Patient Tracker System

Midpoint Deliverable

Riya Deshpande **Riya Danve** **Riya Adsul** **Mitali Juvekar**

GitHub Repository Link: https://github.com/riyaa74/PatientTrackerSystem

1. Requirements

1.1. Overview

The system's purpose is to revolutionize the healthcare ecosystem by providing a digitalized solution that overcomes its current inefficiencies and challenges. It ensures real-time updates for medical records, eliminating the burdens of manual paperwork, and optimizing patient management. The ultimate goal is to enhance healthcare service quality, leading to improved operational efficiency, superior patient care, and heightened patient satisfaction.

Importance of Patient Tracking:

Patient tracking is essential in healthcare because it guarantees precise patient information for informed care decisions. Accurate records help reduce errors and mitigate risks to patients' well-being. In emergencies, quick access to patient data is crucial. It streamlines administrative tasks, allowing healthcare providers to dedicate more time to patient care. Coordination among healthcare providers is improved, encouraging teamwork. Patient engagement is promoted as individuals take an active role in managing their health. Data analysis supports medical research and advances medical knowledge. Compliance with legal data protection requirements is facilitated, ensuring secure handling of patient data. Furthermore, it optimizes resource allocation and enhances operational efficiency in healthcare facilities. Efficient patient care and data management foster patient confidence and satisfaction.

Business Significance:

Users, including doctors and patients, necessitate an efficient and user-friendly platform for managing and accessing medical records, appointments, and other healthcare data. They require a system that simplifies tasks such as adding, editing, or removing patients and appointments, viewing health records, and ensuring the security and privacy of sensitive patient data.

High-Level Application Tasks:

The application serves as a comprehensive patient information management system. It has 3 interfaces. It encompasses user authentication to ensure secure access for doctors, patients, and administrators. For doctors, it facilitates patient data input and updates medical history viewing, and data visualization. Patients can access their personal information and medical records. Real-time notifications and appointment management are provided for doctors and patients. Administrators have the authority to configure user permissions and system settings. Collectively, these tasks enable the efficient management of patient information and enhance healthcare delivery.

Stakeholders:

Doctors: The primary beneficiaries who experience streamlined patient management and real-time patient information access.

Patients: Users who can conveniently manage their appointments and access their healthcare data.

Administrators: Responsible for system configuration and user permission management.

One of the main reasons to switch to digitalized patient tracker is to eliminate the risks posed by manual record keeping. Manual paperwork poses significant challenges, including data inaccuracy, inefficiency, limited accessibility, and security risks. Illegible handwriting, data entry errors, and the potential for misplaced records can lead to medical errors and compromised patient safety. Additionally, healthcare professionals spend valuable time on administrative tasks that could be better utilized for patient care. Physical records stored in different locations limit quick access and coordination among healthcare providers, especially during emergencies. The vulnerability of paper records to theft, unauthorized access, and damage poses security and privacy risks.

A digitalized patient tracking system addresses these challenges by providing real-time updates, ensuring the accuracy and accessibility of medical records, and enhancing patient management. It streamlines administrative tasks, supports coordinated care, and empowers patients to engage in their health management. The system also facilitates data analysis, compliance with legal requirements, and resource optimization, ultimately improving healthcare quality, patient happiness, and overall efficiency in the healthcare ecosystem.

1.2. Features

1. User Management: User management is a crucial feature that allows administrators to control and manage user accounts. It involves creating, modifying, and deactivating user profiles. In the context of the Patient Tracker system, it empowers administrators to assign roles and permissions to doctors, patients, and other healthcare staff. Efficient user management ensures that the right individuals have access to the appropriate data and functions within the system.

2. User Authentication: User authentication is a fundamental security feature. It ensures that only authorized users gain access to the system. In the Patient Tracker system, robust authentication methods, such as usernames and passwords, two-factor authentication, or biometric verification, guarantee the security and privacy of patient information. This authentication layer is essential for maintaining the confidentiality and integrity of medical records.

3. Data Visualization: Data visualization is a powerful tool for healthcare professionals in the Patient Tracker system. It provides doctors with graphical representations of patient health data, including trends, historical information, and key metrics. Through charts, graphs, and dashboards, healthcare providers can quickly assess a patient's health status, identify patterns, and make informed decisions about treatments and interventions.

