Project Meduron FSD

Version 1.0

November, 2024

Table of Contents

1. Introduction

1.1. Project Overview

The "Meduron" platform is an interactive educational application designed to enhance the learning experience for medical students through interactive notes, examination, and problem-based learning (PBL)**.** By presenting realistic medical scenarios and multiple-choice questions (MCQs), Meduron enables students to engage in clinical problem-solving, critical thinking, and active learning. With a focus on accessibility and usability, Meduron targets a broad audience of students seeking a modern, case-based approach to medical education.

1.2. Purpose of the Document

This Functional Specification Document (FSD) outlines the functional requirements and expected behaviors of the Meduron application. It serves as a comprehensive blueprint, ensuring that all stakeholders, including project managers, developers, and testers, have a clear understanding of the system's functionalities and expected outcomes. This document aligns the technical development process with the overarching educational objectives, delivering a solution that meets the needs of its end-users.

1.3. Scope

This FSD details the core functionalities of the Meduron application, including but not limited to:

- User registration, login, and profile management
- Access to educational content, notes, and interactive case scenarios
- Access to MCQs on multiple subjects under the medical profession
- Access to lecture slides from various universities
- Problem-based learning cases with diagnostic tools and guided scenarios
- Subscription and billing management for premium features
- Integration with external tools for seamless content delivery and user engagement

Excluded from this FSD are technical implementation details for underlying algorithms and third-party service integrations, as well as deployment and marketing strategies.

1.4. Assumptions

The following assumptions have been established for this document:

1. **User Competency**:
    a. Users are assumed to have basic technical skills, enabling them to interact with digital learning tools effectively and navigate through the Meduron platform without extensive guidance.
2. **Device Compatibility**:
    a. The application will support multiple device types, including desktops, tablets, and smartphones, to ensure a wide range of accessibility.
3. **Network Availability**:
    a. The application assumes users have a reliable internet connection for accessing interactive learning modules, MCQs, and real-time updates. However, offline functionality will be implemented to allow users to continue

accessing previously loaded content and certain features without an active internet connection.

4. **Standard File Formats**:
    a. Educational materials will be provided in standard digital formats, such as PDF, JPEG, PNG and MP4, ensuring compatibility across various devices and commonly used software.
5. **Security and Data Privacy Compliance**:
    a. The platform will follow industry-standard security practices, including data encryption (in transit and at rest) and secure authentication (OAuth and MFA where applicable). Compliance with data privacy regulations (e.g., GDPR, CCPA) will be maintained, ensuring users' personal information is protected and managed responsibly.
6. **Offline Performance**:
    a. The Meduron platform will support offline functionality for critical features, allowing users to access downloaded or cached content when an internet connection is unavailable. Any data inputs or updates made offline will be synchronized once the user reconnects to the internet.

1.5. Definitions & Acronyms

- **Meduron**: The educational platform developed for medical students, supporting examinations, educational notes and clinical problem-solving through interactive learning.
- **FSD**: Functional Specification Document, outlining the functionalities, features, and interactions of the Meduron system.
- **PBL**: Problem-Based Learning, an educational approach wherein students tackle real-world scenarios to enhance critical thinking skills.
- **MCQ**: Multiple-Choice Questions, a standardized form of assessment used to test medical knowledge.
- **UI**: User Interface, the interactive layer of the application.
- **API**: Application Programming Interface, enabling system interoperability and integration with third-party tools.

2. System Overview

2.1. System Context

The Meduron platform is a comprehensive digital learning solution designed for medical students. It integrates advanced educational tools and interactive content, such as problem-based learning (PBL) case studies and multiple-choice questions (MCQs), into a single, accessible platform. The system includes user management and content sharing functionalities, ensuring that students and educators can engage with and personalize the learning experience to meet their needs.

- **Educational Content Delivery**: Meduron provides PBL case scenarios and MCQs, allowing students to test and apply their clinical knowledge. The platform dynamically adapts to student inputs, presenting relevant educational materials in real-time.
- **User Interaction**: Users interact with the platform through an intuitive user interface that facilitates easy access to content, profile management, progress tracking, and assessments.
- **External Integration**: Meduron integrates with external services and tools, including email for password resets and third-party libraries for enhanced interactivity and content delivery.

2.2. Key Stakeholders

The success of Meduron relies on the collaboration and contribution of several key stakeholders, each playing a vital role in the platform's development and operation:

- **Product Owner**: Responsible for aligning the platform’s development with strategic goals and educational needs, prioritizing features based on user feedback and market requirements.
- **Product Manager**: Defines the product vision and roadmap, ensuring Meduron meets the educational objectives of its target audience. Collaborates closely with the development team to manage timelines and feature prioritization.
- **Tech Lead**: Oversees the technical implementation of the system, ensuring that all components are integrated smoothly and efficiently. The Tech Lead is responsible for architectural decisions and technical problem-solving.
- **Development Team**: Engineers and developers tasked with implementing the features and functionalities specified in this document.
- **UI/UX Designers**: Design an intuitive and accessible interface that enhances user interaction, ensuring a seamless experience across devices.
- **Quality Assurance (QA) Team**: Conducts testing to confirm that Meduron meets all specified requirements, identifying and resolving issues to ensure a stable, reliable platform.
- **Marketing Team**: Focuses on promoting Meduron to medical students, educators, and professionals, driving adoption and engagement through targeted marketing campaigns.
- **End Users**: Medical students, educators, and medical professionals who use Meduron as a learning and teaching tool. User satisfaction and feedback are essential for ongoing improvement and feature development.

2.3. User Persona

Meduron is designed to cater to diverse user personas, each with unique needs and learning goals:

- **Persona 1: Medical Student**
    - **Description**: A student currently enrolled in a medical degree program who uses Meduron for supplemental learning and exam preparation.
    - **Goals**: To engage in interactive PBL scenarios, test knowledge with MCQs, and improve clinical skills through practical applications.
- **Persona 2: Educator**
    - **Description**: A faculty member in a medical institution who uses Meduron as a teaching aid for case-based learning and student assessments.
    - **Goals**: To assign cases, monitor student engagement and progress, and assess knowledge retention through interactive content.
- **Persona 3: Medical Professional (Self-Learner)**
    - **Description**: A practicing professional or recent medical graduate aiming to stay current with clinical skills or prepare for certifications.
    - **Goals**: To access targeted case scenarios and MCQs to maintain proficiency and deepen knowledge in specific areas of medical practice.
- **Persona 4: Admin**
    - **Description**: An administrator responsible for managing platform content, user accounts, and overall system maintenance. The admin ensures that Meduron's educational content is updated, user activity is monitored, and platform compliance with data privacy regulations is upheld.
    - **Goals**:
        - To manage user accounts and permissions, ensuring appropriate access levels for students, educators, and self-learners.
        - To oversee content updates and approve or moderate user-uploaded files and materials.
        - To monitor platform usage, run analytics reports, and address any system issues.
        - To ensure compliance with data privacy laws (e.g., GDPR) and handle user inquiries related to account management or data requests.

3. Functional Requirements

The Meduron platform consists of essential functionalities aimed at delivering an interactive, accessible, and effective learning experience for medical students. This section outlines the primary features, each with specific inputs, processing logic, and outputs.

3.1. Feature List

The key functional features of the Meduron platform include:

- User Registration and Login
- User Profile Management
- Notes Access
- File Sharing System
- Problem-Based Learning (PBL)
- Multiple-Choice Questions (MCQ) System
- Personalized Learning Dashboard
- Subscription Management
- User Dashboard
- Referral Link and Rewards Management
- Content Bookmarking and Favorites
- Analytics and Reporting
- Notification System
- Content Search and Filtering
- In-App Support and Help Desk
- Admin User Management
- Admin Content Management
- Interactive Case Review
- Usage Analytics Dashboard
- Coupon Management

3.2. Feature Descriptions

- User Registration and Login
    - **Description**: This feature provides users with secure account creation, authentication, and password recovery options. New users can create an account via email registration, with an option to use a referral link if provided. The login system validates user credentials, allowing access to the Meduron platform. Users who forget their passwords can initiate a recovery process through email verification.
    - **Inputs**: User inputs for registration include name, email, phone number, password, university, current year and optional referral link. For login, users provide email/username and password. For password recovery, an email address is required.
    - **Processing Logic**:
        - The system checks that all registration fields are filled and validates the email format.
        - Passwords are encrypted and stored securely in the database.

