**CS673 Software Engineering** 

**Team 3 – My Medic App**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Adriel Domingo | Requirements | *Adriel Domingo* | 5/14/2025 |
| [John Gutierrez](mailto:johng13@bu.edu) | All | [*John Gutierrez*](mailto:johng13@bu.edu) | 5/14/2025 |
| Mengliang Tan | QA | *Mengliang Tan* | 5/14/2025 |
| Uzay Isin Alici | Security | *Uzay Isin Alici* | 5/14/2025 |
| Tyler Gonsalves | Configuration/ Design | *Tyler Gonsalves* | 5/14/2025 |
| Indra Sigicharla | Team/Configuration | *Indra Sigicharla* | 5/14/2025 |
| Hongcheng Ding | Design | *Hongcheng Ding* | 5/14/2025 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 0 | Team 3 | 5/14/2025 | N/A |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_aonibv4fd9ny)

[Related Work](#_s6j3waz3bfdw)

[Proposed High level Requirements](#_9yezyy42h484)

[Management Plan](#_ydgl41hdpdqc)

[Objectives and Priorities](#_r3vri48ie2kv)

[Risk Management (need to be updated constantly)](#_hdxu5rthgug7)

[Timeline (need to be updated at the end of each iteration)](#_es1ey4859q91)

[Configuration Management Plan](#_d2x41v5co9tv)

[Tools](#_4orylp69iqd3)

[Deployment Plan if applicable](#_tr7pd9i6owuh)

[Quality Assurance Plan](#_3yp6oo6pjwue)

[Metrics](#_9pptc6y0p6sf)

[Code Review Process](#_n3k1lnws1h46)

[Testing](#_uy308rpldzjb)

[Defect Management](#_b8slztt3vxkr)

[References](#_a8rij6lkh74z)

[Glossary](#_e5qbriq8icsq)

# Overview

**My Medic App** is a **Personal Medical** **System** designed to empower patients by giving them secure and convenient access to their complete medical history with just one click.

**Motivation:**

The growing need for **centralized patient-controlled health records, efficient care coordination, patient convenience,** and **data security**make **My Medic App** a suitable and trusty option to facilitate communication across providers, hospitals, and care systems particularly in a world with increasing mobility, digital healthcare, and chronic condition management.

**Purpose:**  
The main purpose of **My Medic App** is to serve as a **secure and comprehensive digital health record** that patients can use to store, manage, and share their medical data. This includes personal information, medical history, prescriptions, providers, appointments, and surgical records. The system ensures **interoperability** across institutions, consolidates medical records (diagnoses, prescriptions, surgeries, immunizations, lab results), improve patients confidence with a single source of truth for their health data, and enhance care efficiency by providing digital portability.

**Potential Users:**

* **Patients** (individuals managing their own or family members’ health records)
* **Healthcare providers** (clinics, hospitals, providers, insurance companies, etc.) needing patient history
* **Pharmacies** (for prescription tracking)

**Basic Functionality:**

* User registration and authentication (OAuth2/JWT)
* Viewing and editing personal health records
* Managing medical history: conditions, prescriptions, allergies, surgeries
* Scheduling and managing appointments
* Viewing and connecting with providers, clinics, and hospitals
* Notifications for upcoming appointments or prescriptions
* Analytics and reportsthat generate health trend insights.

**Possible Technology Stack:**

* **Backend APIs** Python (Django, Django REST Framework)
* **Frontend** JavaScript
* **Database** SQLite3 Django’s rest framework built-in database models and tools
* **Authentication** OAuth2 Django’s rest framework built-in authentication management

# Related Work

**Similar Software Systems and Comparison**

1. **Novelia**

**Overview:**

Novelia is a digital healthcare platform developed by Santen Pharmaceutical, mainly focused on enhancing ophthalmic care. It includes smart eye-drop dispensers, digital health monitoring, and some integrations with personal health records, primarily for eye care patients.

**Core Features:**

* Eye-drop tracking and reminders
* Integration with optometrist care plans
* Monitoring adherence to medication
* Limited scope of personal health data (mostly ophthalmic)

**Differences from My Medic App:**

| **Feature** | **Novelia** | **My Medic App** |
| --- | --- | --- |
| **Scope of Health Records** | Ophthalmology-specific | All-encompassing medical history |
| **Target Users** | Eye care patients | All patients, providers, pharmacies |
| **Platform Type** | Device-integrated + app | Full-stack cloud-native app |
| **Architecture** | Monolithic | Microservices |
| **User Control & Data Sharing** | Limited sharing | Full control over access |
| **Customization & Integration** | Low (predefined scope) | High (modular services) |

1. **One Medical Passport**

**Overview:**One Medical Passport is a pre-admission software solution used by surgery centers and hospitals. It allows patients to enter their health history before surgery, and providers to streamline the pre-operative workflow.