4. Notifications/Alerts: Notifications and alerts play a critical role in ensuring timely communication within the healthcare system. In the Patient Tracker system, this feature sends real-time notifications to doctors and patients about upcoming appointments, test results, medication reminders, and important health updates. This capability enhances patient engagement and adherence to healthcare plans and helps doctors stay informed about their patients' needs.

5. Appointment Management: Appointment management is a central component of the Patient Tracker system. It allows healthcare providers to efficiently schedule, reschedule, or cancel patient appointments. Doctors can view their appointment calendars, check patient availability, and allocate time for consultations or treatments. Patients can also use this feature to request appointments, view

their scheduled visits, and receive notifications about upcoming medical visits. Effective appointment management ensures that patients receive timely and coordinated care.

1.3. Functional Requirements (Use cases)

- As a doctor, I want to add a new patient to the system.
- As a doctor, I want to edit patient information to ensure it's up-to-date.
- As a patient, I want to view my upcoming appointments.
- As a patient, I want to add or edit my scheduled appointments.
- As an admin, I want to configure user permissions for doctors and patients.
- As an admin, I want to add, edit, or remove doctors from the system.
- As a doctor, I want to view patient health records and graphs for better diagnosis.

1.4. Non-Functional Requirements

Non-functional requirements encompass aspects related to the user experience, system performance, security, and other considerations:

- Security: The system should ensure data encryption for all patient records and communications.
- Reliability: It should be highly reliable, with minimal downtime.
- Performance: The system should provide real-time data updates to ensure the accuracy of medical records.
- Usability: Users should experience a responsive and intuitive interface for ease of use
- Security & Privacy: The system should support multiple user access levels, providing security and privacy for patient data.
- Scalability: It should be scalable to accommodate future growth in the number of patients and doctors.

2. Design

2.1. Architecture Diagram

The language used for the implementation is Python. The high-level architecture diagram of the proposed patient tracker system is as below.

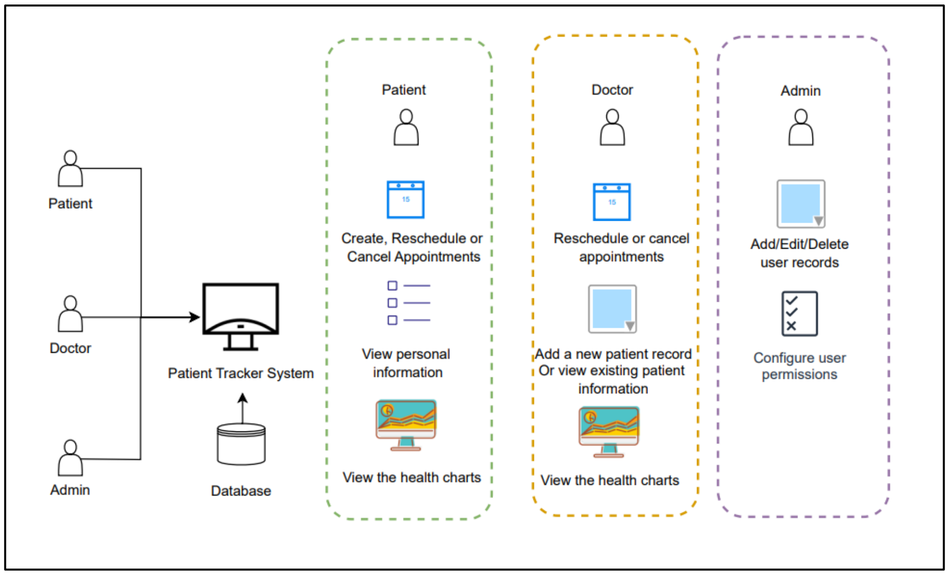

Patient
Doctor
Admin
Patient Tracker System
Database
Patient
Create, Reschedule or Cancel Appointments
View personal information
View the health charts
Doctor
Reschedule or cancel appointments
Add a new patient record Or view existing patient information
View the health charts
Admin
Add/Edit/Delete user records
Configure user permissions

Figure 1: Patient Tracker System Architecture Diagram

2.2. Technology Stack with Justification

We will be implementing our application using the following technology stack:

a. Python

Python is a popular and versatile programming language that is easy to learn and has a large and active community. It offers a wide range of libraries and frameworks, making it suitable for various tasks in application development.

b. Flask

Flask is a lightweight web framework for Python, which is well-suited for building web applications. It is easy to set up and get started with, making it a good choice for a small to medium-sized application like ours.

