Computer science
Project Seminar

DocScheduler

Technical documentation

Mentor:

Prof. Tatjana Zrimec

Students:

Dušan Todorović
Nemanja Cvetić

June 2025.

Table of contents

1. Project Overview

The Medical Appointment Management System represents a comprehensive web-based solution designed to streamline the healthcare appointment booking and management process. This system addresses the growing need for digital transformation in healthcare administration by providing an integrated platform that connects patients, doctors, and healthcare administrators through a unified interface.

The application serves as a centralized hub where patients can register, view their medical records, and schedule appointments with healthcare providers across multiple hospitals. Medical professionals can access their appointment schedules, manage patient consultations, and maintain medical records through the platform.

The system is built upon modern web technologies, utilizing Flask as the primary web framework, Supabase for database management and authentication, and Bootstrap for responsive user interface design.

The core functionality encompasses user registration and authentication, appointment scheduling with real-time availability checking and medical record management. The system implements role-based access control, ensuring that sensitive medical information is protected while allowing appropriate access levels for different user types.

2. System Analysis

Healthcare appointment management traditionally relies on manual processes that are prone to errors, inefficiencies, and patient dissatisfaction. Phone-based booking systems create bottlenecks, while paper-based medical records limit accessibility and increase the risk of data loss. The digital transformation of healthcare services has become essential for improving patient care quality and operational efficiency.

Modern healthcare facilities require integrated systems that can handle the complexity of multi-doctor, multi-hospital environments while maintaining strict security and privacy standards. The challenge lies in creating a system that is sophisticated enough to handle complex scheduling scenarios yet simple enough for users of varying technical proficiency to navigate effectively.

The analysis of existing healthcare management systems reveals common limitations including poor user experience design, inadequate integration capabilities, and inflexible scheduling mechanisms. Many systems fail to provide real-time availability updates, leading to double-booking scenarios and patient frustration. Additionally, the lack of comprehensive medical record integration means that doctors often work with incomplete patient information during consultations.

Our system addresses these challenges by implementing a modern web-based architecture that prioritizes user experience, data integrity, and system reliability. The use of cloud-based database technology ensures data accessibility while maintaining security standards required for healthcare applications. The responsive design approach ensures that the system functions effectively across desktop and mobile platforms, accommodating the diverse ways healthcare professionals and patients interact with technology.

The system architecture follows modern web development practices, through Flask's framework structure. The integration with Supabase provides enterprise-grade database capabilities including automatic backups, real-time synchronization, and built-in security features.

3. Functional Requirements

The development of the Medical Appointment Management System was guided by analysis conducted through consultation with potential end users.

Functional requirements encompass the core capabilities that users expect from the system. Patient registration functionality must support information collection while maintaining simplicity in the registration process. The system must provide secure authentication mechanisms that protect sensitive medical data while allowing convenient access for legitimate users.

Appointment scheduling represents the system's primary functionality, requiring sophisticated calendar management capabilities that can handle multiple doctors across different hospitals. The system must provide real-time availability checking to prevent double-booking scenarios while offering flexible scheduling options that accommodate various appointment types and durations.

Medical record management requires the system to maintain patient histories including previous appointments, medical findings, and treatment outcomes. The system must support the addition of medical findings by healthcare providers while maintaining audit trails for all medical record modifications.