        - An email verification link is sent for account confirmation.
        - During login, credentials are verified, and upon successful authentication, an access token is issued. If incorrect credentials are entered, an error message is displayed.
        - For password recovery, an email with a reset link is sent. The link redirects the user to a password reset page where they can set a new password.
    - **Outputs**: Success messages for registration, login, or password reset, and access to the Meduron platform. Error messages for incomplete forms, incorrect login details, or unverified accounts.
- User Profile Management
    - **Description**: This feature allows users to view, edit, and manage their personal profile details, including their name, email, university, academic year, and profile picture. This information helps personalize the platform experience and maintain accurate records of user demographics.
    - **Inputs**: Profile details provided by the user, such as name, email, university, and profile picture file (if being uploaded).
    - **Processing Logic**:
        - The system verifies the provided inputs to ensure completeness and format validity (e.g., proper email format, acceptable image size and type).
        - Upon confirmation, the updated information is securely stored in the database.
        - If the user uploads a new profile picture, the system checks the file size and format before storing it.
    - **Outputs**: Confirmation message for successfully updated profile information, or error messages if there is invalid input.
- Note Access
    - **Description**: Allows users to access a library of curated notes and study materials organized by topics and categories. This feature enables students to study, bookmark, and quickly navigate between notes on specific medical subjects, supporting enhanced learning experiences.
    - **Inputs**: User-selected topic, category, or keyword.
    - **Processing Logic**:
        - The system searches for and retrieves relevant notes based on the user's selection criteria.
        - Display options include bookmarks for saving notes, highlights for marking important sections, and a personalized "My Notes" section for quick access.
        - The system caches recently accessed notes to improve loading times for frequently used materials.

        - **Outputs**: Displayed notes content, saved bookmarks, and quick-access links in the user's "My Notes" section also support vocabulary .
- File Sharing System
    - **Description**: The file-sharing feature allows users to upload, share, and access learning resources, including lecture slides, lab materials, and other academic files relevant to medical studies. This feature also enables filtering files by categories like university and subject.
    - **Inputs**: Uploaded files with metadata (file title, university, module, subject, etc.).
    - **Processing Logic**:
        - The system validates file type and size to ensure compatibility with platform standards.
        - Files are queued for admin approval before they become publicly accessible.
        - Users can search and filter shared files based on tags, university, or subject to locate specific resources.
    - **Outputs**: Confirmation message for successful uploads, search results for specific queries, and error messages for invalid uploads or access issues.
- Problem-Based Learning (PBL)
    - **Description**: PBL allows users to work through interactive clinical case scenarios, mimicking real-life diagnostic and treatment processes. Students can make choices throughout the case to simulate clinical reasoning and after that give hint to the user or is the answer is also correct it'll provide alternative solutions
    - **Inputs**: User selections and diagnostic inputs for each step of the case.
    - **Processing Logic**:
        - Cases are presented in stages, with each stage revealing additional details based on previous user choices.
        - The system evaluates each user input to provide tailored hints, feedback, and corrective guidance, helping users understand each decision's impact.
    - **Outputs**: Progressive case feedback, hints, and final case results with explanations.
- Multiple-Choice Questions (MCQs) System
    - **Description**: The MCQ system presents users with examination to test their understanding of key medical concepts and retain clinical knowledge.
    - **Inputs**: User-selected answers to each question.
    - **Processing Logic**:
        - The system verifies answers, records progress, and calculates scores.

        - Immediate feedback is provided for correct/incorrect answers, and explanations are shown to reinforce learning.
    - **Outputs**: Quiz scores, detailed feedback on answers, and explanations.
- Personalized Learning Dashboard
    - **Description**: The personalized dashboard tracks user progress, saves learning history, and provides quick access to recent activities and recommended content.
    - **Inputs**: User's saved data on completed cases, quizzes, and bookmarked materials.
    - **Processing Logic**:
        - The system compiles user activity data to display completed cases, quiz scores, and overall learning progress.
        - Recommendations are generated based on past interactions and preferences.
    - **Outputs**: Dashboard displaying recent activities, learning metrics, and recommended materials.

- Subscription Management
    - **Description**: This feature enables users to manage their subscription plans, which may include different tiers with access to premium content or features. Users can view, upgrade, or renew their subscriptions directly through the platform.
    - **Inputs**: User-selected subscription plan, payment details, and any applicable discount/coupon codes.
    - **Processing Logic**:
        - The system validates the selected plan and payment details.
        - If a coupon code is applied, it checks for validity and adjusts the subscription cost accordingly.
        - Upon successful payment, the system updates the user's subscription status and grants access to relevant features.
    - **Outputs**: Subscription confirmation and receipt, updated user access to subscription-based features, or error messages if payment fails or if a coupon is invalid.
- User Dashboard
    - **Description**: A centralized user interface that provides access to various features, resources, and updates relevant to the user's account. The dashboard displays recent activities, bookmarked content, and personalized recommendations.
    - **Inputs**: User profile data, activity logs, saved content, and subscription status.

        - **Processing Logic**:
            - The system retrieves recent user activities, bookmarks, and recommended materials.
            - It dynamically displays content based on user engagement, including notifications, updates, and reminders for relevant features.
        - **Outputs**: Display of recent activities, bookmarks, learning recommendations, and notifications tailored to the user.
- Referral Link and Rewards Management
    - **Description**: Allows users to refer others to the Meduron platform through a unique referral link. Users can earn rewards, such as discounts or credit, for successful referrals.
    - **Inputs**: User-generated referral link, actions of referred users (e.g., new sign-up, subscription).
    - **Processing Logic**:
        - The system tracks the usage of referral links, verifying successful referrals (e.g., completed registrations).
        - Rewards are granted based on set criteria (e.g., referral results in a subscription).
    - **Outputs**: Notifications for successful referrals and rewards earned, or error messages if the referral conditions are not met.
- Content Bookmarking and Favorites
    - **Description**: Allows users to save specific content as bookmarks or mark it as a favorite for easy access. This feature enables users to quickly locate frequently referenced materials.
    - **Inputs**: User selection of content to bookmark or favorite.
    - **Processing Logic**:
        - The system records the bookmarked or favorited items under the user's profile.
        - It provides quick-access links to saved content on the user's dashboard and profile.
    - **Outputs**: Success messages for saved bookmarks and favorites, and display of bookmarked content in the user's profile or dashboard.
- Analytics and Reporting
    - **Description**: Provides data analytics on user engagement and performance, allowing users to monitor their own progress and administrators to assess platform usage and content effectiveness.
    - **Inputs**: User activity logs, content interaction data, and performance metrics.
    - **Processing Logic**:
        - The system aggregates and analyzes data to create reports and charts on user engagement and content effectiveness.

        - Reports are available to users and administrators based on access levels.
    - **Outputs**: Display of analytics reports, performance metrics, and visual representations of user engagement.
- Notification System
    - **Description**: Sends notifications and reminders to users regarding platform updates, content releases, subscription renewals, and other significant events.
    - **Inputs**: Trigger events such as new content, expiring subscriptions, or completed quizzes.
    - **Processing Logic**:
        - The system generates notifications based on trigger events and user settings.
        - Notifications can be sent in-app, via email, or SMS based on the user's preferences.
    - **Outputs**: Notifications delivered to users about relevant updates and events, with messages tailored to their preferences and actions.
- Content Search and Filtering
    - **Description**: Enables users to search and filter through a vast repository of educational content, making it easy to locate specific topics, notes, or files.
    - **Inputs**: User-entered search terms, filter selections (e.g., by category, subject, or university).
    - **Processing Logic**:
        - The system performs a keyword search or applies filters to narrow down the content results.
        - Results are ranked by relevance and displayed to the user.
    - **Outputs**: Filtered search results or an error message if no matching content is found.
- In-App Support and Help Desk
    - **Description**: Offers in-app support options, including a help desk, chat support, and FAQ section, enabling users to resolve issues and get assistance with the platform.
    - **Inputs**: User queries, support tickets, or FAQ searches.
    - **Processing Logic**:
        - The system categorizes and routes user queries to the appropriate support channels.
        - Chat support is provided in real-time, while FAQ responses are automated based on user questions.
    - **Outputs**: Responses to user queries via chat or helpdesk, and relevant FAQ suggestions based on queries.
- Admin User Management
    - **Description**: Allows administrators to manage and monitor user accounts, including actions like account creation, updates, suspension, and deletion.