**Core Features:**

* Patient intake forms
* Pre-surgical health history collection
* Integration with surgery centers and hospitals
* Provider-side scheduling and risk analysis

**Differences from My Medic App:**

| **Feature** | **One Medical Passport** | **My Medic App** |
| --- | --- | --- |
| **Use Case Focus** | Pre-surgical workflow | Complete personal medical history |
| **Ownership of Records** | Provider-centric | Patient-centric |
| **Data Sharing** | Provider-controlled | Patient-controlled |
| **Continuity of Care** | Episode-based | Lifelong health history |
| **Architecture** | Traditional web system | Modern microservices stack |
| **Integration Scope** | Mostly institutional | Includes personal, pharmacy, hospitals |

# Proposed High level Requirements

* 1. **Functional Requirements**

1. **Essential Features (the core features that you definitely need to finish):**

| **Feature Title** | **Description** | **Person Hours** |
| --- | --- | --- |
| **User Registration and Login** | As a **patient**, I want to **sign up/create an account and log in securely**, so that **I can access my medical records.** | 8–12 hours |
| **Mockup** | | |
| **View/Edit Personal Info** | As a **patient,** I want to **view and update my personal information,** so that **my records are up to date.** | 8–12 hours |
| **Mockup** | | |

1. **Desirable Features (the nice features that you really want to have too):**

| **Feature Title** | **Description** | **Person Hours** |
| --- | --- | --- |
| **Notification System** | As a **patient**, I want to **receive reminders for appointments and prescriptions**, so that **I don’t miss them**. | 8–12 hours |
| **Mockup** | | |
| **Family Accounts** | As a **parent,** I want to **manage my children's health records,** so that **I can track vaccinations and pediatric visits.** | 8–12 hours |
| **Mockup** | | |

1. **Optional Features (additional cool features that you want to have if there is time):**

| **Feature Title** | **Description** | **Person Hours** |
| --- | --- | --- |
| **Voice Assistant** | As a **visually impaired user**, I want to **ask voice queries (“What’s my next appointment?”)**, so that **I can interact more easily**. | 8–12 hours |
| **Mockup** | | |

* 1. **Nonfunctional Requirements**
     1. **Scalability**
     2. **Security requirements**
        1. **Authentication & Authorization**
* The system shall enforce secure authentication using OAuth2.
* The system shall implement role-based access control (RBAC) to ensure that users only access resources permitted to their roles (e.g., Patient, Provider, Admin).
  + - 1. **Data Encryption**
* All data in transit shall be encrypted using AES-256.
* All sensitive data at rest (e.g., personal health records, prescriptions).
  + - 1. **Audit Logging**
* The system shall maintain detailed audit logs of data access and modifications, including timestamps, users, and actions performed.
  + - 1. **Secure API Design**
* All backend APIs must require authentication and validate every request's token before processing.
* APIs must implement rate limiting and input validation to prevent abuse, injection, and overuse.
  + - 1. **Data Privacy and Consent**
* Patients must have full control over their data and must explicitly grant consent before a provider can access their records

1. **Compliance**

**The system must adhere to relevant data protection regulations, such as:**

* HIPAA (for U.S. users)
* Ensure patient rights like data access, rectification, and erasure

1. **Vulnerability Management**

* The system must undergo regular security scans and vulnerability assessments.

1. **Incident Response**

* The system shall define and follow a documented incident response plan for detecting, reporting, and mitigating security breaches**.**

# Management Plan

## Objectives and Priorities

## Successfully Implement All Essential Features

## Develop and integrate core functionality including user authentication, medical history, prescriptions, appointments, and provider management.

## Ensure seamless interaction between Django REST API and React frontend.

## Deploying the Application in a Production Environment

## Package all microservices using Docker and Kubernetes.

## Deploy to a cloud environment most likely by using GCP with proper DNS and HTTPS configuration.

## Ensure High Security Standards

## Implement industry-standard authentication, encryption, role-based access control, audit logging, and data privacy controls (HIPAA compliant).