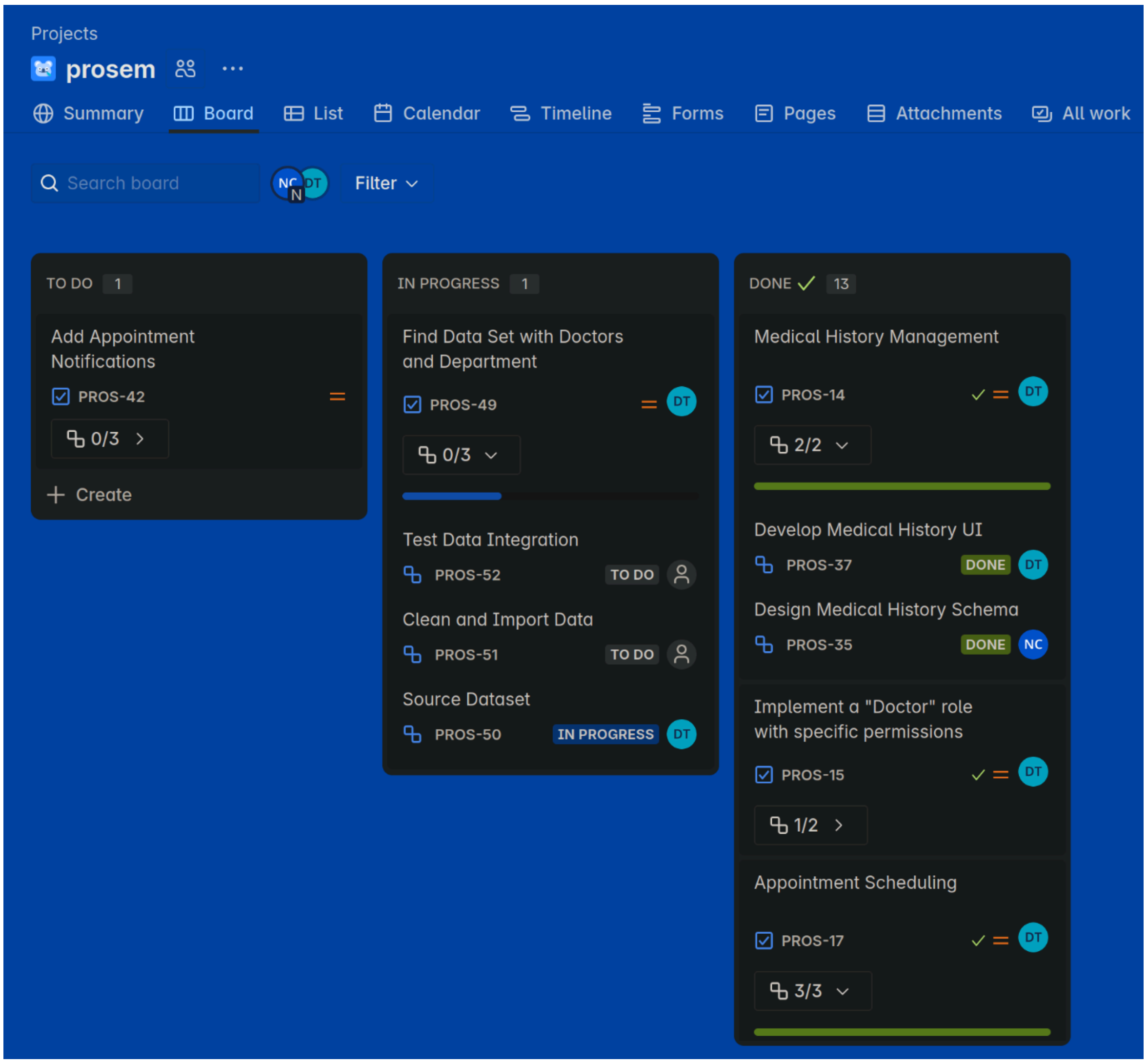
Non-functional requirements focus on system performance, security, and usability characteristics. The system must maintain reasonable response times for all standard operations while supporting concurrent access by multiple users. Data security requirements include encryption of sensitive medical information and secure authentication protocols.

Usability requirements emphasize the need for intuitive interface design that accommodates users with varying levels of technical expertise. The system must provide clear navigation paths, informative error messages, and consistent visual design elements. Mobile responsiveness is essential to support healthcare professionals who frequently access systems through portable devices.

4. Technologies

4.1 Technology Selection Process

The selection of technologies for the Medical Appointment Management System was based on careful evaluation of factors including development efficiency, system scalability, security requirements, and long-term maintainability. The chosen technology stack represents a balance between modern development practices and proven reliability.For issue tracking we used Jira (Figure 1), and for code versioning we used Github (Figure 2).



