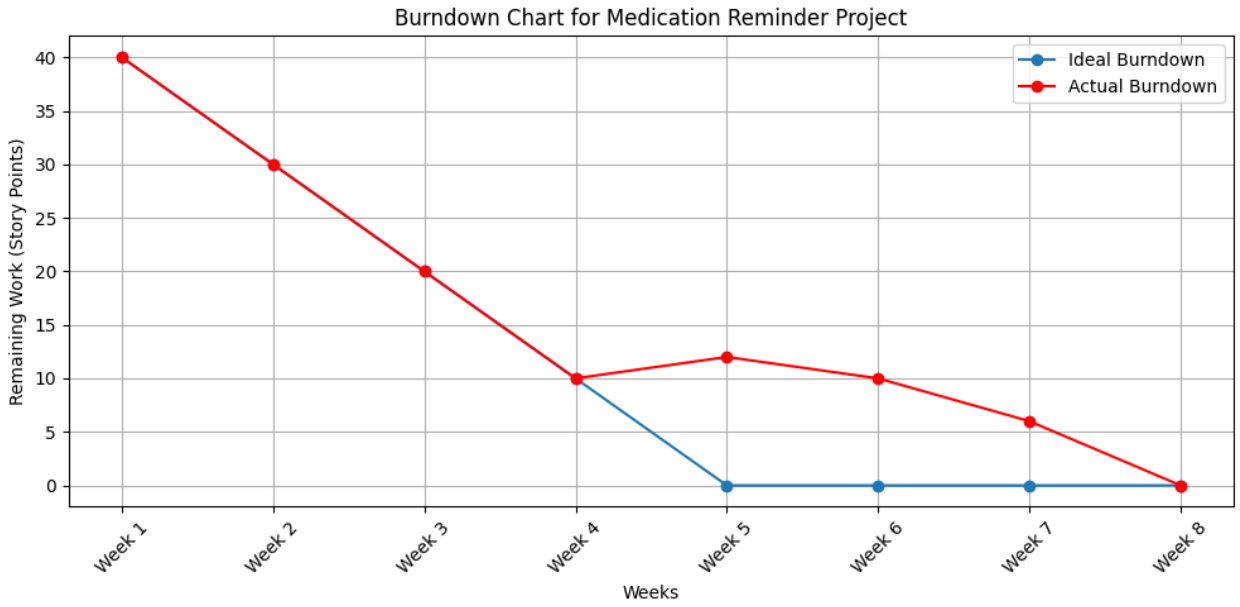
Phase 5: Testing and Quality Assurance

- **Activities**:
    - I will carry out a series of tests individually for each unit I expect to see and interact them as whole Perform basic tests on the registration module, as well as the reminder alerts in this part of our product.
    - Systematically check that everything works according to plan, which also includes the fact that all dependent sections, such as website backend and front-end as well database services, functioning in the way they should be complementary operationally in Russian, Japanese Capitalist script.
    - Check for user satisfaction acceptance testing to make sure customers who are using the software see it beneficial and honorable according to their expectations ascii. ECB Now with up-to-the-minute information.
    - Check compliance with HIPAA, GDPR and other data protection laws by conducting security audits.
- **Outcome**: A fully tested application that is ready for deployment, with any identified bugs or usability issues resolved.

Phase 6: Deployment and Maintenance

- **Activities**:
    - The application is deployed to Google Play Store as well as Apple App Store and acted in accordance with the guidelines of each app store.
    - Cloud hosting on GCP is set up, scaling and monitoring tools now enable stability maintenance for the app.
    - Monitor the post-deployment to track performance, manage user feedback and implement ongoing updates.

Outcome: A deployed application accesible to users, together with a substantial plan for long-term support and future improvements



Cloud Hosting and Database Deployment on GCP

- Configure a secure backend infrastructure on Google Cloud for consistent uptime and scalability to support user growth.
- Deploy MySQL/MariaDB database on GCP with encryption and access restrictions to ensure safe data storage.

