# AWT Tasks Report

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**

(Information and Technology Engineering)


**Submitted by:**

Amrinderdeep Singh

2203796 (URN)

**Submitted to:**

Dr Akshay Girdhar

DEPARTMENT OF INFORMATION & TECHNOLOGY ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

**Task 1:  Simple Text Editor Website**

**1. Introduction**

This project entails the development of a basic web-based text editor. The primary goal of this project is to provide a simple, intuitive platform where users can input and edit text with essential formatting options. The project is designed using HTML, CSS, JavaScript, and Bootstrap for styling.

**2. Objectives**

- To create an editable text area for users to type and format text.
- To implement basic text formatting features like Bold, Italic, Underline, Undo, and Redo.
- To use minimal tools and libraries, keeping the project simple and accessible.
- To design a responsive and visually appealing layout using Bootstrap.

**3. Technology Stack**

- **Frontend**: HTML, CSS, JavaScript
- **Styling Framework**: Bootstrap 5

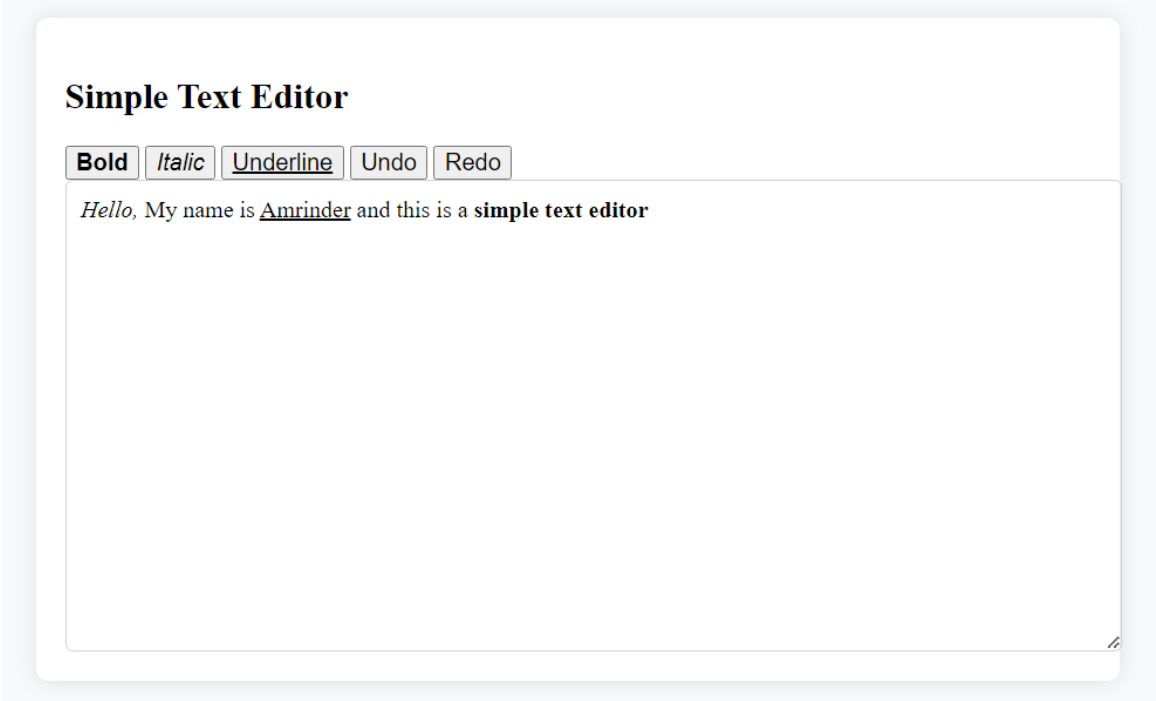
**4. Features**

- **Text Formatting Options**: Users can format the text with options for:
    - **Bold**: Makes the selected text bold.
    - **Italic**: Italicizes the selected text.
    - **Underline**: Underlines the selected text.
    - **Undo/Redo**: Allows users to undo or redo recent actions.
- **Responsive Design**: The editor is styled using Bootstrap, ensuring it is responsive and user-friendly across various screen sizes.

**5. Implementation**

The project is structured as follows:

- **HTML**: Defines the structure with a heading, toolbar, and editable text area.
- **CSS**: Adds custom styling for layout and visual enhancements, including padding, border-radius, and shadow effects.
- **JavaScript**: Uses the `execCommand` method to perform text formatting actions.
- **Bootstrap**: Provides styling for buttons, layout alignment, and responsive behavior.




**6. User Interface**

The user interface consists of:

- A centered heading labeled "Simple Text Editor."
- A toolbar with buttons for text formatting, styled with Bootstrap for consistency.
- A large editable text area where users can type and apply formatting options.