TO DO 1

Add Appointment Notifications
PROS-42 0/3

IN PROGRESS 1

Find Data Set with Doctors and Department
PROS-49 DT 0/3

Test Data Integration PROS-52 TO DO
Clean and Import Data PROS-51 TO DO
Source Dataset PROS-50 IN PROGRESS DT

DONE ✓ 13

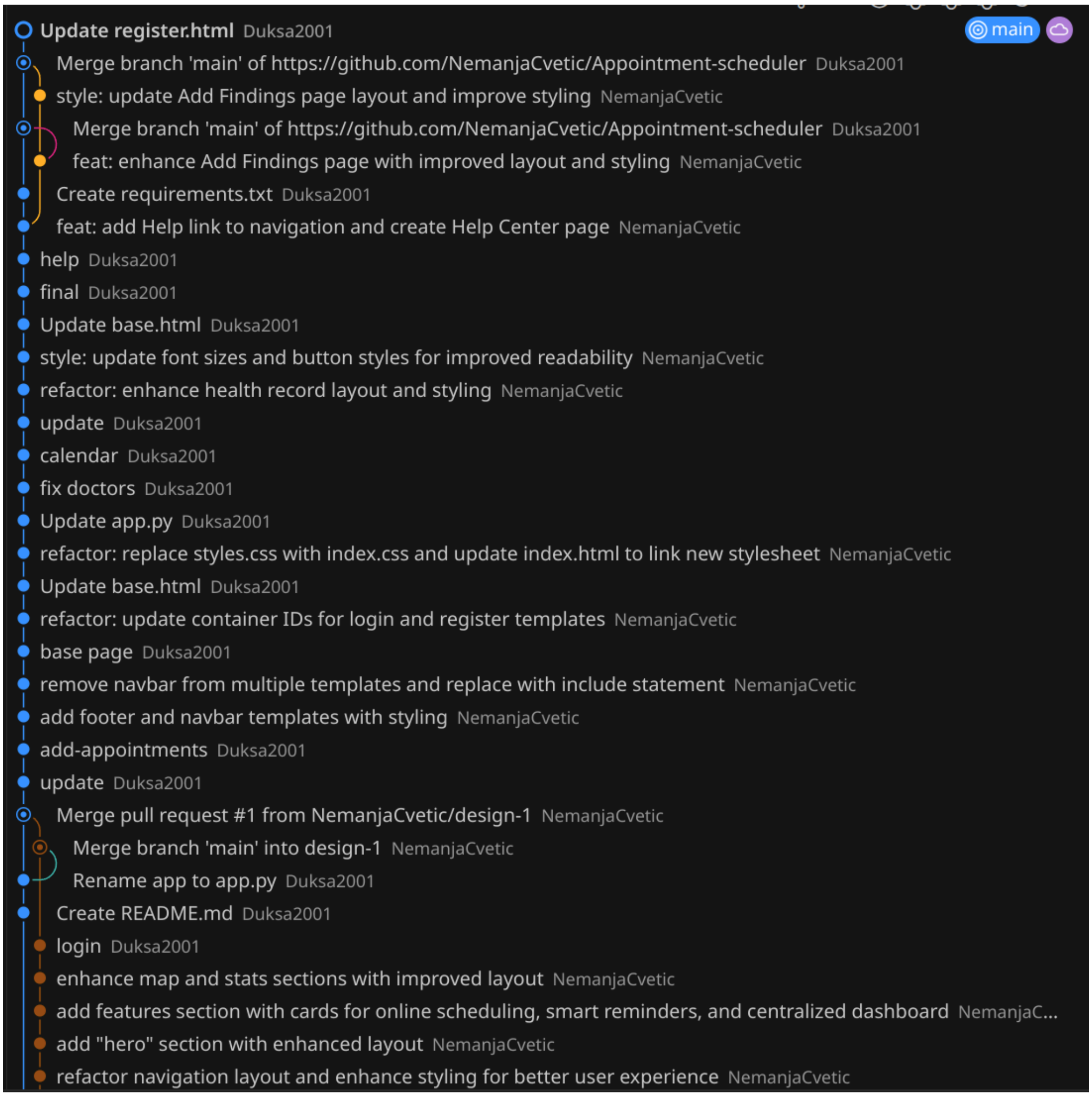
Medical History Management
PROS-14 DT 2/2

Develop Medical History UI PROS-37 DONE DT
Design Medical History Schema PROS-35 DONE NC

