BANNARI AMMAN INSTITUTE OF TECHNOLOGY

An AutonomousInstitution Affiliated to Anna University - Chennai, Accredited by NAAC with A+ Grade
Sathyamangalam - 638401 Erode District, Tamil Nadu, India

Student Name: DHARANI R R

Seat No: 271

Project ID: 17

Project title: ONE CREDIT COURSE EXEMPTION

COMPONENT	TECH STACK
Backend	NODE.js
Frontend	REACT.js
Database	MongoDB
API	REST Ful API

Implementation Timeline

Phase	Deadline	Status	Notes
Stage 1	25/07/2024	Completed	Planning and Requirement Gathering
Stage 2		In progress	Design and UI/UX Prototyping
Stage 3		Not started	Database Design and Implementation
Stage 4		Not started	Backend Development
Stage 5		Not started	Integration and Testing
Stage 6		Not started	Deployment

1. Introduction:

1.1 Purpose:

The purpose of this document is to provide a detailed description of the requirements for the "Complete Three One-Credit Courses for Exemption" application. This application will be developed using the MERN stack, which includes Node.js for the backend, React.js for the frontend, MongoDB for the database, and RESTful API for the API.

1.2 Scope:

The "Complete Three One-Credit Courses for Exemption" application is designed to facilitate students in completing three one-credit courses to qualify for an exemption in one course. The application will allow students to track their progress, access course materials, and submit assignments. It will also enable faculty to manage courses, monitor student progress, and approve exemptions.

1.3 Definitions, Acronyms, and Abbreviations:

- **SRS:** Software Requirements Specification
- **MERN:** MongoDB, Express.js, React.js, Node.js
- **REST:** Representational State Transfer
- **API:** Application Programming Interface
- **Node.js:** A JavaScript runtime built on Chrome's V8 JavaScript engine
- **React.js:** A JavaScript library for building user interfaces
- **MongoDB:** A NoSQL database program
- **Backend:** The server-side part of the application
- **Frontend:** The client-side part of the application

2. Overall Description:

2.1 Product Perspective:

The "Course Exemption Application" will be a standalone web application that interfaces with a database to manage course exemptions. This application will allow students to complete three one-credit courses to get an exemption in one full course.

2.2 User Classes and Characteristics:

1. **Students:** Users who are students at the college, seeking to complete three one-credit courses for an exemption in one full course.
2. **Admin:** Users who are faculty members at the college, responsible for approving course completions and managing exemptions.

2.3 Operating Environment

The application will be accessed through a web browser on both desktop and mobile devices.

2.4 Design and Implementation Constraints:

The application will be developed using the MERN stack:

1. **MongoDB:** For database management to store user data, course information, and exemption records.
2. **Express.js:** For the backend framework to handle server-side logic and RESTful API implementation.
3. **React.js:** For the frontend framework to create an interactive and responsive user interface.
4. **Node.js:** For running the backend server and executing server-side scripts.

3. Specific Requirements

3.1 Functional Requirements

1. **User Registration:** Users should be able to register for an account.
2. **User Login:** Registered users should be able to log in to the application.
3. **Course Enrollment:** Users should be able to enroll in available courses.
4. **Course Completion Tracking:** Users should be able to track the completion status of their enrolled courses.
5. **Course Exemption Request:** Users should be able to request an exemption for a course once they have completed three one-credit courses.
6. **Admin Panel:** Admin users should be able to approve or reject course exemption requests.
7. **Notifications:** Users should receive notifications via email or text message once their exemption request is approved or rejected.
8. **Course Progress Display:** Users should be able to view their progress in enrolled courses in a user-friendly format.

3.2 Non-functional Requirements

1. **Performance:** The application should be able to handle multiple concurrent users without significant slowdowns.
2. **Security:** User data should be stored securely and protected against unauthorized access.
3. **Usability:** The user interface should be intuitive and easy to use.
4. **Reliability:** The application should be reliable and available whenever users need to access it.