c. MySQL

MySQL is a reliable and widely used relational database management system (RDBMS). It provides a structured and organized way to store and manage patient data securely, making it a good choice for our application's relational data storage needs.

d. Matplotlib

Matplotlib is a popular Python library for creating static, animated, or interactive visualizations in your application. It can be used to generate various types of plots and graphs, helping us visualize patient data and medical statistics which we will display on the patient's dashboard.

e. Git

Git is a distributed version control system that will allow us to track changes in your codebase and collaborate with other developers efficiently. It ensures that we can manage the codebase of our patient information application effectively, keep track of changes, and work with the team efficiently.

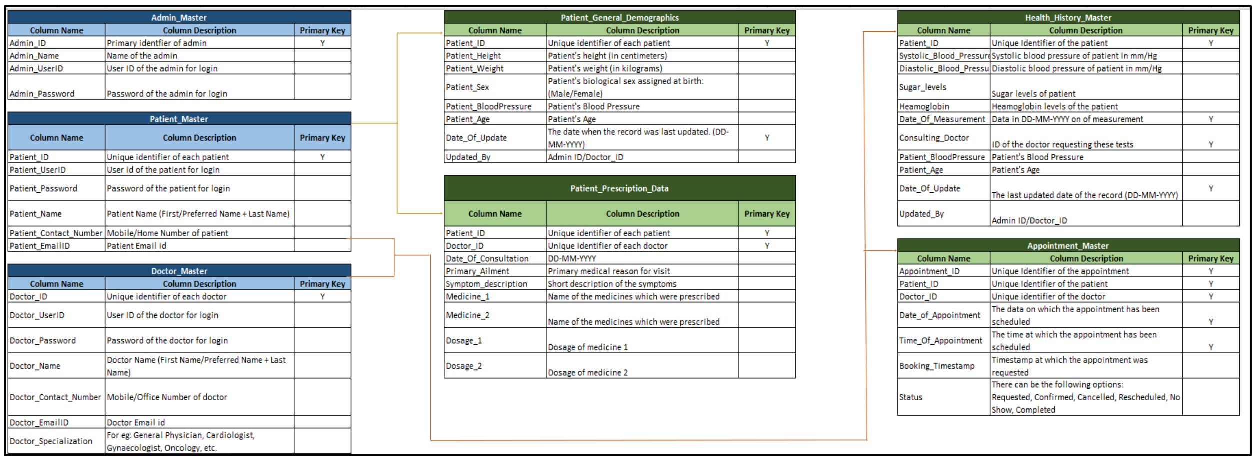
2.3. UI Mockup

The UI mockups designed for our application can be found in our git repository, the link to which is given below:

https://github.com/riyaa74/PatientTrackerSystem/tree/main/Design

2.4. Data Model:

The following diagram below illustrates the data model of the project by showing the entities, their attributes and their interdependence.



Admin_Master		
Column Name	**Column Description**	**Primary Key**
Admin_ID	Primary identfier of admin	Y
Admin_Name	Name of the admin	
Admin_UserID	User ID of the admin for login	
Admin_Password	Password of the admin for login	

Patient_Master		
Column Name	**Column Description**	**Primary Key**
Patient_ID	Unique identifier of each patient	Y
Patient_UserID	User id of the patient for login	
Patient_Password	Password of the patient for login	
Patient_Name	Patient Name (First/Preferred Name + Last Name)	
Patient_Contact_Number	Mobile/Home Number of patient	
Patient_EmailID	Patient Email id	

Doctor_Master		
Column Name	**Column Description**	**Primary Key**
Doctor_ID	Unique identifier of each doctor	Y
Doctor_UserID	User ID of the doctor for login	
Doctor_Password	Password of the doctor for login	
Doctor_Name	Doctor Name (First Name/Preferred Name + Last Name)	
Doctor_Contact_Number	Mobile/Office Number of doctor	
Doctor_EmailID	Doctor Email id	
Doctor_Specialization	For eg: General Physician, Cardiologist, Gynaecologist, Oncology, etc.	