        - **Inputs**: Admin actions on user accounts, user profile data.
        - **Processing Logic**:
            - The system allows admins to create, update, suspend, or delete accounts.
            - The activity of each account is tracked for monitoring purposes.
        - **Outputs**: Confirmation messages for successful account actions, account status changes, and logs for admin records.
- Admin Content Management
    - **Description**: Enables administrators to create, edit, approve, and organize content, such as case studies, notes, quizzes, and user-uploaded files.
    - **Inputs**: Content data (titles, descriptions, categories) and admin-defined settings.
    - **Processing Logic**:
        - Admins can add new content or approve content uploaded by users.
        - The system categorizes and organizes content based on admin input for efficient retrieval.
    - **Outputs**: Display of approved content on the platform, with administrative logs of content actions.
- Interactive Case Review
    - **Description**: Provides tools for educators to review and assess user performance on interactive case scenarios, enabling them to offer guidance and feedback.
    - **Inputs**: User responses and progress within each case scenario.
    - **Processing Logic**:
        - The system compiles data on user responses and time taken at each case stage.
        - Educators can view user performance, provide feedback, and mark areas for improvement.
    - **Outputs**: Case review feedback provided to users, and progress reports for educators to track individual or group performance.
- Usage Analytics Dashboard
    - **Description**: Presents administrators with a real-time dashboard displaying metrics on platform usage, engagement, and content popularity, helping assess overall performance.
    - **Inputs**: User activity logs, content views, and interaction data.
    - **Processing Logic**:
        - The system aggregates user data and content interactions to generate charts and tables.
        - Key metrics, such as active users, most-viewed content, and engagement trends, are visualized on the dashboard.

        - **Outputs**: Analytics dashboard with visualized metrics, allowing admins to view and analyze trends and make data-driven decisions.
- Coupon Management
    - **Description**: Allows admins to create, configure, and manage subscription discount coupons for users. Admins can specify constraints such as maximum uses, expiration date, single-user limit, and applicable subscription plans.
    - **Inputs**: Coupon code, usage limit, expiration date, allowed users, and associated subscription plan.
    - **Processing Logic**:
        - Admin enters coupon details, specifying the expiration date, maximum allowed uses, specific users or user groups allowed to use the coupon, and the subscription plan it applies to.
        - System saves coupon data and enforces these rules at checkout, ensuring each coupon meets defined criteria and can be used only once per user.
        - Users apply coupons at checkout, where the system validates their eligibility and applies discounts if applicable.
    - **Outputs**: Confirmation of coupon creation for admins, coupon application success or failure during checkout, and error messages if conditions are not met (e.g., expired coupon, already used by the user).

4. Non-Functional Requirements

4.1. Usability Requirements

Ensuring Meduron is user-friendly, intuitive, and accessible is essential for providing a positive user experience across a diverse audience.

- **Ease of Use**:
  - The user interface should be intuitive and straightforward, with clearly labeled navigation and controls.
  - Users should be able to perform key actions (e.g., accessing notes, quizzes, and cases) within three clicks from the dashboard.
  - Consistent design patterns and familiar iconography should reduce the learning curve for new users.
- **Accessibility**:
  - Meduron should comply with the Web Content Accessibility Guidelines (WCAG) 2.1 standards to support users with disabilities.
  - Screen readers should be compatible with the platform, with alternative text descriptions for all images and clear labeling for form fields.
  - Support for keyboard navigation should be included, allowing users to interact without relying on a mouse.
  - Adjustable font sizes and high-contrast color themes should be provided for users with visual impairments.
- **User Assistance and Guidance**:
  - Tooltips, contextual help, and a dedicated help section should be available to guide users through the platform.
  - A brief onboarding tutorial should be presented to new users, highlighting key features and actions.

4.2. Reliability & Availability Requirements

Meduron needs to be a dependable, continuously available platform to ensure uninterrupted learning and usage by users.

- **Uptime**:
  - The platform should aim for 99.9% uptime, minimizing downtime and ensuring that it is accessible around the clock except for scheduled maintenance.
- **Data Consistency and Integrity**:
  - All user data, including learning progress, bookmarks, and activity logs, must be consistently stored and synchronized.
  - Regular backups of critical data should be scheduled, with data recovery mechanisms in place to restore information in case of unexpected failures.
- **Redundancy and Failover**:
  - Redundant systems should be deployed to ensure continued service availability. In case of server failure, backup servers should automatically take over to maintain functionality.
  - Data replication across servers should prevent data loss and ensure reliability during high-traffic periods.
- **Scalability**:
  - The platform should support scalable infrastructure to accommodate an increasing number of users without performance degradation. This includes the ability to scale horizontally to handle peak loads.

4.3. Compliance Requirements

Meduron must meet applicable regulatory standards and ensure compliance with privacy laws to protect user rights and data integrity.

- **Legal Compliance**:
  - Meduron should comply with data protection regulations, such as GDPR and CCPA, by ensuring users have control over their personal data, including access, modification, and deletion rights.
  - Clear, accessible terms of service and a privacy policy should be available to all users, detailing data usage, storage, and user rights.
- **Age and Content Regulations**:
  - Implement age-verification checks to ensure compliance with COPPA or other relevant regulations, especially if the platform includes users under 18.
  - Provide appropriate content warnings or filters to prevent access to sensitive or advanced content by younger users if applicable.

4.4. Performance Requirements

To provide a smooth and responsive experience, Meduron should meet defined performance metrics, ensuring efficient response times and load handling capabilities.

- **Response Times**:
  - User interactions, such as loading notes, quizzes, and case studies, should have a maximum response time of 2 seconds.
  - Quiz completion, scoring, and feedback displays should process within 1 second.
- **Throughput**:
  - The platform should support up to 10,000 concurrent users, handling simultaneous activities like quiz submissions, case access, and content browsing.
  - Content upload, particularly for educational material and user notes, should be processed within 5 seconds.
- **Latency**:
  - Data retrieval latency from the database should be optimized to be less than 500ms for primary actions (e.g., accessing user-specific data like bookmarks or recent activities).
  - Network latency for real-time interactions (e.g., chat or support queries) should be minimized to provide near-instantaneous feedback.

4.5. Security Requirements

Meduron must implement robust security measures to protect user data, ensure privacy, and comply with data protection regulations.

- **Authentication and Authorization**:
  - Implement secure, token-based authentication methods (e.g., JWT) for user login sessions, with the option of multi-factor authentication for added security.

    - Role-based access control (RBAC) should be applied to segregate permissions, ensuring users can only access relevant features.
- **Data Encryption**:
    - Sensitive data, such as user credentials and payment details, must be encrypted in transit (using TLS) and at rest.
    - All personal information should be stored in compliance with relevant privacy laws (e.g., GDPR).
- **Regular Security Audits**:
    - Conduct quarterly vulnerability scans and annual penetration testing to identify and mitigate potential security issues.
    - Implement intrusion detection and response protocols to monitor and address unauthorized access attempts.
- **Data Masking and Logging**:
    - Personally identifiable information (PII) should be masked in application logs.
    - Maintain audit logs for critical activities, such as admin actions and login attempts, with regular monitoring for suspicious activities.

4.6. Support & Maintenance Requirements

Providing continuous support and maintenance is essential to ensure Meduron's functionality, security, and usability over time.