**7. Advantages**

- **User-Friendly**: Simplistic and easy-to-navigate interface with basic yet essential formatting options.
- **Responsive Design**: Adapts to different screen sizes, making it accessible on mobile, tablet, and desktop devices.
- **Minimal Dependencies**: Relies only on HTML, CSS, JavaScript, and Bootstrap, making it lightweight.

**8. Conclusion**

The Simple Text Editor is an effective tool for quick text editing, suitable for users who need basic formatting capabilities without additional complexity. The project demonstrates how basic web technologies can be used to create functional and responsive web applications. This editor could serve as a foundation for more advanced features or integration into larger web applications.

**Task 2: Seating Allocation System**

**1. Introduction**

This project is a web-based application designed to simulate a seat selection process for a movie theatre, allowing users to select seats based on their roll number. The allocation rules are simple: users with even-numbered roll numbers can select even-numbered seats, while users with odd-numbered roll numbers can select odd-numbered seats. This restriction ensures a unique experience for each user while maintaining an organized seating arrangement.

**2. Objectives**

- To enable users to select available seats based on their roll number.
- To create an interactive and visually engaging interface that simulates real-time seat selection.
- To enforce seat allocation rules based on even and odd roll numbers.
- To store and display the selected seats for the user before finalizing the selection.

**3. Technology Stack**

- **HTML**: Defines the structure of the page.
- **CSS**: Provides styling for the interface, enhancing user experience by clearly differentiating available, unavailable, and selected seats.
- **JavaScript**: Implements the seating logic, seat selection functionality, and interactive behavior for real-time feedback.

**4. Features**

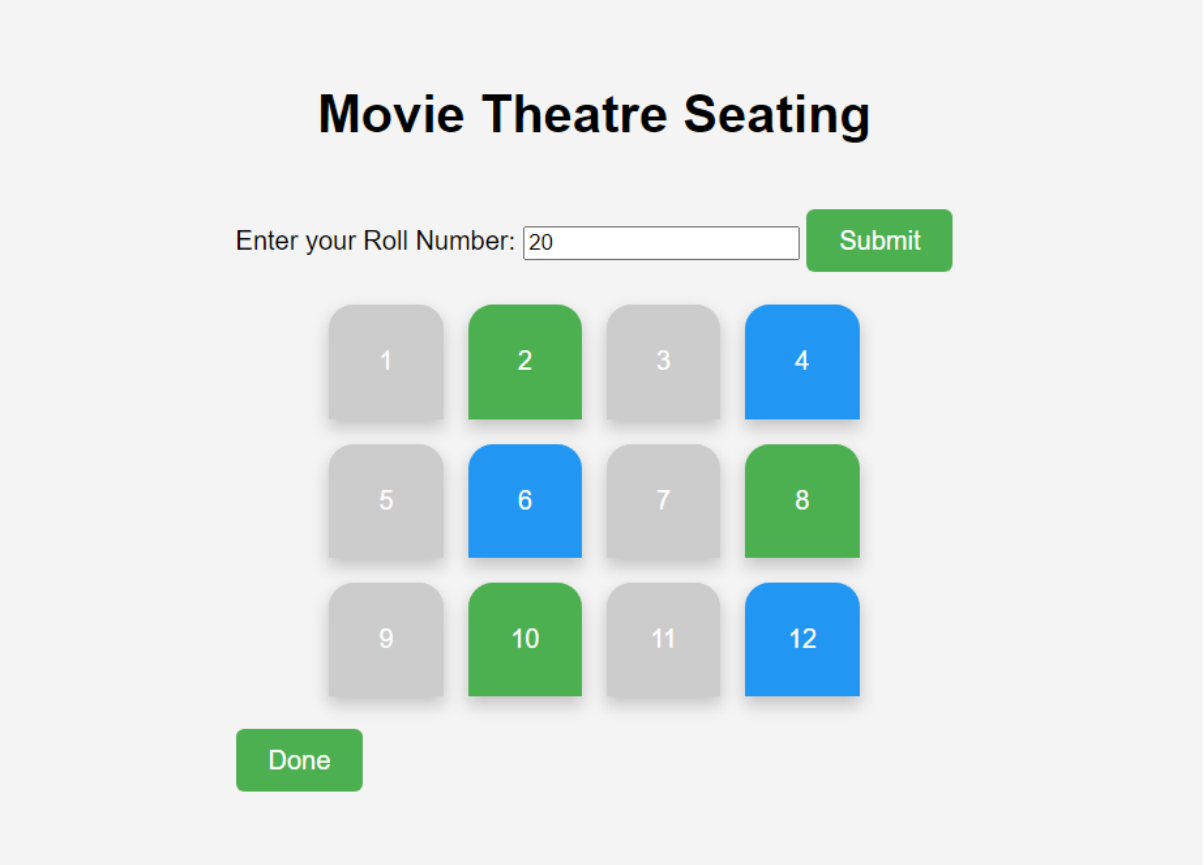
- **Dynamic Seat Allocation**:
    - The user enters their roll number, which determines the availability of seats.
    - Users with even roll numbers can only select even-numbered seats, and users with odd roll numbers can only select odd-numbered seats.
- **Interactive Seat Selection**:
    - Available seats can be clicked to toggle between selected and unselected states.
    - Upon clicking a seat, a message displays the seat number selected, giving real-time feedback.
- **Finalize Selection**:
    - The “Done” button allows users to confirm their selected seats.