Implement a "Doctor" role with specific permissions
PROS-15 DT 1/2

Appointment Scheduling
PROS-17 DT 3/3

Figure 1: Jira Dashboard



Update register.html Duksa2001
Merge branch 'main' of https://github.com/NemanjaCvetic/Appointment-scheduler Duksa2001
style: update Add Findings page layout and improve styling NemanjaCvetic
Merge branch 'main' of https://github.com/NemanjaCvetic/Appointment-scheduler Duksa2001
feat: enhance Add Findings page with improved layout and styling NemanjaCvetic
Create requirements.txt Duksa2001
feat: add Help link to navigation and create Help Center page NemanjaCvetic
help Duksa2001
final Duksa2001
Update base.html Duksa2001
style: update font sizes and button styles for improved readability NemanjaCvetic
refactor: enhance health record layout and styling NemanjaCvetic
update Duksa2001
calendar Duksa2001
fix doctors Duksa2001
Update app.py Duksa2001
refactor: replace styles.css with index.css and update index.html to link new stylesheet NemanjaCvetic
Update base.html Duksa2001
refactor: update container IDs for login and register templates NemanjaCvetic
base page Duksa2001
remove navbar from multiple templates and replace with include statement NemanjaCvetic
add footer and navbar templates with styling NemanjaCvetic
add-appointments Duksa2001
update Duksa2001
Merge pull request #1 from NemanjaCvetic/design-1 NemanjaCvetic
Merge branch 'main' into design-1 NemanjaCvetic
Rename app to app.py Duksa2001
Create README.md Duksa2001
login Duksa2001
enhance map and stats sections with improved layout NemanjaCvetic
add features section with cards for online scheduling, smart reminders, and centralized dashboard NemanjaC...
add "hero" section with enhanced layout NemanjaCvetic
refactor navigation layout and enhance styling for better user experience NemanjaCvetic

Figure 2: GitHub branch visualisation

4.2 Python Programming Language

Python[1] serves as the primary programming language for the application backend, chosen for its readability, extensive library ecosystem, and strong support for web development. Python's syntax promotes clean, maintainable code that can be easily understood and modified by different developers throughout the project lifecycle.

The language's extensive standard library and third-party package ecosystem provide solutions for common web development tasks including form validation, data serialization, and security implementations. Python's strong typing support through type hints enhances code reliability and facilitates debugging during development.

Python's compatibility with various deployment platforms ensures that the application can be deployed in different environments based on organizational requirements. The language's performance characteristics are well-suited for web applications with moderate traffic loads typical of healthcare facility systems.

4.3 Flask Web Framework

Flask serves as the foundation of the web application, providing a lightweight framework for building web applications in Python. Flask's minimalist approach allows for precise control over application architecture while providing essential features such as routing, templating, and session management. The framework's modular design supports the implementation of complex applications through the use of extensions and blueprints.

The choice of Flask[2] over more heavyweight frameworks like Django was driven by the need for flexibility in application design and the desire to avoid unnecessary complexity in areas not relevant to the specific requirements of healthcare appointment management.

4.4 Supabase Database Platform

Supabase provides the backend infrastructure for data storage, authentication, and real-time functionality. As a Backend-as-a-Service platform, Supabase[4] eliminates the need for complex server infrastructure management while providing enterprise-grade database capabilities built on PostgreSQL.

The platform's real-time capabilities enable immediate updates to appointment availability across all connected clients, ensuring that users always see current information when scheduling appointments.

Supabase's automatic backup and disaster recovery features ensure that critical healthcare data remains protected against data loss scenarios. The platform's scalability allows the system to grow with increasing user demands without requiring significant infrastructure changes.

The choice of Supabase over traditional database solutions was influenced by the need for rapid development cycles and the desire to leverage modern cloud infrastructure benefits without the complexity of managing database servers. The platform's API support enables seamless integration with the Flask application while maintaining data security standards.

4.5 Bootstrap Frontend Framework

Bootstrap[3] provides the foundation for responsive user interface design, ensuring that the application functions effectively across different screen sizes and devices. The bootstrap's grid system and pre-built components accelerate development while maintaining consistent visual design standards.

