INWARD REGISTER

By

T.G.RAGURAM

REGISTRATION NO: 952622622025

Of

S.VEERASAMY CHETTIAR COLLEGE OF ENGINEERING AND TECHNOLOGY, PULIYANKUDI.

Project Report

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In Partial Fulfilled of the requirements

For the award of the Degree

OF

MASTER OF COMPUTER APPLICATION

ANNA UNIVERSITY, CHENNAI – 600 025

AUGUST,2024

BONAFIDE CERTIFICATE

Certified that this Project Report titled **(INWARD REGISTER)** is the Bonafide work of **Mr. T.G.RAGURAM(952622622025)** who carried out the Project Report work under my supervision. Certified further, that to the best of our knowledge the work reported here in does not from part of any other Project Report on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Supervisor
Mrs. S.Karthika, MCA.,
Assistant Professor,
Department of Computer Applications
S.Veerasamy Chettiar College of Engg.Tech,
Puliangudi.

Head of the Department
Ms. SHIRIN AYESHA MARIYAM, ME,
Head of the Department,
Department of Computer Applications
S.Veerasamy Chettiar College of Engg.Tech,
Puliangudi.

Submitted for the viva-voce examination held at **S.Veerasamy Chettiar College of Engineering and Technology, Puliangudi** on _____

Internal Examiner **External Examiner**

DECLARATION

I hereby state that the thesis submitted for the degree of

MASTER OF COMPUTER APPLICATIONS

In

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

On

“INWARD REGISTER"

Is my original work and that it has not previously formed the basis for the award of

any Degree, Diploma, Associate ship, Fellowship or any other similar title.

ACKNOWLEDGEMENT

First of all we thank God for his abundant grace and countless blessings in making this Project Report work successful.

We wish to express our sincere thanks to Principal **Dr. K. MUTHULAKSHMI, M.E., PH.D.,** S. Veerasamy Chettiar College of Engineering and Technology, Puliangudi for giving the opportunity to do my Project Report work in the concerned organization.

We express our heartfelt thanks and sincere gratitude to **Ms. SHIRIN AYESHA MARIYAM, ME,** Head of the Department, Department of Computer Applications, S.Veerasamy Chettiar College of Engineering and Technology, Puliangudi, who facilitated us to complete my Project Report successfully.

We express our deep sense of gratitude to **Mrs. S.Karthika, MCA.,** Assistant Professor, Department of Computer Applications, S.Veerasamy Chettiar College of Engineering and Technology, Puliangudi, for her guidance, valuable suggestions and support.

We also wish to record our heartfelt thanks to all the staff members of our MCA Department, parents, friends who have rendered direct or indirect help to complete the Project Report.

ABSTRACT

Inward section exists, whose job is to maintain Inward Register on a daily basis, so as to record each and every letter/document received and sent. Also, to distribute Inward letters to concern section for further necessary action, as per directions of the Head of the office. Inward software is designed and customized for the use of college. The Inward software will help you in managing documents, track letter status, sender and receiver. The inward section receives softcopies/ mails/courier/post and registers them. These letters are then sent to the Principal, Secretary, and Vice Principal. The letters which are approved and signed are dispatched to the concern departments. This application provides record keeping facility of letters Inward in any organization. We can also check the status of letters in the organization. One letter can be sent to multiple addresses at the same time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 SCOPE OF THE PROJECT

The Inward section is a pivotal component of the organization's administrative framework, tasked with the vital role of managing the inflow and outflow of documents. It maintains a meticulously updated Inward Register on a daily basis, which records every letter and document received and sent. This register acts as a comprehensive log that ensures a clear and traceable record of all correspondence, thereby supporting efficient communication and document management across the organization.

The primary responsibility of the Inward section includes the systematic distribution of incoming letters to the relevant sections for further necessary actions. This process is conducted in accordance with the directives issued by the Head of the office, ensuring that each document reaches its intended recipient in a timely and organized manner. By doing so, the Inward section plays a crucial role in maintaining the smooth operation of the organizational workflow.

To enhance the efficiency and effectiveness of its operations, the college has implemented a customized Inward software designed specifically to meet its needs. This software is an advanced tool that aids in managing documents, tracking the status of letters, and recording detailed information about the senders and receivers. With this software, the manual workload of the Inward section is significantly reduced, allowing staff to focus on more critical tasks that require human intervention and decision-making.

The Inward section is equipped to handle various forms of correspondence, including softcopies, emails, courier packages, and postal mail. Upon receipt, these documents are immediately registered in the Inward Register, ensuring that there is a formal and traceable record of their arrival. Following registration, the documents are forwarded to key administrative

personnel, such as the Principal, Secretary, and Vice Principal, for review, approval, and signature. Once approved, these documents are dispatched to the relevant departments for further action.

The customized Inward software offers several key benefits that enhance the overall efficiency of the document management process. It provides a robust document management system that facilitates easy retrieval and reference of documents, significantly reducing the time and effort required to handle physical documents. Additionally, the software's ability to track the status of letters allows users to monitor whether a letter has been received, reviewed, or dispatched, thereby ensuring transparency and accountability.

Furthermore, the software supports record-keeping of all letters handled by the Inward section, creating a digital archive that is easy to access and reference. This digital record-keeping capability not only improves organization but also ensures that important documents are preserved and can be retrieved when needed. The software also enables the simultaneous dispatch of a single letter to multiple addresses, which is particularly useful for disseminating information quickly to various departments or stakeholders.

In conclusion, the Inward section, bolstered by the capabilities of the customized Inward software, is integral to the efficient management of documents within the organization. By maintaining accurate records, distributing documents effectively, and leveraging technology to automate routine tasks, the Inward section ensures that all correspondence is handled promptly and effectively, thereby supporting the smooth and efficient operation of the organization.

1.2 OBJECTIVES OF THE PROJECT

1.2.1 Document Management

The software offers a robust document management system that organizes and categorizes all incoming and outgoing documents. This structured approach allows for easy retrieval and reference, significantly reducing the time and effort needed to handle physical

documents. Users can quickly locate documents using various search criteria, ensuring that important information is always accessible when needed.

1.2.2 Tracking Letter Status

One of the most valuable features of the Inward software is its ability to track the status of each letter or document. Users can monitor whether a letter has been received, is under review, or has been dispatched. This real-time tracking ensures transparency and accountability within the document handling process. By knowing the exact status of a document, staff can provide accurate updates and ensure timely follow-up actions.

1.2.3 Sender and Receiver Information

The software meticulously records detailed information about the sender and receiver of each document. This feature creates a clear audit trail, which is essential for maintaining accountability and transparency. In cases where document authenticity or origin needs to be verified, having access to sender and receiver details is invaluable. This functionality supports internal controls and helps prevent miscommunication.

1.2.4 Efficiency and Automation

By automating routine tasks, the Inward software reduces the manual workload on staff. Tasks such as registering incoming documents, updating status, and dispatching documents are streamlined through automation. This not only speeds up the document handling process but also minimizes the risk of human error. Automation allows staff to focus on more critical tasks that require human judgment and decision-making, thereby improving overall efficiency.

1.2.5 Record-Keeping and Archiving

The software maintains a comprehensive digital archive of all documents handled by the Inward section. This record-keeping capability ensures that all documents are preserved in a structured manner and can be easily retrieved when needed. The digital archive supports compliance with record-keeping policies and enhances the organization’s ability to manage and audit documents.

1.2.6 Multi-Recipient Dispatch

The Inward software allows for a single letter or document to be sent to multiple addresses simultaneously. This feature is particularly useful for disseminating information quickly and efficiently to various departments or stakeholders. It ensures that all relevant parties receive the necessary documents without delay, facilitating better coordination and communication within the organization.

The customized Inward software significantly enhances the document management process within the organization. Its features, including document management, status tracking, detailed sender and receiver information, efficiency through automation, comprehensive record-keeping, and multi-recipient dispatch, collectively contribute to a more streamlined, accurate, and efficient handling of documents. By leveraging this software, the Inward section can ensure that all correspondence is managed promptly and effectively, supporting the overall operational efficiency of the organization.

CHAPTER 2

SYSTEM STUDY

2.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis The feasibility study of the proposed system is to be carried out. This is to ensure that the the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

2.1.1 ECONOMICAL FEASIBILITY

2.1.2 TECHNICAL FEASIBILITY

2.1.3 SOCIAL FEASIBILITY

2.1.1 ECONOMICAL FEASIBILITY

An organization makes good investment on the system. So, they should be worth full for the amount they spend in the system. Always the financial benefit and equals or less the cost of the system, but should not exceed the cost.

- The cost of investment is analyzed for the entire system
- The cost of Hardware and Software is also noted.
- Analyzing the way in which the cost can be reduced

Every organization wants to reduce their cost but at the same time quality of the Service should also be maintained. The system is developed according the estimation of the cost made by the concern. In this project, the proposed system will definitely reduce the cost and also the manual work is reduced and speed of work is also increased.