- **Support Availability**:
    - A dedicated support team should be accessible during peak hours, with options for in-app helpdesk, chat support, and email assistance.
    - Provide a knowledge base with FAQs, user guides, and tutorials to help users resolve common issues independently.
- **Scheduled Maintenance**:
    - Regular platform updates should be scheduled during low-usage periods to minimize disruptions. Users should receive advance notifications about scheduled maintenance.
    - Implement a versioning system to ensure backward compatibility and smooth transitions for new feature releases.
- **Security and Software Patches**:
    - Critical security patches should be deployed as soon as vulnerabilities are identified to protect the platform from emerging threats.
    - Regularly update libraries, frameworks, and dependencies to the latest stable versions to ensure compatibility and address potential security risks.
- **User Feedback Loop**:
    - Implement a user feedback system, allowing users to submit suggestions, bug reports, and usability feedback.
    - The development team should review feedback regularly to prioritize enhancements and improvements in upcoming releases.

5. Use Cases

Use Case 1: User Registration and Login

Name: User Registration and Login
Description: Enables users to register with email /google, log in, and reset passwords for accessing Meduron.
Primary Actor: User
Preconditions:

- User has a valid email address.
- The system is accessible online.

Steps:

1. User selects "Sign Up" and provides name, email, phone number, password, university, academic year and an optional referral link.
2. System validates and encrypts the password, then sends an email verification link.
3. For login, the user provides credentials; the system authenticates and issues an access token.
4. For password recovery, the user requests a reset link, sets a new password upon email verification.

Postconditions:

- User accounts are created, verified, logged in, or password reset.
- Errors are shown for incomplete fields, invalid credentials, or unverified accounts.

Use Case 2: User Profile Management

Name: User Profile Management
Description: Allows users to view and edit profile details like name, email, profile picture and password.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User navigates to the profile page and selects "Edit."
2. Users update fields such as name, email, and upload a new profile picture.
3. System validates and saves changes in the database.

Postconditions:

- User profile is updated successfully, or an error message is shown for invalid input.

Use Case 3: Notes Access

Name: Notes Access
Description: Enables users to access, bookmark, and navigate study notes.
Primary Actor: User
Preconditions:

- User is logged in and has access to the notes feature.

Steps:
1. User searches for or selects a note by topic or keyword.
2. System retrieves and displays the note content.
3. Users can bookmark, highlight, or navigate to other sections within the note.

Postconditions:
- Notes are accessible, bookmarked, and saved in the user's profile.

Use Case 4: File Sharing System

Name: File Sharing System
Description: Allows users to upload, share, and view academic files.
Primary Actor: User
Preconditions:
- User is logged in with an active account.

Steps:
1. User uploads a file and fills in required fields (e.g., file title, university).
2. System validates file type and size, then queues it for admin approval.
3. Upon approval, the file becomes accessible to others.

Postconditions:
- File is successfully uploaded and available for other users after approval.

Use Case 5: Problem-Based Learning (PBL)

Name: Problem-Based Learning (PBL)
Description: Provides interactive clinical case studies for diagnostic learning.
Primary Actor: User
Preconditions:
- User is logged in and has access to PBL.

Steps:
1. User selects a case and starts the PBL process.
2. User makes selections at each stage based on case details.
3. System provides hints, feedback, and progresses the case accordingly.

Postconditions:
- User completes the case, with feedback provided based on their responses.

Use Case 6: Multiple-Choice Questions (MCQ) System

Name: Multiple-Choice Questions (MCQ) System
Description: Enables users to take quizzes to test their medical knowledge.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:

1. User selects an MCQ set to start.
2. User answers each question; the system verifies answers and provides immediate feedback.
3. System calculates the score at the end of the quiz.

Postconditions:

- Quiz results and feedback are available to the user.

Use Case 7: Personalized Learning Dashboard

Name: Personalized Learning Dashboard
Description: Displays user's learning progress and recommendations based on activity.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User navigates to the dashboard.
2. System displays recent activities, scores, and recommendations based on past actions.

Postconditions:

- Dashboard displays recent activities, progress metrics, and content recommendations.

Use Case 8: Subscription Management

Name: Subscription Management
Description: Allows users to manage subscription plans and renewals.
Primary Actor: User
Preconditions:

- User is logged in with a free or active subscription.

Steps:

1. User views subscription options and selects a plan.
2. System verifies payment details and applies any coupon codes.
3. User's subscription status is updated, granting access to premium features if applicable.

Postconditions:

- User subscription is updated, and access is granted accordingly.

Use Case 9: User Dashboard

Name: User Dashboard
Description: Centralises access to recent activities, notifications, and personalized recommendations.

Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User accesses the dashboard from the home screen.
2. System retrieves and displays recent activities, notifications, and recommendations.

Postconditions:

- User views updates, activities, and recommendations tailored to their profile.

Use Case 10: Referral Link and Rewards Management

Name: Referral Link and Rewards Management
Description: Allows users to refer friends and earn rewards.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User generates and shares a referral link.
2. System tracks link usage and rewards users for successful referrals.

Postconditions:

- Rewards are added to the user's account if referrals meet set criteria.

Use Case 11: Content Bookmarking and Favorites

Name: Content Bookmarking and Favorites
Description: Allows users to save content for future reference.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User selects content to bookmark.
2. System saves the content under the user's profile for quick access.

Postconditions:

- Content is accessible from the user's bookmarked section.

Use Case 12: Analytics and Reporting

Name: Analytics and Reporting
Description: Provides insights into user engagement and learning progress.
Primary Actor: User / Admin
Preconditions:

- User or admin has reporting access.

Steps:

1. User or admin views the analytics dashboard.
2. System generates and displays reports on engagement and performance metrics.

Postconditions:

- Analytics and reports are available for user review.

Use Case 13: Notification System

Name: Notification System
Description: Sends alerts to users for platform updates, content releases, and other events.
Primary Actor: User
Preconditions:

- User is logged in and notifications are enabled.

Steps:

1. System detects a trigger event (e.g., new content, quiz result).
2. Notification is sent via in-app message, email, or SMS.

Postconditions:

- Users receive notifications relevant to their account.

Use Case 14: Content Search and Filtering

Name: Content Search and Filtering
Description: Enables users to search and filter educational content easily.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User enters search terms or selects filters.
2. System retrieves and displays relevant content.

Postconditions:

- Filtered search results are displayed to the user.

Use Case 15: In-App Support and Help Desk

Name: In-App Support and Help Desk
Description: Offers help options such as chat support and FAQ.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. Users access help or support.
2. System displays FAQ or connects to chat support.

Postconditions:

- Users receive support or relevant FAQ answers.

Use Case 16: Admin User Management

Name: Admin User Management
Description: Allows administrators to manage user accounts.
Primary Actor: Admin
Preconditions:

- Admin is logged in with the appropriate permissions.

Steps:

1. Admin selects a user account to manage.
2. Admin performs actions (e.g., update, suspend, delete).

Postconditions:

- User account status is updated accordingly.

Use Case 17: Admin Content Management

Name: Admin Content Management
Description: Allows administrators to manage platform content.
Primary Actor: Admin
Preconditions:

- Admin is logged in.

Steps:

1. Admin selects content to edit or approve.
2. System displays content details; the admin makes necessary updates.

Postconditions:

- Content is updated or approved for user access.

Use Case 18: Interactive Case Review

Name: Interactive Case Review
Description: Enables educators to review and assess user performance in case scenarios.
Primary Actor: Educator/Admin
Preconditions:

- Educators have access to case review tools.

Steps:

1. Educator selects a case review.
2. System shows user responses; educators provide feedback.

Postconditions:

- Educator feedback is saved and accessible to the user.

Use Case 19: Usage Analytics Dashboard

Name: Usage Analytics Dashboard
Description: Displays metrics on platform engagement for admin review.
Primary Actor: Admin
Preconditions:

- Admin has access

to the analytics dashboard.

Steps:

1. Admin views metrics and selects specific reports.
2. System displays engagement trends and data visualizations.

Postconditions:

- Admin reviews and analyzes platform usage metrics.

Use Case 20: Coupon Management

Name: Coupon Management
Description: Enables admins to create and manage discount coupons.
Primary Actor: Admin
Preconditions:

- Admin is logged in and has coupon management permissions.

Steps:

1. Admin creates a new coupon with restrictions (expiry date, max uses).
2. System validates and saves the coupon; users apply it at checkout.

Postconditions:

- Coupon is available for eligible users during subscription checkout.

