جامعة عفت

EFFAT UNIVERSITY

Software Engineering Medicomemroy System.

CS 3151 - 1 - Software Engineering

Project Report - Fall 2023

Students:

Leen Sharab - S21107195

Sarah Alshumayri - S20106125

Reema Abdallah - S20106463

Ameera Attiah - S21107316

Lujain Almarri - S20106753

Instructor: **Dr. Akila Sarirete**

Contents

1 Introduction

In the realm of modern healthcare, the integration of technology and medical practice has become paramount. "MedicoMemory," a software system, stands at the forefront of this integration, offering a seamless, efficient, and secure platform for managing patient medical records and healthcare data. This system is designed to revolutionize the way medical information is stored, accessed, and utilized.

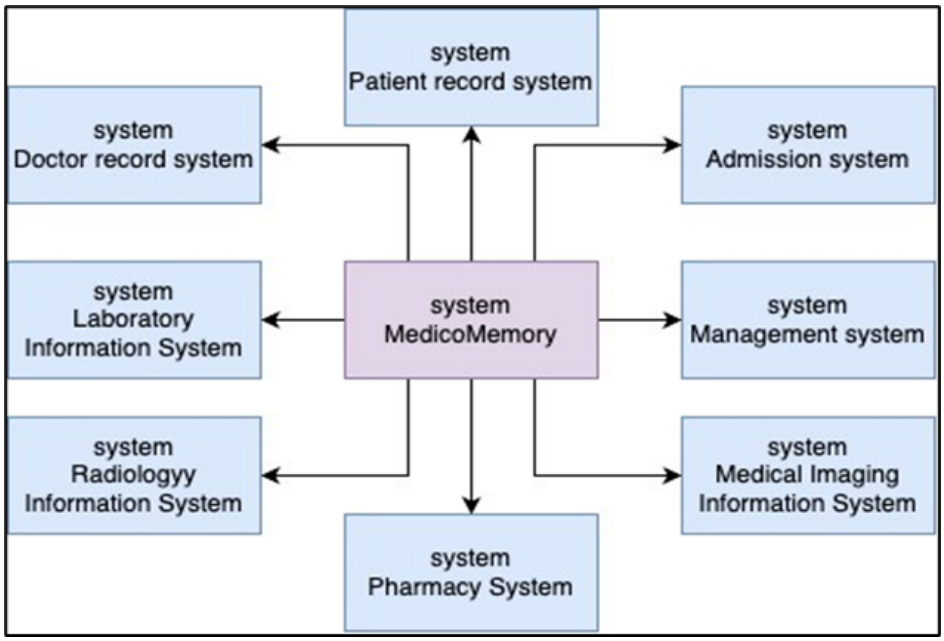
MedicoMemory is a comprehensive solution designed to cater to the diverse needs of patients and healthcare providers. It addresses several key areas in the healthcare industry, including patient record management, admission systems, medical imaging, pharmacy services, and laboratory information systems, among others. The software's capabilities extend to personal information management, medication tracking, and medical report storage, providing a holistic approach to patient care.

One of the software's pivotal features is its user-centric design, enabling patients to access and manage their medical records, verify information provided by doctors, and maintain their medication history. For healthcare professionals, MedicoMemory offers a robust platform for accessing and updating patient records, prescribing medications, and uploading medical reports, thereby facilitating better patient care and operational efficiency.

Furthermore, MedicoMemory adheres to security and privacy standards, ensuring the protection of sensitive patient data. The system is designed with regulatory compliance in mind, aligning with healthcare regulations and data protection laws. Its integration with various healthcare systems and stakeholders enhances communication and data sharing, contributing to an improved healthcare environment.

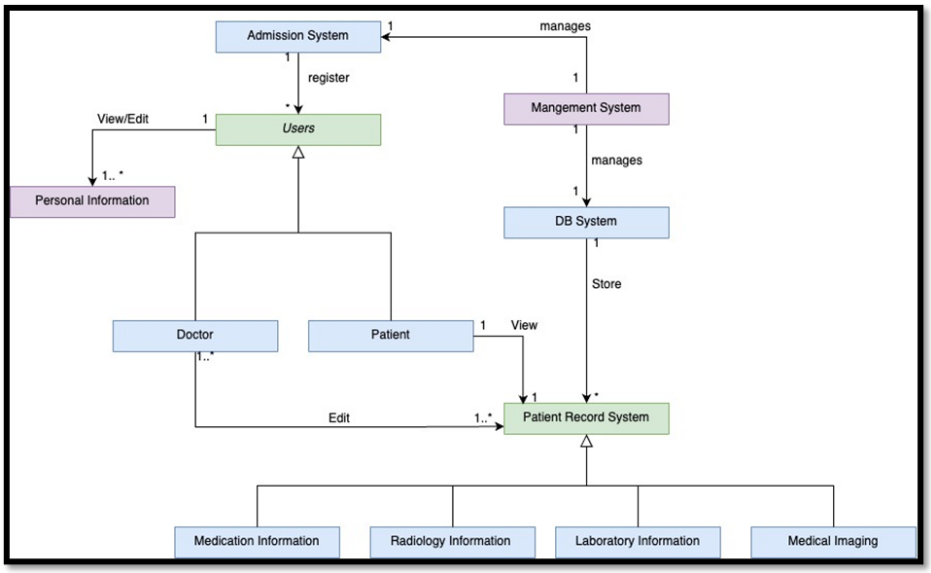
2 Models

2.1 Context Diagram



Description: The diagram shows the integration of various medical information systems centered around "MedicoMemory." This core system interfaces with several others, including the Doctor record system, Patient record system, Laboratory Information System, Radiology Information System, Admission system, Management system, Medical Imaging Information System, and Pharmacy System. These connections indicate a centralized approach where "MedicoMemory" acts as a hub for data exchange and coordination among the various healthcare-related systems.

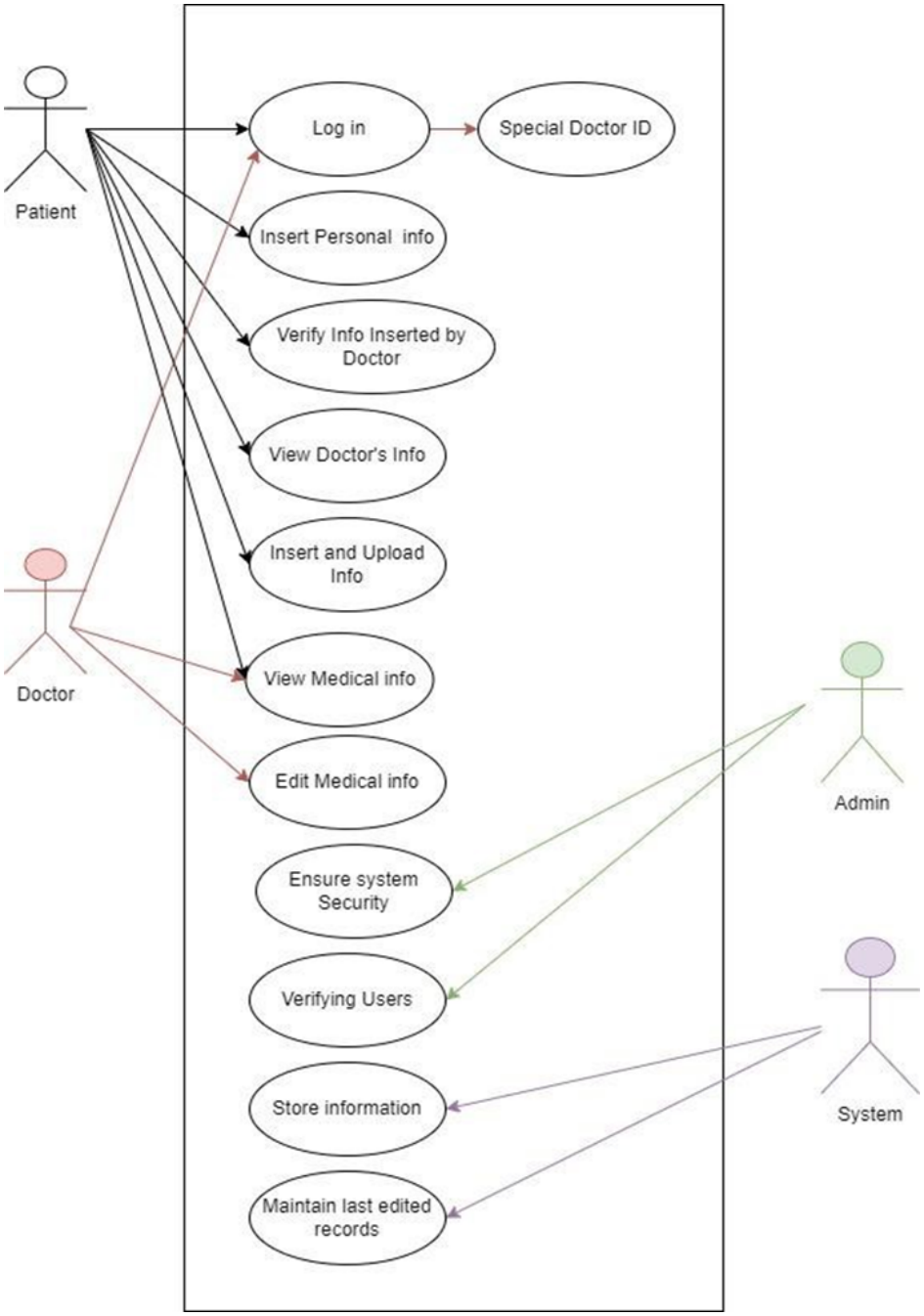
2.2 Class Diagram



Description: The class diagram portrays the structure and relationships within our Medicomemroy system. Central to the diagram is the "Users" class, which can register through the "Admission System." These users are further categorized as "Doctor" or "Patient." Doctors and Patients can view and edit personal information. A "Patient Record System" is in place for viewing patient records, which includes information like medication, radiology, laboratory details, and medical imaging. Furthermore, the "Management System" oversees the "DB System," responsible for data storage. This comprehensive layout signifies the intricacies of managing patient data, user access, and system control within a medical facility.