2.1.2 TECHNICAL FEASIBILITY

The Technical feasibility is the study of the software and how it is included in the study of our project. Regarding this there are some technical issues that should be noted they are as follows:

- Is the necessary technique available and how is it suggested and acquired?
- Does the proposed equipment have the technical capacity to hold the data required using the new system?
- Will the system provide adequate response that is made by the requester atan periodic time interval
- Can this system be expanded after this project development
- Is there a technique guarantees of accuracy, reliability in case of access of data and security

The technical issues are raised during the feasibility study of investigating our System. Thus, the technical consideration evaluates the hardware requirements, software etc. This system uses Android as front end and MySQL as back end. They also provide sufficient memory to hold and process the data. As the company is going to install all the process in the system it is the cheap and efficient technique.

This system technique accepts the entire request made by the user and the response is done without failure and delay. It is a study about the resources available and how they are achieved as an acceptable system. It is an essential process for analysis and definition of conducting a parallel assessment of technical feasibility.

Though storage and retrieval of information is enormous, it can be easily handled by MySQL. As the MySQL can be run in any system and the operation does not differ from one to another. So, this is effective.

2.1.3 SOCIAL FEASIBILITY

Proposed project will be beneficial only when they are turned into an information system and to meet the organization operating requirements. The following issues are considered for the operation:

- Does this system provide sufficient support for the user and the management?
- What is the method that should be used in this project?
- Have the users been involved in the planning and development of the projects?
- Will the proposed system cause any harm, bad result, loss of control and accessibility of the system will be lost?

Issues that may be a minor problem will sometimes cause major problem in the operation. It is the measure of how people can able to work with the system. Finding out the minor issues that may be the initial problem of the system. It should be a user- friendly environment. All these aspect should be kept in mind and steps should be taken for developing the project carefully.

Regarding the project, the system is very much supported and friendly for the user. The methods are defined in an effective manner and proper conditions are given in other to avoid the harm or loss of data. It is designed in GUI interface, as working will be easier and flexible for the user.

CHAPTER 3

SYSTEM ANALYSIS

System analysis is a critical phase in the software development lifecycle, focusing on understanding and defining the requirements of a new system or the improvements needed for an existing one. It involves a thorough examination of the current system, identification of shortcomings, and envisioning the desired functionalities and features. During system analysis, stakeholders collaborate closely with analysts to gather, document, and analyze requirements. This process includes studying user needs, business processes, and technical infrastructure to ensure that the proposed system aligns with organizational goals and user expectations. Additionally, system analysts assess the feasibility of the proposed solution, considering factors such as technical constraints, budgetary limitations, and time constraints. Through techniques like interviews, surveys, and prototyping, system analysts create detailed specifications and design documents that serve as a blueprint for development. Effective system analysis lays the foundation for a successful system implementation, providing clarity and direction for the development team while ensuring that the final product meets the needs of its users and stakeholders.

3.1 Existing System

The existing system refers to any software or technological infrastructure that is currently in place within an organization or environment. In the context of the inward section, the existing system involves the manual handling of incoming letters received via various channels such as mails, couriers, or postal services. The initiator receives these letters and stores them as paper copies, necessitating significant physical storage space and manual effort. This manual process poses challenges in terms of organization, retrieval, and document management. Moreover, the reliance on paper-based storage makes it susceptible to issues such as misplacement or loss of documents. Overall, the existing system lacks efficiency and scalability, hindering the smooth operation of the inward section.

3.2 Proposed System

The proposed inward register software offers a feasible solution to streamline the process of sending and receiving mails within the organization. By digitizing the document management process, the software eliminates the need for extensive physical storage space required for paper documents. Instead, documents are stored electronically, allowing for efficient organization and retrieval. Users can easily access and manage documents through the software interface, enhancing productivity and reducing the risk of document loss or misplacement. The proposed system aims to address the shortcomings of the existing system by providing a centralized platform for managing incoming and outgoing mails effectively. Through features such as document tracking, notifications, and search capabilities, the proposed system offers enhanced efficiency, accuracy, and transparency in document management processes.

CHAPTER 4

SYSTEM SPECIFICATION

To be used efficiently, all computer software needs certain hardware components or the software resources to be present on a computer. These prerequisites are known as computer system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

4.1 HARDWARE REQUIREMENTS:

Processor	:	Intel Core Duo 2.0 GHz or more
RAM	:	1 GB or More
Hard disk	:	80GB or more
Monitor	:	15’’ CRT or LCD monitor
Keyboard	:	Normal or Multimedia
Mouse	:	Compatible mouse

4.2 SOFTWARE REQUIREMENTS:

Front End	:	React JS
Back End	:	Django
Database	:	MYSQL Server
Operation System	:	Ubuntu 22.04.4 LTS

4.3. ABOUT SOFTWARE

4.3.1 React JS:

4.3.1.1 Introduction to React.js

React.js, commonly referred to as React, is a popular JavaScript library for building user interfaces, especially for single-page applications. Developed and maintained by Facebook, React emphasizes a component-based architecture, allowing developers to create reusable UI components. React's efficient rendering using a virtual DOM makes it an excellent choice for building dynamic and high-performance web applications.

4.3.1.2 Key Features of React.js

- **Component-Based Architecture:**
    - React allows developers to build encapsulated components that manage their own state and compose them to create complex UIs. This modularity promotes reusability and maintainability, as each component can be developed and tested independently.
- **Virtual DOM:**
    - React uses a virtual DOM to optimize rendering performance. When the state of a component changes, React updates the virtual DOM first, calculates the most efficient way to update the real DOM, and then applies those changes. This process minimizes the number of DOM manipulations, leading to faster and smoother UI updates.
- **Declarative Syntax:**
    - React’s declarative approach makes it easy to reason about the state of the UI at any given time. Developers describe what the UI should look like based on the current state, and React takes care of updating the UI to match this state, simplifying the development process.

- **JSX (JavaScript XML):**
    - React introduces JSX, a syntax extension that allows writing HTML-like code within JavaScript. JSX enhances readability and makes it easier to visualize the structure of UI components, bridging the gap between HTML and JavaScript.
- **One-Way Data Binding:**
    - React uses a unidirectional data flow, where data is passed from parent components to child components through props. This one-way data binding makes it easier to track data changes and debug applications, as the data flow is more predictable.
- **Rich Ecosystem and Community:**
    - React has a vast ecosystem of libraries and tools, such as Redux for state management, React Router for navigation, and various UI component libraries. Additionally, the large and active community contributes to continuous improvements and a wealth of resources for learning and troubleshooting.

4.3.1.3 Usage of React.js in Project

In your project, React.js is used to build dynamic and responsive user interfaces. Here's how React.js supports your project:

- **Component-Based UI Development:**
    - React’s component-based architecture allows for the creation of reusable UI components. Each component encapsulates its own logic and presentation, making it easy to develop, test, and maintain the UI. This modular approach enables developers to build complex UIs by composing smaller, reusable components.
- **Efficient Rendering:**
    - React’s virtual DOM ensures that the UI updates are efficient and performant. By minimizing direct DOM manipulations, React provides a smooth user experience even in applications with frequent state changes and dynamic content.

- **State Management:**
    - React's built-in state management within components enables developers to manage UI state effectively. For more complex state management needs, libraries like Redux or Context API can be integrated, providing a scalable solution for managing application state across multiple components.
- **Declarative UI:**
    - React’s declarative syntax allows developers to describe how the UI should look based on the current state. This approach simplifies the development process by abstracting away the direct manipulation of the DOM, making it easier to reason about the UI and its behavior.
- **Integration with Backend APIs:**
    - React can seamlessly integrate with backend APIs to fetch and display data. By using JavaScript’s fetch API or libraries like Axios, React components can make asynchronous calls to APIs, handle responses, and update the UI accordingly.

4.3.1.4 Benefits of Using React.js

- **Modularity and Reusability:**
    - React’s component-based architecture promotes modularity and reusability. Components can be reused across different parts of the application, reducing code duplication and improving maintainability
- **Performance:**
    - React’s virtual DOM and efficient diffing algorithm ensure high performance and fast UI updates. This makes React suitable for building applications with complex and dynamic user interfaces.
- **Developer Experience:**
    - React’s declarative syntax, rich ecosystem, and extensive documentation enhance the developer experience. Tools like React DevTools aid in debugging and profiling, making development and troubleshooting more efficient.

- **Community and Ecosystem:**
    - React's large and active community provides a wealth of resources, tutorials, and third-party libraries. This extensive ecosystem allows developers to find solutions to common problems and extend React's functionality as needed.

- **Scalability:**
    - React's architecture and state management solutions, such as Redux, enable the development of scalable applications. React can handle applications of varying sizes and complexities, making it a versatile choice for both small projects and large-scale enterprise applications.

4.3.2 Django Framework:

4.3.2.1 Introduction to Django

Django is a high-level Python web framework that enables rapid development and clean, pragmatic design. Created by experienced developers, Django takes care of much of the hassle of web development, allowing developers to focus on writing their app without needing to reinvent the wheel. It's free and open source, boasting a wide range of features and a large, active community that contributes to its ongoing improvement.