Post-Deployment Monitoring and Maintenance

- Monitoring: Utilize the GCP monitoring services to monitor the servers performance, errors and latency. Apply and launch notifies to prompt an instant correction of aspects that affect user interaction.
- User Feedback Collection: Construct and frequently collect and analyze customers feedback through the in-app surveys, app stores, and support contacts. Optimize frequent updates and problems solving according to their reported occurrence by users.
- Security Patches and Updates: The security features should be updated on a constant basis in order to deal with new threats and other open weaknesses. Such changes include changing the app in order to adapt new rules in the field of healthcare or new laws regulating the use of data.

Part – 7 -Testing

A testing phase is important in developing Medication Reminder Task Management System especially for the purpose of checking functionality of all parts. Since this app is target to help improve the healthcare systems, much testing has to be done in order to ensure that security, accuracy and usability of the application is has been improved. Further, this part defines different testing phases, processes and findings that guarantees compliance to the functional , security and performance characteristics required in the system.

Testing Strategies

Testing is divided into numerous independent phases to present full range scenario and each phase deals with specific aspect of the system component and specific performance criteria. The QA team will perform a manual and automated testing to reduce time and ensure high quality speed up the process is paramount.

Testing Phases and Expected Outcomes

Manual Testing:

1. **Unit Testing**
    - Objective: To isolate the components for a verification and checked if all elements are functioning as intended.
    - Methodology: Each of the said modules is for individual testing. H1 is an evaluation of input, process, and output correctness under various conditions through descriptive unit testing.
    - Tools: Jest and Mocha for unit testing in the frontend; PHP Unit for backend PHP components.
    - Expected Outcome: The concept of Integrated Component Testing means that all inputs and outputs of a certain component should be absolutely fine when the component is tested separately; if there are some errors or some inconsistencies, these points must be found and fixed immediately.

Sample Unit Testing Table:

Module	Test Case	Expected Outcome	Actual Outcome	Status
Registration	Validate input fields	Successful registration on valid inputs	Registration completed as expected	Pass
Medication Input	Add new medication	Medication added to database successfully	Data stored accurately	Pass

Reminder Notification	Verify timing of notifications	Reminders sent at scheduled times	Notifications received on time	Pass

2. **Integration Testing**

Make sure there is a clean and direct integration between the frontend and backend and the database section.ase components.

- Methodology: It keeps flowing test data between the components such as user profile details, medication details, and reminder. The APIs are then employed to ensure that any exchange of data in between the various modules is carried out correctly and without compromise to security.
- Tools: API testing needs to be done through Postman, and the Selenium tool for frontend-backend integration testing.
- Expected Outcome: Integration testing brings out all the problems relating to data flow and interfaces so that at the end the user will be able to be assured of a fuller and more reliable ride as well as the data.

Sample Integration Testing Table:

Module	Test Case	Expected Outcome	Actual Outcome	Status
Frontend-Backend Connection	Retrieve medication data via API	Accurate data display on the UI	Data displayed accurately	Pass
Reminder Notification System	Verify timing and accuracy	Notifications appear at set times	Notifications accurate and on time	Pass
Database Synchronization	Update adherence history	Data reflects accurately in adherence logs	Data synchronization successful	Pass

3. **System Testing**

- Objective: Check and verify the complete product and make sure that every component within it should act or behaviors in a precise real life environment.
- Methodology: The software components of the application that are tested in system testing include the user registration, scheduling of prescription, check on the level of prescription compliance and prescription alerts.
- Tools: It involves testing done on the systems and on the scripts for repetitive chores.
- Expected Outcome: It should be able to perform the tasks in the best way possible, through all the interfaces in a manner that they all respond in one way giving the user interface a clean and interrupted look.

Sample System Testing Table:

Workflow	Description	Expected Outcome	Actual Outcome	Status
Complete Workflow Execution	Register, login, add medication	Error-free navigation and functionality	System fully functional	Pass
Reminder Notification Workflow	Schedule and receive notifications	Notifications received at correct intervals	Notifications consistent and accurate	Pass
Data Logging and Persistence	Verify data logging in adherence history	Adherence history updated and accessible	Data recorded as expected	Pass

4. **User Acceptance Testing (UAT)**

- Objective: Ensure that the system meets user requirement for usability, accessibility and use.
- Methodology: Real life users are used in testing the system for instance using the elderly, caretakers, as well as individuals who require to manage many medications. Notification, recognition of the drug as well as checking the compliance record are examples of test cases.
- Tools: They are such items as feedback forms or questionnaires, surveys made in application, and direct observation of the users' interaction.
- Expected Outcome: The application should not be complicated while implementing the design elements that will act in its best interest of supporting a variety of users.