Patient_General_Demographics		
Column Name	**Column Description**	**Primary Key**
Patient_ID	Unique identifier of each patient	Y
Patient_Height	Patient's height (in centimeters)	
Patient_Weight	Patient's weight (in kilograms)	
Patient_Sex	Patient's biological sex assigned at birth: (Male/Female)	
Patient_BloodPressure	Patient's Blood Pressure	
Patient_Age	Patient's Age	
Date_Of_Update	The date when the record was last updated. (DD-MM-YYYY)	Y
Updated_By	Admin ID/Doctor_ID	

Patient_Prescription_Data		
Column Name	**Column Description**	**Primary Key**
Patient_ID	Unique identifier of each patient	Y
Doctor_ID	Unique identifier of each doctor	Y
Date_Of_Consultation	DD-MM-YYYY	
Primary_Ailment	Primary medical reason for visit	
Symptom_description	Short description of the symptoms	
Medicine_1	Name of the medicines which were prescribed	
Medicine_2	Name of the medicines which were prescribed	
Dosage_1	Dosage of medicine 1	
Dosage_2	Dosage of medicine 2	

Health_History_Master		
Column Name	**Column Description**	**Primary Key**
Patient_ID	Unique Identifier of the patient	Y
Systolic_Blood_Pressure	Systolic blood pressure of patient in mm/Hg	
Diastolic_Blood_Pressu	Diastolic blood pressure of patient in mm/Hg	
Sugar_levels	Sugar levels of patient	
Heamoglobin	Heamoglobin levels of the patient	
Date_Of_Measurement	Data in DD-MM-YYYY on of measurement	Y
Consulting_Doctor	ID of the doctor requesting these tests	Y
Patient_BloodPressure	Patient's Blood Pressure	
Patient_Age	Patient's Age	
Date_Of_Update	The last updated date of the record (DD-MM-YYYY)	Y
Updated_By	Admin ID/Doctor_ID	

Appointment_Master		
Column Name	**Column Description**	**Primary Key**
Appointment_ID	Unique Identifier of the appointment	Y
Patient_ID	Unique Identifier of the patient	Y
Doctor_ID	Unique identifier of the doctor	Y
Date_of_Appointment	The data on which the appointment has been scheduled	Y
Time_Of_Appointment	The time at which the appointment has been scheduled	Y
Booking_Timestamp	Timestamp at which the appointment was requested	
Status	There can be the following options: Requested, Confirmed, Cancelled, Rescheduled, No Show, Completed	

Figure 1: Data Model of the proposed Patient Tracker System

3. Implementation

1. Outline the coding process, methodologies to be followed, and potential challenges.:

Process:

- Understanding Requirements and Planning: Our first step is to thoroughly understand the project requirements and create a detailed plan. This includes defining the user roles (admin, doctor, patient), their functionalities, and the data they need to interact with. We should also clarify the data schema for our SQL database.
- Setting up the Development Environment: To begin coding, we need to set up our development environment. This includes installing Flask, a SQL database, and any other necessary libraries. We'll choose a code editor or integrated development environment (IDE) and create a version control repository for our project (e.g., using Git).
- User Authentication and Login Page: Implementing user authentication is a crucial first step. We'll use Flask-Login or a similar extension to manage user sessions and create a login page. It's important to ensure that only authorized users can access their respective interfaces.
- Building the Admin Interface: The admin interface should be capable of adding, editing, and removing doctors, patients, and appointments. We will also need a configuration system to manage user permissions. Flask-WTF can be used to create forms for data input.
- Creating the Doctor Interface: The doctor interface should allow doctors to add, edit, and remove patients and appointments. We should also provide functionality for doctors to add/edit patient records and view health records and graphs. Data visualization libraries such as Matplotlib or Plotly can help with graph generation.
- Developing the Patient Interface: For patients, we'll build a user interface that allows them to add/edit/remove appointments and view their information and graphs. Data visualization will play a key role in presenting their health records.
- User Management: Implement a user management system to handle user profiles, roles, and permissions. Flask-Principal or Flask-User can help manage user roles and access control.
- Database Setup and Interaction: Set up the SQL database and define the tables for doctors, patients, appointments, and health records. Use an Object-Relational Mapping (ORM) library like SQLAlchemy to interact with the database from your Flask application.
- Data Visualization: Implement data visualization using appropriate libraries to generate health graphs and records. Matplotlib or Plotly can be integrated into the Flask application to create these visualizations.
- Notifications/Alerts: Integrate a notification system for appointment reminders and alerts. This could be done using email notifications or SMS services.