4.3.2.2 Key Features of Django

- **Object-Relational Mapping (ORM):**
    - Django's ORM provides a powerful and intuitive way to interact with the database. Developers can define their database schema using Python classes, and Django handles the underlying SQL queries. This abstraction allows for easy manipulation of the database and supports multiple database backends.
- **Automatic Admin Interface:**
    - One of Django's most praised features is its automatically generated admin interface. By registering models, developers can instantly get a fully functional

admin panel for managing application data, significantly speeding up the development process.

- **Security:**
    - Django comes with built-in protections against many common security threats, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking. It also includes a robust authentication system that handles user accounts, passwords, and sessions securely.
- **Scalability and Performance:**
    - Django is designed to help developers scale applications seamlessly. Its modularity and ability to integrate with various caching mechanisms make it suitable for handling high traffic and large volumes of data. Django's architecture supports horizontal scaling, allowing for efficient load balancing and performance optimization.
- **Comprehensive Documentation:**
    - Django's extensive and thorough documentation is one of its greatest strengths. It includes a wide range of tutorials, guides, and references that make it accessible for both beginners and experienced developers.
- **Versatile Templates:**
    - Although your project does not use Django's template system, it's worth mentioning that Django includes a powerful templating engine for rendering dynamic HTML content. This feature supports separation of business logic and presentation, promoting clean and maintainable code.

4.3.2.3 Usage of Django in Project

In your project, Django is primarily used for building APIs, focusing on backend functionalities rather than the traditional template-based web development. Here's how Django supports your project:

- **API Development:**

  Django's robust framework supports the creation of APIs through its views and serializers. This allows for efficient handling of HTTP requests and responses, making it an ideal choice for developing RESTful APIs.

- **Database Management:**
  - Using Django's ORM, you can define your data models and interact with the database in a seamless manner. The ORM handles data migrations, schema evolution, and complex queries, simplifying the process of database management.

- **Middleware and Routing:**
  - Django's middleware framework allows for processing requests globally, such as handling authentication or logging. The URL dispatcher (routing) in Django enables you to map URL patterns to specific views, ensuring clean and maintainable URL structures.
- **Security:**
  - Django's built-in security features protect your API endpoints from common vulnerabilities. The framework's emphasis on security helps ensure that your application adheres to best practices for web security.
- **Testing:**
  - Django includes a comprehensive testing framework that allows you to write unit tests and integration tests for your application. This ensures that your APIs work correctly and maintain high reliability.
- **Admin Interface (for Data Management):**
  - Even though your primary focus is on APIs, you can still leverage Django's admin interface to manage backend data efficiently. This can be particularly useful for administrative tasks and data entry without needing to build custom interfaces.

4.3.2.4 Benefits of Using Django

- **Rapid Development:**
    - Django’s framework encourages rapid development by providing many pre-built components and tools, reducing the time needed to build complex applications.
- **Security:**
    - Django's emphasis on security ensures that developers can avoid many common security pitfalls, providing a solid foundation for building secure web applications.
- **Scalability:**
    - Designed to handle high traffic, Django’s architecture supports scaling both vertically and horizontally, making it suitable for projects of all sizes.

4.3.3 MySQL:

4.3.3.1 Introduction to MySQL

MySQL is a widely-used open-source relational database management system (RDBMS). It is known for its reliability, performance, and ease of use. MySQL was initially developed by MySQL AB, a Swedish company, and is currently owned by Oracle Corporation. It uses structured query language (SQL) for accessing and managing the data within its databases, providing a robust and scalable solution for data storage and retrieval.

4.3.3.2 Key Features of MySQL

- **Relational Database Management:**
    - MySQL organizes data into tables with rows and columns, following the principles of a relational database. This structured format allows for efficient data retrieval and manipulation through SQL queries, making it ideal for managing large datasets and complex relationships.
- **Performance and Scalability:**
    - MySQL is designed to handle high traffic and large volumes of data. It supports various storage engines like InnoDB and MyISAM, each optimized for different

use cases. MySQL's architecture allows for easy scaling both vertically (by increasing server resources) and horizontally (through replication and sharding).

- **Security:**
    - MySQL includes robust security features to protect data integrity and prevent unauthorized access. It supports user authentication, role-based access control, and encryption. Fine-grained privileges can be set for different users and databases, ensuring secure data management.
- **ACID Compliance:**
    - MySQL supports ACID (Atomicity, Consistency, Isolation, Durability) properties, especially when using the InnoDB storage engine. This ensures reliable transaction processing, maintaining data integrity even in the event of system failures.

- **Replication and High Availability:**
    - MySQL provides various replication options, including master-slave and master-master replication, to ensure high availability and data redundancy. Replication allows data from one MySQL database server (the master) to be copied to one or more MySQL database servers (the slaves), facilitating load balancing and failover solutions.
- **Comprehensive Documentation and Community Support:**
    - MySQL has extensive documentation covering all aspects of the database system, from installation and configuration to advanced features and troubleshooting. Additionally, its large and active community offers a wealth of resources, forums, and third-party tools to assist developers and database administrators.

4.3.3.3 Usage of MySQL in Project

In your project, MySQL serves as the primary database system for storing, managing, and retrieving data. Here's how MySQL supports your project:

- **Data Storage and Management:**
    - MySQL provides a reliable and efficient way to store structured data. Tables, indexes, and foreign keys ensure that data is organized and accessible. MySQL's support for various data types and advanced SQL queries allows for complex data operations and analytics.
- **Database Design and Normalization:**
    - Using MySQL, you can design your database schema to ensure data consistency and avoid redundancy through normalization. By organizing data into related tables and defining primary and foreign keys, MySQL helps maintain data integrity and supports complex relationships between different data entities.

- **Transaction Management:**
    - MySQL’s ACID compliance ensures reliable transaction processing. By using transactions, you can group multiple SQL operations into a single, atomic unit of work, which either completes entirely or has no effect, ensuring data consistency.
- **Security and Access Control:**
    - MySQL’s robust security features allow you to define user roles and set granular permissions for accessing and modifying data. This ensures that only authorized users can perform specific operations, protecting sensitive data from unauthorized access.
- **Backup and Recovery:**
    - MySQL provides tools for data backup and recovery, ensuring data durability and protection against data loss. Regular backups can be scheduled, and in the event of data corruption or loss, recovery procedures can restore the database to its previous state.

4.3.3.4 Benefits of Using MySQL

- **Performance and Reliability:**
    - MySQL's optimized storage engines and efficient indexing provide high performance for both read and write operations. Its reliability and stability make it a trusted choice for many large-scale applications.
- **Scalability:**
    - MySQL's ability to handle high volumes of data and concurrent transactions ensures that it can scale with the growth of your application. Features like replication and clustering further enhance its scalability and availability.
- **Flexibility and Ease of Use:**
    - MySQL's flexibility allows it to be used in various environments, from small applications to large enterprise systems. Its ease of use, combined with comprehensive documentation, simplifies the learning curve for new users and developers.

CHAPTER 5

SYSTEM DESIGN

In designing a news app system, numerous considerations are vital to ensure its effectiveness, scalability, and reliability. The system architecture should comprise components like a front-end client interface, a back-end server, and a database. The client interface should offer intuitive navigation, personalized content delivery, and efficient search functionalities. Meanwhile, the back-end server must handle user requests, authenticate users securely, and communicate with external APIs to fetch news articles. A robust database structure is crucial for storing and retrieving articles, user preferences, and session data. Scalability concerns necessitate employing cloud-based infrastructure and microservices architecture to accommodate growing user bases and fluctuating traffic demands. Additionally, the system should prioritize real-time updates, ensuring users receive timely news notifications. Security measures such as encryption, authentication, and authorization mechanisms are imperative to safeguard user data and prevent unauthorized access. Lastly, the system design should prioritize performance optimization techniques, such as caching frequently accessed data and implementing asynchronous processing for resource-intensive tasks, to deliver a seamless and responsive user experience.

5.1 USE CASE DIAGRAM:

A use case diagram serves as a visual representation of the functional requirements of a system from the perspective of its users. It illustrates the various interactions between actors (users or external systems) and the system itself, showcasing how users interact with the system to achieve specific goals. In a news app, the use case diagram would outline the different actions users can perform and how they relate to the system's

functionalities.

The use case diagram would depict these interactions as actors (e.g., "User") interacting with the system (represented by a box) through various use cases (represented by ovals). Arrows between actors and use cases illustrate the flow of actions or information. Additionally, relationships between use cases, such as "extends" and "includes" relationships, can be represented to show dependencies or optional functionalities.

Overall, a detailed use case diagram provides a clear overview of the system's functionalities and helps stakeholders understand how users interact with the system to achieve their goals. It serves as a valuable tool for requirements elicitation, system design, and communication between stakeholders and development teams throughout the software development lifecycle.

A use case diagram at its simplest is a representation of a user’s interaction with the system.