Sample UAT Testing Table:

Aspect	User Feedback	Action Taken	Result
Interface Usability	Interface is intuitive and easy to navigate	No major changes needed	Pass
Accessibility Features	Users appreciated large text and simplified options	Enhanced button sizes for visibility	Pass
Reminder Notifications	Users found reminders helpful and appropriately timed	No changes required	Pass

5. **Security Testing**

- Objective: Identify all threats so that no threat exposes the user information to attacks, data breaches or compliance mishaps that would cause a lot of havoc.
- Methodology: All the techniques include Vulnerability Scanning, Penetration testing, and access control tests which assist in identifying all the possible threats that face the information as well as the ways of protecting them. Encryption measures, role based access control, our HIPAA and GDPR compliances checks and reviews are all proper in place or in check.

- Tools: In vulnerability scanning, manual pen testing, and even encryption validation, the OWASP ZAP is used.
- Expected Outcome: It should also be protected from other mundane cyber challenges such as okayed data encryption and access control as shown below.

Sample Security Testing Table:

Aspect	Test Case	Expected Outcome	Actual Outcome	Status
Data Encryption	Verify data encryption protocols	Sensitive data encrypted in database	Encryption verified and secure	Pass
Authentication Security	Unauthorized access attempts	Unauthorized access blocked	Access control effective	Pass
Compliance Verification	Check for HIPAA and GDPR compliance	Compliance with regulatory standards	Compliance verified	Pass

6. **Regression Testing**

- Objective: Ensure that nothing changes from the previous code that had been incorporated in the previous code or added new code base will interfere with functionalities that have been tested severally.
- Methodology: As mentioned, the objective of regression testing is to ensure the basic functionality stability for the application after every major as well to upgrade it. The other advantage is that you can run again the regression scripts in a very short time while testing critical features including reminders and compliance.
- Tools: The above steps are used for rerun testing where we use Selenium for the same.
- Expected outcome: Consistent with this functionality, general functionality operations can remain more or less stable after upgrades when tested and if new problems crop up, they should remain easy to identify and solve.

7. **Sample Regression Testing Table**:

Feature	Test Case	Expected Outcome	Actual Outcome	Status
User Registration	Re-test registration post-update	Registration continues to work as expected	Registration works as expected	Pass
Reminder Notifications	Test reminder system after update	Reminder delivery remains consistent	Notifications are accurate	Pass
Adherence Tracking	Check adherence log after update	Adherence history updates correctly	History logs are complete	Pass

Automation Testing:

1. **Set Up Your Test Environment**:

    - Framework: Decide on test frameworks for example JUnit.
    - Selenium WebDriver: It is important you have the WebDriver set for the browser (Chromedriver for Chrome).
    - Programming Language: Type a code in a language supported by selenium (Python).
    - IDE: Use an IDE like IntelliJ.

2. **Create Test Cases**:

- **Define Test Cases**: Write down the objectives for each test, such as "Verify login functionality with valid credentials".
- **Identify Elements**: Use tools like browser developer tools to inspect and find unique locators (e.g., id, name, XPath).
- **Write Test Code**

3. Run the Tests:

- Execute tests using the command line or directly from the IDE.
- Use batch or shell scripts to automate test execution in CI/CD pipelines (e.g., using Jenkins).

Aspect	**Test Case**	**Expected Outcome**	**Actual Outcome**	**Status**
Login with correct details	User logs in right cred	User successfully logs in	User logged in successfully	PASS
Login with wrong password	User logs in with wrong cred	Error message shown	Error message shown	PASS
Password reset	User requests password reset	Confirmation message displayed	Confirmation message displayed	PASS

Logout function	User logs out	User is redirected to login page	User redirected to login page	PASS
Search feature	User searches for "medication"	Search results are shown	Results shown correctly	PASS
Form with empty fields	User submits empty form	Error messages shown	Error messages shown	PASS
Notification timing	Reminder set for 10:00 AM	Notification arrives on time	Notification arrived on time	PASS

Evaluation Metrics

To measure the system's success and quality, numerous assessment indicators are used, including performance, usability, and compliance. The following metrics provide information on how successfully the application accomplishes its objectives.