The bootstrap's extensive component library includes form controls, navigation elements, and modal dialogs that are essential for modern intuitive application interfaces. The bootstrap's widespread adoption ensures that maintenance and future enhancements can be performed by developers familiar with standard web development practices.

5. Database Design

The database design (represented in figure 3) for the Medical Appointment Management System follows relational database principles to ensure data integrity, consistency, and efficient query performance. The schema is designed to support the relationships between patients, doctors, hospitals, and appointments while maintaining flexibility for future enhancements.

1. **Appointments table** represents the scheduled appointments in the system, linking patients with doctors and hospitals for specific dates and times. It has a one-to-many relations with hospitals, doctors, and patients tables as hospitals can have multiple appointments scheduled in them as well as patients and doctors having multiple scheduled appointments.

2. **Doctors table** represents all doctors working at hospitals. It has a many-to-one relation with the appointments table as one doctor can have multiple appointments, and a one-to-many relation with the hospitals table as a doctor can work in one hospital at a time, but a hospital can have multiple doctors.

3. **Hospitals table** represents all available hospitals in the system. It has a one-to-many relation with appointments table as one hospital can have multiple appointments, and a one-to-many relation with the doctors table as one hospital can have multiple doctors working in it.

4. **Patients table** represents all registered patients in the system. It has a one-to-many relation with an appointments table, as one patient can have multiple appointments.

Database constraints ensure data integrity through primary key definitions, foreign key relationships, and field validation rules.