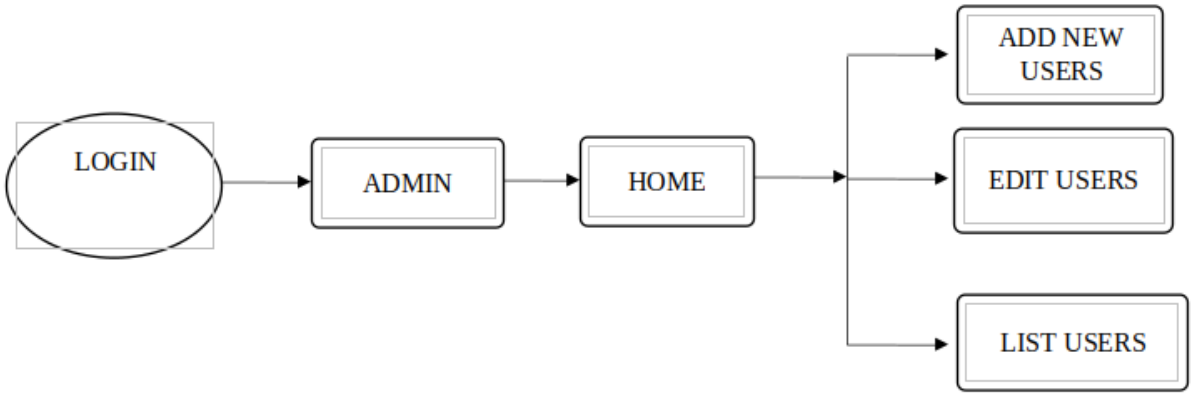

Fig 5.1.1 DATAFLOW DIAGRAM LOGIN FOR ADMIN

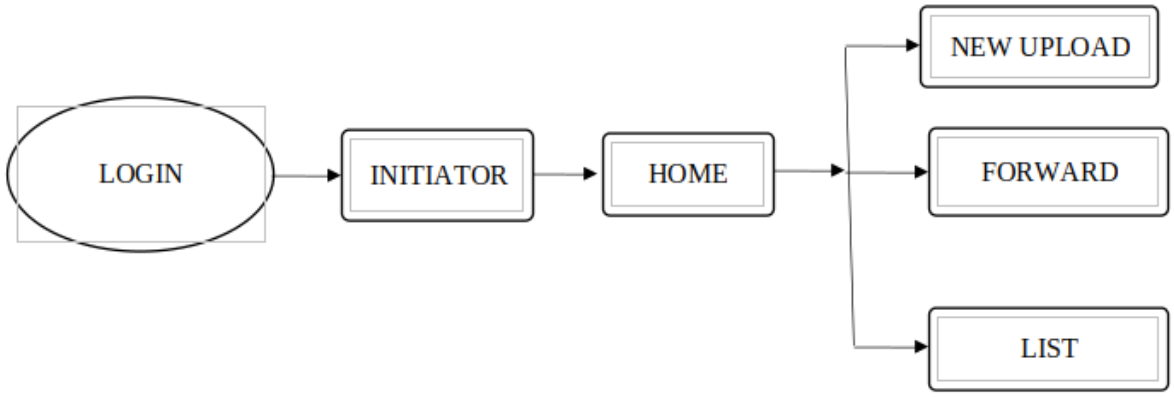

Fig 5.1.2 DATAFLOW DIAGRAM LOGIN FOR INITIATOR

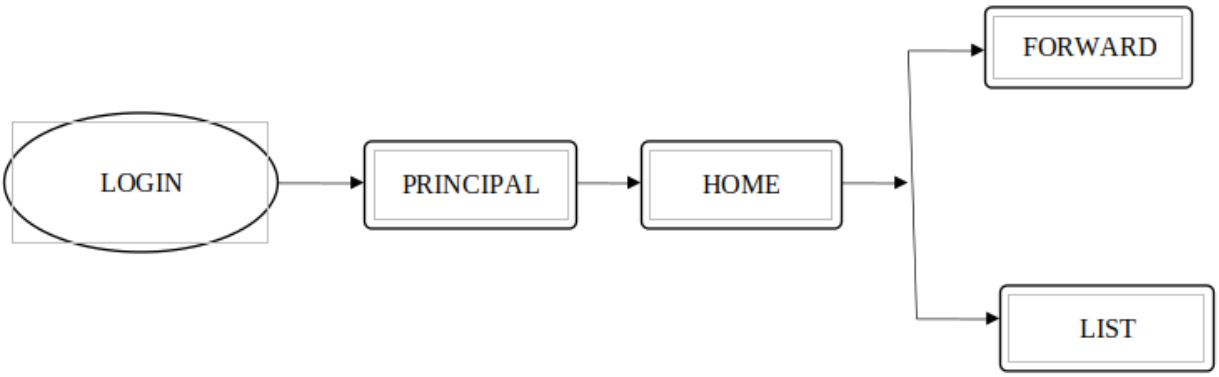

Fig 5.1.3 DATAFLOW DIAGRAM LOGIN FOR PRINCIPAL

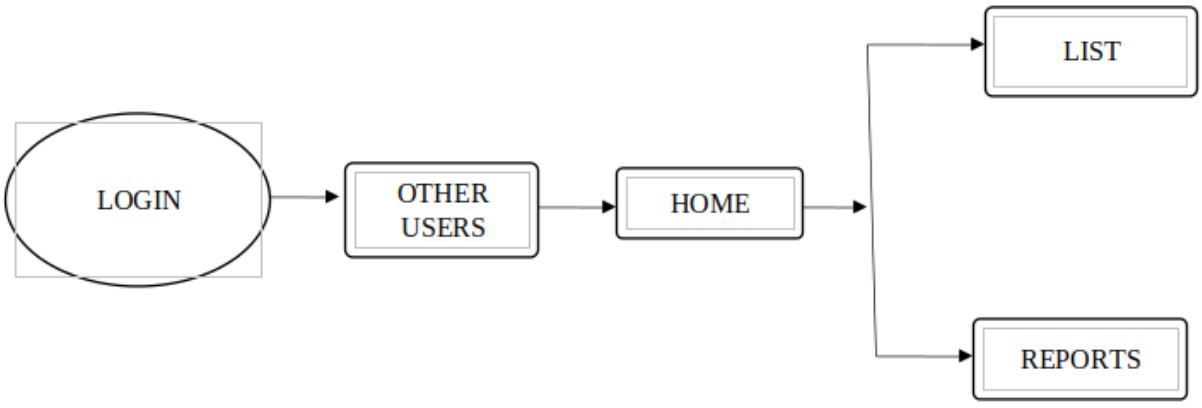

Fig 5.1.3 DATAFLOW DIAGRAM LOGIN FOR OTHERS

5.2 DATABASE TABLE

FIELD NAME	DATA TYPE
user	Foreign key (Django user table)
email	Varchar(30)
role	Varchar(30)
department	Foreign key (Department table)
created_on	DateTimeField
updated_on	DateTimeField

Table 5.2.1 User Role

FIELD NAME	DATA TYPE
department_name	Varchar(225)
department_type	Varchar(30)

Table 5.2.2 Department

FIELD NAME	DATA TYPE
File	Filefield
departments	Many to many key from Department
Replayable	BooleanField
replays	Many to many key from replay table

Table 5.2.3 Inward

CHAPTER 6
MODULE DESCRIPTION

6.1 Login

The login module is the first point of interaction for users with the system, offering a secure authentication process to ensure only authorized individuals gain access. Upon navigating to the login page, users are prompted to input their credentials, typically consisting of a username and password. These credentials undergo rigorous validation against the system's database. If the entered information matches an existing user record, the system grants access, directing the user to the system's dashboard or homepage. However, incorrect or mismatched credentials prompt the system to deny access and display an error message, guiding the user to reattempt the login process. This meticulous authentication mechanism forms the cornerstone of system security, safeguarding sensitive data and ensuring only legitimate users can access system functionalities.

Additionally, the login module incorporates sophisticated security features to bolster user authentication. These may include measures such as CAPTCHA verification, multi-factor authentication, or password strength requirements. By integrating such advanced security protocols, the module fortifies the system against potential security threats like brute-force attacks or unauthorized access attempts. Moreover, the login module adheres to industry best practices and compliance standards, such as OWASP guidelines or GDPR regulations, to ensure the highest level of data protection and user privacy. Through continuous monitoring and updates, the module evolves to mitigate emerging security risks and uphold the integrity of the system's authentication process. Overall, the login module serves as a critical checkpoint, effectively balancing user accessibility with robust security measures to maintain the system's integrity and trustworthiness.

6.2 Admin Controls

The admin controls module empowers designated administrators with comprehensive tools and privileges to oversee and manage various facets of the system. Administrators access a

dedicated admin panel or dashboard, providing centralized control over system administration tasks. Within this interface, administrators can perform a myriad of functions, including user management, system configuration, and activity monitoring. User management capabilities allow administrators to create, modify, or delete user accounts, assigning specific roles and permissions tailored to organizational requirements. Through role-based access control (RBAC), administrators can enforce access restrictions and allocate permissions based on user roles, ensuring the principle of least privilege and enhancing data security.

Furthermore, the admin controls module facilitates granular configuration of system settings, enabling administrators to customize features and functionalities according to organizational needs. This may entail defining password policies, configuring session timeouts, or establishing data retention rules. Administrators also play a pivotal role in monitoring system activities, leveraging audit logs and reporting functionalities to track user actions, system modifications, and security incidents. By maintaining a vigilant watch over system operations, administrators can promptly detect anomalies, investigate potential security breaches, and implement corrective measures. Overall, the admin controls module serves as the linchpin of effective system governance, empowering administrators with the tools and insights needed to uphold data integrity, enforce compliance, and ensure the seamless operation of the system.