Use Case 21: Coupon Management for Subscription

Name: Coupon Management for Subscription
Description: Allows admins to create and manage coupons for subscription discounts with restrictions such as user limit, expiration date, single-use, and specific subscription eligibility.
Primary Actor: Admin, User
Preconditions:

- Admin is logged in with permissions for coupon management.
- Coupon is configured for one specific subscription.

Steps:

1. **Create Coupon:**
    a. Admin inputs coupon code, user restrictions (e.g., max users or specific users), and expiration date.
    b. Admin specifies applicable subscription for the coupon.
2. **User Redemption:**

a. User enters the coupon code during subscription purchase.
b. System verifies coupon validity, checks user eligibility, and applies the discount if conditions are met.

Postconditions:
- Coupon is applied if valid, or error is displayed if invalid or expired.

Use Case 22: Content Moderation by Admin

Name: Content Moderation
Description: Admins review and manage user-uploaded content to ensure adherence to platform guidelines.
Primary Actor: Admin
Preconditions:
- Admin is logged in.

Steps:
1. Admin accesses content moderation panel.
2. Admin reviews content and selects actions (approve, edit, or delete).
3. System updates content visibility based on the admin's decision.

Postconditions:
- Content is available or removed based on admin actions.

Use Case 23: Subscription Renewal Notification

Name: Subscription Renewal Notification
Description: Notifies users about upcoming subscription renewals.
Primary Actor: User
Preconditions:
- Users have an active subscription approaching expiration.

Steps:
1. System detects an approaching subscription expiration.
2. System sends an in-app notification and/or email to the user.
3. User selects notification to review renewal options.

Postconditions:
- Users are informed about renewal and can take action as needed.

Use Case 24: In-App Support Ticket Submission

Name: Support Ticket Submission
Description: Users submit a support ticket for platform assistance.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:

1. User accesses support and submits a ticket with issue details.
2. System logs and prioritizes the ticket.
3. Admin reviews and responds.

Postconditions:
- Users receive response and resolution options for their issue.

Use Case 25: Admin Dashboard Access

Name: Admin Dashboard Access
Description: Allows admins to access a dashboard with system metrics, user activity, and subscription data.
Primary Actor: Admin
Preconditions:
- Admin is logged in.

Steps:
1. Admin navigates to the dashboard.
2. System displays metrics and analytics relevant to user activity and content management.

Postconditions:
- Admin accesses the platform's key metrics.

Use Case 26: Recommendation System for Learning Content

Name: Learning Content Recommendations
Description: System suggests content based on the user's recent activities and interests.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. System analyzes the user's recent interactions.
2. System recommends similar or relevant content on the dashboard.

Postconditions:
- Users receive personalized content recommendations.

Use Case 27: Account Deactivation Request

Name: Account Deactivation Request
Description: Users can deactivate their accounts from the settings page.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. User navigates to account settings and selects "Deactivate Account."

2. System prompts for confirmation and deactivates the account upon user confirmation.

Postconditions:
- User account is deactivated and inaccessible.

Use Case 28: View MCQ Progress History

Name: MCQ Progress History
Description: Allows users to view their history and scores of completed MCQ quizzes.
Primary Actor: User
Preconditions:
- User has taken at least one MCQ quiz.

Steps:
1. Users access MCQ history from their profile.
2. System displays past quizzes, scores, and feedback.

Postconditions:
- Users can review and assess their performance.

Use Case 29: Bookmarking Content

Name: Bookmarking Content
Description: Allows users to bookmark notes, files, or cases for future reference.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. User selects “Bookmark” on desired content.
2. System adds the item to the user’s bookmarked list.

Postconditions:
- Content is accessible in the user’s bookmark section.

Use Case 30: Edit Personal Profile Information

Name: Edit Profile Information
Description: Allows users to update their profile details, such as email or profile picture.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. User navigates to profile settings and selects "Edit Profile."
2. Users update relevant information.
3. System saves changes upon confirmation.

Postconditions:

- User profile information is updated successfully.

Use Case 31: Content Tagging and Organization

Name: Content Tagging
Description: Allows admins to tag and organize content by topics, making it easier for users to search.
Primary Actor: Admin
Preconditions:

- Admin is logged in.

Steps:

1. Admin selects content and adds tags.
2. System categorizes content based on the tags.

Postconditions:

- Content is organized and searchable by tags.

Use Case 32: Access Usage Analytics for Admins

Name: Usage Analytics Access
Description: Allows admins to view usage analytics to monitor user engagement.
Primary Actor: Admin
Preconditions:

- Admin is logged in.

Steps:

1. Admin accesses the analytics section.
2. System displays usage metrics (e.g., active users, content interactions).

Postconditions:

- Admin reviews analytics data.

Use Case 33: Generate Referral Link for User Rewards

Name: Referral Link Generation
Description: Generates unique referral links for users to share. Rewards are provided for successful referrals.
Primary Actor: User
Preconditions:

- User is logged in and has no active referral link.

Steps:

1. User requests a referral link.
2. System generates and displays a unique link.

Postconditions:

- Referral link is generated and available for sharing.

Use Case 34: Enable/Disable In-App Notifications

Name: Manage In-App Notifications
Description: Users can manage notification preferences.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User accesses notification settings.
2. User toggles notification preferences.

Postconditions:

- Notification settings are updated based on user preferences.

Use Case 35: Content Flagging and Reporting

Name: Flag Content
Description: Allows users to flag inappropriate or inaccurate content.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User selects “Flag” on content and provides a reason.
2. System logs report for admin review.

Postconditions:

- Content report is created for administrative review.

Use Case 36: Subscription Referral Link Creation and Management

Name: Subscription Referral Link Creation and Management
Description: Allows admins to generate referral links for employees and track their effectiveness in bringing new subscriptions.
Primary Actor: Admin
Preconditions:

- Admin is logged in and has permissions to manage subscriptions.

Steps:

1. Admin accesses the subscription management page.
2. Admin selects "Create Referral Link" and enters necessary details.
3. System generates and displays the referral link.
4. Admin tracks referral usage statistics.

Postconditions:

- Referral link is created, and referral statistics are available.

Use Case 37: Admin Review of User Comments

Name: Admin Review of User Comments
Description: Admins manage user comments on shared files, including the ability to delete inappropriate comments.
Primary Actor: Admin
Preconditions:

- Admin is logged in with comment management permissions.

Steps:

1. Admin navigates to the comments section of a shared file.
2. Admin reviews user comments and selects inappropriate ones.
3. System prompts for confirmation before deletion.
4. Admin confirms, and system deletes the selected comments.

Postconditions:

- Comments are moderated, maintaining the quality of interactions.

Use Case 38: User Grades Entry on Dashboard

Name: User Grades Entry on Dashboard
Description: Allows users to enter and track their academic grades within the platform.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. Users navigate to the "Grades" section on their dashboard.
2. User inputs grades for subjects/modules in a valid format.
3. System verifies entries and saves them to the user’s profile.

Postconditions:

- User grades are saved and displayed on the dashboard.

Use Case 39: File Bulk Download by User

Name: File Bulk Download
Description: Enables users to download multiple files at once based on selected filters (e.g., by subject, university).
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User selects filters for desired files (e.g., by subject or university).
2. System displays the list of matching files.
3. User selects "Download All" to initiate bulk download.

Postconditions:

- Selected files are downloaded in bulk for user access.

Use Case 40: Admin User Account Deactivation

Name: Admin User Account Deactivation
Description: Allows admins to deactivate user accounts as needed.
Primary Actor: Admin
Preconditions:
- Admin is logged in with account management permissions.

Steps:
1. Admin accesses the user management page.
2. Admin selects a user account and chooses "Deactivate."
3. System confirms action; admin confirms deactivation.

Postconditions:
- User account is deactivated, preventing further access.

Use Case 41: User Reports File Issue

Name: Report File Issue
Description: Users report issues with files they have accessed.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. User selects a file and chooses "Report Issue."
2. User provides details of the issue.
3. System records and forwards the report to the admin for review.

Postconditions:
- Issue report is submitted for admin review.

Use Case 42: In-App Content Search with Filters

Name: Content Search with Filters
Description: Users search and filter content within the platform.
Primary Actor: User
Preconditions:
- User is logged in.