- A message shows the list of seats selected or a notification if no seats were chosen.

**5. Implementation**

The implementation follows these key steps:

- **Seat Allocation Logic**:
  - The `setSeatAvailability` function checks the user's roll number to identify even or odd status.
  - Based on the roll number, only even or odd seats are made selectable, while the others are marked as unavailable.
- **Seat Selection and Finalization**:
  - The `selectSeat` function toggles the selected class on a clicked seat. When selected, it adds the seat to a list of `selectedSeats`.
  - The `doneSelecting` function provides an alert with the selected seats, or a message if no seats are chosen.



**6. User Interface**

The UI is straightforward and consists of:

- **Roll Number Input**: An input field for the user to enter their roll number.
- **Seating Layout**: A grid layout displaying numbered seats. Selected seats are highlighted, unavailable seats are grayed out, and available seats remain clickable.
- **Done Button**: A button to finalize the selection, which remains hidden until seat availability is set.

**7. Advantages**

- **User-Friendly**: Clear indication of seat availability based on roll number, making the process intuitive.
- **Real-Time Interaction**: The immediate visual feedback and seat-selection alerts enhance the user experience.
- **Organized Seating**: Restricting seat selection based on roll number prevents overcrowding of popular seats and ensures a more organized distribution.

**8. Future Enhancements**

Potential improvements include:

- **Enhanced Validation**: Adding checks for valid roll numbers and seat limits per user.
- **Styling Improvements**: Using more vibrant colors or icons to improve the UI.
- **Database Integration**: Allowing persistent storage of selected seats for multiple users.

**9. Conclusion**

The Movie Theatre Seating Allocation System provides a functional and interactive way to manage seat allocation based on user-specific criteria. The system effectively demonstrates how JavaScript can be used to create a simple yet engaging interface, making the project a useful exercise in web development and user interface design.

## Task 3: File Uploader Website

### *1. Introduction*

This project involves creating a simple file uploader website that allows users to upload files to a server. The goal is to provide a user-friendly interface where users can select a file, upload it, and receive feedback on the success of the upload. This project is built using HTML, CSS, JavaScript for the frontend, and Node.js with the Multer middleware for handling file uploads on the backend.

### *2. Objectives*

- To create a basic user interface that allows users to upload files.
- To provide feedback to the user after a file is successfully uploaded.
- To handle file storage on the server and provide easy access to uploaded files.
- To implement the backend using Node.js and Multer for file handling.

### *3. Technology Stack*

- **Frontend:**
  - HTML: For structuring the webpage.
  - CSS: For styling and layout of the file upload interface.
  - JavaScript: For interactive elements like file selection and upload submission.
- **Backend:**
  - Node.js: A JavaScript runtime used to handle HTTP requests and manage server-side operations.
  - Multer: A Node.js middleware used to handle `multipart/form-data`, which is used for uploading files.
- **Other Tools:**
  - Bootstrap: A front-end framework used to ensure the website is responsive and visually appealing.

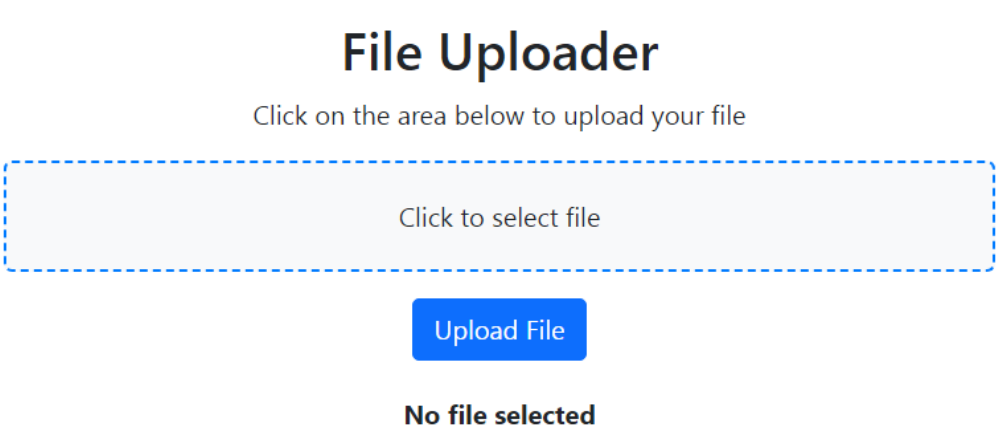
### *4. Features*

- **File Upload:**
  - Users can select files through a simple file input button.
  - The system allows only one file to be uploaded at a time.
- **Responsive Design:**
  - The user interface is responsive and adapts to various screen sizes, ensuring that it works well on both desktop and mobile devices.

