BLG 411E SOFTWARE ENGINEERING
ADVANCED HOSPITAL MANAGEMENT SYSTEM
PROJECT PLAN

TEAM NAME:
GOFIES

MEMBERS:

Abdullah Shamout

Berkay Gürsu

İbrahim Halil Marsil

Mustafa Can Çalışkan

Rasim Berke Turan

Sarper Öztorun

Yusuf Emir Sezgin

Yusuf Şahin

27.10.2024

Table of Contents

1. INTRODUCTION

The Advanced Hospital Management System project aims to create a comprehensive, secure, and scalable platform tailored to modern healthcare needs. It will integrate essential hospital management functions such as patient, doctor, lab, and administrative services, supported by advanced features like role-based access control, and extensive data encryption. Leveraging a monolithic architecture, the system will employ RESTful APIs, JWT-based authorization, and a multilingual, responsive design for accessibility. Additionally, the project will incorporate an AI-driven doctor expert system to enhance patient interaction, backed by large language models for realistic, expert-like responses, and a robust DevOps pipeline to streamline deployment and maintenance.

1.1 Scope

+ Users can securely log into the system using JWT-based authorization.
+ Users can access specific features based on their role, such as Patient, Doctor, Lab Staff, or Admin.
+ Patients can view their medical records, appointments, and test results.
+ Doctors can access patient history, add notes, and update records for patients under their care.
+ Lab staff can upload test results and manage lab data in the system.
+ Admins can monitor system activities, review audit logs, and manage user access levels.
+ Users can interact with an AI chatbot that provides general medical information and doctor-like responses for initial inquiries.
+ Users receive real-time notifications on appointments, test results, and updates via in-app alerts.
+ The system supports multilingual functionality and customizable color themes (e.g., night mode).
+ The interface is fully responsive, ensuring usability on mobile, tablet, and desktop devices.
+ Regular CI/CD deployment and Docker container orchestration ensure continuous system updates and cloud-based scalability.

1.1.1 Out of Scope:

- The application does not include in-app payments or billing for hospital services.
- Users cannot view or share information about other users' profiles or data due to strict data privacy controls.
- The project does not involve creating new databases; it only manages and retrieves data from existing hospital databases but we will have a database designed and implemented with dummy data for proof of concept purposes.
- Users cannot directly upload or modify system data without appropriate roles (e.g., only lab staff can upload test results).

1.1.2 Optional Scope:

- Admins can monitor and analyze general user actions to gain insights into how the system is used, improving user experience and efficiency.
- The system can support Progressive Web App (PWA) capabilities, allowing offline access and enhanced user experience on mobile devices.
- The system includes lazy loading and code splitting for improved performance.
- Users can access interactive UI animations and feedback to enhance the visual experience.
- A reverse proxy load balancer can be implemented for cloud deployment, improving system resilience and uptime.

1.2 Deliverables

Deliverable Number	Deliverable	Delivery Date
1	Project Plan	27/10/2024
1.1	Scope of Project	27/10/2024
1.2	WBS of Project	27/10/2024
2	Requirements Specification	12/11/2024
3	UI/UX Designs Specification	02/12/2024
4	Backend Services	16/12/2024
5	Frontend Services	16/12/2024
8	Database Services	16/12/2024
6	AI Services	27/12/2024
7	Testing	27/12/2024

1.3 Functional Requirements

Deliverable Number	**Requirement**
4	User Authentication
4	Lab Technician Services
4	JWT-based Login
4	IT Technician Services
4	Admin Dashboard
4	Lab Results
4	API- Database Integration
4	Role-Based Access Control
4	Patient Services
4	Viewing Medical Records
4	Appointment Management

4	Doctor Services
4	Access to Patient Records
4	Patient Note Taking
4	Lab Staff Services
4	Prescriptions
6	Doctor Expert System
6	Feeding LLM with Specialized Models
6	AI Integration / AIOps
3	UI/UX Customization
3	Multilingual Support
3	Color Theme Options
8	Scalability and Backup of Medical Records
8	Medical Data Security and Encryption within Database
8	Automated Archiving and Audit Logging for Medical Records
8	Distributed Data Storage and Backup of Medical Data
7	Unit and Integration Testing
7	API Testing
7	Code Coverage Analysis
7	Security and Penetration Testing

1.4 Non-functional issues

Performance: The system is optimized for handling moderate to high traffic. However, in case of increased simultaneous usage, the development team will monitor and address any performance bottlenecks.