2.3 Use Case Diagrams

Actors	Functionalities
Patient	1. Log in to the system.
	2. Insert personal information (name, DOB, address, etc.).
	3. View own medical information.
	4. Verify the information inserted by the doctor.
	5. View doctor's information (name, specialty, phone number, email).
Doctor	1. Log in to the patient's account with a special ID.
	2. Edit the medical information of the patient.
	3. Insert required medications for the patient.
	4. Upload documents of medical reports and imaging information.
Admin	1. Provide authentication and verification of patient accounts.
	2. Ensure system security.
System	1. Store all patient record information, including patient information, medications, laboratory, radiology/non-radiology info, etc.
	2. Maintain a record of the last edited name (doctor's name) when changes are made to the records.



Description: The use case diagram shows the interactions among Patients, Doctors, Admin, and a System within a Medicomemroy platform. Patients have functionalities such as logging in (with a special doctor ID option), inputting personal information, verifying the information provided by the doctor, and viewing the doctor's details. Alternatively, Doctors can insert, upload, view, and edit medical information. The Admin role emphasizes system security, verifying users, and overseeing information storage. Lastly, the System is responsible for securely storing data and maintaining records of the most recent edits. Each role's actions intersect, showcasing the interdependence and collaborative nature of the platform.

2.4 Activity Diagram

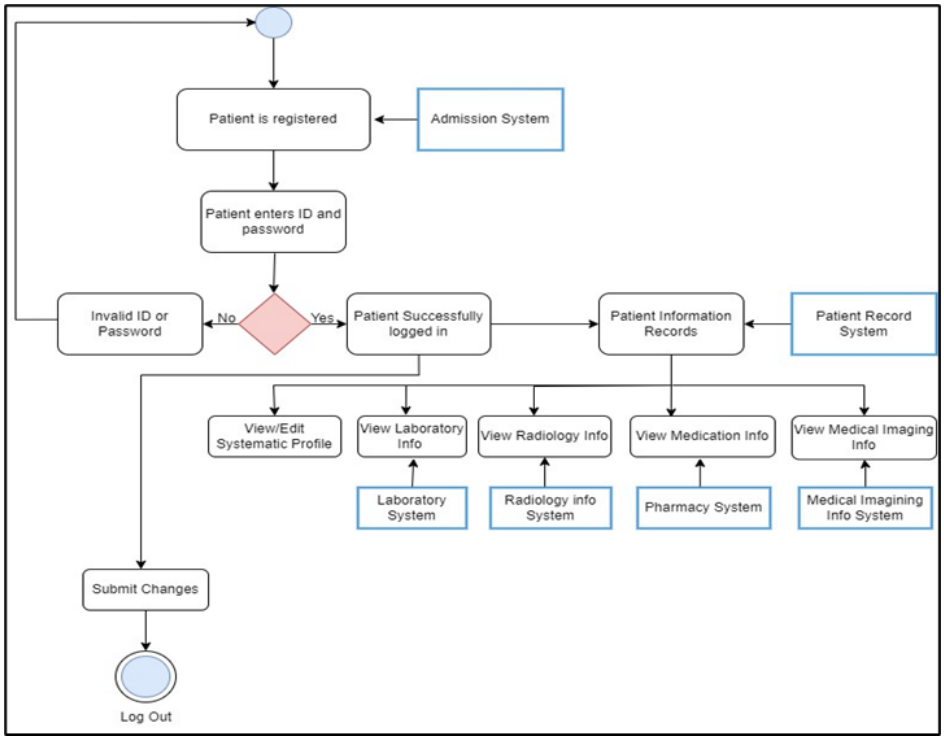


Figure 1: patient's interaction

Description: The diagram outlines a patient's interaction with our Medicomemroy information system. Initially, the patient's registration is verified by the Admission System. Following this, the patient logs in by entering their ID and password. If the credentials are correct, the patient gains access to the Patient Information Records within the Patient Record System. From here, the patient can view or edit their systematic profile, access laboratory data from the Laboratory System, review radiology information from the Radiology info System, check medication details from the Pharmacy System, and view medical imaging data from the Medical Imaging Info System.

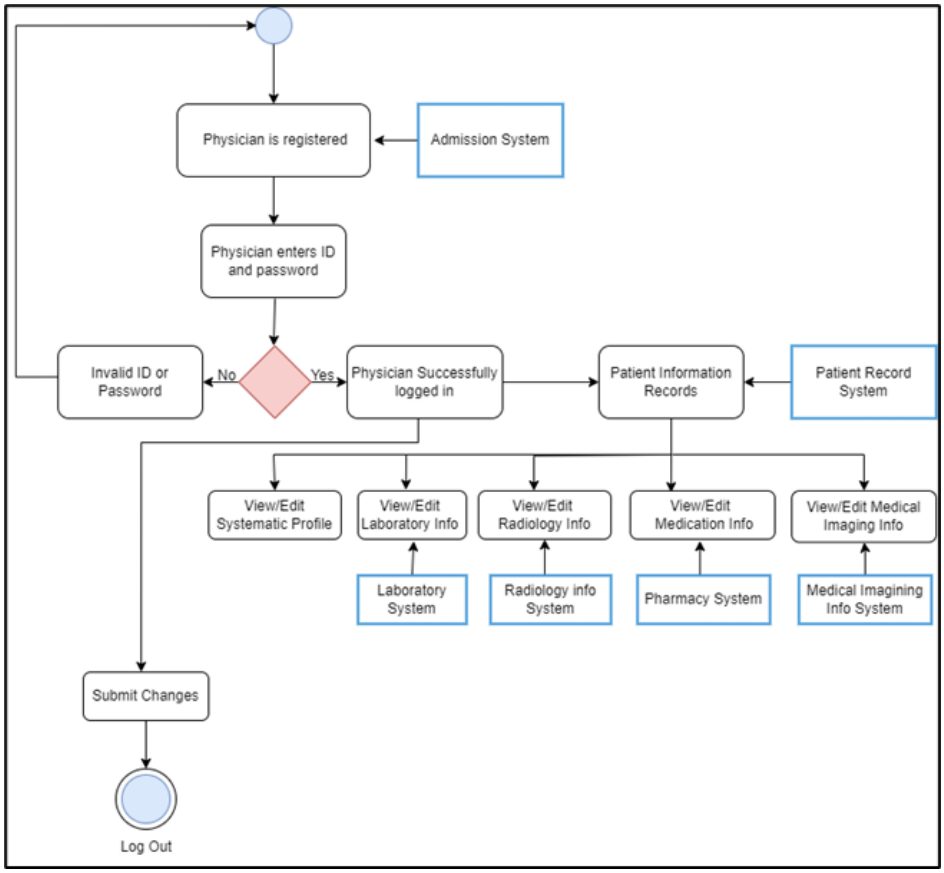


Figure 2: Physician's interaction

Description: The diagram showcases a physician's interaction flow with a Medicomemroy information system. Upon verification of the physician's registration by the Admission System, they can log in using their ID and password. Successful authentication provides the physician access to the Patient Information Records within the Patient Record System. From this point, the physician has the capability to view and edit various data sections: the systematic profile, laboratory details via the Laboratory System, radiology information from the Radiology info System, medication particulars from the Pharmacy System, and medical imaging data from the Medical Imaging Info System. Any modifications made can be finalized by submitting changes, and the physician can subsequently log out when done.