1. **Medication Adherence Improvement:**
    - **Metric:** Tracking users adherence rates before and after using the application to assess its overall effectiveness in increasing medication and consumption consistency.
    - **Expected outcome:** The application's ability to support regular prescription leads to greater usage rates among users.
2. **User Satisfaction:**
    - **Metric:** Gather input on general satisfaction of the users, focusing on the simplicity of use, reminder dependability, and a user-friendly interface design.
    - **Expected outcome:** High satisfaction over diverse demographics, including senior users who are experienced and those managing complex prescriptions.
3. **System Performance and Responsiveness:**
    - Metric: To calculate the response time, the load time and whether or not notifications where delivered on time.
    - Expected outcome: Another requirement is to deliver the answer responding in 1 to 2 seconds and, at the same time, containing plans associated with notice delivery.
4. Security and Compliance:
    - Metric: Perform security scans in order to see whether the healthcare data is compliant with those standards set by the HIPAA and GDPR and among others.

- Expected outcome: The expected outcome is the people's compliance to healthcare data rules that will maintain the safe and legal use of such information.

5. Reliability:
    - Metric: Safety might also be measured by the validity in terms of time and frequency, errors and the capacity to expel an error.
    - Expected outcome: The reliability rate not being lower than 99.5% it is a basic requirement but important additional functions, like reminders, should perform without any delays.

Part – 7 - Maintenance

For this Medication Reminder Task Management System to be around for some time, it has to be maintained to guarantee its availability and reliability. This phase comprises ordinary upkeep, to maintain steady performance; future plans to update to fit the user demands and to keep up with new technologies and development. The maintenance plan outlines how input from the users shall be obtained, how technical difficulties encountered shall be resolved, and how the functioning of the system shall be gradually enhanced.

1. Maintenance Objectives

The main objectives of the maintenance phase are:

1. Maintain System Reliability: Ensure all function operates effectively, however compulsory the essential functions like reminder and compliance check.

2. Improve Security Protocols: Security of the data must be guaranteed while the measures must be altered from time to time if the company deals with the healthcare data fulfilling HIPAA and GDPR legislation.

3. Adapt to Evolving User Needs: Improve the user experience from the feedback received and also because new technologies are continuing to be developed in the field of medicine.

4. Minimize Downtime: This means that technological challenges, problems, disruptions, and glitches, which can be resolved as soon as possible with minimal disruption of time to the users.

2. Planned Enhancements

1. Predictive Analytics and Adherence Pattern Recognition

- Objective: Employ machine learning:Based on available data on users' medication adherence monitor such patterns to be estimated and appropriate recommendations to be forwarded.
- Description: Priorization data is also considered by the system, such as prior missed doses and preferred time for administrating a particular drug. With the information, the system can suggest changes in the adherence-enhancing techniques (for example, alteration in reminder timings).
- Implementation plan: Use TensorFlow to host machine learning models which were trained on the adherence data to identify and predict. This, in turn, may require the adjustment of the recommendation algorithms to targeting recommendations at each user based on her information.
- Expected outcome: Increase of the proportion of consumers developing better treatment adherence by increasing communication of individualized reminders and recommendations.

2. Expanded AR Integration for Comprehensive Pill Identification

- **Objective:** Improve the AR-based pill identification functionality by including visual clues such pill color, shape, and size.
- **Description:** Improved AR capabilities to allow multiple prescriptions and similar-looking drugs. This functionality will help users by preventing medication mistakes.
- **Implementation Plan**: Update the AR module (powered by Vuforia SDK) with a larger collection of pill properties to increase identification accuracy through diversified picture training.
- **Expected outcome:** Improved user trust in medicine identification and reduced medication mistakes.