Portability: The application is designed to be cross-platform and will be accessible on various devices (e.g., mobile, desktop).

Reliability: As the application depends on external hospital databases, its reliability is linked to the availability of those data sources.

Reusability: Core functions such as data retrieval, searching, and user authentication are modular and can be adapted for other projects if required.

Security: All user interactions are tightly controlled with no direct database modifications by users, ensuring data security. Sensitive data is encrypted both in transit and at rest.

Usability and Appearance: Usability is a high priority, with continuous improvement based on front-end design expertise and user feedback to maintain an intuitive and appealing interface.

2. PROJECT PLAN

1. Project Management

- (a) Project Planning (L)
  - i. Define project goals and objectives (M)
  - ii. Establish project scope and constraints (M)
  - iii. Develop project schedule and timelines (L)
- (b) Risk Management (M)
  - i. Identify potential project risks (M)
  - ii. Develop risk mitigation strategies (M)
  - iii. Monitor risk factors throughout the project (S)
- (c) Resource Allocation (M)
  - i. Assign tasks to team members (M)
  - ii. Track resource availability and utilization (S)
  - iii. Adjust allocations based on project needs (M)
- (d) Progress Tracking (M)
  - i. Conduct weekly status meetings (S)
  - ii. Update project timelines (S)
  - iii. Document progress reports (M)
- (e) Quality Assurance (L)
  - i. Define quality standards (M)
  - ii. Perform regular quality checks (L)
  - iii. Address and resolve quality issues (M)

2. Database Development

- (a) Database Architecture Design (L)
  - i. Design database schema (L)
  - ii. Establish database relationships and constraints (M)
  - iii. Create data models (L)
- (b) Security and Encryption Implementation (M)
  - i. Implement encryption for data at rest (M)
  - ii. Secure data in transit (M)
- (c) Distributed Data Storage and Backup (M)
  - i. Configure distributed storage (M)
  - ii. Set up regular data backup schedules (M)
- (d) Multi-tenancy and Scalability (L)
  - i. Design multi-tenant architecture (L)
  - ii. Implement scalability strategies (L)
- (e) Automated Data Archiving and Audit Logging (M)
  - i. Create automated archiving scripts (M)

ii. Set up audit log retention policies (M)

(f) Testing and Optimization (M)

i. Perform database performance testing (M)

ii. Optimize database queries (M)

3. Backend Development

(a) Design of API Architecture (L)

i. Define API endpoints and methods (M)

ii. Establish data serialization formats (e.g., JSON, XML) (M)

(b) Patient Services and Functions (L)

i. Implement patient registration and management (L)

ii. Develop appointment scheduling features (M)

(c) Doctor Services and Functions (L)

i. Implement doctor profile management (M)

ii. Appointment schedule and patient information screen.

iii. Add prescription and diagnosis functions (M)

(d) Lab Services and Functions (L)

i. Integrate lab test ordering and results (L)

(e) Admin Services and Functions (L)

i. Develop a screen for all management system data

ii. Develop system configuration settings (M)

(f) Authentication and Authorization (JWT & RBAC) (L)

i. Configure JWT authentication (M)

ii. Set up role-based access controls (M)

iii. Configure access control policies (M)

(g) Logging Setup (M)

i. Configure logging for API requests (M)

4. Frontend Development

(a) Implementation of Patient Pages and Components (L)

i. Design patient dashboard layout (M)

ii. Develop patient registration form (M)

(b) Doctor Pages and Components (L)

i. Create doctor dashboard interface (M)

ii. Implement doctor-patient interaction features (L)

(c) Lab Staff Pages and Components (L)

i. Design lab test management interface (M)

ii. Add lab result entry forms (M)

(d) Admin Dashboard (L)

i. Develop admin control panel (L)

ii. Integrate system analytics (M)

(e) State Management (M)

i. Configure state management libraries (e.g., Redux) (M)

ii. Set up data flow between components (M)

(f) Error Handling and User Feedback (M)

i. Implement global error handling (M)

ii. Add user notifications for actions (S)

(h) Multi-language and Accessibility Support (M)

i. Add language selection options (M)
ii. Configure accessibility settings (M)