- **Server-Side File Storage:**
  - Files are uploaded to the server and stored in the `uploads` directory.
  - Each uploaded file is given a unique filename to avoid collisions.
- **Feedback:**
  - After the file upload, the user is provided with a success message containing the uploaded file's name.

### *5. Implementation*

- **Frontend (index.html):**
  - The HTML provides the structure for the file upload interface, including the upload button and file input.
  - CSS is used to style the file upload area and make the UI attractive and user-friendly.
  - JavaScript manages file selection and upload actions, including showing feedback after a file is uploaded.
- **Backend (server.js):**
  - Node.js with the Express framework handles the HTTP requests.
  - Multer is used for file uploading, where it saves the uploaded files in the `uploads` folder.
  - Upon a successful upload, a message is sent back to the frontend, confirming the file's successful upload.




### *6. User Interface*

The user interface consists of the following elements:

- **Title:** A heading that displays "File Uploader."

- **File Upload Area:** A clickable area where users can select a file. The area displays a "Click to select file" message.
- **Upload Button:** A button that initiates the file upload process.
- **File Name Display:** A text area that shows the name of the selected file after the user picks one.

### *7. Advantages*

- **Simple and Easy to Use:** The file upload process is intuitive, with clear instructions and buttons.
- **Responsive Layout:** The design adapts to different screen sizes, making the site accessible on both mobile and desktop devices.
- **Efficient File Handling:** The backend efficiently handles file uploads using Multer and stores files securely on the server.
- **Minimalistic Design:** The website is minimal and focused solely on the task of file uploading, making it quick to use.

### *8. Conclusion*

The File Uploader website is a simple and efficient tool for uploading files. It provides users with an easy-to-use interface and ensures that files are handled securely on the server. The project demonstrates how to integrate frontend and backend technologies for file handling and offers a solid foundation for more complex file upload systems in future applications.

### *9. Future Enhancements*

- **Multiple File Upload:** Implementing support for uploading multiple files at once.
- **File Type Validation:** Adding functionality to limit uploads to specific file types (e.g., images, documents).
- **Progress Bar:** Adding a progress bar to show the upload status in real-time.
- **File Download Feature:** Allowing users to download their uploaded files from the server.

## Task 4: Personal Portfolio Website

### *1. Introduction*

This project involves the development of a personal portfolio website using React. The primary goal of this portfolio website is to showcase my skills, projects, and experience in web development. It provides potential employers or clients with an overview of my work and allows them to easily navigate through sections such as about me, projects, skills, and contact information.

### *2. Objectives*

- To create a visually appealing, responsive portfolio website that showcases my personal and professional information.
- To highlight key projects, skills, and experiences to demonstrate my expertise in web development.
- To integrate React components for a dynamic and interactive user experience.
- To make the portfolio accessible on all devices by ensuring it is fully responsive.

### *3. Technology Stack*

- **Frontend:**
    - React: The core JavaScript library used for building dynamic and reusable UI components.
    - HTML & CSS: Used for the basic structure and styling of the website.
    - Bootstrap: A front-end framework used for responsive design and to quickly style elements.
    - React Router: Used for navigation between different sections of the website.
- **Other Tools:**
    - GitHub: For version control and code collaboration.
    - Node.js: For running the React development server.