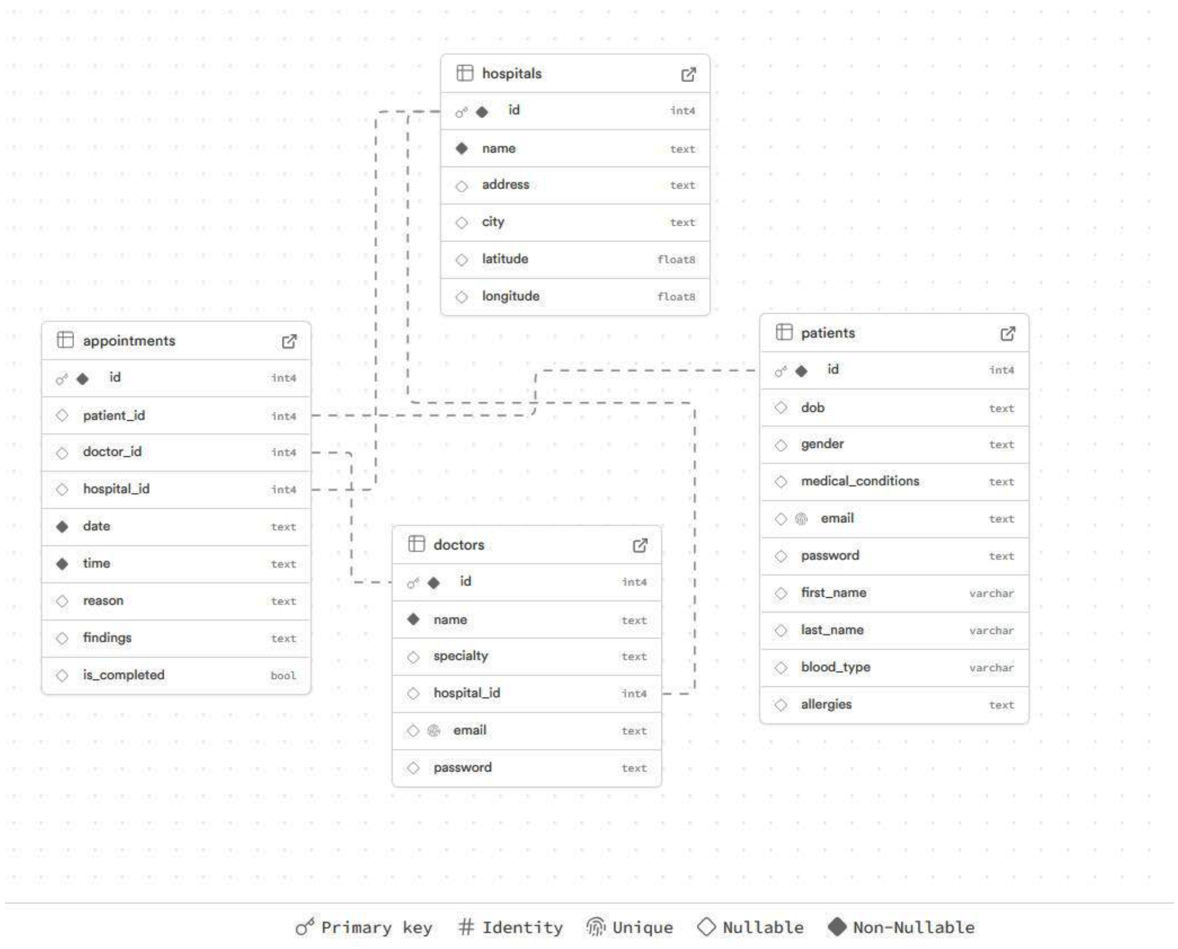


Figure 3: Database schema

6. Implementation

The implementation of the Medical Appointment Management System follows a structured approach that prioritizes functionality, security, and maintainability. The development process began with the establishment of the core application framework and gradually expanded to include advanced features and user interface enhancements.

Route handlers manage HTTP requests and coordinate between data access layers and template rendering. The application utilizes Flask's built-in session management for user state tracking and implements custom user authentication logic that integrates with the Supabase authentication system.

User authentication is implemented through a custom User class that extends Flask-Login's UserMixin, providing the necessary methods for user session management. The authentication system supports multiple user roles including patients, doctors, and administrators, with each role having appropriate access permissions and interface customizations.

The appointment scheduling system represents the most complex implementation component, featuring real-time availability checking and conflict prevention mechanisms. The system generates time slot availability by comparing requested appointment times against existing bookings, ensuring that double-booking scenarios are prevented. Calendar integration provides visual representations of appointment availability, with color-coded indicators showing fully booked, partially available, and completely free days.

Data access is implemented through direct integration with the Supabase client library, providing secure and efficient database operations. The implementation includes comprehensive error handling to manage database connection issues and data validation failures. Transaction management ensures that complex operations such as appointment creation maintain data consistency even in the event of system failures.

The medical records functionality enables healthcare providers to add findings and treatment notes to patient appointments, creating comprehensive medical histories that support continuity of care. The system maintains audit trails for all medical record modifications, ensuring compliance with healthcare record-keeping requirements.

The application implements responsive design principles through Bootstrap integration, ensuring that the interface adapts effectively to different screen sizes and devices. This capability is essential for healthcare environments where users may access the system through various devices including desktop computers, tablets, and mobile phones.

7. Work assignment

Here is how we split the work:

- Dušan: Requirement analysis, problem analysis, backend development, backend maintenance and bug fixing, documentation writing, issue reporting

- Nemanja: Requirement analysis, problem analysis, frontend development, frontend maintenance and design, documentation writing, issue reporting

8. Conclusion

For this project we made a web app which allows patients and doctors to easily schedule and manage medical appointments. The application was developed using python programming language with flask framework for backend and bootstrap framework for the front end design.
The primary goal of this project was to build an easy to use application that would allow patients and doctors alike an easy management, scheduling and tracking of appointments. By using python programming language we ensured the relatively easy maintenance and future upgrades of the application.

Speaking of the possible future upgrades there are definitely some things that can be improved in future versions of the application. For starters the security of the authentication process can be massively improved by implementing the hash functions for stored passwords. Another thing that can be implemented is an email notification system that will remind the users about upcoming appointments.

9. References

[1] Python language documentation url: https://www.python.org/doc/ [Accesed 13-03-2025]

[2] Flask framework documentation url: https://flask.palletsprojects.com/en/stable/ [Accesed 18-03-2025]

[3] Bootstrap framework documentation url: https://getbootstrap.com/ [Accesed 18-03-2025]

[4] Supabase database url: https://supabase.com/docs [Accesed 14-03-2025]

10. Appendices

GitHub repository link: https://github.com/NemanjaCvetic/Appointment-scheduler