Methodologies:

- Agile Development: We'll follow an agile development methodology, breaking the project into sprints with specific goals and features for each sprint. Regular meetings and feedback loops will be essential.
- Version Control: We'll use Git and platforms like GitHub to manage and collaborate on the codebase, allowing multiple team members to work on different parts of the project simultaneously.

- Testing: Comprehensive unit and integration testing will be a priority to ensure the application works as expected. We'll consider using testing frameworks like PyTest.
- Documentation: We'll maintain documentation for code, database schemas, and user guides. This will aid in project management and future maintenance.

Potential Challenges:

- Security: Ensuring the system is secure and protecting sensitive patient data from unauthorized access.
- Scalability: As the number of users and data grows, the application should remain performant.
- User Experience: Designing a user-friendly interface for doctors and patients to efficiently manage appointments and health records.
- Data Integrity: Ensuring the accuracy and integrity of patient health records and appointment data.
- Regulatory Compliance: Adhering to healthcare regulations and data privacy laws, such as HIPAA if applicable.
- Notifications: Implementing a reliable notification system that delivers alerts and reminders to users on time.
- Data Visualization: Creating informative and visually appealing health graphs and records.
- Database Management: Managing database migrations and backups as the application evolves.

3.1. Security and Risks

1. **User Authentication and Authorization:**
   - Threat: Unauthorized access to the system by someone impersonating a legitimate user.
   - Vulnerability: Weak password policies and lack of multi-factor authentication.
   - Prevention: Implement strong password policies, encourage users to use unique and complex passwords, and consider implementing multi-factor authentication to enhance security.
2. **Data Storage and Access Control:**
   - Threat: Unauthorized users gaining access to sensitive patient data.
   - Vulnerability: Insufficient access controls and improper database configurations.
   - Prevention: Use proper role-based access control (RBAC) to limit what each user type (admin, doctor, patient) can access and ensure the database is properly configured with appropriate permissions.
3. **Data Encryption:**

- Threat: Data interception during transmission between the client and the server.
- Vulnerability: Unencrypted data transmission.
- Prevention: Implement SSL/TLS for secure data transmission between clients and the server to encrypt data in transit.

4. **SQL Injection:**
    - Threat: Malicious users exploiting vulnerabilities in SQL queries to access or modify the database.
    - Vulnerability: Insufficient input validation and lack of prepared statements.
    - Prevention: Sanitize user inputs, use prepared statements, and employ ORM (Object-Relational Mapping) tools to prevent SQL injection attacks.
5. **Data Backups:**
    - Threat: Data loss due to system failures, accidental deletion, or other unforeseen events.
    - Vulnerability: Inadequate or non-existent data backup procedures.
    - Prevention: Regularly backup the database and implement backup and recovery procedures to ensure data availability.
6. **Session Management:**
    - Threat: Unauthorized access to another user's session.
    - Vulnerability: Insecure session management.
    - Prevention: Use secure session management practices, such as generating and storing session tokens securely.
7. **Notification and Alert Security:**
    - Threat: Unauthorized users receiving sensitive notifications or alerts.
    - Vulnerability: Lack of access control for notifications.
    - Prevention: Ensure that notifications and alerts are only sent to authorized users based on their roles and permissions.
8. **Cross-Site Scripting (XSS):**
    - Threat: Malicious scripts injected into web pages viewed by other users.
    - Vulnerability: Inadequate input validation and output encoding.
    - Prevention: Implement proper input validation and output encoding to mitigate XSS attacks.
9. **Data Visualization Security:**
    - Threat: Unauthorized access to patient health records and graphs.
    - Vulnerability: Inadequate access controls for data visualization.

- Prevention: Apply access controls to patient health records and graphs, ensuring only authorized users can access this information.

To prevent these potential threats and vulnerabilities in our patient tracking system, we will follow best practices for security, including proper authentication, access control, encryption, and data validation. Regularly updating and patching the software and conducting security testing will also be essential for maintaining the security of our system.

3.2 HIPAA Compliance (in case of Patient Tracker)

The Health Insurance Portability and Accountability Act (HIPAA) is a crucial federal law in the United States that was enacted to protect patients' sensitive health information. HIPAA sets standards and regulations for the security and privacy of protected health information (PHI). Our project, the patient tracking system, is closely related to healthcare and involves the storage and management of patient data, making HIPAA compliance extremely relevant. The key aspects of HIPAA that pertain to our project are:

1. **Privacy Rule:** HIPAA's Privacy Rule mandates the protection of patients' PHI by restricting access to authorized personnel only. It is vital for ensuring that patients' personal information, medical history, and health records remain confidential and secure.
2. **Security Rule:** HIPAA's Security Rule outlines the requirements for safeguarding electronic PHI (ePHI). Since our system will involve storing patient records in a database, we must adhere to these security standards to prevent data breaches or unauthorized access.