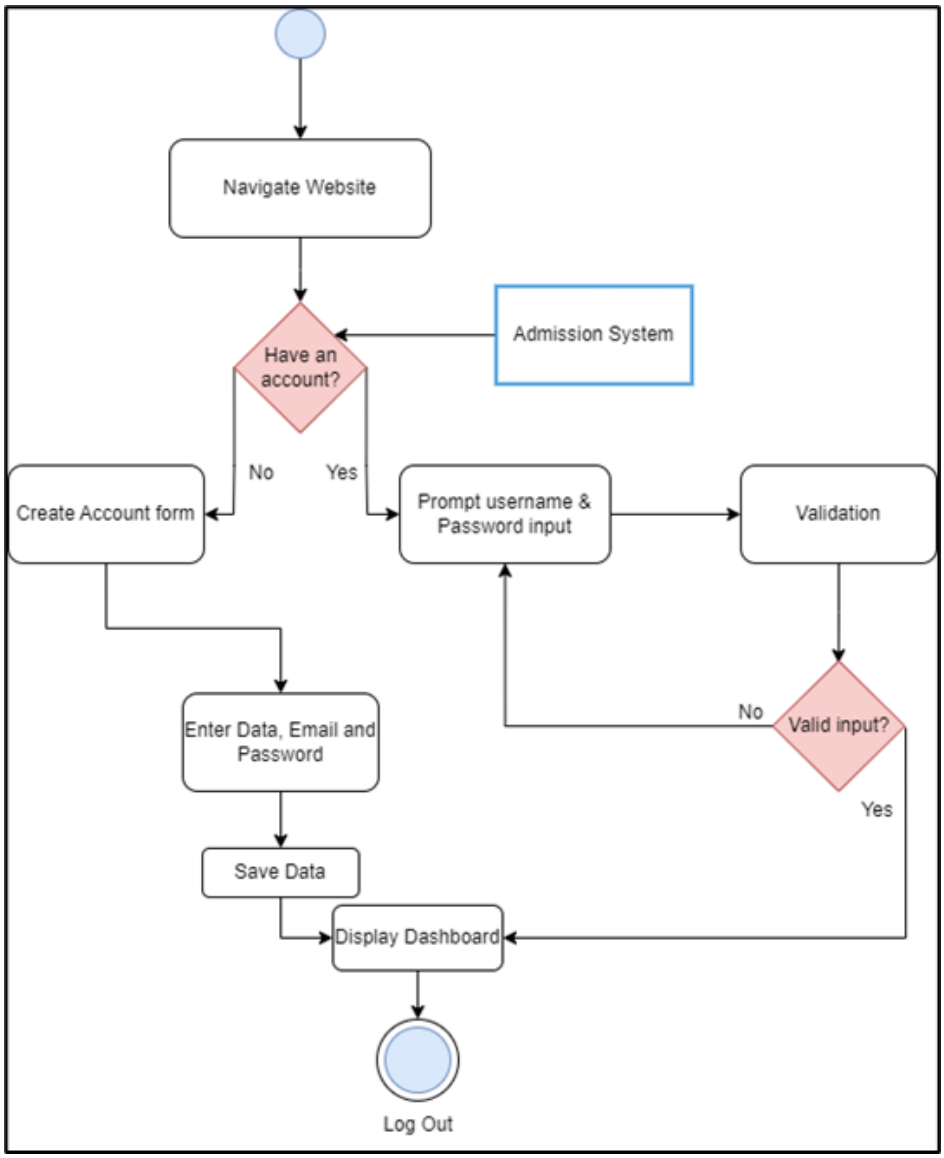


Figure 3: User flow for website navigation

Description: The diagram illustrates the user flow for website navigation and authentication through the Admission System. When a user navigates to the website, they're presented with a query to determine if they already have an account. Users without an account are directed to a "Create Account" form, where they input data, including email and password, which gets saved. Alternatively, existing account holders are prompted for their username and password. These credentials undergo a validation process. If the credentials are validated successfully, users are directed to the dashboard; otherwise, they're prompted about the invalid input and asked to re-enter their details. From the dashboard, users have the option to log out.

2.5 Sequence Diagram

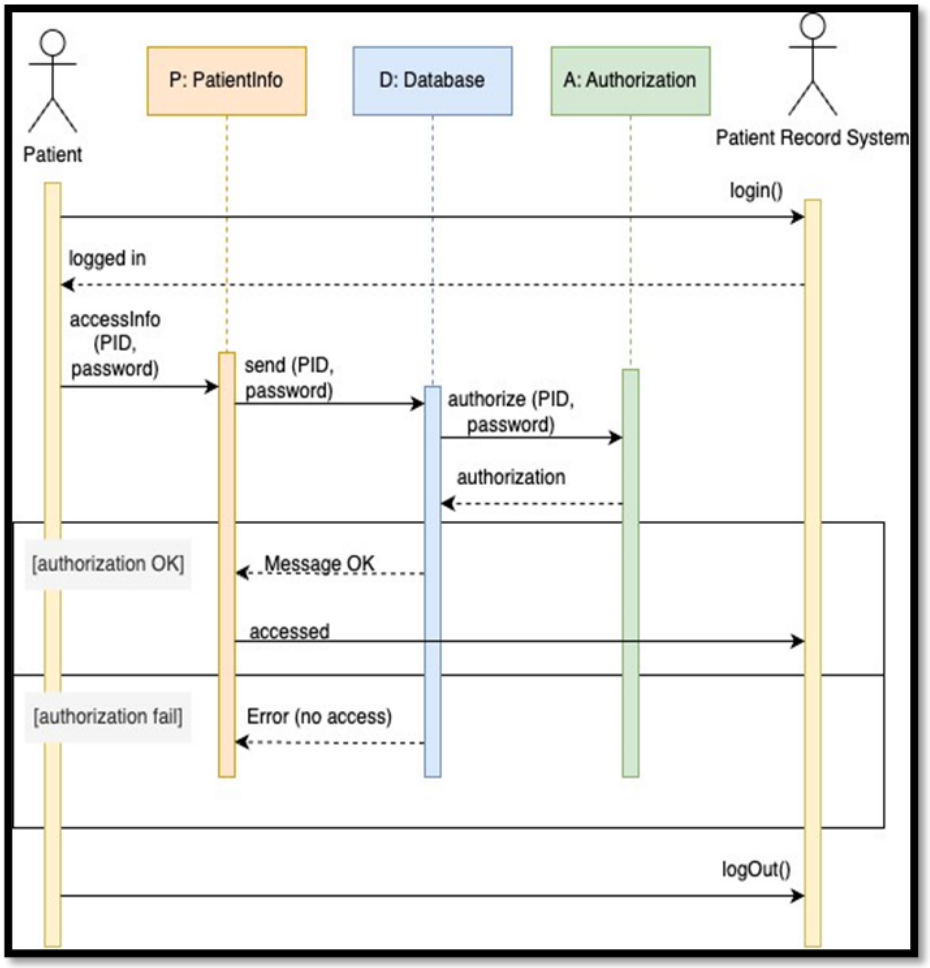


Figure 4: Login authorization process for a patient

Description: The sequence diagram illustrates the login authorization process for a patient trying to access their records in the Patient Record System. The patient, represented on the left, initiates the process by sending their personal identification (PID) and password to the "PatientInfo" system. This information is then transmitted to the "Database," which communicates with the "Authorization" module to validate the credentials. If authorization is successful, the patient receives a "Message OK" indicating successful access to their records, and they are logged into the Patient Record System. Conversely, if the authorization fails, the patient receives an "Error (no access)" message. The sequence concludes with the patient logging out of the system.