## Conduct penetration testing and secure API practices.

## Maintain Code Quality and Automated Testing

## test coverage, integration, and end-to-end tests.

## Apply CI/CD pipelines with automated testing, linting, and building validation.

## Deliver the Software with Zero Known Critical Bugs

## Resolve all critical and high-severity issues before deployment.

## Perform thorough manual QA testing to validate system behavior.

## Support Scalable, Modular Development Through Microservices

## Structure the backend into decoupled services (e.g., User, Appointment, Records) to allow teams to work independently and scale components as needed.

## Ensure Continuous Integration and Continuous Deployment (CI/CD)

## Implement CI/CD workflows using GitHub Actions or GitLab CI to automate testing, building, and deployment.

## Monitor System Health and Performance

## Integrate logging and monitoring tools (Prometheus, Grafana, ELK) for proactive alerting and diagnostics.

## Risk Management (need to be updated constantly)

The main risks identified in the development of the **My Medic App** include **integration complexity**, **data privacy/security compliance**, **deployment issues**, and **team coordination across microservices**. Integration across multiple services (e.g., authentication, medical records, appointments) introduces challenges in maintaining consistent APIs and data flow. To mitigate this, we will use **well-defined API contracts and Swagger documentation**, along with frequent integration testing. Given the sensitive nature of health data, compliance with regulations like **HIPAA** is critical. We will manage this by implementing strong encryption, audit logs, and access control mechanisms from the start. Deployment risks will be addressed by using **CI/CD pipelines, containerization (Docker), and (Kubernetes)** for repeatable, controlled deployments. Finally, to manage distributed teamwork, we will enforce clear **module ownership**, regular standups, and use **version control best practices (Git branching strategies)** to avoid conflicts.

**Risk Management Sheet Link:**

<https://docs.google.com/spreadsheets/d/1n1xKI0jsKUVCqNSs9bQoeuef1tG9JOeyPWJcRhjUrVE/edit?gid=0#gid=0>

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Desirable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | User Registration and Login(Essential) | * UML Diagrams * Write acceptance tests * Create wireframes * Create database | 20-30 |
| 2 | View/Edit personal Info (Essential) | * UML Diagrams * Write acceptance tests * Security tests (HIPAA) * Create wireframes | 8-12 |
| 3 | Notification System (Desirable) | * UML Diagrams * Write acceptance tests | 8-12 |
| 3 | Family Accounts (Desirable) | * UML Diagrams * Write acceptance tests | 8-12 |
| 3 | Voice Assistant (optional) | * UML Diagrams * Write acceptance tests | 50+ |

# Configuration Management Plan

## Tools

In addition to Git and GitHub for version control, **My Medic App** project will utilize a range of development and DevOps tools to ensure high code quality, security, and efficient deployment. **Visual Studio Code** and **PyCharm** will be the primary IDEs for backend (Django) and frontend (React) development. For CI/CD, we will use **GitHub Actions** to automate testing, linting, Docker image builds, and deployment workflows. Microservices will be containerized using Docker, with **Kubernetes** orchestrating the deployment and management of these services across staging and production environments. **Terraform** will handle infrastructure provisioning and management as **Infrastructure-as-Code (IaC)**. **AWS** will be the primary cloud provider, leveraging **Amazon EKS** for Kubernetes clusters, and Amazon S3 for object storage. For security, we will incorporate **SonarQube** for static application security testing (SAST) and **OWASP ZAP** for dynamic application security testing (DAST). Container security will be enhanced with **Trivy** for vulnerability scanning. We will also use **Prometheus and Grafana** for system monitoring, and **ELK Stack** for centralized logging. Additionally, **GitHub Copilot** and **ChatGPT APIs** may be employed to boost productivity through AI-assisted coding and documentation.

* 1. Code Commit Guideline and Git Branching Strategy

For the **My Medic App** project, we will adopt the **TrunkFlow** . The main branch includes main (for production-ready code). New features will be developed in feature/<feature-name> branches, bug fixes in bugfix/<bug-name>, and urgent production fixes in hotfix/<issue>. All changes will be submitted via **pull requests (PRs)**, which must pass automated CI checks and be reviewed by at least one team member before being merged to main. Commits should be atomic, follow a clear naming convention, and include meaningful messages. This strategy ensures a clean, traceable codebase and smooth integration of contributions from multiple teams working on different microservices.