Steps:
1. User enters search terms and selects filters (e.g., by subject, university).
2. System retrieves and displays results based on the filters.

Postconditions:
- Filtered content results are displayed to the user.

Use Case 43: Content Rating by User

Name: Content Rating
Description: Allows users to rate shared files or case studies.
Primary Actor: User
Preconditions:

- User is logged in and has accessed the content.

Steps:

1. User selects "Rate" on the content page.
2. User provides a rating (e.g., 1–5 stars).
3. System records the rating and updates the content's average rating.

Postconditions:

- User rating is saved, and the content's overall rating is updated.

Use Case 44: User Profile Picture Upload

Name: Profile Picture Upload
Description: Users can upload or update their profile picture.
Primary Actor: User
Preconditions:

- User is logged in.

Steps:

1. User accesses profile settings and selects "Upload Profile Picture."
2. System validates file format and size.
3. System saves and displays the new profile picture.

Postconditions:

- Profile picture is updated successfully.

Use Case 45: Admin Review of File Reports

Name: Admin Review of File Reports
Description: Admin reviews user-reported issues with files and resolves them.
Primary Actor: Admin
Preconditions:

- Admin is logged in and has file management permissions.

Steps:

1. Admin accesses the file report section.
2. Admin reviews reports and chooses appropriate actions (e.g., delete, notify user).
3. System implements the admin's selected actions.

Postconditions:

- File reports are addressed, and actions are recorded.

Use Case 46: Apply Coupon for Subscription

Name: Coupon Management
Description: User applies a coupon code for subscription discounts.
Primary Actor: User
Preconditions:
- User has a valid coupon.

Steps:
1. User selects a subscription and enters the coupon code.
2. System validates the code and applies the discount.

Postconditions:
- Discounted subscription price is displayed.

6. Acceptance Criteria

1. **User Registration and Login**

- **Functionality**: Enable users to register, log in, and reset passwords.
- **Usability**: Forms should be intuitive, with field validation and feedback for errors.
- **Performance**: Login and registration should process within 1-2 seconds.
- **Security**: Secure login credentials and prevent brute-force attacks.
- **Validation**: Verify email format, password length, and provide error messages for invalid entries.

2. **User Profile Management**

- **Functionality**: Allow users to update personal information.
- **Usability**: Editable fields should be accessible with guided tooltips.
- **Performance**: Profile updates should complete within 1 second.
- **Security**: Protect sensitive information in transit and at rest.
- **Validation**: Validate email, phone number, and profile picture format.

3. **Notes Access**

- **Functionality**: Provide access to curated study notes by topic.
- **Usability**: Notes should be easily searchable and bookmarkable.
- **Performance**: Notes load within 1 second.
- **Security**: Restrict access to premium notes based on subscription level.
- **Validation**: Validate bookmark actions and ensure smooth navigation.

4. **File Sharing System**

- **Functionality**: Enable file upload and sharing with metadata categorization.
- **Usability**: Clear instructions for uploading and searching files.
- **Performance**: Uploads and file access should complete within 3-5 seconds.
- **Security**: Validate file types and sizes to prevent malicious uploads.
- **Validation**: Ensure that only approved files are accessible to others.

5. **Problem-Based Learning (PBL)**

- **Functionality**: Provide interactive medical case scenarios for learning.
- **Usability**: Cases should progress in stages with clear user prompts.
- **Performance**: Load each case stage within 2 seconds.
- **Security**: Track user activities securely for case completion.
- **Validation**: Validate user inputs at each stage and provide immediate feedback.

6. **Multiple-Choice Questions (MCQs) System**

- **Functionality**: Present quizzes with real-time scoring and feedback.
- **Usability**: Clear quiz layout and explanations for answers.
- **Performance**: Each response should process within 1 second.
- **Security**: Store quiz results securely to ensure user privacy.

- **Validation**: Validate selected answers and score accuracy.

7. **Personalized Learning Dashboard**

- **Functionality**: Display user-specific progress and recommendations.
- **Usability**: Organized, accessible layout for recent activities.
- **Performance**: Dashboard loads within 1-2 seconds.
- **Security**: Restrict dashboard access to the logged-in user.
- **Validation**: Validate bookmarks and recently accessed content.

8. **Subscription Management**

- **Functionality**: Manage subscriptions with payment processing.
- **Usability**: Show detailed information on subscription options.
- **Performance**: Transactions should process within 3-5 seconds.
- **Security**: PCI-compliant storage and encryption of payment information.
- **Validation**: Verify coupon codes, payment details, and display errors.

9. **User Dashboard**

- **Functionality**: Centralize access to activities, bookmarks, and notifications.
- **Usability**: Easy-to-navigate layout with recent activities displayed.
- **Performance**: Load within 1-2 seconds.
- **Security**: Ensure data visibility only to the user.
- **Validation**: Validate each user action and show updates accordingly.

10. **Referral Link and Rewards Management**

- **Functionality**: Generate referral links and reward successful referrals.
- **Usability**: Simplified referral generation with share options.
- **Performance**: Link generation takes less than 1 second.
- **Security**: Track referral activity to prevent misuse.
- **Validation**: Validate rewards and conditions for eligibility.

11. **Content Bookmarking and Favorites**

- **Functionality**: Allow users to bookmark content for future access.
- **Usability**: Bookmarked content should be easy to find and retrieve.
- **Performance**: Bookmarks load within 1 second.
- **Security**: Ensure access only to the logged-in user.
- **Validation**: Prevent duplicate bookmarks and confirm success.

12. **Analytics and Reporting**

- **Functionality**: Generate reports on user engagement and progress.
- **Usability**: Accessible report interface with filtering options.
- **Performance**: Reports generated within 3 seconds.
- **Security**: Protect analytics data and limit access by role.
- **Validation**: Ensure data accuracy and meaningful metrics.

13. **Notification System**

- **Functionality**: Notify users of platform updates and subscriptions.
- **Usability**: Allow users to toggle notification preferences.
- **Performance**: Send notifications within 1 second of trigger.
- **Security**: Send sensitive notifications over secure channels.
- **Validation**: Validate notification triggers and preferences.

14. **Content Search and Filtering**

- **Functionality**: Search and filter educational content by various criteria.
- **Usability**: Simple, clear search bar and filter options.
- **Performance**: Results load within 1 second.
- **Security**: Protect search results based on user permissions.
- **Validation**: Display accurate results and handle no-match scenarios.

15. **In-App Support and Help Desk**

- **Functionality**: Provide user support via in-app chat and FAQs.
- **Usability**: Support access should be clearly marked.
- **Performance**: Chat support should load within 1 second.
- **Security**: Protect user support data and track interactions.
- **Validation**: Log user issues and ensure timely responses.

16. **Admin User Management**

- **Functionality**: Enable admins to create, update, suspend, or delete users.
- **Usability**: Simple admin dashboard for quick actions on user accounts.
- **Performance**: Admin actions should complete within 1 second.
- **Security**: Role-based access for managing user permissions.
- **Validation**: Verify admin authority before modifying accounts.

17. **Admin Content Management**

- **Functionality**: Allow admins to review, approve, and manage content.
- **Usability**: Organized dashboard with clear action buttons for content review.
- **Performance**: Actions complete within 1 second.
- **Security**: Limit management capabilities to authorized admins.
- **Validation**: Ensure only approved content is displayed.

18. **Interactive Case Review**

- **Functionality**: Allow educators to review and provide feedback on cases.
- **Usability**: Feedback options should be clear and intuitive.
- **Performance**: Case data should load within 2 seconds.
- **Security**: Track educator activity and feedback securely.
- **Validation**: Confirm feedback saves successfully for future review.

19. **Usage Analytics Dashboard**

- **Functionality**: Provide a real-time admin dashboard for user analytics.
- **Usability**: Clear display of metrics, trends, and activity logs.
- **Performance**: Refreshes within 2 seconds.
- **Security**: Restrict access to authorized personnel.
- **Validation**: Validate accuracy and timeliness of displayed data.

