

SHARE TO

**A PROJECT REPORT
for
Mini Project (KCA353)
Session (2024-25)**

Submitted by

**Vishal Sharma
(2300290140205)
Yatharth Kaushik
(2300290140212)
Tapasya
(2300290140193)
Shivam Singh
(2300290140171)**

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Ms. Annu Yadav
(Assistant Professor)**



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(DECEMBER 2024)

CERTIFICATE

Certified that **Vishal Sharma (2300290140205), Yatharth Kaushik (2300290140212), Shivam Singh (2300290140171), Tapasya (2300290140193)** have carried out the project work having “**Share-To**” (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

MS. Annu Yadav
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Share-To
(Vishal Sharma)
(Yatharth Kaushik)
(Shivam Singh)
(Tapasya)

ABSTRACT

Share-To is a web-based application designed to provide a secure, efficient, and user-friendly platform for sharing files over the internet. Built using React.js for the frontend, this project leverages modern web technologies to ensure seamless performance and an interactive user experience.

The primary objective of Share-To is to enable users to upload, download, and share files with other users or groups while maintaining high standards of data security and accessibility. Key features include drag-and-drop file upload functionality, secure file encryption for data privacy, and dynamic sharing options via unique links or user invitations.

The backend of the application can be powered by Node.js with a MongoDB database to store file metadata and user details, ensuring scalability and real-time data synchronization. Additionally, cloud storage services like AWS S3 or Firebase can be integrated to handle large files efficiently.

This project is ideal for scenarios where team collaboration, document sharing, or personal file storage is needed, providing a streamlined alternative to traditional file-sharing platforms. By utilizing React's modular and component-based architecture, Share-To ensures flexibility, reusability, and enhanced development experience.

Share-To stands as a modern solution for file-sharing needs, highlighting the potential of React.js in creating dynamic, scalable, and user-centric applications.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Annu Yadav** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Vishal Sharma

**Yatharth
Kaushik**

Tapasya

Shivam Singh

TABLE OF CONTENTS

Certificate	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v-vi
List of Figures	vii
1 Introduction	1-3
1.1 Overview	1
1.2 Problem Statement	1
1.3 Objective	1
1.4 Scope	2
1.5 Features	2
1.6 Background	3
2 Feasibility Study	4-7
2.1 Economic Feasibility	4
2.2 Technical Feasibility	5
2.3 Operational Feasibility	5
2.4 Behavioral feasibility	6
3 Software requirement specification	8-14
3.1 Functionalities	8
3.2 User and Characteristics	9
3.3 Features of project	10
3.4 Feature of Admin	11
3.4 Features of User	12
4 System Requirement	15-21
4.1 Functional Requirement	15
4.2 Non-Functional Requirement	16
4.3 Design Goal	17
4.4 System Sequence Diagram	18
4.5 ER Diagram for a File Transfer	20
5 System Design	22-24
5.1 Primary Design Phase	22
5.2 Secondary Design Phase	23
5.3 User Interface	23
6 Architecture	25-28
6.1 Layered Architecture	25
6.2 Real Time Update	26

6.3	Security Architecture	27
7	Project Screenshots	29-32
8	Code Screenshots	33-35
9	Conclusion	36

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
4.1	System Sequence Diagram	18
4.2	E-R Diagram	21
7.1	Dashboard Page	28
7.2	Login Page	28
7.3	Signup Page	29
7.4	User Page	29
7.5	Help and Support	30
7.6	Sender Page	30
7.7	Receiver Page	31
8.1	User Schema	33
8.2	Router	34
8.3	Protected Auth	34
8.4	User Controller	35

CHAPTER 1

INTRODUCTION

1.1 Overview

Share-To is a cutting-edge peer-to-peer (P2P) desktop application designed to redefine how users transfer files between two personal computers. Unlike traditional methods that depend on external storage devices or cloud-based services, Share-To offers a direct, serverless solution for file sharing. The application leverages the power of the MERN stack (MongoDB, Express.js, React.js, Node.js) alongside advanced technologies such as WebRTC for real-time communication, Socket.IO for event-driven data exchange, and JWT (JSON Web Token) for secure and robust user authentication.

With Share-To, users can experience fast, efficient, and secure file transfers within local networks or over the internet, bypassing the risks and limitations of third-party involvement. The app ensures privacy by preventing file data from being stored on external servers. It combines simplicity with high functionality, providing an intuitive interface suitable for users with varying levels of technical expertise. By addressing common challenges like security risks, device dependency, and complex user workflows, Share-To delivers a seamless and dependable platform for modern file-sharing needs.

1.2 Problem Statement

In the rapidly evolving landscape of online commerce, there is a growing demand for robust and scalable e-commerce platforms. Traditional methods of shopping are being replaced by digital experiences, creating a need for feature-rich and user-friendly online stores. Existing e-commerce solutions often come with limitations, making it challenging for businesses to tailor their platforms to specific needs and requirements.