6.3 Upload Page

The upload page module facilitates the seamless transfer of files and documents into the system, streamlining the document management process. Users, typically initiators or senders, access the upload page from the system's interface, where they can easily upload files from their local devices. This intuitive interface simplifies the upload process, allowing users to select multiple files simultaneously and initiate the upload with a single click. As files are uploaded, the module performs validation checks to ensure data integrity, verifying file types, sizes, and formats against predefined criteria. Any discrepancies or errors are promptly flagged, providing users with immediate feedback and guidance for resolution.

Moreover, the upload page module supports various file management features to enhance user experience and efficiency. For instance, users can assign metadata to uploaded files, such as titles, descriptions, or tags, facilitating categorization and searchability within the system.

Additionally, the module may offer version control capabilities, enabling users to upload new versions of existing documents while preserving previous iterations. This versioning functionality ensures users access the most up-to-date information while retaining historical records for reference. By incorporating these advanced features and user-friendly interfaces, the upload page module empowers users to seamlessly upload, organize, and manage documents within the system, fostering collaboration and productivity across the organization.

6.4 Track

The track module provides users with comprehensive visibility and control over the status and progress of their documents within the system. Users can access the tracking feature from the system's interface, where they are presented with real-time updates on the lifecycle of their uploaded documents. This includes tracking when documents are received, viewed, or acted upon by other users within the system. The module may employ status indicators, such as timestamps or status labels, to convey document progress and milestones effectively. Additionally, users can set up customizable alerts and notifications to stay informed about significant document events, such as when a document is approved, rejected, or requires further action.

Furthermore, the track module offers robust reporting and analytics capabilities, allowing users to derive valuable insights from document activity data. Users can generate detailed reports summarizing document interactions, access patterns, and user engagement metrics. These reports enable users to identify trends, assess document performance, and make informed decisions to optimize document management processes. Moreover, the track module may support compliance auditing requirements by providing audit trails and documentation of document-related activities. By offering transparency, accountability, and actionable insights, the track module empowers users to effectively monitor, manage, and optimize document workflows within the system, driving efficiency and productivity across the organization

6.5 Logout

The logout module ensures secure termination of user sessions and safeguards system integrity upon user exit. Users can access the logout feature from any page within the system's

interface, typically through a designated logout button or link. Upon initiating the logout process, the module executes a series of actions to ensure comprehensive session termination. First, it invalidates the user's session token, rendering it unusable for subsequent access attempts. Simultaneously, the module clears session data stored in the server's memory, including user authentication credentials and session-specific variables, effectively erasing any traces of the user's presence within the system.

Furthermore, the logout module enhances user experience by providing a seamless transition from active session to logout confirmation. After successfully terminating the session, the module may redirect the user to a logout confirmation page or the system's login page, providing visual feedback that the logout process has been completed. This confirmation step adds an extra layer of assurance to users, assuring them that their session has been securely closed. Additionally, the module may incorporate measures to prevent session fixation attacks or session hijacking attempts, further fortifying system security. By implementing robust logout procedures, the module ensures the confidentiality and integrity of user sessions, mitigates security risks, and upholds user privacy within the system.

CHAPTER 7

SYSTEM TESTING

Testing is the process of executing then programs with then programs with the intention of finding out errors. During the process, the project is executed with the set of tests and the output of the website is evaluated to determine if the project is performing as expected. Testing makes a logical assumption that if all the parts of the module are correct then the goal will be successfully achieved. Testing includes after the completion of the coding phase. The project was tested from the very beginning and also at each step by entering different types of data. In the testing phase some mistakes were found, which did not come to knowledge at the time of coding the project. Then changes were made to the project coding so that it may take all the relevant data and give the required result. All the forms were tested one by one and made all the required changes.

Testing is vital to the success of the system. Testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. A small system error can conceivably explode into a much larger problem. Effective testing early in the process translates directly into long term cost savings from a reduced number of errors. For the verification and validation of data various-nesting tasks are performed. Testing is itself capable of finding the syntactical mistakes in the system for logical checking.

7.1 LEVELS OF TESTING:

The aim of the testing process is to identify all the defects in the website. It is not practical to test the website with respect to each value that the input request data may assume. Testing provides a practical way of reducing defects in the website and increasing the user's confidence in a developed system. Testing consists of subjecting the website to a set of test inputs and observing if the program behaves as expected. If the program fails to test behave as expected then conditions under which failure occurs are noted for later debugging and correction. The following things are associated with testing:

Failure is a manifestation of an error. But the mere presence of an error may not necessarily lead to a failure. A test case is the triplet [I, S, O] where I am data input to the

system. S is the state of the state of the system at which the data is input, O is the expected output of the system, a test suite is the set of all test cases with which a given software product is to be tested.

7.2 FUNCTIONAL TESTING:

Here the system is a black box whose behavior is determined by studying its inputs and related outputs. The key problem is to select the inputs that have a huge probability of being members of a set in May case; the selection of those test cases is based on the previous studies.

7.3 STRUCTURAL TESTING:

A great deal can be learnt about the strength and the limitation of the application by examining the manner in which the system breaks this type of testing has two limitations.

It tests failure behavior of the system; circumstances may arise through an unexpected combination of events where the node placed on the system exceeds the maximum anticipated load.

The structure of each module was checked at every step. Some structures were firstly wrong, which came to notice at the time of the connectivity.

7.4 UNIT TESTING:

In unit testing the entire individual functions and modules were tested independently. By following this Strategy all the errors in coding were identified and corrected. This method was applied in combination with the white and black box testing techniques to find the errors in each module.Unit testing is normality considered an adjunct to the coding step.

Unit test case design was started after source level code had been developed, reviewed, and verified for correct syntax. A review of design information provides guidance for establishing test cases that were likely to uncover errors in each of the categories discussed above. Each test case was coupled with a set of expanded results.

Tested admin login form:-

This form is used for login of administrator of the system. In this we enter the

username and password if both are correct administration page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask for username and password

Test for Admin Module

User Account Addition:-

In this section the admin can verify user details from academic info and then only add student details to the main library database. It contains add and delete buttons if user clicks add button data will be added to student database and if he clicks delete button the student data will be deleted.

Book Addition:-

Admin can enter details of the book and can add the details to the main book table and he can view the books requests.

Test for User Login Module

Test for User Login Form:-

This form is used for login of user. In this we enter the, username and password if all these are correct User login page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask for username and password.

Test for account creation:-

This form is used for new account creation when the user does not fill the form completely, it asks again to fill the whole form when he fill the form fully gets redirected to a page which shows waiting for confirmation message as his data will be only added by administrator after verification.

7.5 INTEGRITY TESTING:-

Integrity phases the entire module using the bottom-up approach and tests them. Integrity testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective was to take unit tested modules and build a program structure that has been dictated by design.

CHAPTER 8
CONCLUSION

With the theoretical inclination of our syllabus it becomes very essential to take the almost advantage of any opportunity of gaining practical experience that comes along. The building blocks of this major project " NEWS APP" was one of those opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of term spirit and coordination in a successful project. The project also provided us the opportunity of interacting with our teachers and to gain from their best experience.

CHAPTER 9

FUTURE ENHANCEMENT

Future enhancements for this project will focus on several key areas to ensure it continues to meet user needs and technological advancements. Integrating advanced analytics and reporting tools will provide deeper insights and customizable reports, enhancing decision-making. Improving the user experience through responsive design, personalization, and multi-language support will make the system more accessible and user-friendly. Optimizing the MySQL database, implementing horizontal scaling, and leveraging cloud integration will boost performance and scalability. Enhancing security with advanced features, compliance with regulations, and robust disaster recovery solutions will protect data integrity and privacy. Expanding the Django API, integrating third-party services, and implementing Single Sign-On (SSO) will enhance interoperability and user convenience. These enhancements will ensure the system remains competitive, secure, and capable of handling increased demand and complexity.