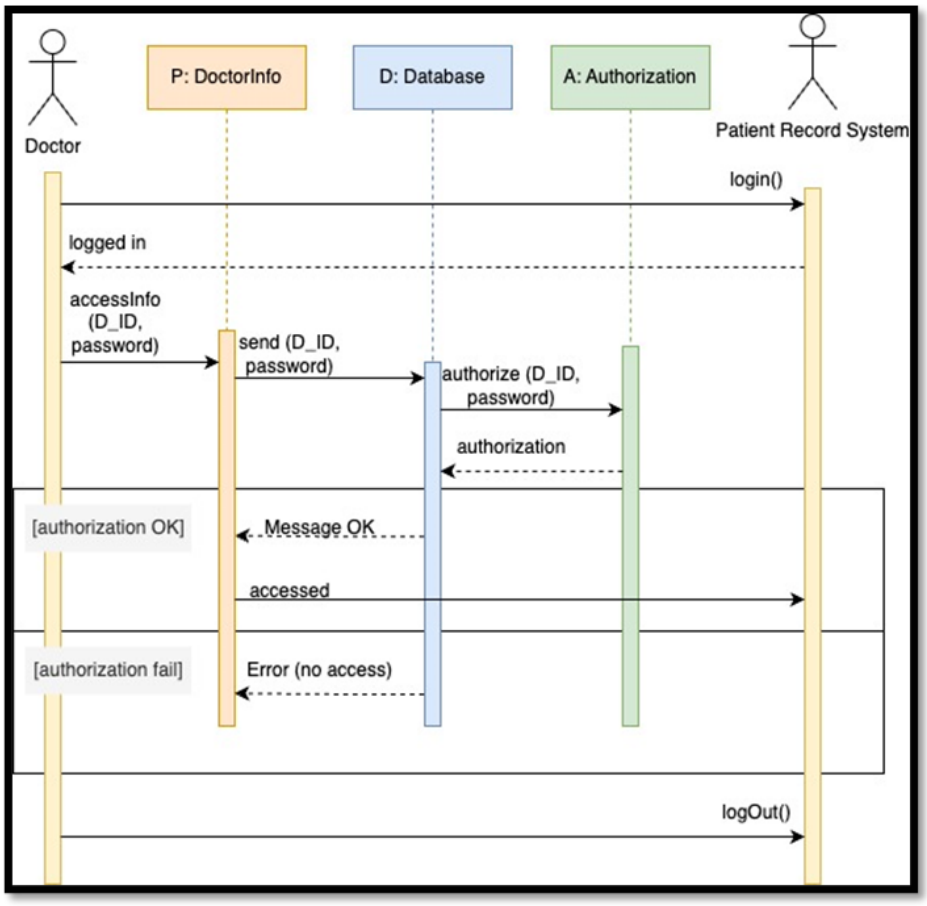


Figure 5: The doctor undergoes to access the Patient Record System

Description: The sequence diagram showcases the process a doctor undergoes to access the Patient Record System. The doctor, already logged into the system, attempts to retrieve doctor-specific information by submitting their unique identification (D_ID) and password to the "DoctorInfo" module. This module relays the credentials to the "Database" for validation. In tandem, the "Database" communicates with the "Authorization" module to confirm the doctor's identity. If the authorization is successful, a "Message OK" signal indicates the doctor has been granted access; however, if there's a failure in authorization, an "Error (no access)" message is relayed. The process concludes with the doctor logging out of the system.

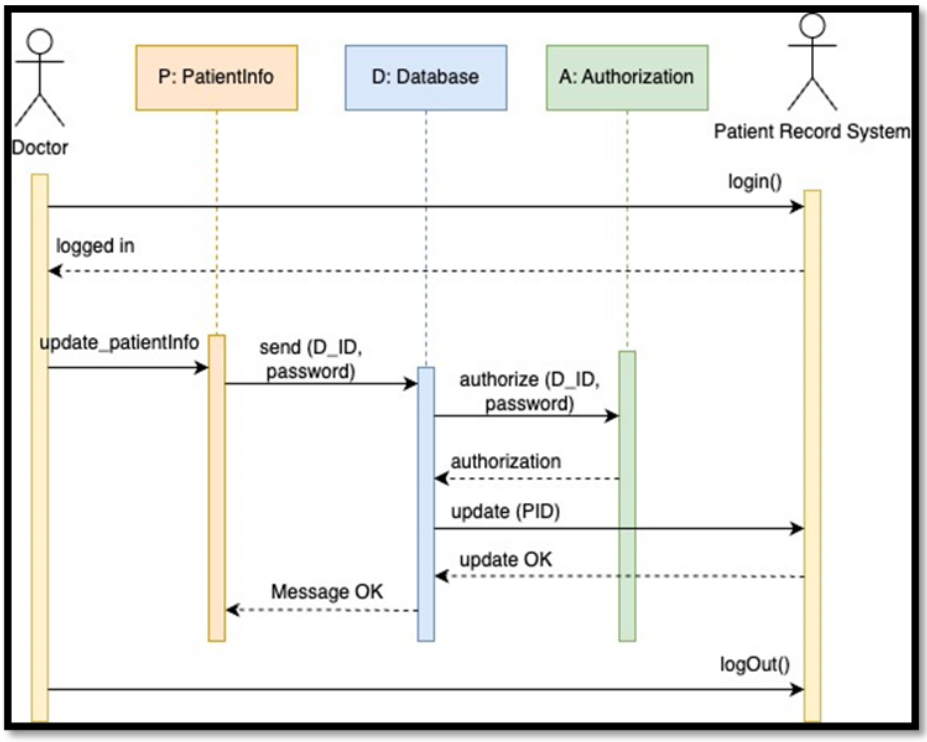


Figure 6: doctor's interaction

Description: The sequence diagram depicts a doctor's interaction with the Patient Record System to update patient information. Initially, the doctor is logged into the system. When the doctor decides to update a patient's information, they interact with the "PatientInfo" module, which then forwards the doctor's identification (D_ID) and password to the "Database" for verification. The "Database" collaborates with the "Authorization" module to authenticate the doctor's credentials. Upon successful authorization, the doctor is permitted to update the patient's record, identified by their personal identification (PID). The update is confirmed with an "update OK" message, and the sequence culminates with the doctor logging out of the system.

2.6 State Diagram

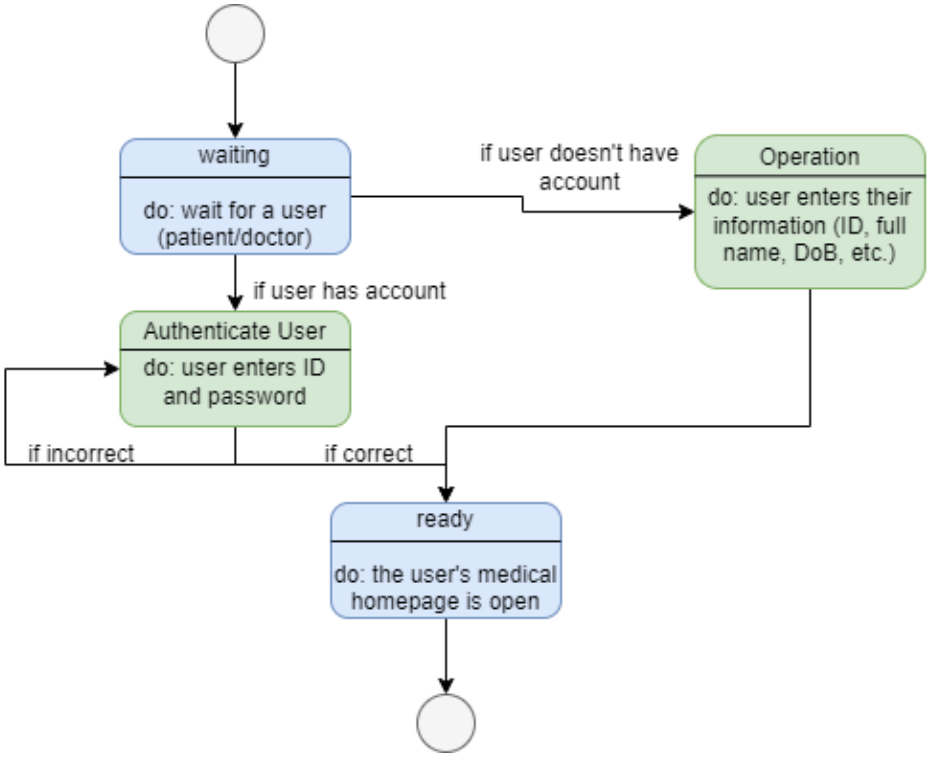


Figure 7: authentication process

Description: The State diagram illustrates the authentication process for a user, whether they are a patient or a doctor, accessing a medical system. Initially, the system is in a waiting state, anticipating a user's interaction. When a user approaches, the system checks whether the user already possesses an account. If they do, they're prompted to input their ID and password. In cases where the provided credentials are accurate, the user is directed to their medical homepage. Conversely, if the credentials are incorrect, the process loops back to the user entering their details. If the user doesn't have an existing account, they are provided with full permissions to register, necessitating the input of personal details like ID, full name, and date of birth to create a new account.