1.3 Objectives

The main objective of the Project on Online E-commerce System is to manage the details of Clothes, Item Category, Shopping Cart, Customer, Order. It manages all the information about Clothes, Delivery Address, and Order. The project is totally built at administrative end and thus only the administrator is guaranteed the access.

1.4 Scope

- The scope of the "ShareTo" project encompasses the development of a web-based solution that enables seamless wireless file transfer between two personal computers (PCs) over a local network or Wi-Fi. The application will utilize Socket I/O technology for real-time communication, ensuring fast, secure, and reliable file transfers without the need for physical devices such as USB drives or external storage solutions. The project aims to create a solution that is efficient, user-friendly, and accessible to individuals, small businesses, or teams that require easy and quick data sharing.
- Key features within the scope of the project include:
- **Wireless File Transfer:** The core functionality will allow users to transfer files between two PCs connected to the same local network or Wi-Fi without cables or external hardware.
- **Socket I/O Integration:** The use of Socket I/O will enable real-time data transmission, ensuring fast and continuous communication between the devices, which is key to optimizing transfer speeds and minimizing latency.
- **User Interface (UI):** The application will feature an intuitive, web-based interface designed for simplicity and ease of use, allowing users to drag-and-drop files and manage transfers effortlessly.
- **Security Measures:** The solution will incorporate secure file transfer protocols, such as encryption, to protect user data during the transfer process. Authentication mechanisms will also be put in place to prevent unauthorized access.
- **Cross-Platform Compatibility:** The web application will be designed to work across different operating systems, including Windows, macOS, and Linux, as long as both devices are connected to the same network.
- **File Size and Type Support:** The application will support a wide range of file types, including documents, images, videos, and more, with optimizations to handle large files efficiently.
- **Performance Optimization:** The transfer speed and stability will be optimized to ensure smooth and fast transfers, even under variable network conditions.

By meeting these objectives, "ShareTo" will address the challenges of traditional data-sharing methods and offer an innovative, wireless solution for quick and secure file sharing.

1.5 Feature

1. Wireless file transfer over a local network or Wi-Fi without the need for cables or external devices.
2. Real-time communication using Socket I/O technology for fast and efficient data transmission between devices.
3. A user-friendly, web-based interface designed for easy drag-and-drop file selection and transfer management.

4. Cross-platform compatibility, supporting various operating systems like Windows, macOS, and Linux without requiring special software installation.
5. Support for transferring files of varying sizes, from small documents to large media files.
6. Handling a wide range of file types, including images, videos, documents, and more.
7. High-speed transfer optimized to minimize latency and ensure smooth data sharing.
8. Secure file transfer with encryption and security protocols to protect privacy and data integrity.
9. Authentication mechanism to ensure that only authorized devices can initiate and receive transfers.
10. File transfer queue for batch transfers without interruptions.
11. Robust error handling and progress tracking to manage failed or incomplete transfers.
12. Pause and resume functionality for flexibility in case of interruptions.
13. Network efficiency optimization to maintain stable performance under varying network conditions.

1.6 Background

With the growing reliance on digital communication and the increasing volume of data exchanged daily, efficient data transfer methods have become essential. Traditional methods of file sharing, such as using USB drives, external hard disks, or email attachments, often come with limitations. These include the need for physical devices, slow transfer speeds, security risks, and file size constraints. Additionally, users often face challenges when sharing files between different devices or operating systems.

In response to these challenges, the demand for wireless data transfer solutions has surged. Wireless technologies, including Wi-Fi and Bluetooth, have revolutionized how we connect devices, but many existing solutions still lack the convenience, speed, or security needed for seamless file transfers.

The "ShareTo" project was conceived to address these limitations by providing an easy-to-use, efficient, and secure wireless data transfer solution. By leveraging Socket I/O technology and a web-based interface, the project aims to enable fast and reliable file sharing between two personal computers (PCs) connected to the same local network or Wi-Fi. The solution eliminates the need for external devices or software installation while ensuring robust security measures for user data privacy.

CHAPTER 2

FEASIBILITY STUDY

2.1 ECONOMICAL FEASIBILITY

Economic feasibility assesses whether the project can be implemented within a reasonable budget without compromising on quality or performance. This project leverages cost-effective solutions by relying on open-source technologies such as **React JS**, **Node JS**, and **MongoDB**, which significantly reduce licensing expenses.

1. Open-Source Advantages:

- **React JS** is used for building dynamic and interactive frontends without incurring any licensing fees.
- **Node JS** and **Express** provide a powerful yet lightweight backend, eliminating the need for expensive enterprise-grade solutions.
- **MongoDB**, a NoSQL database, handles data efficiently and offers flexible hosting plans, minimizing operational costs.