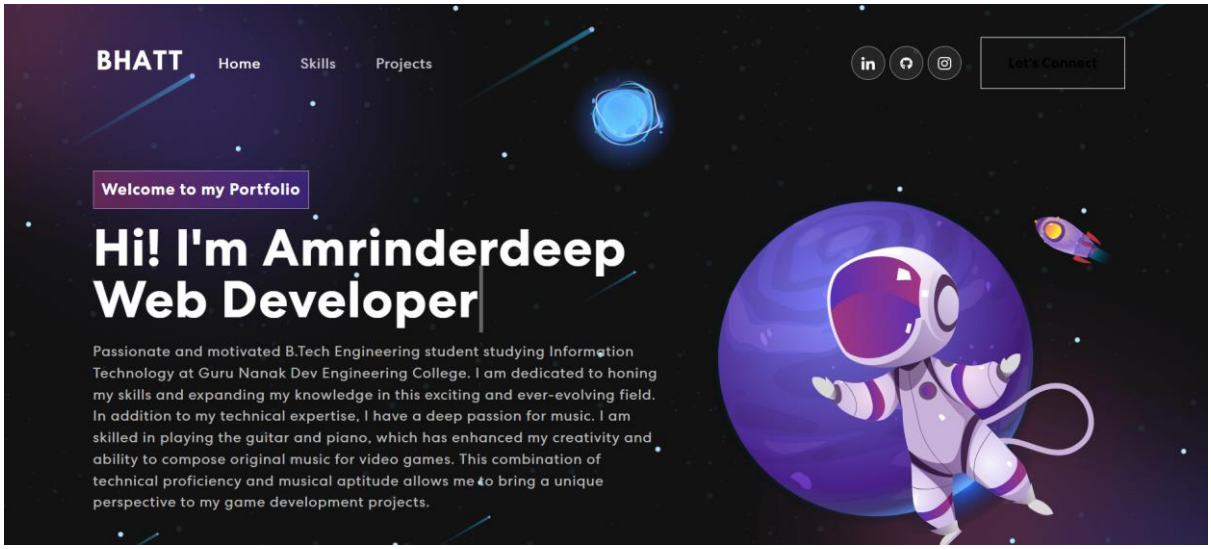
### *4. Features*

- **Responsive Design:**
    - The portfolio website is built to be responsive across various screen sizes, from mobile devices to desktop monitors. This ensures it provides a good user experience on any device.
- **Navigation:**

    - Easy-to-use navigation allows visitors to move between sections such as Home, About Me, Projects, Skills, and Contact.
    - React Router enables smooth, dynamic transitions between different sections of the website.
- **Showcase of Projects:**
    - The portfolio features a section dedicated to showcasing personal projects, each with a title, description, and links to view more or visit live projects. This section demonstrates the work I have done in web development.
- **Skills and Experience:**
    - A section that highlights my technical skills in various programming languages, frameworks, and tools. It also features my professional experiences and educational background.
- **Contact Form:**
    - A contact form for potential clients or employers to get in touch. The form collects the user's name, email, and message and includes basic form validation.

### *5. Implementation*

- **React Components:**
    - **Header Component:** Displays the navigation bar with links to different sections of the portfolio.
    - **Home Component:** The landing page featuring an introduction to myself and a professional image.
    - **About Component:** Details about my background, education, and professional journey.
    - **Projects Component:** Showcases various projects with descriptions and links.
    - **Skills Component:** Highlights technical skills in areas like web development, programming languages, and frameworks.
    - **Contact Component:** A contact form for users to reach out.
- **Styling:**
    - The styling is implemented using CSS and Bootstrap for responsive grid layouts, typography, and buttons. Custom styles are used to give the website a personal and professional appearance.
- **React Router:**
    - React Router is used to handle navigation between the different sections of the portfolio without requiring page reloads, offering a seamless user experience.




### *6. User Interface*

The user interface is designed with a clean and professional layout:

- **Header** with links to sections.
- **Home Section** with an introduction and a professional image.
- **About Me Section** showcasing the personal background.
- **Skills Section** displaying various technical abilities.
- **Projects Section** with links to various projects and detailed descriptions.
- **Contact Section** with a simple contact form for inquiries.

### *7. Advantages*

- **User-Friendly Interface:** The portfolio features an easy-to-navigate interface with a clear structure, allowing visitors to quickly find information.
- **Responsive Design:** The website automatically adjusts to fit any screen size, ensuring a positive experience across devices.
- **Professional Look and Feel:** The combination of modern design and clear content presents a professional image, ideal for a personal portfolio.
- **Interactive and Dynamic:** Using React, the website offers smooth transitions and a dynamic user experience.

### *8. Conclusion*

The Personal Portfolio website serves as an effective tool to showcase my skills, projects, and professional journey. Built using React, it is dynamic, interactive, and fully responsive across devices. This portfolio highlights my web development expertise and

serves as an essential part of my personal brand. It also lays the groundwork for further enhancements, including the addition of a blog or an integrated project management system.

### *9. Future Enhancements*

- **Blog Section:** Adding a blog to share technical knowledge and personal experiences.
- **Dark Mode:** Implementing a dark mode for a better user experience, especially in low-light environments.
- **Animations and Interactivity:** Adding subtle animations and interactivity to enhance the user experience and make the portfolio more engaging.
- **SEO Optimization:** Implementing proper SEO practices to improve search engine ranking and visibility.

## Task 5: CRUD Operations (Social Media)

### *Objective:*

To develop the Snippets social media platform, focusing on backend development using Express.js and MongoDB. The goal is to create a functional website for users to share short pieces of information (snippets) while demonstrating backend skills such as user authentication, CRUD operations, and database management.

### *Prerequisites:*

- Knowledge of JavaScript and Node.js for backend development.
- Familiarity with Express.js for building APIs and managing routing.
- Basic understanding of MongoDB and Mongoose for data management.
- Experience with EJS (Embedded JavaScript) templating for dynamic rendering of content.
- Basic knowledge of HTML and CSS for front-end structure and styling.