3. Voice Command Integration for Accessibility

- Objective: Allows users to receive instructions on how to perform a certain task through voice commands especially to those who find it hard to move their arms or rather have a poor sight.
- Description: Communication through voice means allows users to set reminder, check dosing schedules and make use of the pill identifier without the intervention of a person. This feature will be intended for the users that experience difficulties within the paradigm where nearly all manipulations are executed by touching the screen.
- Implementation Plan: Propose a speech recognition API (e.g., Google Cloud Speech-to-Text) into critical application functions and make prompts' and replies' interfaces voice-friendly.

- Expected Outcome: Increased accessibility means that more people in this population are going to be able to engage with the program effectively and unaided.

4. Comprehensive Report Generation and Sharing Features

- Objective: Ensure that the users are provided with complete adherence reports for the purpose of presenting it to one's healthcare practitioner and career.
- Description: The system produces reports that together present medicine ingestion patterns, adherence rates, and the missing doses. These reports can be forwarded by email or can be downloaded in pdf format.
- Implementation Plan: A backend reporting module needs to be developed where you collate data on adherence and convert into easy to read PDFs. Ensure the patient has choices of secure ways, which would ensure that information exchange in this platform is compliant with the provisions of HIPAA.
- Expected Outcome: Enhanced support for co-ordinated health care to allow clinicians and careers to detect and facilitate patient compliance efficiently.

3. Maintenance Types and Protocols

Ongoing maintenance activities are categorized into four types: corrective, adaptive, perfective and preventive maintenance.

Corrective Maintenance

• Objective: Ais and bugs that may surface after deployment require strategic correction so that there is non-interrupted smooth running of the program.

• Activities:

o Bug Fixes: Bugs reported by users or those seen by the monitoring system require fixing. These are in general simple bugs that may cause issues such as; missed notifications or data synchronization issues or application crashes.

o Incident Response: Create response procedures for contingencies that may impend system accessibility, staff or information credibility, with the capability of rapid resolution for minimal inconvenience to the audience.

• Tools: It also helps to use bug tracking tools to track and prioritize the issues, as well as ,it will remains the technical team's responsibility to solve them, (e.g. Jira).

Adaptive Maintenance

- Objective: Upgrade the application to align with new conditions in the outside environment like new rules, the new operating system or API.
- Activities:

    o Compliance Updates: Make sure updating the app to achieve the new requirements of the healthcare legislation such as HIPAA, GDPR, etc.

    o Compatibility Updates: Charge 2011: 'Make sure your app is compatible with the new iOS, Android, and other platform updates'. This requires adapting component inputs and presentation, notifiers, and services based on the target upgraded devices to achieve compliance.

    o API Updates: Change API integration if there are some changes in 3rd party APIs (for example, changes in AR, voice recognition or notification APIs).

- Expected Outcome: A system that stays active and In synch with existing changes, to avoid the consequences of non-compliance penalties while guaranteeing a uniform user interface.

Perfective Maintenance

- Objective: Update existing functionalities based on the feedback received by users, the measuring of models performance, and research on new and better practices.
- Activities:

    o UI/UX Enhancements: This involves frequently changing components of the user interface according to the comments made by users regarding improvement such as the menus or colors for the visually impaired to include options like the dark theme.

    o Feature Optimization: Build upon the current aspects of the application such as the reminder settings and the degree of compliance of the patients with the set dose regimes as sampled from the user base.

    o Performance Tuning: Optimize and improve the current speed and efficiency of the presentation in response to submitted performance metrics such as load times and notification accuracy.

    • Expected Outcome: An improved technical infrastructure with features and looks that should be generally upgraded to correspond to the preferences of users.

Preventive Maintenance

- Objective: Of course, among the approaches that should be used in order to minimize and, in fact, eradicate instances of system failures, there is the consistent, accurate

monitoring of the system's performance and addressing all the potential causes of system failure when they are still potential, not real.

- Activities:

  o Regular Audits and Monitoring: Schedule ongoing tests of the system and it includes security scans of the system, analysis of system's functions and coding convention check.

  o Database Optimization: Once in a while there is the need for its defragmentation in a way that makes it easy for data to be retrieved and as well in an endeavor to utilise as little storage space as possible.

  o Backup and Disaster Recovery Planning: Duplicate the system at least once a week, or more often if possible, and have an operable recovery plan if the system fails or if data has been lost.

  • Expected Outcome: Greater system reliability and reduced likelihood of fatal problem/s system downtime so that the users of the system develop confidence.