## CI/CD Plan

To ensure rapid and reliable delivery of updates, we will implement a **CI/CD pipeline using GitHub Actions**. Upon every commit or pull request to the develop or main branches, the pipeline will automatically run unit and integration tests, perform static code analysis, and build Docker images for all microservices. Once the tests pass and the build is successful, the pipeline will push Docker images to a container registry (e.g., Docker Hub or GitHub Container Registry). Deployment and container orchestration will utilize Kubernetes, with environments managed on AWS EKS. Argo CD or Helm will continuously synchronize Kubernetes manifests from the Git repository, automating the deployment process to staging and production environments. This automated CI/CD setup ensures consistent, traceable, and reliable deployments, significantly reducing manual intervention and enhancing overall application stability and performance.

# Quality Assurance Plan

## Metrics

**UNDER INVESTIGATION**

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

| **Metric Name** | **Description** |
| --- | --- |
| LOC | Lines of code written, modified, or removed |
| Stories completed | At least 1 user story completed per week |
| Test case pass rate | Pass rate must be greater 80% |
| Defect rate | 20 bugs or less per 1000 lines of code (KLOC) |

* 1. Coding Standard

**UNDER INVESTIGATION**

For **My Medic App** project, we will adhere to established coding standards to ensure consistency, readability, and maintainability across all codebases. For the backend (Python/Django), we will follow **PEP 8** guidelines, including proper naming conventions, indentation, and documentation practices. The frontend (JavaScript/React) will follow widely accepted best practices such as using clear component structures, consistent naming conventions, and proper use of hooks and state management. All code must be reviewed through pull requests before merging, and **linters** such as flake8 (Python) and ESLint (JavaScript) will be integrated into the CI pipeline to automatically enforce these standards. In addition, code comments, docstrings, and consistent formatting will be required to support long-term maintainability and collaboration across multiple teams.

## Code Review Process

**UNDER INVESTIGATION**

(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other’s code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)

## Testing

**UNDER INVESTIGATION**

For **My Medic App**, both manual and automated testing will be employed to ensure thorough quality assurance. The testing will involve unit testing, integration testing, and user acceptance testing. **Unit testing** will be performed by developers using tools such as **JUnit** for Java or **pytest** for Python, ensuring each function or method behaves as expected. **Integration testing** will be led by the QA leader, utilizing tools like **Postman** for API testing and **Selenium** for end-to-end testing of the UI. **Automated testing** will be conducted with **JUnit**, **Selenium**, and **Cypress** for front-end testing, and **Jest** for JavaScript. The team will run unit tests during the development phase, and integration testing will be conducted after all modules are integrated. Manual testing, including exploratory testing and user acceptance testing, will be performed before deployment to identify any issues not covered by automated tests. The objectives are to ensure that the app functions seamlessly across all components, is free of bugs, and meets user requirements. The testing will occur in parallel with development cycles, with specific testing sprints before release stages to ensure high-quality standards.

## Defect Management.

**UNDER INVESTIGATION**

Defect management for **My Medic App** will be handled using **GitHub Issues** as the primary tool. This tool allows for efficient tracking, categorization, and prioritization of defects. The types of defects to be monitored include **functional defects** (issues with the app’s features), **performance defects** (slow response times or crashes), **usability defects** (issues with the user interface or experience), and **security vulnerabilities**. The QA leader will oversee the defect management process, ensuring that all defects are logged with clear descriptions and appropriate severity levels. Developers will be responsible for addressing and resolving the defects assigned to them, with periodic reviews to ensure timely fixes. Once defects are resolved, they will undergo retesting before being marked as closed. Additionally, **automated workflows** will be used to streamline the tracking process, ensuring that no defects are overlooked. The defect management process will be integrated into the development cycle to maintain a smooth workflow and prevent critical issues from reaching production.

# AI usage Log

**UNDER INVESTIGATION**

You are allowed and even encouraged to use AI tools to help you generate the project idea, plan it and build it, but you need to clearly describe 1) What tools were used? 2) for what specific tasks and 3) Is it helpful? 4) how did you evaluate or modify AI-generated content? Additionally, you should submit the exported AI chat history as an appendix or share that with the instructor and facilitators.

| Tools | Who | Tasks | helpful | Evaluation/modification | links |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# References

(Any references/citations that you have used)

# Glossary

(Any acronym used in the document should be explained here)