2. Minimal Hardware Requirements:

- End-users require only basic hardware to access the platform, making it accessible to a broader audience, including individuals and small businesses.
- This approach ensures affordability and promotes widespread adoption.

3. Cost Breakdown:

- **Hosting Services:** The platform uses cloud hosting solutions, which are economical and scalable based on usage.
- **Development Efforts:** The project involves moderate costs for development and testing but remains within manageable limits, especially due to the use of open-source frameworks.

By focusing on open-source tools and minimal hardware requirements, the project achieves a balance between affordability and high-quality performance.

2.2 TECHNICAL FEASIBILITY

The technical feasibility of the project focuses on whether the proposed technologies and architecture can meet the platform's requirements for scalability, reliability, and performance.

1. Technical Stack:

- **React JS:** Provides dynamic, interactive frontends, ensuring a responsive and engaging user experience.
- **Node JS and Express:** Offer a lightweight backend capable of handling concurrent users and processing requests efficiently.
- **MongoDB:** A scalable and flexible database solution that manages large datasets seamlessly, ensuring smooth data retrieval and storage.

2. Real-Time Capabilities:

- **Socket.IO** enables real-time updates for file-sharing activities, such as notifications for uploads or downloads.
- **WebRTC** supports peer-to-peer file sharing, providing a fast and efficient mechanism for transferring data.

3. Cross-Platform Compatibility:

- The platform is designed to work across multiple operating systems and devices, including desktops, smartphones, and tablets.
- Its compatibility ensures that users can access the system from anywhere, making it convenient for both individual and enterprise users.

With a robust technical foundation, the platform ensures smooth deployment and reliable performance, catering to a wide range of users and use cases.

2.3 OPERATIONAL FEASIBILITY

Operational feasibility assesses how well the platform can be integrated into users' daily workflows and whether it meets their operational needs effectively.

1. User-Friendly Design:

- The platform includes intuitive modules such as dashboards, file history logs, and help/support sections to enhance usability.
- The navigation is designed to be simple and self-explanatory, minimizing the need for training or technical expertise.

2. Streamlined Workflows:

- Users can perform core functions like uploading, sharing, and tracking files with minimal effort.
- The system's operational flows ensure seamless execution of tasks, saving time and improving productivity.

3. Accessibility:

- The platform is lightweight and requires minimal system resources, ensuring that even users with older hardware can access its features without any issues.

By focusing on ease of use and accessibility, the platform ensures that users of all skill levels can adopt and benefit from its features without requiring specialized training.

2.4 BEHAVIOURAL FEASIBILITY

Behavioural feasibility examines how well the system aligns with user preferences and promotes positive interaction and engagement.

1. User-Centric Design:

- The platform prioritizes simplicity and usability through a clean interface and personalized user experience.
- Features like activity history, file notifications, and role-based access controls allow users to feel in control of their data and actions.

2. Personalization and Engagement:

- Role-based dashboards cater to different user types, such as admins and general users, providing tailored functionalities.
- Help modules and real-time feedback ensure that users remain engaged and confident in using the platform.

3. Enhancing User Behaviour:

- The inclusion of interactive elements, such as file-sharing notifications and activity tracking, encourages users to adopt the platform for regular use.
- Positive user experiences are further enhanced by support systems and an overall focus on convenience.

Behavioural feasibility emphasizes creating a system that not only meets technical requirements but also fosters long-term user satisfaction and engagement.

CONCLUSION

The feasibility study demonstrates that the platform is economically viable, technically robust, operationally efficient, and behavioural engaging. By addressing all critical aspects, the project lays a solid foundation for successful implementation and widespread adoption. This detailed feasibility ensures that the platform meets the diverse needs of its users while maintaining scalability and adaptability for future enhancements.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

The product specified in the SRS document is a software application for seamless peer-to-peer file sharing between two personal computers. Unlike traditional methods reliant on external storage or third-party cloud services, ShareTo delivers a direct, encrypted, and highly secure file transfer solution.

This application is built to cater to the modern need for instantaneous and secure file sharing without compromising user privacy or data integrity. ShareTo can be accessed via a desktop environment, ensuring that users can quickly and efficiently exchange files, whether on a local network or across the internet. The application's focus is on creating a frictionless and private sharing experience, eliminating the inconvenience of physical storage devices and the security risks of cloud-based transfers.