20. **Coupon Management**

- **Functionality**: Admins can create and manage discount coupons for subscriptions.
- **Usability**: Admins should have a simple interface for coupon details.
- **Performance**: Coupon creation and validation should complete within 1 second.
- **Security**: Limit coupon visibility and validity based on defined criteria.
- **Validation**: Validate coupon constraints (e.g., expiration date, user limits) and prevent expired coupon use.

7. Assumptions and Constraints

8. Dependencies

8.1. Development Tools and Frameworks

These tools and frameworks support the core development and ongoing maintenance of the Meduron platform.

1. **Frontend Framework**
    - **Dependency**: A modern frontend framework, such as React or Angular, provides the structure for building the user interface, ensuring a responsive and interactive experience.
    - **Impact**: The framework enables rapid UI development and helps maintain a consistent look and feel across devices. Framework issues or compatibility limitations could affect the usability and accessibility of the platform.
2. **Backend Framework**
    - **Dependency**: A backend framework, such as NestJS (or Express), facilitates the development of server-side logic, user authentication, and data handling.
    - **Impact**: The backend framework supports efficient processing and integration with the database. Limitations or updates in the backend framework could impact overall platform performance.
3. **Database Management System**
    - **Dependency**: A reliable database system like PostgreSQL or MongoDB is required to store user data, content, and analytics securely.
    - **Impact**: Data integrity, availability, and retrieval speeds depend on the chosen database system. Any database issues could affect user data retrieval, performance, and analytics.
4. **Version Control System**
    - **Dependency**: Git and GitHub for code versioning, collaboration, and history tracking.
    - **Impact**: Ensures source code integrity, enables collaboration, and allows for rollback to previous versions if issues arise. Any service interruptions could disrupt the development workflow.
5. **UI Component Libraries**
    - **Dependency**: Component libraries (e.g., Material UI, Bootstrap) streamline UI development with pre-built design components.
    - **Impact**: Reduces frontend development time, maintaining consistent UI design. Any library updates or deprecations may impact the UI's consistency and functionality.

8.2. Third-Party Services and APIs

Meduron relies on external services and APIs to handle specific functions such as payments, communication, and analytics.

1. **Payment Gateway API**

    - **Dependency**: Services like Chapa to process user subscription payments securely.
    - **Impact**: Payment gateways facilitate secure transactions. Issues with these services could disrupt premium feature access.
2. **Email Service API**
    - **Dependency**: Email providers like SendGrid or Mailgun to send verification emails, notifications, and updates to users.
    - **Impact**: Account verification and user notifications depend on email delivery. Service interruptions could impact communication and user onboarding.
3. **Authentication API**
    - **Dependency**: OAuth, JWT, or Firebase Authentication for secure login and user session management.
    - **Impact**: Ensures secure user access to the platform. Downtime or issues with authentication providers could reduce security and limit access.
4. **Push Notification API**
    - **Dependency**: Services like Firebase Cloud Messaging (FCM) to send notifications for reminders and updates.
    - **Impact**: Enhances user engagement by delivering timely notifications. Any disruptions could reduce platform interactivity and user engagement.
5. **Analytics and Tracking API**
    - **Dependency**: Google Analytics or Mixpanel for tracking user behavior and engagement.
    - **Impact**: Provides insights into user engagement, enabling data-driven decisions for improvement. Lack of reliable analytics may reduce insight into platform performance and usage.

8.3. Project Management and Collaboration Tools

Effective collaboration and organization are essential for team productivity and project success.

1. **Project Management Tool**
    - **Dependency**: Tools like Jira or Trello for tracking project progress, assigning tasks, and managing deadlines.
    - **Impact**: Facilitates task management, milestone tracking, and team accountability. Unavailable project management tools could hinder coordination and visibility into project timelines.
2. **Collaboration and Documentation Tools**
    - **Dependency**: Platforms like Confluence or Notion for team documentation, knowledge sharing, and collaboration.
    - **Impact**: Allows centralized access to project documentation and guidelines, improving team collaboration. Documentation gaps may arise if these tools are inaccessible.
3. **Communication Tool**
    - **Dependency**: Real-time communication platforms like Slack or Microsoft Teams for team discussions, updates, and quick decision-making.

- **Impact**: Enables efficient communication and swift issue resolution. Outages could impact team coordination and response time to development issues.

8.4. Testing and Deployment

Testing frameworks and deployment tools ensure Meduron's reliability and readiness for use in a production environment.

1. **Automated Testing Framework**
   - **Dependency**: Testing frameworks like Jest, Mocha, or Cypress for unit, integration, and end-to-end testing.
   - **Impact**: Ensures the platform functions as expected, detecting issues early. Delayed or inaccurate testing results could lead to undetected bugs.
2. **Continuous Integration/Continuous Deployment (CI/CD)**
   - **Dependency**: Tools like Jenkins or GitHub Actions automate testing and deployment.
   - **Impact**: CI/CD tools support streamlined testing and deployment, enabling faster iterations and updates. CI/CD tool outages may delay development cycles and reduce the frequency of updates.
3. **Staging Environment**
   - **Dependency**: A staging environment to test new features and updates in a controlled setting before releasing to production.
   - **Impact**: Enables testing in an environment close to production, reducing potential bugs in live releases. Lack of a staging environment could lead to more production bugs and downtime.
4. **Data Backup and Recovery**
   - **Dependency**: Backup solutions for regularly storing copies of critical data and user information.
   - **Impact**: Protects against data loss due to system failures or accidental deletions. Lack of reliable backup could risk permanent data loss.

8.5. Legal and Compliance

Compliance tools and guidelines are critical for meeting legal standards, ensuring data protection, and managing user privacy.

1. **Data Protection Compliance**
   - **Dependency**: Compliance with regulations like GDPR and CCPA for data handling and user privacy rights.
   - **Impact**: Ensures that user data is managed in accordance with privacy laws, protecting the platform from legal issues. Non-compliance could result in fines or penalties.
2. **User Consent Management Tool**

- **Dependency**: Consent management for tracking user agreement to terms of service, privacy policies, and data usage.
- **Impact**: Maintains records of user consent, supporting transparency and regulatory compliance. Inadequate consent tracking could lead to privacy violations.

3. **Terms of Service and Privacy Policy Updates**
   - **Dependency**: Regular updates to terms and policies to reflect platform changes or legal requirements.
   - **Impact**: Ensures that users are informed of their rights and any data usage changes. Lack of updated policies could result in misunderstandings or legal issues.

9. Architecture Diagrams

System Architecture Diagram

- **Objective**:
  To illustrate the overall architecture of the Meduron platform, including the interaction between various components: React (frontend), NestJS (backend), Prisma (database management), and cloud services.
- **Components**:
  - **User Devices**:
    - **Mobile Devices**: Smartphones and tablets running web-based interfaces through browsers.
    - **Web Browsers**: Primarily desktop and mobile browsers accessing the Meduron platform.
    - **Features**: User registration, subscription management, accessing PBL cases, viewing notes, taking quizzes, and managing profile settings.
  - **Frontend (React)**:
    - **User Interface (UI)**: The primary interactive layer for users, handling navigation, content display, and responsive design.
    - **Learning Tools**: Components to display quizzes, case studies, and notes with interactive elements.
    - **API Calls**: Executes HTTP requests to interact with the backend (NestJS) for various functionalities like user authentication, content retrieval, and subscription management.
  - **Backend (NestJS)**:
    - **API Gateway**:
      - **Endpoints**: Provides API endpoints for user registration, authentication, subscription management, content delivery, and analytics.
      - **Authentication**: Manages user login and registration, utilizing JWT and OAuth for secure sessions.
    - **Business Logic**:
      - **Content Management**: Handles requests for educational materials, user data, bookmarks, and quizzes.
      - **Subscription Management**: Processes subscription status updates, payment confirmations, and access controls.
    - **Database Integration**:
      - **Database Server**: Stores user profiles, progress data, quizzes, and educational content.
      - **Database Type**: SQL (e.g., PostgreSQL) managed through Prisma ORM, allowing complex data queries and relationships.
  - **Cloud Services**:
    - **Cloud Storage**:
      - **Storage Solutions**: Services like AWS S3 or Google Cloud Storage are used to store user-uploaded files, notes, and other resources.