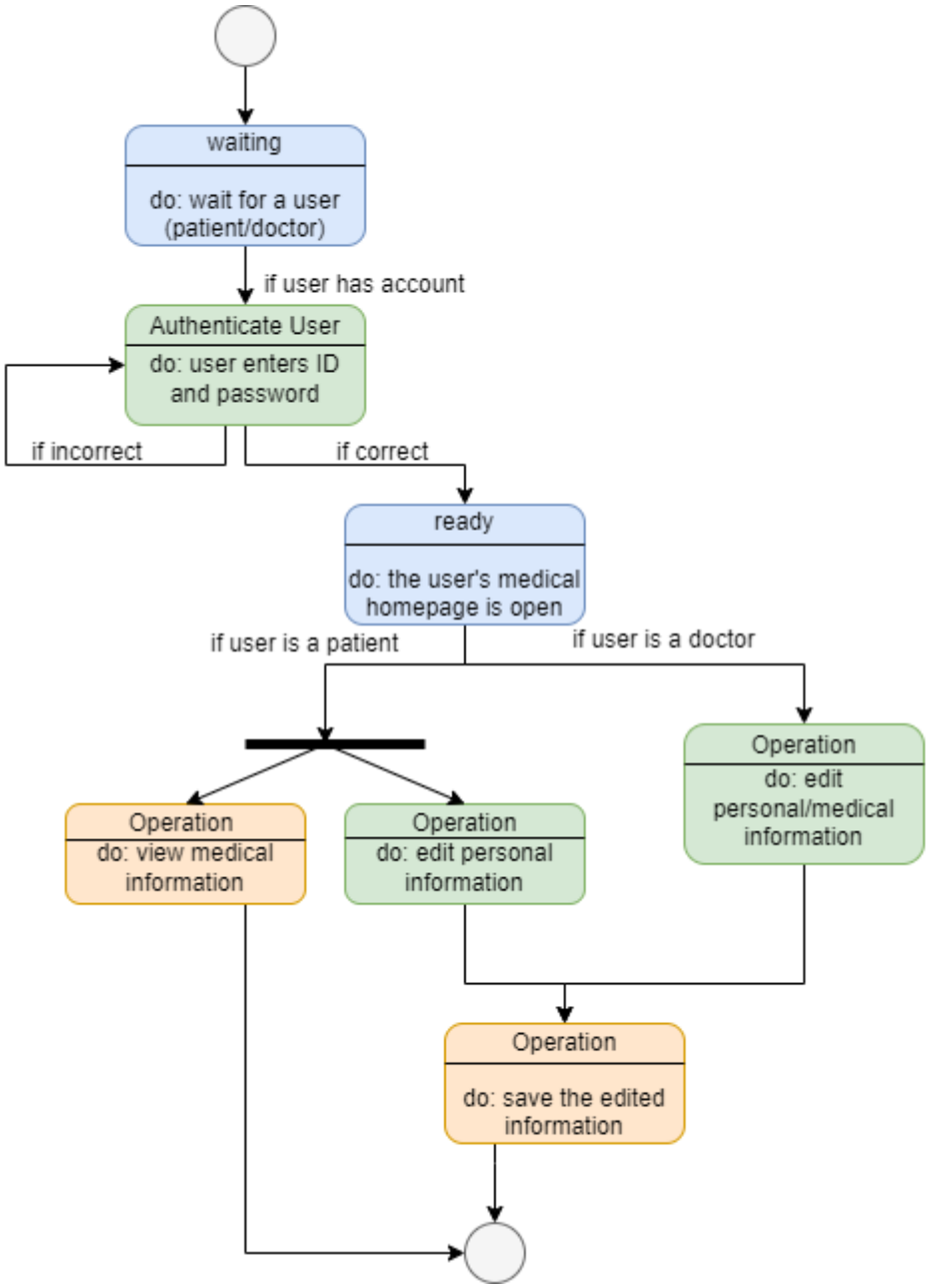


Figure 8: user input

Description: The system remains in a waiting state for user input. Once a user, either a patient or doctor initiates interaction and possesses an account, they are prompted to input their ID and password. Upon successful authentication, they access their medical homepage. For users identified as patients, there are differentiated access levels: they can modify personal details but only view medical data. After making potential updates, patients can save the altered information.

3 Design Architecture

3.1 Architectural Overview:

MedicoMemory is a comprehensive web-based platform designed to facilitate the management of medical records. The system is structured on a multi-tier architecture that segregates the user interface, data processing, and data storage into separate layers. This separation enhances security and allows for scalable updates and maintenance.

3.2 System Components and Interaction:

- **User Interface (UI):** The UI is responsive and accessible via web browsers, providing interfaces for various user roles including patients, doctors, and administrators.
- **Business Logic Layer:** Core functionalities such as user authentication, record management, and data validation are processed in this layer.
- **Data Access Layer:** This layer handles communication with the database, ensuring data integrity and transaction management.
- **Database Layer:** A relational database management system (RDBMS) stores all the data with encryption for sensitive information.

3.3 Technology Stack:

- **Frontend:** HTML, CSS, and JavaScript with a framework called wix.com .
- **Backend:** PHP language was used to manage the frontend operations.
- **Database:** A secure DBMS like MySQL.
- **Hosting/Cloud Services:** Services like AWS for hosting, storage, and additional cloud functionalities like serverless computing.

3.4 System Charactaristics

3.4.1 Performance

The design of MedicoMemory focuses on speed and efficiency. By keeping important tasks within close reach of each other, the system minimizes the need for extensive data exchange. Also, the system uses powerful, well-integrated parts to handle complex tasks quickly.

3.4.2 Security

A layered security approach is adopted, where core assets are safeguarded within inner layers, and strict access control is enforced through authentication and authorization mechanisms.

3.4.3 Safety and Reliability

The design of MedicoMemory ensures that all features crucial for safety are contained in just a few separate areas of the system. This makes it easier to manage and protect them. Additionally, the system has backup components in place, which means it can continue to operate smoothly even if something goes wrong, leading to fewer disruptions and a more dependable service.

3.4.4 Maintainability

The system architecture emphasizes modularity and replaceability, allowing for easy updates, enhancements, and repairs without significant system downtime.

3.5 Architectural Views:

To ensure a comprehensive understanding among all stakeholders, the architecture of MedicoMemory is documented from multiple specific viewpoints, each addressing particular concerns:

- **Conceptual View:** This view outlines the high-level structure and organization of MedicoMemory, providing stakeholders with a map of the system's abstract components and their relationships.
- **Logical View:** The Logical View delves into MedicoMemory's system requirements and showcases how various components and layers work together to fulfill these requirements, detailing service interfaces and data models.
- **Process View:** In the Process View, we describe the dynamic behavior of MedicoMemory's architecture, illustrating how different processes within the system interact, synchronize, and communicate, as well as how data is processed and flows through the system.
- **Development View:** The Development View focuses on MedicoMemory's programming and deployment infrastructure, detailing the setup of the development environment, the structure of the source code, and the build and deployment processes.

4 Prototype

4.1 Introduction

Our website, MedicoMemory, is a simple and secure place where anyone can keep track of their medical history online. It is especially helpful for people who want to have all their health information in one spot, easy to access and manage.

4.2 Platform and Creation

We built our website with Wix, a user-friendly tool that lets you make websites without needing to know how to code. It's like using building blocks to create a webpage, which means we were able to put together a professional-looking site pretty quickly.

4.3 Features and Functionality

- **Keep Your Medical Records:** You can type in your health details and keep them updated.
- **Family Health History:** There is a special area to note down health information about your family, which can be important to know about.
- **Upload Reports:** If you have got X-rays or blood tests, you can upload them and keep them organized.
- **Track Shots:** You can keep a record of vaccinations for you and your family.
- **Easy to Use:** The website is set up to be super easy to navigate, so you won't get lost trying to find what you need.

4.4 Design and User Experience

We made sure the website looks clean and is easy to walk through. Big buttons, clear labels, and a smart layout mean you won't have to hunt for the information you need.

4.5 Access and Usage

Anyone can visit MedicoMemory at https://samalshumayri.wixstudio.io/my-site-3. It works well on a computer, tablet, or phone, so you can check your health info at home or on the go.

5 Test Cases/User Stories

5.1 Login Feature

5.1.1 User Story:

As a registered user, I want to log into my account with my username and password so that I can access my personal dashboard and manage my information securely.

5.1.2 Test Case 1: Valid Login

- **Objective:** To verify that the system allows access with valid user credentials.
- **Steps:** Navigate to the login page, enter valid Patient ID and Password, and click 'Log in'.
- **Expected Result:** User is successfully logged in and redirected to the homepage/dashboard.

5.1.3 Test Case 2: Invalid Login

- **Objective:** To ensure the system denies access when incorrect credentials are used.
- **Steps:** Navigate to the login page, enter invalid Patient ID and Password, and click 'Log in'.
- **Expected Result:** User is not logged in and an error message is displayed.

5.1.4 Test Case 3: Password Visibility Toggle

- **Objective:** To check the functionality of the password visibility toggle.
- **Steps:** Enter a password in the password field and click on the visibility toggle icon.
- **Expected Result:** The password is displayed in plain text when the toggle is active.

5.1.5 Test Case 4: Forgot Password

- **Objective:** To verify the forgot password feature is working.
- **Steps:** Click on 'Forgot Password?' and follow the prompted steps.
- **Expected Result:** User should be able to initiate a password reset process.

5.2 Sign Up Feature

5.2.1 User Story:

As a new user, I want to sign up for an account by providing necessary personal details so that I can register and start using the services offered by the platform.

5.2.2 Test Case 1: Valid Sign Up

- **Objective:** To verify that the system allows the creation of a new account with valid details.
- **Steps:** Navigate to the sign-up page, enter all required details including a unique Patient ID, and click 'Sign up'.
- **Expected Result:** A new account is created and the user is directed to a confirmation page or logged in directly.

5.2.3 Test Case 2: Sign Up with Existing User Details

- **Objective:** To confirm that the system does not allow duplicate accounts.
- **Steps:** Attempt to sign up using an already registered Patient ID.
- **Expected Result:** The system should not allow the sign-up and should display an error message regarding the duplicate ID.

5.2.4 Test Case 3: Mandatory Fields

- **Objective:** To ensure all mandatory fields must be filled for successful sign-up.
- **Steps:** Leave one or more mandatory fields empty and attempt to sign up.
- **Expected Result:** The system should prevent sign-up and indicate which mandatory fields are missing.