### *Tools and Software Required:*

- **Backend Development:** Node.js, Express.js
- **Database:** MongoDB
- **Frontend Development:** Vanilla HTML, CSS
- **Templating:** EJS
- **Version Control:** Git, GitHub
- **Development Environment:** Visual Studio Code, Postman (for API testing)

### *Introduction:*

Snippets is a simple social media platform that allows users to share short, insightful pieces of information. The project focuses primarily on backend functionality, showcasing features like user authentication, CRUD operations for snippets, and seamless database interaction. The platform also includes an admin panel for managing user data and content.

### *Technology Stack:*

- **Frontend:** Vanilla HTML, CSS
- **Backend:** Node.js, Express.js
- **Database:** MongoDB, Mongoose

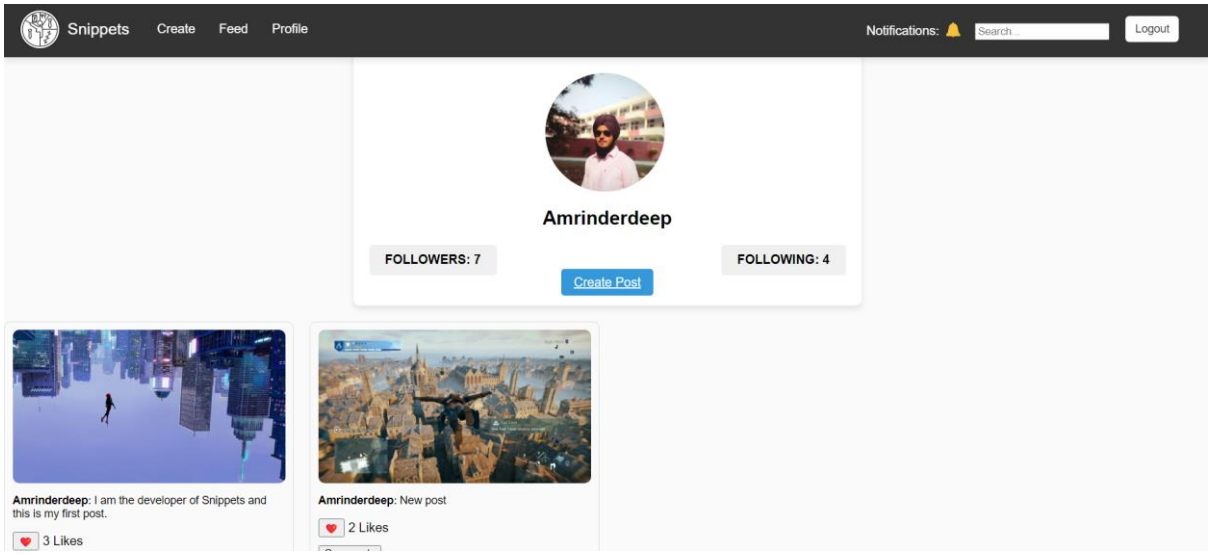
- **Templating:** EJS

### *Features:*

- **User Authentication and Authorization:** Secure login and registration using session management.
- **Create and View Snippets:** Users can post snippets and view others' snippets.
- **MongoDB Integration:** MongoDB stores user data and snippets, with Mongoose used for database operations.
- **Admin Panel:** Admin users can manage snippets and user data.

### *Implementation:*

- **Frontend:** The user interface is basic, using HTML for layout and CSS for styling. It includes features like login, snippet creation, and viewing shared snippets.
- **Backend:** Express.js handles routing for user authentication and snippets. MongoDB and Mongoose are used for data storage and retrieval.
- **Database Structure:**
    - **Users Collection:** Stores user credentials.
    - **Snippets Collection:** Stores snippets created by users.




### *User Interface:*

- **Public Interface:** Homepage displaying shared snippets and an option to create new snippets. Login and snippet creation placeholders are included.
- **Admin Panel:** Admin can log in to manage users and snippets, using a session-based authentication system.

### *Advantages:*

- **User-Friendly Interface:** Simple navigation for users and admins.
- **Dynamic Content Management:** Admins can easily manage snippets and users.
- **Database Integration:** Data is stored and retrieved efficiently from MongoDB, ensuring smooth operation.

### *Conclusion:*

The Snippets project serves as a foundational backend development project, demonstrating key concepts such as user authentication, CRUD functionality, and MongoDB integration. The project provides a base for future improvements in both backend and frontend development.

### *Future Enhancements:*

- **Enhanced UI:** Improving the user interface for better engagement.
- **Real-Time Updates:** Implementing WebSockets for live updates of snippets.
- **Advanced Authentication:** Adding features like password reset and email verification.