5. UI/UX Design

(a) Design of Patient Interfaces (M)
i. Create wireframes for patient pages (M)
ii. Apply design feedback for usability (M)
(b) Design of Doctor Interfaces (M)
i. Develop doctor page layouts (M)
(c) Design of Lab Staff Interfaces (M)
i. Integrate lab test result views (M)
(d) Admin Interface Design (M)
i. Design admin control interfaces (M)
ii. Add configuration options (S)
(e) Night Mode and Accessibility Settings (M)
i. Implement dark mode support (M)
ii. Optimize for color blindness (S)
(f) Iterative Design Adjustments (L)
i. Make design adjustments based on feedback (M)

6. AI and Machine Learning

(a) Development of Doctor Imitating Chatbot (L)
i. Define chatbot conversation flows (M)
ii. Implement NLP capabilities (L)
(b) Integration of Specialized Models with LLM (L)
i. Train specialized models (L)
ii. Integrate models into LLM-based services (L)
(c) Training and Fine-tuning of AI Models (L)
i. Collect training data (M)
ii. Perform model validation and testing (L)
(d) Deployment of AI Features in Production (M)
i. Set up AI deployment pipelines (M)
ii. Monitor AI performance in production (M)
(e) Ongoing AI Performance Monitoring (M)
i. Set up monitoring dashboards (M)
ii. Implement alerting for AI anomalies (S)

7. Testing and Quality Assurance

(a) Unit and Integration Testing (L)
i. Write unit test cases (M)
ii. Perform integration testing for modules (L)
(b) API Testing (M)
i. Test API endpoints for correctness (M)
ii. Validate response formats and error handling (M)
(c) Code Coverage Analysis (M)
i. Measure code coverage during testing (M)
ii. Improve coverage where necessary (M)
(d) Security and Penetration Testing (L)

        - i. Perform security audits (L)
        - ii. Conduct penetration testing (L)

8. DevOps and Deployment

- (a) Setup CI/CD Pipeline (L)
    - i. Configure build automation (M)
    - ii. Implement deployment scripts (M)
- (b) Docker Container Orchestration (L)
    - i. Set up Docker environment (L)
    - ii. Manage container orchestration using docker compose(L)
- (c) Cloud Deployment (L)
    - i. Configure cloud infrastructure (L)
    - ii. Deploy application to cloud environment (L)
- (d) Reverse Proxy and Load Balancing Setup (M)
    - i. Configure Nginx as reverse proxy (M)
    - ii. Set up load balancing (M)
- (e) Monitoring and Alerting Configuration (M)
    - i. Set up application performance monitoring (M)
    - ii. Configure alerting for system issues (M)
- (f) Setup a makefile (S)

9. Final Delivery and Documentation

- (a) Technical Documentation (M)
    - i. Document API specifications (M)
    - ii. Record system architecture details (M)
- (b) Final Testing and Bug Fixes (M)
    - i. Conduct final round of testing (M)
    - ii. Resolve outstanding bugs (M)
- (c) Project Handover (M)
    - i. Conduct handover meetings (M)

3. ESTIMATES

Man/Week Estimations

- **Tiny (T):** 0.5 man/weeks
- **Small (S):** 1 man/week
- **Medium (M):** 2 man/weeks
- **Large (L):** 3-5 man/weeks
- **Extra Large (XL):** 6-8 man/weeks

Data Functions

- ***Internal Logical Files (ILF):***
    1. Database for patient records: High (7)
    2. Doctor profiles database: Average (5)
    3. Lab results database: High (7)
    4. Admin user roles: Average (5)
- ***External Interface Files (EIF):***

1. Integration with external AI services: High (7)
2. External medical data sources: Average (5)

Transactional Functions

- ***External Input (EI):***
  1. Patient registration: Average (4)
  2. Appointment scheduling: Average (4)
  3. Doctor profile updates: Average (4)
  4. Lab result submission: High (6)
  5. Admin user role management: Average (4)
  6. State management updates: Low (3)
- ***External Output (EO):***
  1. Display patient dashboard: Average (5)
  2. Generate lab reports: High (7)
  3. Doctor interaction summaries: Average (5)
  4. System analytics dashboard: High (7)
  5. Multi-language support configuration: Low (4)
- ***External Query (EQ):***
  1. Search patient records: Average (5)
  2. Query lab results: Average (5)
  3. User access queries: Low (3)

Type	Complexity	Count	Weight	Total FP
ILF	Low	0	7	0
	Average	2	10	20
	High	2	15	30
EIF	Low	0	5	0
	Average	1	7	7
	High	1	10	10
EI	Low	1	3	3
	Average	4	4	16
	High	1	6	6
EO	Low	1	4	4
	Average	2	5	10
	High	2	7	14
EQ	Low	1	3	3
	Average	2	4	8
	High	0	6	0