5.2.5 Test Case 4: Password Strength Validation

- **Objective:** To validate that the system enforces a certain password strength.
- **Steps:** Enter various passwords that do not meet the presumed complexity requirements.
- **Expected Result:**The system should reject weak passwords and prompt the user to create a stronger one.

5.3 Personal Information Form

5.3.1 User Story:

As a user, I need to be able to enter and update my personal information such as date of birth, contact details, and medical history to ensure my profile is up-to-date for health management purposes.

5.3.2 Test Case 1: Mandatory and Format Validation

- **Objective:** To check mandatory fields and validate the format for inputs like email and phone number.
- **Steps:** Fill in all fields but leave one mandatory field empty. Enter invalid formats for Email and Cell phone. Attempt to submit the form.
- **Expected Result:**The form submission is blocked, prompting the user to fill the mandatory field and correct the formats for Email and Cell phone.

5.3.3 Test Case 2: Date and Age Consistency Check

- **Objective:** To verify that Birthday, Age and their inter-field consistency are validated.
- **Steps:** Enter a Birthday that doesn't match the Age field, including an invalid date such as February 29 on a non-leap year. Attempt to submit.
- **Expected Result:**The form should prevent submission and indicate both the invalid date and the inconsistency between Birthday and Age.

5.3.4 Test Case 3: Input Validation and Length Restrictions

- **Objective:** To ensure all input fields accept only appropriate types of data and adhere to length restrictions.
- **Steps:** Enter numerical values in text fields, text in numerical fields, excessively long strings, and include special characters in Name and Occupation fields. Attempt to submit the form.
- **Expected Result:**The form should reject inappropriate data types, truncate or reject excessively long inputs, and handle special characters according to field specifications.

5.3.5 Test Case 4: Comprehensive Positive Flow

- **Objective:** To ensure the form submits successfully when all fields are filled correctly, including valid selections for dropdowns.
- **Steps:**Fill in every field correctly, ensuring that all data types and formats are valid, and make proper selections for Gender and Marital status. Submit the form.
- **Expected Result:** The form submits without any error, and a success message or confirmation page is displayed.

5.4 Radiology Page

5.4.1 User Story:

As a patient, I want to view and manage my radiology reports and images so that I can keep track of my diagnostic history and share information with my healthcare providers as needed.

5.4.2 Test Case 1: Complete Workflow Validation

- **Objective:** To verify that all form elements and uploads work correctly when used together in a complete workflow.
- **Steps:**
  1. Enter valid dates in different formats in each row.
  2. Fill in the procedure names with a mix of normal and edge case inputs (long names, special characters).
  3. Upload various file types to their corresponding sections, including borderline acceptable and unacceptable file sizes and types.
  4. Attempt to submit the form.
- **Expected Result:**
  - The form accepts only the correct date formats and procedure names within character limits.
  - The upload functionality accepts correct file types and sizes within the limit.
  - An error message is displayed for any invalid input or file.
  - The form submits successfully if all inputs are valid.

5.4.3 Test Case 2: Mandatory Fields and Error Handling

- **Objective:** To ensure the form enforces mandatory fields and correctly handles errors across multiple entries.
- **Steps:**
  1. Leave some mandatory fields empty across different rows.
  2. Enter invalid dates and procedure names.
  3. Attempt to upload incorrect file types and oversized files.
  4. Try to submit the form.
- **Expected Result:**
  - The form submission is blocked.
  - The user is prompted to fill in the mandatory fields.
  - Clear error messages are displayed for each invalid entry or file upload issue.
  - The form does not submit until all errors are resolved.

5.5 Medication Page

5.5.1 User Story:

As a patient, I need to log and track my medication details, including dosages, frequency, and treatment duration, to manage my prescriptions effectively and ensure adherence to my treatment plan.

5.5.2 Test Case 1: Field Validation and Form Submission

- **Objective:** To verify that all input fields accept valid data and the form submits successfully.
- **Steps:**
  1. Enter alphabetic characters in the 'Drug Name' field.
  2. Input numeric and unit-based values in the 'Dosage' and 'Frequency' fields.
  3. Select valid 'Start Date' and ensure 'End Date' is after the 'Start Date'.
  4. Fill out 'Duration of Treatment' with numeric values and units.
  5. Enter a descriptive text in 'Any Reaction?' field.
  6. Type a condition name in 'Medicine For?' field.
  7. Click the 'Submit' button.
- **Expected Result:**
  - Each field should accept the appropriate data type without error.
  - The form should submit and navigate to a confirmation message or page.

5.5.3 Test Case 2: Date Functionality and Layout Verification

- **Objective:** To check the date picker functionality and the proper alignment of form fields for usability.
- **Steps:**
  1. Click on the 'Start Date' and 'End Date' to open the date picker.
  2. Attempt to select a date prior to the current date for 'Start Date'.
  3. Observe the alignment of all fields in the form.

- **Expected Result:**
  - Date pickers for 'Start Date' and 'End Date' should function correctly.
  - The date picker should not allow selection of past dates for 'Start Date'.
  - All form fields should be properly aligned with equal spacing.

5.5.4 Test Case 3: Compatibility and Security Checks

- **Objective:** To ensure the form is compatible across different browsers and is secure against common web threats.
- **Steps:**
  1. Open and submit the form in various browsers: Chrome, Firefox, Safari, and Edge.
  2. Enter a string that includes SQL code in each field.
  3. Attempt to submit the form with this data.
- **Expected Result:**
  - The form should work without issues on all tested browsers.
  - The form should handle SQL code inputs by sanitizing, rejecting, or escaping special characters to prevent SQL injection.

5.6 Medical Information Page

5.6.1 User Story:

As a patient, I want to enter my medical details into the medical information form so that my health records are up-to-date and easily accessible by healthcare professionals.

5.6.2 Test Case 1: Form Functionality and Data Integrity

- **Objective:** To ensure the form's dropdowns work correctly, BMI is calculated or validated, and form data is processed accurately.
- **Steps:**
  1. Verify that dropdowns open and can have an option selected.
  2. If BMI is automatically calculated, check the calculation by entering known height and weight values.
  3. If BMI needs to be entered manually, ensure that only valid numeric entries are allowed.
  4. Fill in all fields, note the entered data, and submit the form.
- **Expected Result:**
  - Dropdown menus should open upon interaction and allow for selection.
  - The BMI should either be calculated correctly or manually entered without errors.
  - Upon form submission, the data should be correctly transmitted to the server or database without loss or corruption.

5.7 Vaccination Page

5.7.1 User Story:

As a healthcare provider or patient, I want to enter and review vaccination records for each specific age or stage milestone to ensure that the patient, regardless of age, is following the recommended vaccination schedule and to keep an up-to-date immunization record.

5.7.2 Test Case 1:Vaccination Schedule Data Entry and Submission Integrity

- **Objective:** To ensure that vaccination entries are correctly inputted, saved for each milestone, and that the entire form can be submitted accurately, maintaining data integrity.
- **Steps:**
  1. For each age milestone (e.g., 'AT BIRTH', '2 MONTHS', '4 MONTHS', etc.), input the appropriate vaccines.
  2. Confirm that each entry is saved correctly within the form.
  3. After all data is entered, click the 'Submit' button.
  4. Check the database or confirmation screen for the accuracy of submitted data.
- **Expected Result:**Vaccination data is retained correctly after entry for each milestone, and upon submission, the data is stored accurately in the database with confirmation provided to the user.

5.7.3 Test Case 2:Cross-Compatibility and User Experience

- **Objective:** To confirm that the vaccination form provides a consistent user experience across different devices and browsers and that the form has proper error handling for data validation.
- **Steps:**
  1. Access the form on various browsers (Chrome, Firefox, Safari, Edge) and on different devices (desktop, tablet, smartphone).
  2. Evaluate the form's usability in terms of layout, readability, and ease of navigation.
  3. Intentionally enter invalid data or leave required fields empty and attempt to submit the form.
  4. Observe the system's response, including error messages and prompts for correction.
- **Expected Result:**The form should be responsive and function uniformly across all platforms. Invalid data entries or missing information should trigger clear, helpful error messages guiding the user to resolve the issues.

5.8 Family History

5.8.1 User Story:

As a patient, I want to input and update medical history details for my family members on a dedicated health information page, So that I can provide a detailed family medical history to healthcare providers, ensuring better personalized care for my family and me.

5.8.2 Test Case

- **Objective:** Ensure that the family information page allows for accurate data entry, handles validations correctly, and supports multiple family member entries.
- **Steps:**
  1. Navigate to the family information page.
  2. For the first family member:
     - Select the relationship from the dropdown.
     - Enter the name, date of birth, sex, and occupation.
     - Input any relevant health conditions and hereditary diseases.
     - If applicable, enter the cause of death and verify if the age at time of death can be entered.
     - Attempt to save with some mandatory fields empty to test validation.

  - Save the completed information and confirm it is displayed correctly.
3. Add a second family member entry and repeat the above steps.
4. Confirm that both entries are saved and displayed correctly without interference.