CHAPTER 10
APPENDICES

10.1 React JS Code

```
import React, { useState } from "react";
import logo from '../assets/logo2.png'
import { Link, useNavigate } from "react-router-dom";
import './../App.css'
import "../assets/style.css"

const LoginPage = () => {
  const [loginCreds, setLoginCreds] = useState({ username: "", password: "" });
  const [isPassword, setIsPassword] = useState(true);
  const navigate = useNavigate();

  const inputChange = (e) => {
    const { id, value } = e.target;
    setLoginCreds((prevState) => ({ ...prevState, [id]: value }));
  };

  const onEyeClick = () => {
    setIsPassword(!isPassword);
  };

  const loginHandler = (e) => {
    e.preventDefault();
    if (loginCreds?.username && loginCreds?.password) {
      navigate("/dashboard");
    }
```

```
};

return (
  <div className="container d-flex align-items-center justify-content-center vh-100">
    <div className="account-box">
      <div className="account-logo">
        {/* <img src={logo} alt="Logo" /> */}
      </div>
      <div className="account-wrapper">
        <div>
          <form onSubmit={(e) => loginHandler(e)}>
            <div form-group>
              <div className="input-group mb-3">
                <span className="input-group-text">
                  <i className="bi bi-person-fill" />
                </span>
                <input
                  type="text"
                  className="form-control"
                  id="username"
                  placeholder="Enter your username"
                  value={loginCreds?.username}
                  onChange={inputChange}
                  required={true}
                />
              </div>
            </div>
            <div form-group>
              <div className="input-group mb-3">
                <span className="input-group-text">
                  <i className="bi bi-lock-fill" />
```

```
    </span>
    <input
      type={isPassword ? "password" : "text"}
      className="form-control"
      id="password"
      placeholder="Enter your password"
      value={loginCreds?.password}
      onChange={inputChange}
      required={true}
    />
    <span className="input-group-text">
      <i
        className={isPassword ? "bi bi-eye-slash" : "bi bi-eye"}
        onClick={onEyeClick}
      />
    </span>
  </div>
</div>
<div className="row">
  <div className="text-right">
    <p className="text-body-secondary " to="/forgotpassword">
      Forgot password?
    </p>
  </div>
</div>
{/* <br/> */}
<div className="form-group text-center">
  <button type="submit" className="btn btn-primary account-btn">
    Login
  </button>
</div>
```

```
          </form>
        </div>
      </div>
    </div>
  </div>
 );
};

export default LoginPage;

import React, { useEffect, useState } from "react";
import { Layout, Menu } from "antd";
import { Link, useLocation, useNavigate } from "react-router-dom";
import "../assets/style.css";

const { Sider } = Layout;

const Sidebar = ({ collapsed }) => {
  const location = useLocation();
  const navigate = useNavigate();
  const [selectedKeys, setSelectedKeys] = useState(["dashboard"]);
  const [openKeys, setOpenKeys] = useState([]); // Default open keys
  // const [collapsed, setCollapsed] = useState(false);

  // const toggleCollapsed = () => {
  //   setCollapsed(!collapsed);
  // };

  useEffect(() => {
    // Retrieve selected key and open keys from storage
    const storedSelectedKey = localStorage.getItem("selectedKey");
```

```
  const storedOpenKeys = JSON.parse(localStorage.getItem("openKeys")) || [];

  if (storedSelectedKey) {
    setSelectedKeys([storedSelectedKey]);
  }

  if (storedOpenKeys) {
    setOpenKeys(storedOpenKeys);
  }
}, []);

const handleMenuClick = ({ key, keyPath }) => {
  // Store selected key and open keys in storage
  localStorage.setItem("selectedKey", key);
  localStorage.setItem("openKeys", JSON.stringify(keyPath));
  setSelectedKeys([key]);
  setOpenKeys(keyPath);
};

return (
  <Sider
    style={{
      minHeight: "100vh",
      // overflow: "auto",
      // position: "fixed",
      // left: 0,
      // zIndex: 1,
    }}
    trigger={null}
    collapsible
    collapsed={collapsed}
```

```
  // onCollapse={toggleCollapsed}
  breakpoint="lg"
>
  <div className={collapsed ? "sidebar-box-collapsed" : "sidebar-box"}>
    <div className="account-logo">
      {/* <img src={logo} alt="Logo" /> */}
    </div>
  </div>
  <Menu
    theme="dark"
    mode="inline"
    // defaultSelectedKeys={["1"]}
    selectedKeys={selectedKeys}
    openKeys={openKeys}
    onOpenChange={(keys) => setOpenKeys(keys)}
    onSelect={handleMenuClick}
    items={[
      {
        key: "dashboard",
        icon: (
          <span>
            <i class="bi bi-speedometer" style={{ fontSize: "20px" }} />
          </span>
        ),
        label: "Dashboard",
        onClick: () => navigate("/dashboard"),
      },
      {
        key: "inward-upload",
        icon: (
          <span>
```

```
      <i
        class="bi bi-diagram-3-fill"
        style={{ fontSize: "20px" }}
      ></i>
    </span>
  ),
  label: "Inward Upload",
  children: [
    {
      key: "add-new",
      label: "Add New",
      onClick: () => navigate("/inward-upload"),
    },
    {
      key: "list-all",
      label: "List All",
      onClick: () => navigate("/inward-lists"),
    },
  ],
},
{
  key: "3",
  icon: (
    <span>
      <i
        class="bi bi-diagram-3-fill"
        style={{ fontSize: "20px" }}
      ></i>
    </span>
  ),
  label: "Outward Upload",
```

```
  children: [
    {
      key: "o1",
      label: "Add New",
      onClick: () => navigate("/dashboard"),
    },
    {
      key: "o2",
      label: "List All",
      onClick: () => navigate("/dashboard"),
    },
  ],
},
{
  key: "users-departments",
  icon: (
    <span>
      <i
        class="bi bi-diagram-3-fill"
        style={{ fontSize: "20px" }}
      ></i>
    </span>
  ),
  label: "Users & Departments",
  children: [
    {
      key: "users",
      label: "Users",
      onClick: () => navigate("/users-lists"),
      icon: (
        <span>
```

```
                <i
                  className="bi bi-people-fill"
                  style={{ fontSize: "20px" }}
                ></i>
              </span>
            ),
          },
          {
            key: "departments",
            label: "Departments",
            onClick: () => navigate("/departments-lists"),
            icon: (
              <span>
                <i
                  className="bi bi-building-fill-add"
                  style={{ fontSize: "20px" }}
                ></i>
              </span>
            ),
          },
        ],
      },
    ]}
  />
 </Sider>
 );
};

export default Sidebar;

import React, { useContext, useState } from "react";
```

```
import { Layout, Menu, Button } from "antd";
import { Link } from "react-router-dom";
// import { MenuOutlined } from "@ant-design/icons";
import "../assets/style.css"
import AuthContext from "../store/AuthContext";

const { Header } = Layout;

const CustomHeader = ({ toggleSidebar }) => {
  // const { onLogout } = useContext(AuthContext)

  const onLogout = () => {
    localStorage?.clear()
  }
  return (
    <Header className="header">
      <div className="container-fluid">
        <div className="row align-items-center">
          <div
            className="col-auto"
            onClick={toggleSidebar}
            style={{
              fontSize: "30px",
              width: 64,
              height: 64,
              marginLeft: "0 16px",
              cursor: "pointer",
            }}
          >
            <i class="bi bi-justify" style={{ color: "cyan" }} />
          </div>
```

```
        <div className="col text-center">
          <h1 className="text-white">Inward Register</h1>
        </div>
        <div className="col-auto ml-auto">
          <div className="nav-item dropdown" style={{ color: "whitesmoke" }}>
            <div
              className="dropdown-toggle"
              type="button"
              data-bs-toggle="dropdown"
            >
              Admin
            </div>
            <div className="dropdown-menu dropdown-menu-right">
              <Link to="/" className="dropdown-item" onClick={onLogout}>Logout</Link>
            </div>
          </div>
        </div>
      </div>
    </Header>
  );
};

export default CustomHeader;
```