4. User Feedback and Continuous Improvement

To ensure the application remains aligned with user needs, a structured feedback mechanism will be established to gather insights and suggestions for improvements:

• In-App Surveys: Occasionally the users with the brief and targeted questions are to receive feedback on the particular item or overall satisfaction and submit them through feedback in the app.

• App Store Reviews: Check the number of stars and comments received from the customers in Google Play and Apple App Store, then reply to the customers' feedback in a positive way and know which request or complaint is frequently appearing from the customers.

• Support Channels: Keep a support page specially dedicated to the product where users can report issues personally and also converse about the features in great detail.

• Usage Analytics: Gather user metrics without requiring any personal information and focus on how often an element is used ad how strictly people follow an application's suggested habits.

With regards to feedback, the team can accumulate and come up with feedbacks then identify which of the update makes the most significant impact with regards to the satisfaction of users or the effectiveness of the system.

5. Security and Compliance Maintenance

Because the data under the healthcare system is very sensitive data, very strict measure need to be taken on the user end for the support of the users.

- Data Security Audits: Security check should be performed from time to time to determine the weaknesses, for instance the software with outdated patches or a defective access control.

- HIPAA and GDPR Compliance Checks: Set up time for a periodic review somewhere around one year to check compliance with the new rules such as HIPAA, GDPR, and others. Any compliance documentation will be modified when there are changes to data collecting and using protocols.

- Access Control Reviews: Periodically ensure that appropriate personnel has rights to access patients' sensitive health information, as well as organizational privileges.

- Encryption Updates: Check into encryption algorithms to make sure all stored and transmitted information is protected to the best of current standards.

6. Documentation and Training

To facilitate the effective maintenance, comprehensive documentation will be created and updated throughout the entire maintenance phase for users ease:

- Technical Documentation: Documentation of the API to be deployed along with the structure of the database and the coding structure will be maintained in details for future reference.

- User Guides and Tutorials: Instructive guides, tutorials, and tips that were especially created for users of the service will assist them to get the most of it and solve inherent difficulties on their own.

- Developer Training: Updates will be done to train the development team to encourage consistencies of the current security practices, changes in regulations, and improved practices.

Part – 9 - Roles and Responsibilities

Every individual had key duties that were essential to the achievement of this task. The teamwork enhanced proper organization of tasks, timely accomplishment as well as compliance with the quality standards.

Roles:

1. **Dharani Satwika Komaravolu (1002148504)**
    - **Role**: Project Manager
    - **Responsibilities**:
        - Responsible for managing project timetables and deliverables.
        - Attend coordination meetings to exchange communication ideas with team members.
        - Maintain a library of project documentation and report on it.

2. **Sai Puneeth Thummaluru (1002158041)**

    - **Role**: Frontend Developer
    - **Responsibilities**:
        - Develop the user interface for the Medication Reminder application.
        - Make your web page responsive and user-friendly.
        - Collaborate with the backend developer on API integration.
        - After receiving user input, launch the UI for testing.

3. **Rohith Reddy Papareddy (1002120248)**

    - **Role**: Backend Developer
    - **Responsibilities**:
        - Contribute to server-side logic and database development.
        - Create interfaces to facilitate communication between the backend and frontend.
        - Ensure excellent data security and integrity in applications.
        - Work with the frontend developer to ensure smooth integration.

4. **Sai Prasanna Rachakonda (1002090739)**

    - **Role**: Quality Assurance (QA) Tester
    - **Responsibilities**:
        - Write and execute test scenarios to identify issues and ensure functioning.
        - Conduct a usability test to ensure that the program meets user expectations.
        - Collaborate with developers to discover and address issues, and enhance app quality.
        - Identify document testing processes, test them, and report the findings.