- **Expected Result:**
  - The system should save entered details for each family member with no errors.
  - Validation messages should appear for any missing mandatory fields.
  - The 'Age at time of death' field should only accept input if the 'Cause of Death' is filled.
  - Multiple family member entries are managed and displayed without conflict.

6 Configuration Management

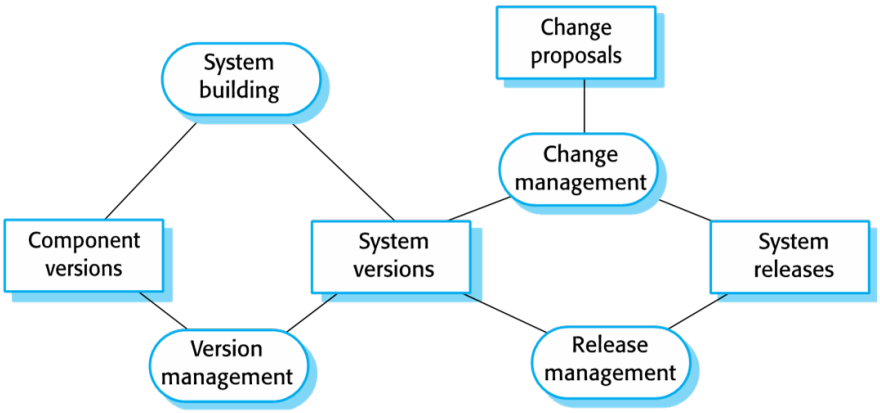


Figure 9: CM Activities

Configuration management (CM) is concerned with the policies, processes, and tools for managing changing software systems.

6.1 Version management (VM)

(VM) Showing the progression of our software versions. Include Branching and Merging Strategies. Major, minor, and patch updates, demonstrate how each new version of our software builds upon the previous one. We used a timeline or a version tree can be used to visually represent the chronological order of changes, updates, and bug fixes for each version.

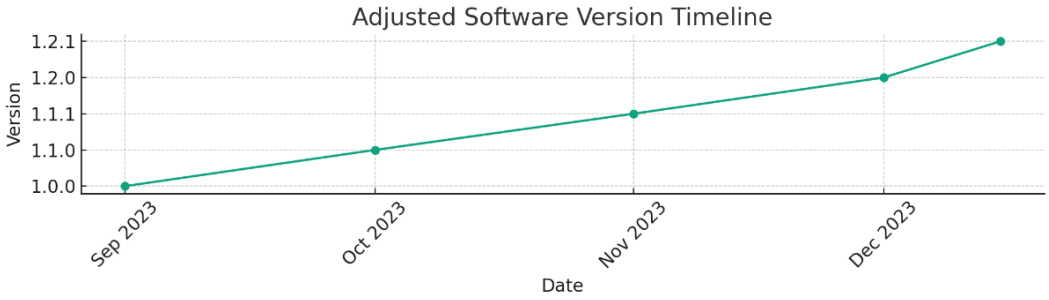


Figure 10: Timeline Diagram for the software versioning scheme

This adjusted timeline provides a clear view of the versioning process of our software over the recent months, highlighting the development and release pattern of our software.

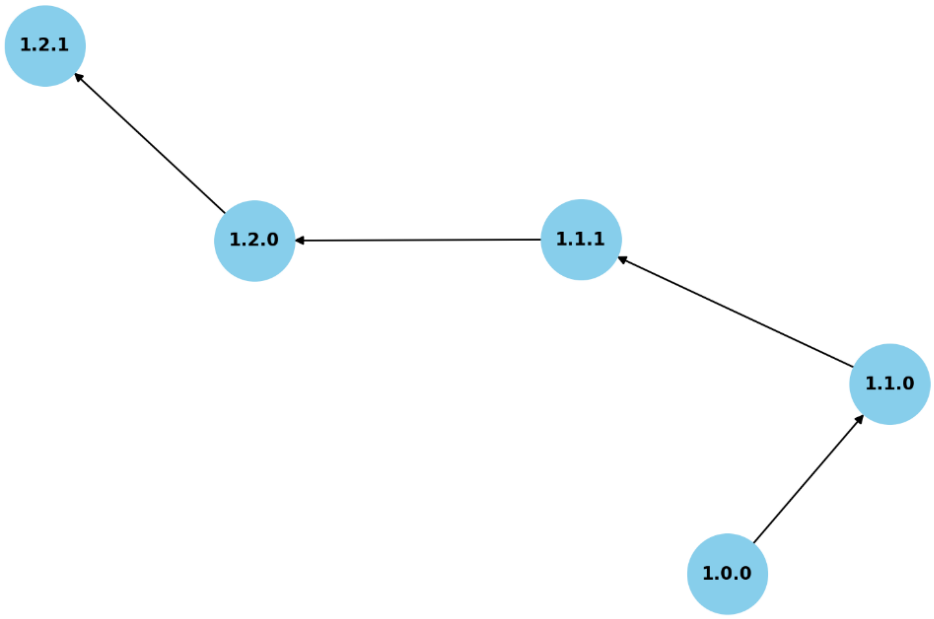


Figure 11: Version tree diagram

6.1.1 System version

The system's versioning begins with the major release 1.0.0, followed by a series of planned updates: minor update 1.1.0 adding new features, patch 1.1.1 for bug fixes, another minor update 1.2.0 introducing further enhancements, and a subsequent patch 1.2.1 for additional refinements. This versioning hierarchy demonstrates a structured progression, with each release building on the previous one to evolve the system's functionality and reliability.

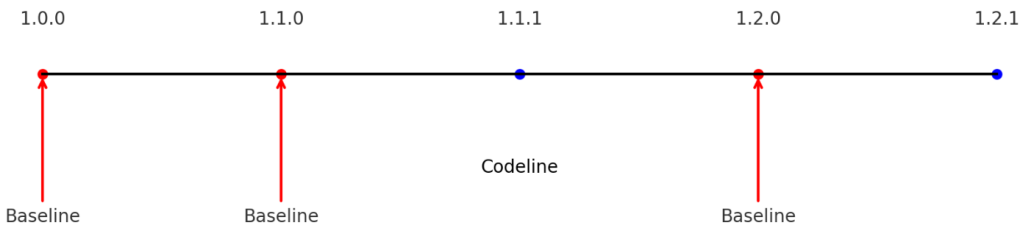


Figure 12: Codelines and baselines in the versioning

Codelines are branches in a version control system. Each codeline represents a stream of changes to a set of files, leading to different versions or releases of our software system. **Baselines** are specific points within a codeline that are designated as stable and serve as reference points. It typically represents a completed phase in development, such as the end of a sprint or a release milestone.

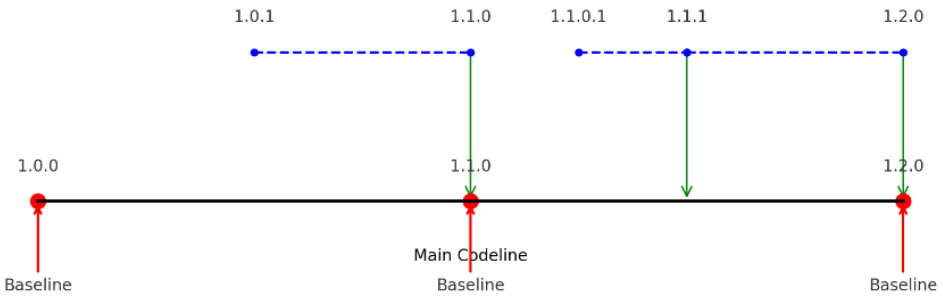


Figure 13: Branching and Merging

Branching allows us as developers to diverge from the main code line (the baseline) to work on different tasks simultaneously without interfering with the stable version of the software. This is critical when developing new features (like going from 1.0.0 to 1.1.0), addressing bugs (as in moving from 1.1.0 to 1.1.1), or experimenting with new ideas. Each branch represents an isolated environment where changes can be made, tested, and refined independently.

Merging is the process of integrating the changes from these branches back into the main code line, ensuring that the updates are consolidated and that the software evolves cohesively. This practice is vital for maintaining the integrity of the software, as it allows for the systematic combination of different development efforts. For instance, after developing and testing new features in a separate branch, these can be merged back to form a new baseline version (like merging the changes from 1.1.1 to create a new baseline at 1.2.0), thereby updating the main code line with the latest enhancements.

Throughout the project, we have successfully applied these strategies to maintain a high standard of code integrity and application reliability. For instance, our 'feature-appointment-scheduling' branch allowed us to develop a complex appointment scheduling system without disrupting the main application flow. It was only merged into the master after extensive testing and review, ensuring a seamless integration. By meticulously applying branching and merging, we have established a development environment that is both flexible and stable, supporting our team's collaborative efforts and our software's continuous evolution.