# Minor Project Report: IRCS Ludhiana Website

## *1. Objective*

To develop a modern, responsive website for the Indian Red Cross Society (IRCS) Ludhiana to promote their initiatives, provide relevant information to the public, and streamline administrative tasks via an admin panel.

## *2. Prerequisites*

- Knowledge of **React** for building the frontend.
- Experience with **Bootstrap** for styling and responsive design.
- Proficiency in **Node.js** and **Express.js** for backend development.
- Familiarity with **MongoDB** for database management.
- Understanding of **JWT (JSON Web Tokens)** for user authentication.

## *3. Tools and Software Required*

- **Frontend Development**: React, Bootstrap, HTML, CSS, JavaScript
- **Backend Development**: Node.js, Express.js
- **Database**: MongoDB
- **Authentication**: JWT
- **Version Control**: Git, GitHub
- **Development Environment**: VS Code, Postman (for API testing)

## *4. Introduction*

The IRCS Ludhiana website aims to serve as a digital presence for the organization, providing information about their initiatives, events, and appeals for donations. The platform includes a public-facing interface for visitors and an admin panel for managing content.

Key Features:

- User-friendly public interface.
- Secure admin panel for managing data.
- Database integration for storing dynamic content like notices and user details.

- JWT-based authentication for secure access to administrative functions.

### *5. Technology Stack*

**Frontend:**

- **React**: For building interactive user interfaces.
- **Bootstrap**: Ensures the website is responsive and visually appealing.

**Backend:**

- **Node.js**: Handles server-side logic.
- **Express.js**: Manages routes and API endpoints.

**Database:**

- **MongoDB**: Stores dynamic content, user details, and notices.

**Authentication:**

- **JWT**: Provides secure login for the admin panel.

### *6. Features*

**Public User Interface:**

- **Appeals Section**: Highlights ongoing donation and volunteer appeals.
- **Notices Section**: Displays important updates and announcements.
- **Donation Button**: Allows users to contribute directly.

**Admin Panel:**

- **Dashboard**: Overview of website statistics and recent activities.
- **Manage Notices**: Add, edit, or delete notices dynamically.
- **Manage Donations**: View donor details and manage payment records.

**Authentication:**

- Admin access is protected using JWT-based authentication.

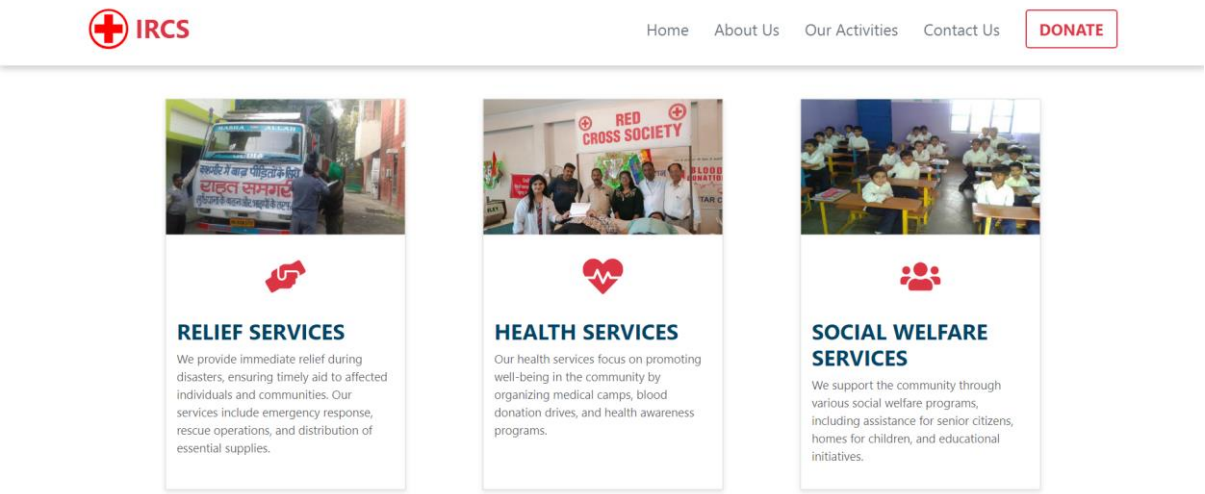
**Database Integration:**

- Notices, user details, and donation records are stored and retrieved dynamically from MongoDB.