Total UFP = 131

Rating General System Characteristic (GSC) on a scale of 0-5:

- **Data communications:** 4
- **Distributed data processing:** 3
- **Performance:** 4
- **Heavily used configuration:** 4
- **Transaction rate:** 3
- **On-line data entry:** 4
- **End-user efficiency:** 5
- **On-line update:** 3
- **Complex processing:** 4
- **Reusability:** 5
- **Installation ease:** 3
- **Operational ease:** 3
- **Multiple sites:** 2
- **Facilitate change:** 4

Total **GSC** = 51

VAF = 0.65 + 0.01 × 51 = 1.16

Calculate Adjusted Function Points (AFP)

AFP = 131 × 1.16 = 151.96

4. RESOURCES

4.1. Team Members

Member Name	**Member ID**	**Member Role**
Abdullah Shamout	150200919	Project Leader / AI-Engineer / DevOps Engineer
Berkay Gürsu	150200046	Front-end Developer
İbrahim Halil Marsil	150200026	Back-end Developer / Scrum Master
Mustafa Can Çalışkan	150200097	Database Engineer / Tester
Rasim Berke Turan	150200042	Back-end Developer / Client Representative
Sarper Öztorun	150200071	Meetings Organiser / Front-End Developer
Yusuf Emir Sezgin	150200066	UI / UX Designer

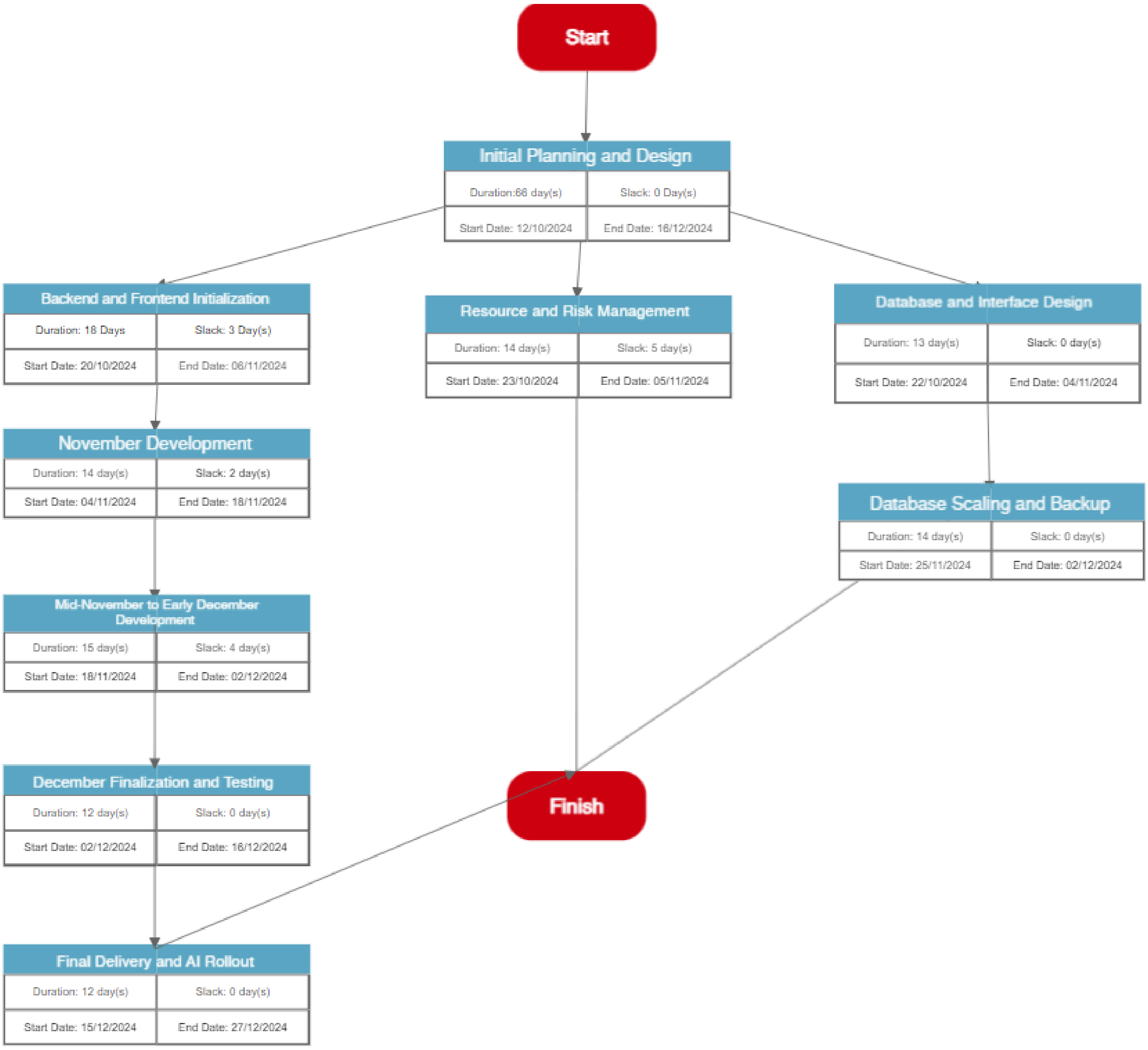
Yusuf Şahin	150200016	Tester / AI-Engineer