3.3 User Interfaces

- **Login Page:** Allows users to log in to their accounts.
- **Course Enrollment Page:** Allows users to view and enroll in available courses.
- **Course Progress Page:** Allows users to view their progress in enrolled courses.

- **Course Exemption Request Page:** Allows users to request an exemption for a course.
- **Admin Panel:** Allows admin users to approve or reject course exemption requests.

3.4 System Interfaces

- **Database:** The application will interface with a MongoDB database to store user, course, and enrollment information.

4. Application Features

4.1 User-side Features

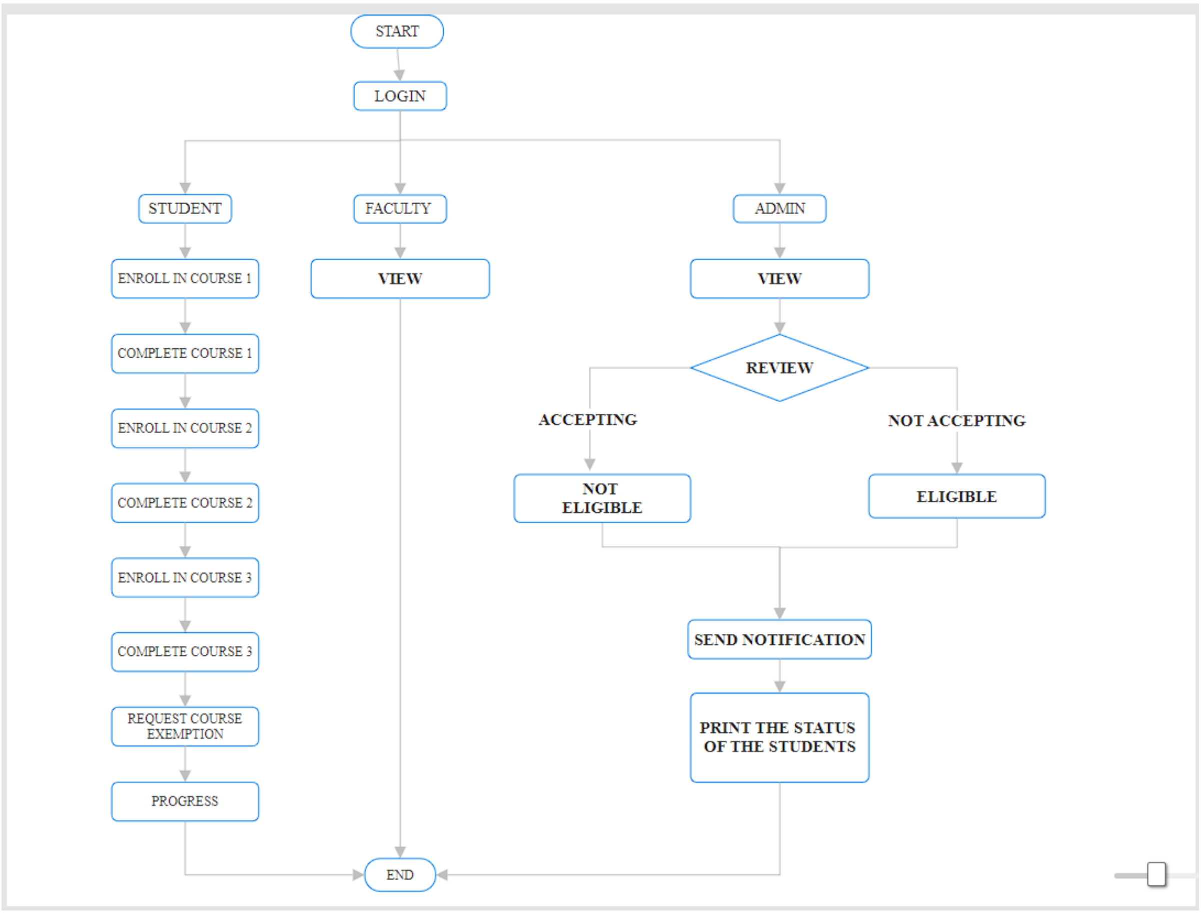
- **Course Enrollment:** Allow users to enroll in available courses, track their progress, and view course details such as description, duration, and credits.
- **Sort and Filter Courses:** Enable users to sort and filter courses based on various criteria such as category, duration, and credit points.
- **Bulk Booking (Modified**): Allow users to book multiple classes within a course or across different courses simultaneously, selecting specific start and end dates for each booking.
- **Progress Tracking:** Provide users with the ability to track their progress in each enrolled course, including completed modules and remaining tasks.
- **Editing and Modification:** Allow users to edit or modify their course enrollment details, such as changing the course start date, or time, if they made a mistake or need to make changes.
- **Booking Approval Status**: Display the status of the course enrollment request, indicating whether it is pending, approved, or rejected. Users can track the progress of their enrollment requests.
- **Notification on Approval:** Send a notification to the user when their course enrollment request is approved by the admin, confirming their successful registration.
- **Course Completion and Exemption Status:** Allow users to view the status of their course completions and see if they qualify for an exemption based on the completion of three one-credit courses.