10.2 Django Code:

users/views.py

```
from django.shortcuts import render
from django.http import HttpResponse, JsonResponse
from users.decorators import admin_required
from users.models import UserRole, Department
from django.views.decorators.csrf import csrf_exempt

from users.models import User, UserRole
# Create your views here.

@csrf_exempt
def admin_department_creation(request):
    if request.method != 'POST':
        return JsonResponse({"message": "Method not allowed"}, status=405)
    name = request.POST.get('department_name')
    depart_type = request.POST.get('department_type')

    if name and depart_type:
        try:
            if depart_type == Department.SELF:
                depart_type = Department.SELF
            elif depart_type == Department.AIDED:
                depart_type = Department.AIDED
            else:
                return JsonResponse({"message": "Department type match not found"}, status=404)
```

```
            department, create = Department.objects.get_or_create(department_name=name,
department_type=depart_type)
            if create:
                return JsonResponse({"message": f"{name} - {depart_type} Department created
successfully."}, status=200)
            else:
                return JsonResponse({"message": f"{name} - {depart_type} Department already
exist."}, status=400)
        except Exception as e:
            return JsonResponse({"message": f"Error: {e}"})
    else:
        return JsonResponse({"message": "POST parameters missing"}, status=400)

@csrf_exempt
def department_list(request):
    if request.method != 'GET':
        return JsonResponse({"message": "Method not allowed"}, status=405)

    departments = [Department.SELF, Department.AIDED]
    department_dict = []

    for dept_type in departments:
        data =
list(Department.objects.filter(department_type=dept_type).values('department_name',
'department_type'))
        for dept in data:
            department_dict.append({"name": dept['department_name'], "type":
dept['department_type']})

    return JsonResponse(department_dict, status=200, safe=False)
```

```
@csrf_exempt
def user_creation_choice_field(request):
    if request.method != 'GET':
        return JsonResponse({"message": "Method not allowed"}, status=405)
    type_of = request.GET.get('type')
    department_dict = []

    if 'department' in type_of.lower():
        departments = [Department.SELF, Department.AIDED]
        data =
list(Department.objects.filter(department_type__in=departments).values('department_name',
'department_type'))
        for dept in data:
            department_dict.append({"name": dept['department_name'], "type":
dept['department_type']})
    if 'cell' in type_of.lower():
        data =
list(Department.objects.filter(department_type=Department.Cell).values('department_name',
'department_type'))
        for dept in data:
            department_dict.append({"name": dept['department_name'], "type":
dept['department_type']})
    if 'committee' in type_of.lower():
        data =
list(Department.objects.filter(department_type=Department.Committee).values('department_nam
e', 'department_type'))
        for dept in data:
            department_dict.append({"name": dept['department_name'], "type":
dept['department_type']})
    return JsonResponse(department_dict, safe=False, status=200)
```

```
@csrf_exempt
def choice_field(request):
    if request.method != 'GET':
        return JsonResponse({"message": "Method not allowed"}, status=405)

    type_of = request.GET.get('type')
    department_dict = []

    if 'department' in type_of.lower():
        self_departments =
list(Department.objects.filter(department_type=Department.SELF).values('department_name'))
        aided_departments =
list(Department.objects.filter(department_type=Department.AIDED).values('department_name'))
        department_dict.append({
            "name": "Self",
            "children": [{"name": dept['department_name']} for dept in self_departments]
        })
        department_dict.append({
            "name": "Aided",
            "children": [{"name": dept['department_name']} for dept in aided_departments]
        })

    if 'cell' in type_of.lower():
        cell_departments =
list(Department.objects.filter(department_type=Department.Cell).values('department_name'))
        department_dict.append({
            "name": "Cells",
            "children": [{"name": dept['department_name']} for dept in cell_departments]
```

```
        })

    if 'committee' in type_of.lower():
        committee_departments =
list(Department.objects.filter(department_type=Department.Committee).values('department_nam
e'))
        department_dict.append({
            "name": "Committee",
            "children": [{"name": dept['department_name']} for dept in committee_departments]
        })

    return JsonResponse(department_dict, safe=False, status=200)



@csrf_exempt
@admin_required
def user_creation(request):
    if request.method != 'POST':
        return JsonResponse({"message": "Method not allowed"}, status=405)

    email = request.POST.get('email')
    role = request.POST.get('role')
    password = request.POST.get('password')
    department_type = request.POST.get('department_type')
    department = request.POST.get('department')
    name = request.POST.get('user_name')

    # Validate the input
    if not email or not role or not password or not name:
        return JsonResponse({"message": "Missing required fields"}, status=400)
```

```
    # Check if the role is valid
    if role not in dict(UserRole.ROLES_CHOICE).keys():
        return JsonResponse({"message": "Invalid role"}, status=400)

    # Create the user
    try:
        if role.lower() == "admin":
            user = User.objects.create_superuser(username=email, email=email, password=password,
first_name=name)
        else:
            user = User.objects.create_user(username=email, email=email, password=password,
first_name=name)
    except Exception as e:
        return JsonResponse({"message": f"Error creating user: {str(e)}"}, status=400)

    # Create the user role
    try:
        if role == 'Department':
            department_obj = Department.objects.filter(department_type=department_type,
department_name=department)
            if department_obj.exists():
                department_obj = department_obj.first()
                user_role = UserRole.objects.create(user=user, email=email, role=role,
department=department_obj)
            else:
                return JsonResponse({"message": "Department details is missing"}, status=400)
        else:
            user_role = UserRole.objects.create(user=user, email=email, role=role)
    except Exception as e:
        # If creating user role fails, delete the created user
        user.delete()
```

```
        return JsonResponse({"message": f"Error creating user role: {str(e)}"}, status=400)

    return JsonResponse({"message": "User and role created successfully"}, status=201)
```

login/views.py:

```
from django.contrib.auth import authenticate
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from rest_framework.authtoken.models import Token
from users.models import UserRole  # Make sure to adjust this import based on your actual
models
import json

@csrf_exempt
def login_view(request):
    if request.method == 'POST':
        try:
            # data = json.loads(request.body)
            email = request.POST.get('email')
            password = request.POST.get('password')
            email = email.strip() if email else None

            user = authenticate(request, username=email, password=password)

            if user is not None:
                token, created = Token.objects.get_or_create(user=user)
                user_details = UserRole.objects.filter(user=user).values(
                    'user__id', 'user__email', 'role', 'department__department_name',
'department__department_type', 'user__first_name'
                ).first()
```

```
            if user_details:
                data = {
                    "user_id": user_details.get('user__id'),
                    "user_name": user_details.get('user__first_name'),
                    "email": user_details.get('user__email'),
                    "role": user_details.get('role'),
                    "department_type": user_details.get('department__department_type'),
                    "department_name": user_details.get('department__department_name'),
                    "token": token.key
                }
                return JsonResponse({'message': 'Login successful', 'data': data}, status=200)
            else:
                return JsonResponse({'message': 'User role details not found'}, status=404)
        else:
            return JsonResponse({'message': 'Invalid credentials'}, status=401)
    except json.JSONDecodeError:
        return JsonResponse({'message': 'Invalid JSON'}, status=400)
return JsonResponse({'message': 'Only POST method is allowed'}, status=405)
```

users/urls.py:

```
from users.views import admin_department_creation, choice_field, user_creation,
department_list
from django.urls import path

urlpatterns = [
```

```
    path('department/create', admin_department_creation, name='department_creation'),
    path('choice_field/', choice_field, name='choice_field'),
    path('create-user/', user_creation, name='create-user'),
    path('department/list', department_list, name="department_list")
]
```

inward_register/urls.py

```
"""
URL configuration for inward_register project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('new/', include('users.urls')),

]
```