Part – 10 - References

Academic Resources

We referenced some information from previously submitted files (Project Proposal and progress report)

1. **Sommerville, I.** (2011). *Software Engineering.* Pearson.
    - This book provided foundational knowledge on software engineering principles, which were instrumental in structuring the development process, including requirements gathering, design, testing, and maintenance phases.
2. **Alenezi, A., & Alharthi, S.** (2020). *Agile Software Development: A Systematic Review of the Literature.* Journal of Software Engineering Research and Development.
    - This literature review helped establish the framework for Agile-based project management, which supported iterative development and facilitated effective response to user feedback and evolving requirements.
3. **Mukherjee, A., & Peterson, T.** (2019). *Machine Learning in Healthcare.* Journal of Data Science Applications.
    - This article explored the use of machine learning in healthcare applications, supporting the implementation of AI-driven adherence predictions and personalized medication recommendations in the system.
4. **Zhou, X., & Wang, H.** (2021). *Security Compliance in Healthcare Information Systems.* Information Security Journal.
    - This journal article provided essential insights into security compliance in healthcare applications, including HIPAA and GDPR, which guided the development of security protocols and data protection measures in this project.

Technical Documentation and Standards

1. **React Documentation** (2023). *React.* Retrieved from https://reactjs.org/docs/getting-started.html
    - The official React documentation was used as a reference for building a responsive, cross-platform frontend interface that prioritizes accessibility and user interaction.
2. **Laravel Documentation** (2023). *Laravel.* Retrieved from https://laravel.com/docs
    - Laravel's documentation provided guidance on API creation, database management, and security protocols, facilitating robust backend development for secure data handling.
3. **Google Cloud Platform Documentation** (2023). *GCP.* Retrieved from https://cloud.google.com/docs
    - GCP documentation was instrumental in configuring cloud hosting, enabling secure and scalable storage solutions, and setting up monitoring tools for real-time performance tracking.
4. **TensorFlow Documentation** (2023). *TensorFlow.* Retrieved from https://www.tensorflow.org/learn

    - TensorFlow documentation provided resources on implementing machine learning algorithms, specifically for adherence prediction and personalized recommendations based on user behavior.
5. **Vuforia SDK Documentation** (2023). *Vuforia.* Retrieved from https://library.vuforia.com/
    - Vuforia's SDK documentation supported the development of the AR-based pill identification feature, ensuring accurate object recognition and enhancing medication identification functionality.
6. **Firebase Cloud Messaging (FCM) Documentation** (2023). *Firebase.* Retrieved from https://firebase.google.com/docs/cloud-messaging
    - Firebase Cloud Messaging was referenced for integrating a reliable notification system, supporting real-time reminders for medication intake to improve adherence.

Web Resources

1. **What is Agile?** Agile Alliance. Retrieved from https://www.agilealliance.org
    - This resource provided an overview of Agile methodologies, including principles of flexibility, continuous feedback, and incremental development, which were essential in the project's iterative approach.
2. **Constructive Cost Model (COCOMO II)**. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/COCOMO
    - Information on COCOMO II helped in estimating project costs, based on the complexity and scale of the application's development.
3. **HIPAA Compliance Guide**. HHS.gov. Retrieved from https://www.hhs.gov/hipaa
    - This guide was used to ensure that the application complies with HIPAA regulations, informing data encryption, access control, and data retention policies.
4. **GDPR Compliance in Healthcare**. GDPR.eu. Retrieved from https://gdpr.eu/healthcare/
    - This source provided guidelines on GDPR compliance in healthcare, which helped shape the data protection measures and user consent practices within the application.
5. **Open Source Libraries and Tools for Healthcare Applications**. GitHub. Retrieved from https://github.com/
    - Various open-source libraries hosted on GitHub contributed to feature implementation, including libraries for data encryption, accessibility improvements, and notification services.

Development and Testing Tools

1. **Jest**. *JavaScript Testing Framework*. Retrieved from https://jestjs.io/
    - Jest was used extensively for unit testing of the frontend components, ensuring individual functions performed accurately.
2. **Selenium**. *Browser Automation Tool*. Retrieved from https://www.selenium.dev/

    - Selenium provided an effective solution for integration testing, verifying smooth interaction between frontend and backend components.
3. **OWASP ZAP**. *Vulnerability Scanning Tool*. Retrieved from https://www.zaproxy.org/
    - This tool supported security testing by identifying vulnerabilities in the system, enabling preemptive security enhancements.
4. **Postman**. *API Testing Tool*. Retrieved from https://www.postman.com/
    - Postman was essential for testing RESTful API endpoints, verifying data consistency and seamless integration between system modules.