## *7. Implementation*

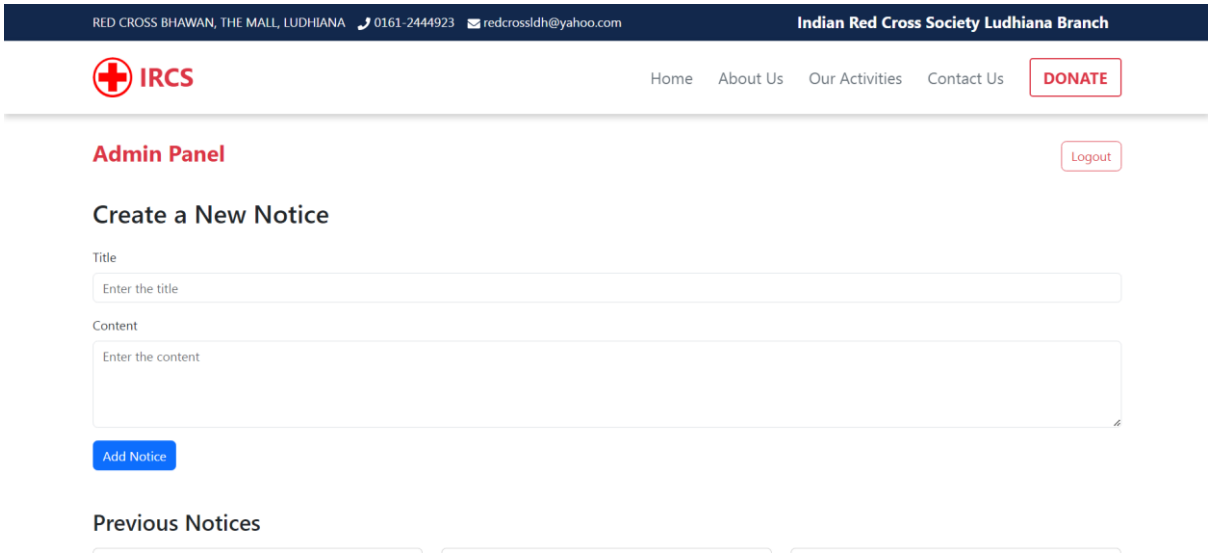
**Frontend:**

- **React Components**: Modular components for appeals, notices, and the admin panel.
- **CSS Styling**: Custom styles integrated with Bootstrap for responsiveness.
- **Dynamic Rendering**: Notices and content fetched from the database are rendered dynamically.




**Backend:**

- **Node.js and Express.js**: Backend APIs handle CRUD operations for notices, donations, and user authentication.
- **JWT Authentication**: Ensures only authorized users can access the admin panel.



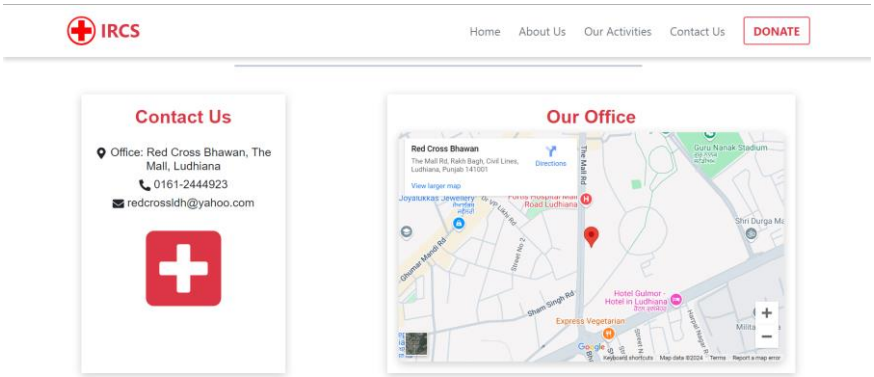

**Database:**

- **MongoDB Collections**:
  - `notices`: Stores information about notices.
  - `users`: Stores admin login credentials.
  - `donations`: Tracks donation records.

## *8. User Interface*

**Public Interface:**

- **Homepage**: Showcases the IRCS mission, recent activities, and appeals.
- **Appeals Section**: Highlights donation and volunteering opportunities.
- **Notices Section**: Displays a list of recent updates.
- **Responsive Design**: Ensures compatibility across devices.




**Admin Panel:**

- **Login Page**: Protected with JWT authentication.
- **Dashboard**: Displays website stats and controls for content management.
- **Notices Management**: Admin can add, edit, or delete notices via a form.
- **Donation Management**: Allows admin to view and organize donations.

### *9. Advantages*

- **User-Friendly Interface**: Simplifies navigation for visitors and admins.
- **Dynamic Content Management**: Enables real-time updates via the admin panel.
- **Secure Access**: JWT ensures admin functionality is restricted.
- **Responsive Design**: Ensures usability on both desktop and mobile devices.

### *10. Conclusion*

The IRCS Ludhiana website bridges the gap between the organization and its stakeholders, providing an efficient platform for public outreach and administration. By leveraging modern technologies, the project demonstrates a robust full-stack development approach.

### *11. Future Enhancements*

- **Multi-Language Support**: Adding regional languages for a broader audience.
- **Real-Time Chat**: Live chat integration for visitor queries.
- **Advanced Analytics**: Admin panel enhancements for tracking user interactions.
- **Payment Gateway Integration**: Enable secure online donations directly through the website.