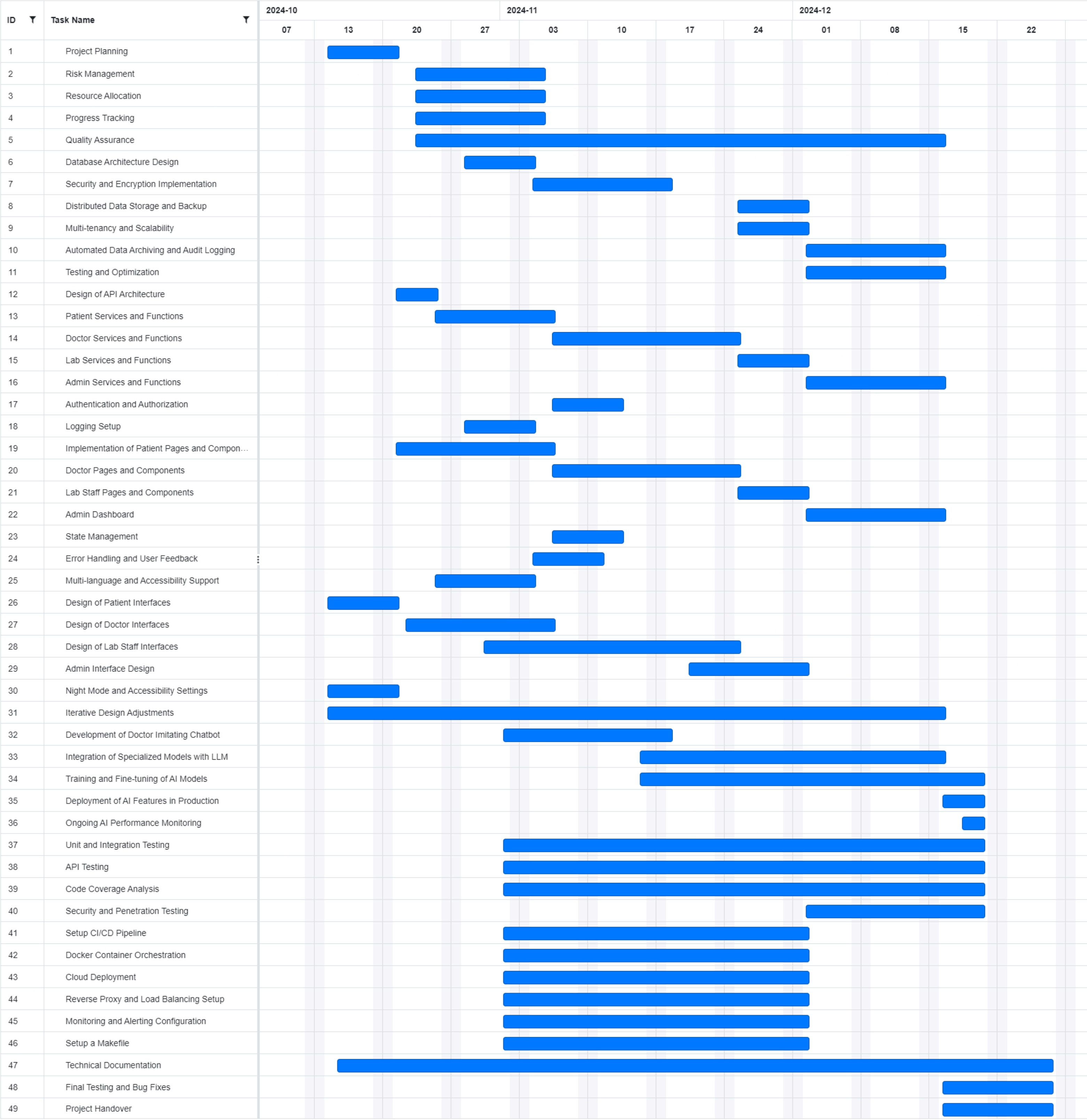
4.2. Mapping Tasks To Member

Member Name	**Assigned Task**
Abdullah Shamout	CI/CD pipeline Orchestration of Docker containers Cloud Deployment Doctor imitating Expert System chat bot Integration between different technologies Web server setup Port and data forwarding and routing
Berkay Gürsu	Implementation of layouts and components using React.js State management using Redux Implementation of frontend - backend integration Frontend Team Leader
İbrahim Halil Marsil	Create Backend Tasks Ensuring communication between backend and frontend Design Backend API's Scrum Master Backend Tasks reviews Implement Backend Functions And API's
Mustafa Can Çalışkan	Database architecture design and setup Data security and encryption within the database context Distributed data storage and backup Automated data archiving and audit log Responsible for facilitating communication between the backend and database teams.
Rasim Berke Turan	Implement backend security protocols Ensuring all backend features meet quality and user requirements Implement backend - database integration Implement Backend Functions And API's
Sarper Öztorun	Integration of notifications and multi-language options Dark Mode and Theme Implementations Pentesting Test Case Developer Scheduling meetings
Yusuf Emir Sezgin	Design front-end user interface with React Lead UI/UX design to ensure an intuitive, user-friendly interface. Designing visual prototypes Frontend API Integration

Yusuf Şahin	Running and managing tests & test cases Reporting bugs Database and Dackend assistant Feeding LLM with specialized models