4.2 Admin-Side Features

- **Course Management:** Enable admins to create, update, and manage courses, including setting course details such as title, description, duration, and credit points.
- **Enrollment Management:** Enable admins to handle course enrollment requests, reviewing and approving them based on the availability and user eligibility.
- **Progress Review:** Allow admins to monitor the progress of users in their enrolled courses, ensuring they meet the necessary criteria for course completion.
- **Notification on Enrollment Approval:** Notify users when their course enrollment request is approved, confirming their successful registration.
- **Course Availability Management:** After approval, automatically mark the enrolled courses as unavailable for other users during the enrolled time slots, preventing double enrollments.

- **Exemption Management:** Provide admins with the ability to review and approve or reject exemption requests from users who have completed three one-credit courses.
- **Booking Details and Reason Review:** Admins can view the enrollment details and the reason provided by the user for enrolling in the courses, ensuring the enrollments align with college policies and regulations.

5. FLOW CHART



6. Technology Stack

6.1 Backend

- **Node.js:** Used for server-side logic and handling HTTP requests.
- **Express.js:** Framework for building the backend API and handling routing.

6.2 Frontend

- **React.js:** Used for building the user interface, providing a dynamic and responsive experience for users.

6.3 Database

- **MongoDB:** NoSQL database used for storing resource information, booking history, and user data.

6.4 Additional Libraries and Components

- **React Router:** For client-side routing in the React.js application, enabling navigation between different views.
- **Material-UI:** React component library for building a modern and visually appealing user interface.
- **Full Calendar:** JavaScript calendar library integrated into the system for displaying course schedules, deadlines for course completions, and important dates for exemption requests.

7. Role of Each Component:

7.1 Node.js and Express.js

- **Backend Logic:** Handle user authentication, course completion tracking, and interactions with the database. Manage the logic for determining course exemptions based on completed courses.
- **API Development:** Create RESTful APIs for communication between the frontend and backend. These APIs will handle user data, course enrollment, course completion status, and exemption eligibility.

7.2 React.js

- **User Interface:** Build a dynamic and interactive user interface for managing course enrollments, tracking course progress, and applying for exemptions.
- **State Management:** Manage application state and data flow using React's state and context APIs, ensuring smooth interaction between various components and real-time updates.

7.3 MongoDB

- **Database Storage**: Store resource information, booking history, and user data in a flexible and scalable manner.

7.4 React Router

- **Client-Side Routing:** Enable navigation between different views in the application, such as course lists, user profiles, progress tracking, and exemption status, without a page reload.

7.5 Material-UI

- **UI Components:** Use pre-built components for designing a modern and visually appealing user interface and feel throughout the application.

7.6 Full Calendar

- **Course Schedule:** Display course schedules, completion deadlines, and key dates in a calendar format for easy visualization, helping users manage their course timelines effectively.