/inward_register/settings.py:

```
"""
Django settings for inward_register project.

Generated by 'django-admin startproject' using Django 5.0.3.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-xthyg4-dsjgkg_mc3h%i$n#lq-__sklj*71w39c%o$faf_b-q@'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'users',
    'login'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'inward_register.urls'

TEMPLATES = [
    {
```

```
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'inward_register.wsgi.application'



# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': BASE_DIR / 'db.sqlite3',
#     }
# }


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
```

```
        'NAME': 'mydb',
        'USER': 'root',
        'PASSWORD': 'password',
        'HOST':'localhost',
        'PORT':'3306',
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

10.3 SNAPSHOT

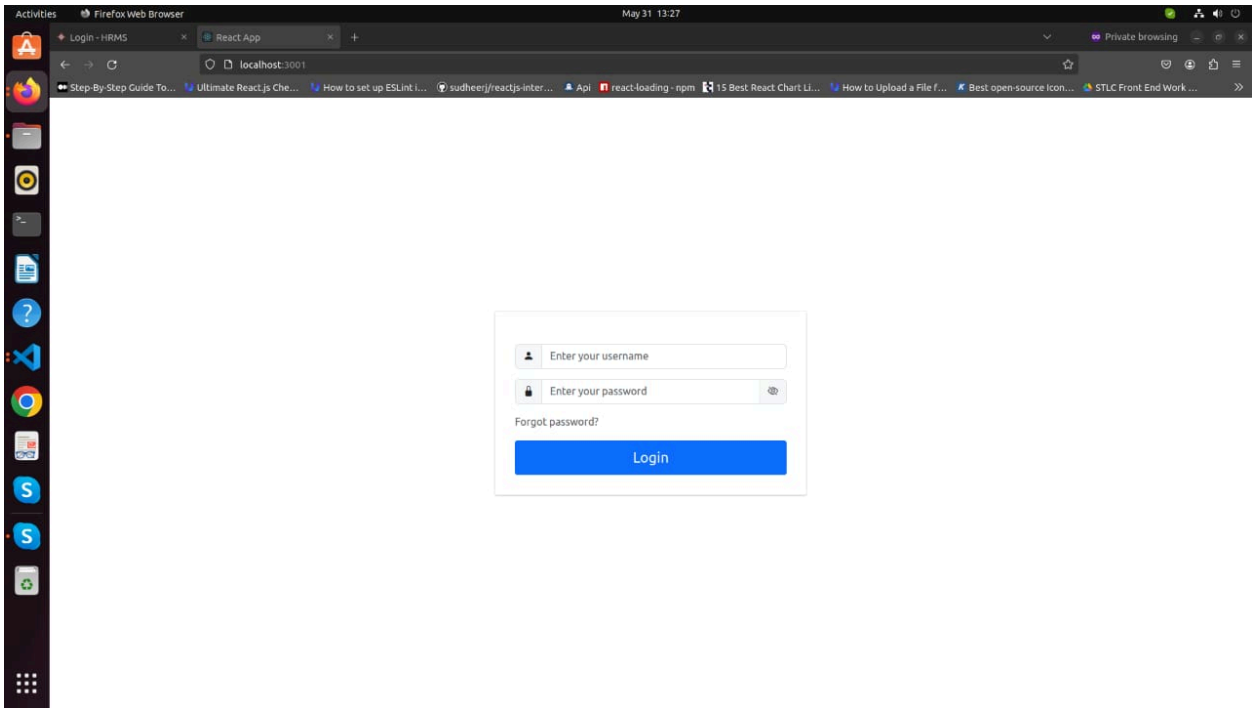

Fig 10.3.1 Login

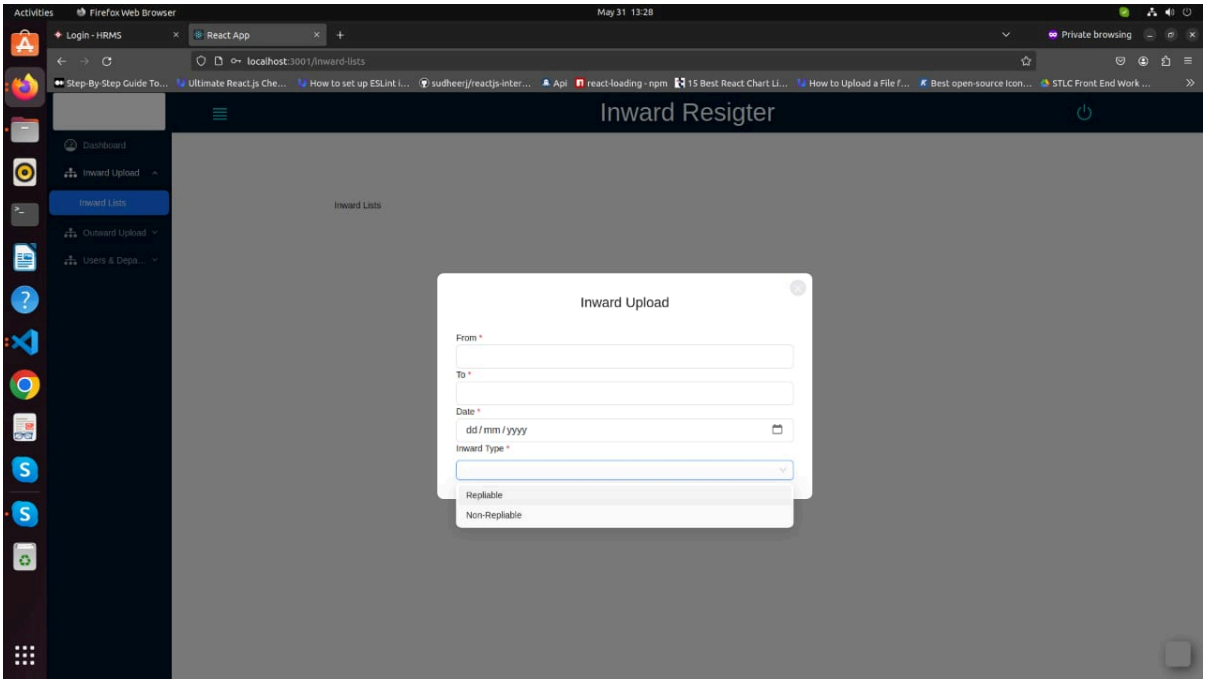

Fig 10.3.2 Inward creation

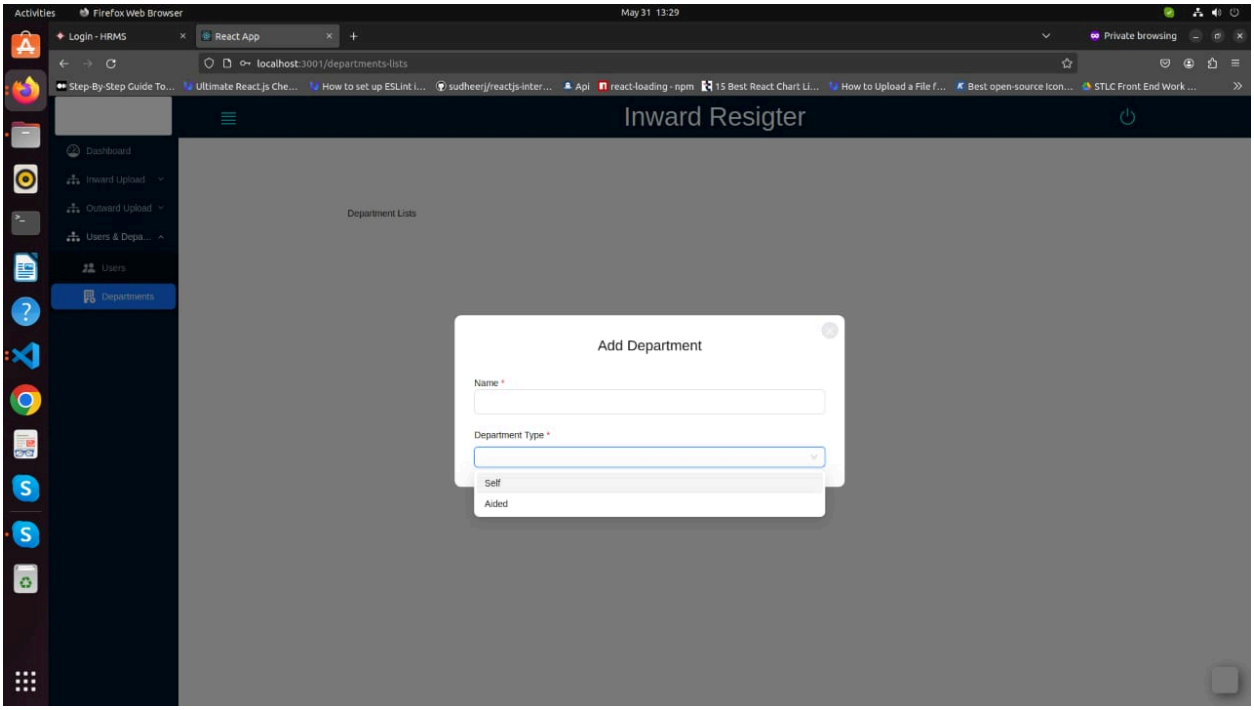

Fig 10.3.3 Admin create user

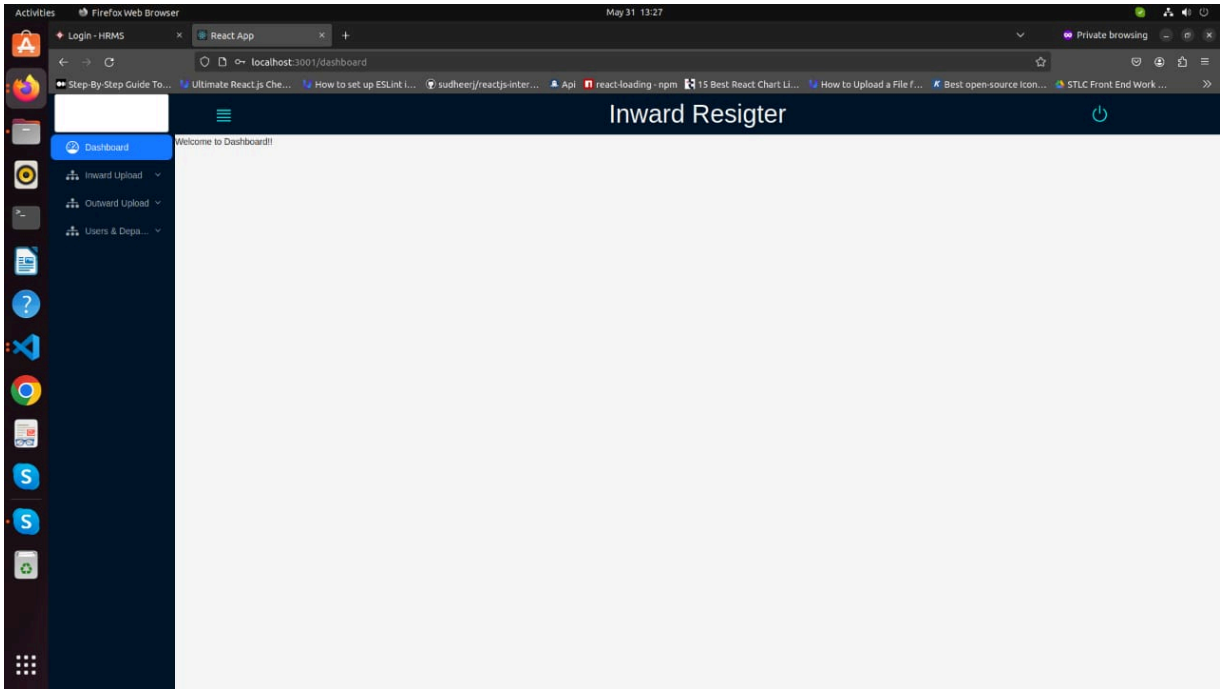

Fig 10.3.4 Dashboard

CHAPTER 11

REFERENCES

https://youtu.be/qFd1_4uVtsg

https://www.w3schools.com/css/default.asp

https://www.youtube.com/channel/UCWv7vMbMWH4-V0ZXdmDpPBA

https://www.dcpehvpm.org/E-Content/BCA/BCA-II/Web%20Technology/the-complete-reference-html-css-fifth-edition.pdf

https://education.fsu.edu/wp-content/uploads/2015/04/Learning-PHP-MySQL-JavaScript-and-CSS-2nd-Edition-1.pdf