5. SCHEDULE



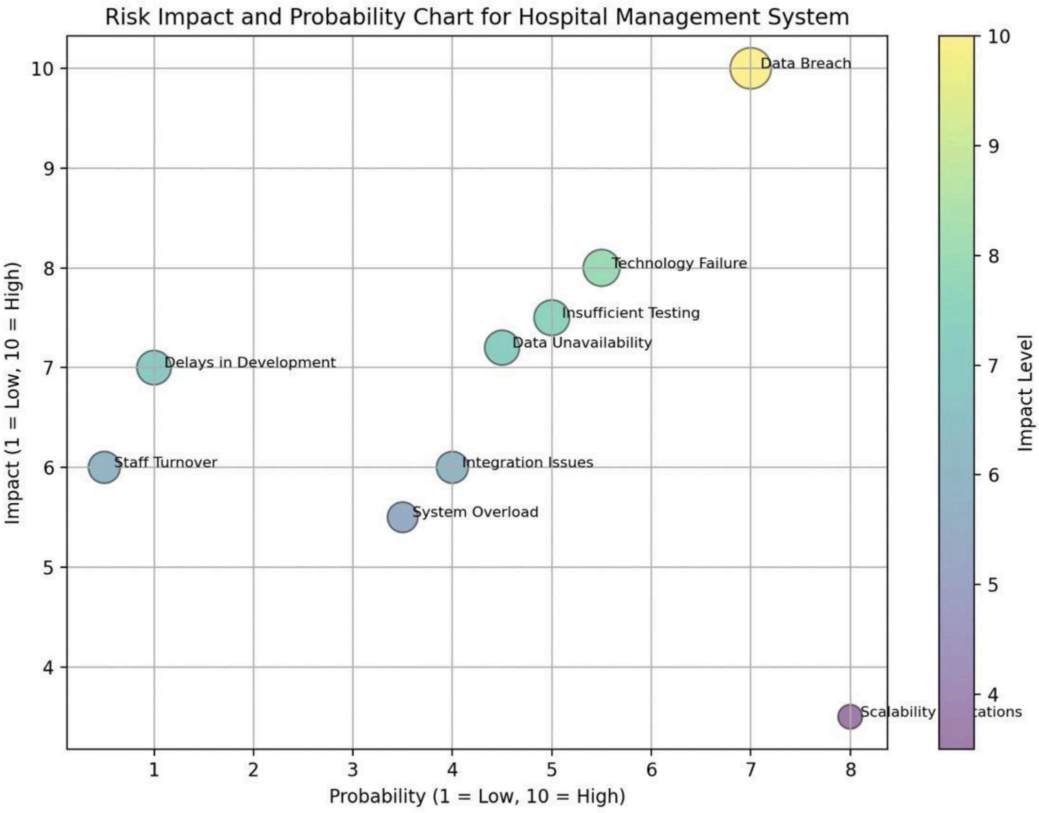


ID	Task Name
1	Project Planning
2	Risk Management
3	Resource Allocation
4	Progress Tracking
5	Quality Assurance
6	Database Architecture Design
7	Security and Encryption Implementation
8	Distributed Data Storage and Backup
9	Multi-tenancy and Scalability
10	Automated Data Archiving and Audit Logging
11	Testing and Optimization
12	Design of API Architecture
13	Patient Services and Functions
14	Doctor Services and Functions
15	Lab Services and Functions
16	Admin Services and Functions
17	Authentication and Authorization
18	Logging Setup
19	Implementation of Patient Pages and Compon…
20	Doctor Pages and Components
21	Lab Staff Pages and Components
22	Admin Dashboard
23	State Management
24	Error Handling and User Feedback
25	Multi-language and Accessibility Support
26	Design of Patient Interfaces
27	Design of Doctor Interfaces
28	Design of Lab Staff Interfaces
29	Admin Interface Design
30	Night Mode and Accessibility Settings
31	Iterative Design Adjustments
32	Development of Doctor Imitating Chatbot
33	Integration of Specialized Models with LLM
34	Training and Fine-tuning of AI Models
35	Deployment of AI Features in Production
36	Ongoing AI Performance Monitoring
37	Unit and Integration Testing
38	API Testing
39	Code Coverage Analysis
40	Security and Penetration Testing
41	Setup CI/CD Pipeline
42	Docker Container Orchestration
43	Cloud Deployment
44	Reverse Proxy and Load Balancing Setup
45	Monitoring and Alerting Configuration
46	Setup a Makefile
47	Technical Documentation
48	Final Testing and Bug Fixes
49	Project Handover

6. RISKS

RISK IMPACT AND PROBABILITY

Risk	Description	Impact	Probability
Staff Turnover	Loss of key project members which may delay project milestones.	Moderate	Very Low
Delays in Development	Delays due to complexity, lack of expertise, or unforeseen circumstances.	High	Very Low
System Overload	Overloading of server resources leading to downtime or performance issues.	Moderate	Low
Integration Issues	Problems in integrating various technologies like Express JS, React, MongoDB, etc.	Moderate	Moderate