            - **Backup**: Regular backups of database and user data for reliability and disaster recovery.
        - **Compute Resources**:
            - **Server Instances**: Virtual machines or containers running backend and database services.
            - **Scalability**: Auto-scaling mechanisms in place to manage demand during peak usage periods.
        - **Authentication Services**:
            - **OAuth Providers**: Integration with providers like Google or Facebook to streamline the user registration and login process.
    - **External Systems**:
        - **Third-Party APIs**:
            - **Payment Gateways**: Services like Stripe or PayPal are integrated to handle secure transactions for user subscriptions.
            - **Email Service**: Providers like SendGrid or Mailgun manage email-based user notifications, verification, and password recovery.
            - **Analytics Platform**: Services such as Google Analytics track user behaviour and engagement, providing insights into content usage and platform performance.

Diagram Layout:

```
[User Devices]
      |
      V
[Frontend (React)]
      |
      V
[Backend (NestJS)] <--> [Database (Prisma with PostgreSQL)]
      |
      V
[Cloud Storage]
      |
      V
[External Systems]
```

- **User Devices** interact with the **Frontend (React)**, enabling users to access Meduron's features.
- **Frontend (React)** communicates with the **Backend (NestJS)** via API calls, handling user interactions and data retrieval.
- **Backend (NestJS)** manages business logic and interacts with the **Database (Prisma with PostgreSQL)** for storing user data, content, and progress.
- **Backend (NestJS)** also stores and retrieves data from **Cloud Storage** for files and educational resources.
- **External Systems** such as payment gateways, email providers, and analytics platforms support additional functionalities for user subscriptions, notifications, and engagement tracking.

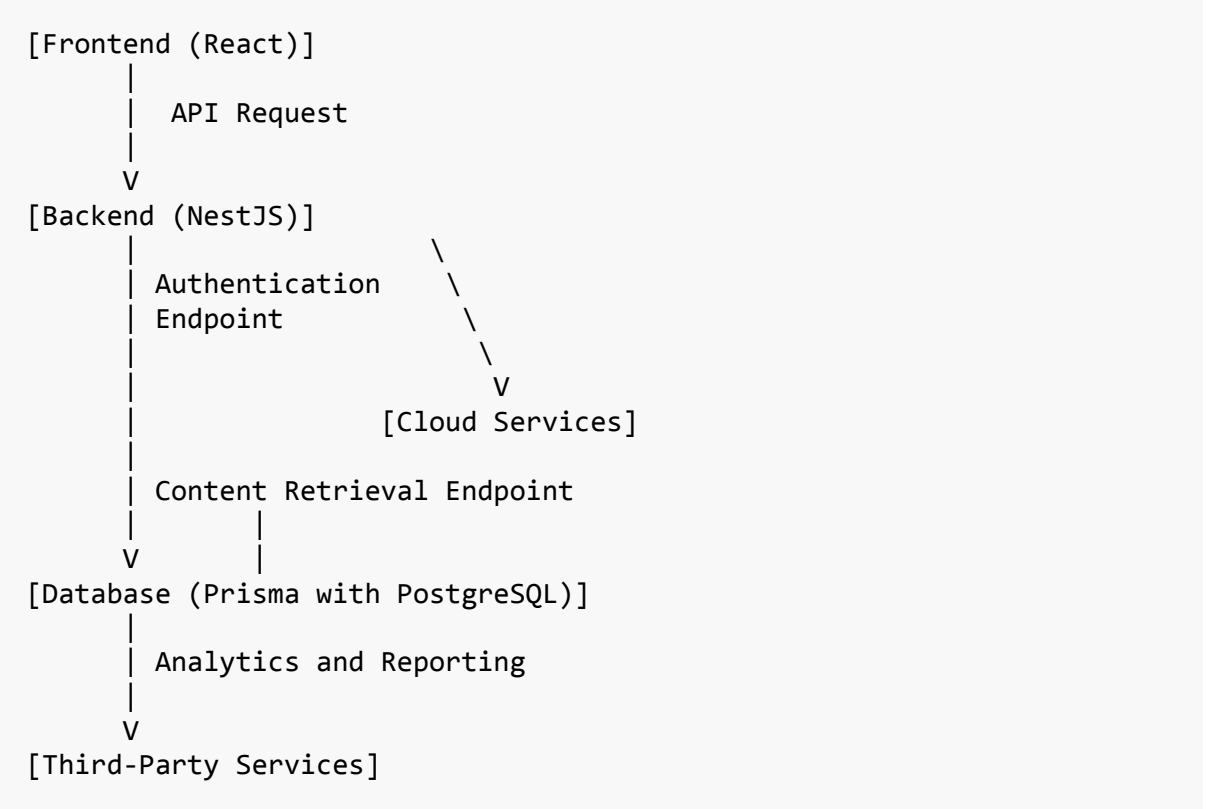
10. Integration Diagrams

API Integration Diagram

- **Objective**:
  To illustrate how different components of the Meduron platform communicate through APIs, including endpoints for user authentication, data retrieval, content management, and payment processing.
- **Components**:
  - **Frontend (React)**:
    - **UI Layer**: Provides user interactions, such as registering accounts, accessing educational content, and managing subscriptions.
    - **API Calls**: Executes HTTP requests to the backend for various operations, such as login, retrieving study materials, and submitting quiz answers.
  - **Backend (NestJS)**:
    - **API Gateway**: Manages incoming requests from the frontend and routes them to the appropriate services or databases.
    - **Endpoints**:
      - **Authentication**: Endpoint for user login, registration, password reset, and multi-factor authentication.
      - **Content Retrieval**: Endpoint for fetching user-specific content, including PBL cases, notes, and quizzes.
      - **Subscription Management**: Endpoint for processing user subscriptions, managing payments, and updating access levels.
      - **Analytics and Reporting**: Endpoint to retrieve or submit user activity and progress data for reporting and analytics.
  - **Database (Prisma with PostgreSQL)**:
    - **User Data Storage**: Stores user profiles, progress, subscription status, and bookmarks.
    - **Content Storage**: Holds data on educational content, quizzes, and PBL cases, retrieved by the backend when requested by the frontend.
  - **Cloud Services**:
    - **Storage Service**:
      - **Data Storage and Retrieval**: API for storing and retrieving files and media content, such as notes and user-uploaded materials.
    - **Authentication Service**:
      - **OAuth Providers**: Integration with external providers like Google or Facebook for user authentication and registration.
  - **Third-Party Services**:
    - **Payment Gateway**: Provides API for handling payments and managing user subscription transactions.
    - **Email Service**: Manages user notifications and communication, including account verification, password recovery, and platform updates.

- **Analytics Platform**: Tracks user behaviour and engagement metrics for content optimization and platform improvement.

Diagram Layout:



- **Frontend (React)**:
  - Sends requests to the **Backend (NestJS)** via various API endpoints to perform actions like user authentication, retrieving educational content, and updating subscriptions.
- **Backend (NestJS)**:
  - **Authentication Endpoint**: Manages login, registration, and session management, utilizing OAuth from **Cloud Services** when required.
  - **Content Retrieval Endpoint**: Fetches user-specific content and materials from the **Database**.
  - **Analytics and Reporting Endpoint**: Tracks and submits user activity to **Third-Party Analytics Platforms** for insights into platform usage.
  - **Subscription Management Endpoint**: Processes payments with **Payment Gateway** integration, managing user access levels and subscription status updates.
- **Database (Prisma with PostgreSQL)**:
  - Stores user information, educational content, and activity logs, retrieving data when requested by the backend.
- **Cloud Services**:
  - Provides file storage and retrieval for user-uploaded content and media resources.

    - **Authentication Service** handles third-party logins through OAuth, supporting login and registration via external providers.
- **Third-Party Services**:
    - **Payment Gateway**: Processes payments and subscriptions.
    - **Email Service**: Sends critical notifications, including account verification and password recovery.
    - **Analytics Platform**: Monitors and tracks user engagement metrics for content optimization.

Revision History

Version	Date	Description	Author
1.0	Nov 10	Initial draft	haptome2@gmail.com
1.1	Nov 11	Admin persona,Offline Performance,other security compliance	haptome2@gmail.com
1.2	Nov 13	review	leulykpro@gmail.com