6.2 Change Management

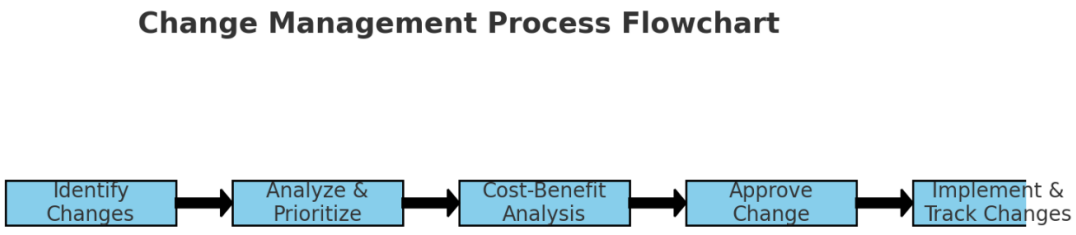


Figure 14: Change Management Process

The Change Management Process outlines a strategic framework for updating our software systems. This process provides a disciplined approach to managing changes, from inception through to completion, facilitating controlled system evolution.

6.2.1 Identification of Changes:

Originating from user feedback, new healthcare regulations, or identified bugs, changes are first identified. For example, modifications might include updates to patient record management or enhancements to the system's integration with healthcare facilities.

6.2.2 Analysis and Prioritization:

Each proposed change undergoes a thorough analysis, considering factors like urgency, user benefits, and resource availability. For instance, a suggestion to enhance the medication tracking feature is evaluated based on its potential to improve patient outcomes against the associated development and implementation costs.

6.2.3 Cost-Benefit Analysis:

we conduct a cost-benefit analysis for high-priority changes. For example, enhancing the medication tracking feature might be evaluated for its potential to improve patient outcomes against the development and implementation costs.

6.2.4 Formal Approval:

Changes that clear the cost-benefit analysis are formally approved. This step might involve a review by the project's Product Owner and Scrum Master to align changes with the overall product vision and project timeline.

6.2.5 Implementation and Tracking:

Approved changes are implemented. For example, a new feature for lab data integration would be developed, tested, and deployed. Post-deployment, the change is tracked for effectiveness and to ensure it aligns with performance goals and user needs. Throughout this process, we as a team of Medico Memory utilize Agile methodologies, ensuring flexibility and responsiveness to changing requirements or unforeseen challenges. Regular team meetings and clear roles, as outlined in the project report, support effective communication and collaboration during the change management process.

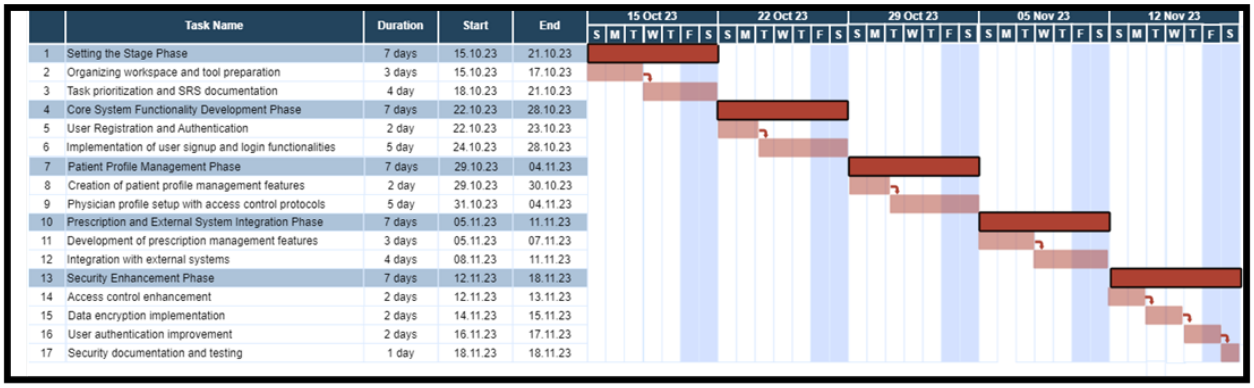
7 Teamwork

We worked together on a project like a really good team. We all talked a lot and shared our ideas and problems. This made us trust and respect each other, which helped us work well together. Everyone had different skills and ways of thinking, which made our team creative and good at solving problems. We gave each other tasks that matched what we were good at and liked doing. This made us work better and feel happy about what we were doing. We had regular meetings to make sure we all knew what was going on and could handle any changes easily. Working together like this, we all helped push the project towards success, even doing better than we thought we would. Doing this project as a team not only helped us reach our goals but also made us feel like we were part of something special. The journey was just as great as finishing the project.

Team Member	Role
Leen Sharab	Management, Wix Site Designer, Tester
Sarah Alshumayri	Graphic Designer, Wix Site Designer, Tester
Reema Abdallah	Content Strategist, Wix Site Designer, Tester
Ameera Attiah	Graphic Designer, Wix Site Designer, Tester
Lujain Almarri	Content Strategist, Page Integrations, Tester

Table 1: Teamwork

8 Project Management



	Task Name	Duration	Start	End
1	Setting the Stage Phase	7 days	15.10.23	21.10.23
2	Organizing workspace and tool preparation	3 days	15.10.23	17.10.23
3	Task prioritization and SRS documentation	4 day	18.10.23	21.10.23
4	Core System Functionality Development Phase	7 days	22.10.23	28.10.23
5	User Registration and Authentication	2 day	22.10.23	23.10.23
6	Implementation of user signup and login functionalities	5 day	24.10.23	28.10.23
7	Patient Profile Management Phase	7 days	29.10.23	04.11.23
8	Creation of patient profile management features	2 day	29.10.23	30.10.23
9	Physician profile setup with access control protocols	5 day	31.10.23	04.11.23
10	Prescription and External System Integration Phase	7 days	05.11.23	11.11.23
11	Development of prescription management features	3 days	05.11.23	07.11.23
12	Integration with external systems	4 days	08.11.23	11.11.23
13	Security Enhancement Phase	7 days	12.11.23	18.11.23
14	Access control enhancement	2 days	12.11.23	13.11.23
15	Data encryption implementation	2 days	14.11.23	15.11.23
16	User authentication improvement	2 days	16.11.23	17.11.23
17	Security documentation and testing	1 day	18.11.23	18.11.23

Figure 15: Gantt chart

The Gantt chart clearly shows our project's full schedule, from start to finish. It carefully marks each phase of the project with specific start and end dates, making it easy to see the project's timeline at a glance.

8.1 Project Planning Excellence

Our project plan is very detailed, with each task clearly stated and arranged in order. We start with a phase called "Setting the Stage Phase," which prepares the groundwork for everything else. After that, the "Core System Functionality Development Phase" and the "Patient Profile Management Phase" focus on important early steps, making sure the main parts of the project are set up first. This way, we can easily add more features and improvements later on.

8.2 Risk Management

In our risk management, we used Agile methods, which let us adjust to changes and unexpected problems easily. Our Gantt chart shows some phases overlapping. This was done on purpose to have extra time as a safety net for delays we didn't plan for. This flexibility was really helpful, especially when we had delays in the "Patient Profile Management Phase." We could change our short-term goals without messing up our overall schedule. Even with these issues, our team stayed flexible and strong.

8.3 Adherence to Timelines

The project timeline shows that we are very strict about following our schedule. Tasks are set up to happen one after the other or at the same time, which helps us use our time and resources well. The darker parts of the task bars (which show how much we have done) show that we are carefully sticking to our important goals. However, it's important to remember that knowing how well we are sticking to the schedule will depend on tracking our actual progress compared to what we planned.

9 User Roles Assignment

9.1 Product Owner

Responsibilities: Leen Sharab, as the Product Owner, is responsible for defining the product vision and prioritizing the product backlog. This involves understanding customer needs, analyzing market trends, and deciding which features the development team should focus on next.

9.2 Scrum Master

Responsibilities: Sarah Alshumayri serves as the Scrum Master, facilitating the Scrum process and ensuring the team adheres to Agile practices. Her role includes removing impediments to progress, coaching team members in Agile methodologies, ensuring effective communication and collaboration, and resolving conflicts within the team.

9.3 Developers

Responsibilities: The developers, including Reema Abdullah, Lujain Almarri, and Ameera Attiah, are tasked with writing and testing code, implementing features, and fixing bugs. They work collaboratively to develop and refine the product according to the specifications and guidance from the Product Owner.

9.4 Testers

Responsibilities: As testers, Sarah Alshumayri, Leen Sharab, Reema Abdullah, Lujain Almarri, and Ameera Attiah are responsible for testing the software to identify and report bugs. Their role involves executing test cases to ensure new features work as expected and maintaining the quality of the final product.

Each team member in this project has clearly defined responsibilities, contributing to an organized and efficient workflow.

10 Conclusion

In conclusion, the Software Engineering course under Dr. Akila Sarirete has been profoundly beneficial, significantly enhancing our skills and knowledge in the field. The practical focus of the course, enabled us to deeply understand and apply core software engineering concepts. Through hands-on projects, we not only improved our technical abilities but also developed essential soft skills like teamwork, communication, and problem-solving. This course has been instrumental in preparing us for real-world challenges in software engineering, equipping us with both the confidence and competence to excel in our future endeavors.