Scalability Limitations	Inability of the system to handle increased loads, potentially leading to performance degradation.	Low	Very High
Data Unavailability	Potential loss of access to essential data for AI functionalities due to technical or operational failures.	High	Moderate
Insufficient Testing	Bugs and errors due to inadequate testing phases.	High	Moderate
Technology Failure	Failure of backend, frontend, or AI components due to technical malfunctions.	High	Moderate
Data Breach	Unauthorized access to sensitive data due to security vulnerabilities.	Very High	High



Risk	Mitigation Strategy	Mapped Activities

Staff Turnover	Plan for knowledge transfer, maintain up-to-date documentation, and possibly have contingency staffing.	Management and Scrum Master tasks
Delays in Development	Use agile project management, frequent reviews, and adjust timelines as necessary.	Project management and team leader tasks
System Overload	Optimize code, scale resources dynamically, and use load balancers.	Backend - Frontend optimizations and DevOps tasks
Integration Issues	Use standardized APIs, ensure compatibility tests, and maintain a flexible integration architecture.	Backend, frontend, and AI integration tasks
Scalability Limitations	Design system architecture for scalability using load balancers and elastic cloud resources.	Infrastructure planning, application of scalable technologies, and regular performance testing tasks.
Data Unavailability	Implement comprehensive data management and redundancy systems to ensure data is always accessible.	Data governance, regular backups, and system monitoring tasks.
Insufficient Testing	Increase test coverage, include integration and system testing, use automated testing tools.	Testing phase and continuous integration tasks
Technology Failure	Implement robust error handling, regular backups, and redundancy systems.	Backend -Frontend Meeting and DevOps tasks
Data Breach	Enforce strict access controls, use encrypted communications, and conduct regular security audits.	Security implementations in backend and database