Steps and protocols to ensure HIPAA compliance in our project:

1. **Access Control:** Implement robust access control mechanisms to ensure that only authorized users, such as healthcare professionals and administrators, can access the system. Role-based access control (RBAC) can be employed to restrict access to specific functions and data.
2. **Encryption:** Encrypt data both in transit and at rest. Use secure protocols (e.g., HTTPS) to protect data during transmission and encryption mechanisms to secure data stored in the database.
3. **User Authentication and Authorization:** Implement strong user authentication methods, such as two-factor authentication (2FA), to verify the identity of users. Authorize users to access only the data and functions necessary for their roles.
4. **Incident Response Plan:** Develop a comprehensive incident response plan to address potential data breaches. This plan will include protocols for reporting and mitigating breaches while complying with HIPAA breach notification requirements.
5. **Regular Audits and Assessments:** Periodically conduct security risk assessments and audits to identify vulnerabilities and ensure ongoing compliance. Address any issues promptly.
6. **Documentation:** Maintain thorough records of policies, procedures, and security measures to demonstrate compliance in case of audits or investigations.

4. Work Plan

-Project Execution Roadmap:

Phase 1 - Project Initiation (Week 1-2):

Define project scope and objectives.

Identify stakeholders and their requirements.

Set up project team roles and responsibilities.

Conduct initial research on healthcare systems.

Responsibilities:

Group Member 1: Gather project requirements and conduct initial research.

Group Member 2: Create the project charter, outline objectives, and identify stakeholders.

Group Member 3: Define high-level timelines and project milestones.

Group Member 4: Assist in requirements gathering and stakeholder analysis.

Phase 2 - Planning and Design (Week 3-5):

Develop a detailed project plan, including timelines and deliverables.

Create user stories and use cases for the application.

Design the user interface and database schema.

Choose the technology stack and tools for development.

Responsibilities:

Group Member 1: Develop the project plan and timeline.

Group Member 2: Create user stories and use cases.

Group Member 3: Design the user interface and database schema.

Group Member 4: Assist in technology stack selection and tool evaluation.

Phase 3 - Development (Week 6-10):

Begin coding the Patient Tracker System, following the project plan.

Implement user authentication and basic user management.

Build the core functionalities of patient data management.

Integrate data visualization and notifications/alerts features.

Responsibilities:

Group Member 1: Work on user authentication and user management.

Group Member 2: Implement patient data management features.

Group Member 3: Integrate data visualization and notifications/alerts.

Group Member 4: Collaborate on core functionality development and testing.

Phase 4 - Testing and Quality Assurance (Week 11-12):

Conduct unit testing to ensure individual components work correctly.

Perform integration testing to verify interactions between system modules.

Carry out user acceptance testing with potential end-users.

Address and resolve any identified issues and bugs.

Responsibilities:

Group Member 1: Lead unit testing efforts.

Group Member 2: Oversee integration testing and coordinate user acceptance testing.

Group Member 3: Collaborate on bug identification and resolution.

Group Member 4: Assist in user acceptance testing and bug tracking.

Phase 5 - Deployment and Rollout (Week 13-14):

Prepare the system for production use.

Deploy the Patient Tracker System in a controlled environment.

Train end-users (doctors, patients, administrators) on system usage.

Monitor the system's performance and gather user feedback.

Responsibilities:

Group Member 1: Manage deployment tasks.

Group Member 2: Organize user training sessions.

Group Member 3: Oversee system performance monitoring and gather user feedback.

Group Member 4: Assist in system deployment and user training.

High-level Timeline:

Week 1-2: Project initiation and requirements gathering.

Week 3-5: Planning, design, and technology selection.

Week 6-10: Development and core functionality implementation.

Week 11-12: Testing and issue resolution.

Week 13-14: Deployment, training, and monitoring.

This updated roadmap includes responsibilities and planned contributions for all four group members, ensuring a well-organized and collaborative approach to the project development.