3.1 Functionalities:

The ShareTo application is designed to facilitate secure and seamless peer-to-peer file-sharing operations for both users and administrators. Let's break down the key components and functionalities:

- **File Sharing Setup:** The system enables users to establish secure file-sharing sessions, allowing direct communication between two PCs via WebRTC.
- **Real-Time File Transfer:** Powered by WebRTC and Socket.IO, the application ensures fast and reliable data exchange, supporting small to large file sizes.
- **Secure Authentication:** The application employs JWT for user authentication, ensuring that only authorized users can initiate file-sharing sessions.
- **Session Management:** Users can start, pause, resume, and terminate file-sharing sessions with complete control over the transfer process.
- **Cross-Network Connectivity:** The system facilitates file sharing over both local networks and the internet without relying on third-party servers.

- **User-Friendly Interface:** A simple, intuitive interface allows users of all technical levels to navigate and utilize the application efficiently.

These functionalities ensure that ShareTo is a robust, secure, and user-friendly solution for modern file-sharing needs.- **File Sharing Setup:** The system enables users to set up secure file-sharing sessions, where files can be shared directly between two PCs using WebRTC.

- **Real-Time Communication:** Powered by WebRTC and Socket.IO, the system ensures fast and reliable data exchange during file transfer sessions.
- **User Authentication:** The application uses JWT for secure user login and authentication, ensuring only authorized users can access file-sharing features.
- **Cross-Network Connectivity:** Share files over local networks or the internet without intermediary servers.
- **Session Management:** Users can start, pause, resume, and terminate file-sharing sessions with ease.

3.2 User and Characteristics:

For this application, two types of users are defined:

1. **Standard User:** Individuals who initiate and receive file transfers using the ShareTo application.
 - Can register and authenticate securely.
 - Initiates secure peer-to-peer file transfer sessions.
 - Manages session controls including pause, resume, and termination.
 - Can access transfer history and logs for personal reference.
2. **Administrator:** Responsible for managing and maintaining the application environment.
 - Oversees user activity and ensures system integrity.
 - Configures system-level settings and logs.
 - Manages security protocols, including encryption and user permissions.
 - Monitors real-time transfer activities for system performance.
 - Has authority to troubleshoot and resolve technical issues.

Additional Characteristics

- **User Accessibility:** Both types of users can access the application via desktop interfaces compatible with Windows, macOS, or Linux.
- **Secure Interactions:** All interactions between users and the system are secured through JWT authentication and data encryption.
- **User Feedback Integration:** Provides avenues for users to report issues or suggest improvements, enhancing system usability and reliability.

These characteristics ensure that ShareTo offers an efficient and secure environment tailored to diverse user needs.

3.3 Features of the project:

- **User Registration and Login:** Provides users with the ability to create unique accounts and log in securely. The system uses JWT for authentication to ensure secure sessions and protect user data.
- **Session Encryption:** Employs end-to-end encryption to safeguard file transfers, ensuring data remains private and inaccessible to unauthorized parties.
- **Real-Time Status Updates:** Offers live updates on the progress of file transfers, including file size, time remaining, and transfer completion.
- **Cross-Platform Support:** Ensures compatibility with popular operating systems such as Windows, macOS, and Linux, allowing for broad accessibility.
- **Error Handling:** Incorporates robust mechanisms to detect, report, and recover from errors during file transfers, minimizing disruptions.
- **Drag and Drop Interface:** Simplifies the user experience by allowing files to be selected and added for transfer through an intuitive drag-and-drop feature.
- **Multiple File Support:** Enables simultaneous transfer of multiple files, streamlining the process for users with larger workloads.
- **Bandwidth Optimization:** Dynamically adjusts transfer speeds based on network conditions to maintain stable performance and reduce bottlenecks.
- **Offline Mode:** Supports pre-configuration of file transfer sessions, which activate automatically when both users are online, providing flexibility and convenience.
- **Notification System:** Sends alerts and notifications for session status updates, including completion or failure, ensuring users remain informed at all times.
- **Customizable Settings:** Allows users to personalize settings such as transfer speed limits, default file save locations, and notification preferences, tailoring the experience to their needs.
- **Session Expiry Management:** Automatically detects and terminates inactive sessions, enhancing security and optimizing system resources.
- **Integration with System Storage:** Seamlessly integrates with the system's file explorer to facilitate quick and easy file selection and organization.

- **File Size Optimization:** Handles large file transfers efficiently without compromising speed or data integrity, making it ideal for sharing heavy multimedia or datasets.
- **Session Resume Support:** Allows interrupted transfers to restart from the point of failure, saving time and preventing redundancy.
- **Cross-Device Compatibility:** Supports file transfers between devices with different operating systems, ensuring seamless interoperability for users.
- **Auto-Update Mechanism:** Keeps the application up-to-date with the latest features, performance improvements, and security patches, ensuring reliability.
- **User Analytics:** Tracks usage patterns and user interactions to help developers improve the application and enhance user experience.
- **File Preview and Metadata Display:** Provides users with a preview of files and associated metadata such as size, type, and date before initiating transfers, ensuring accuracy and clarity.
- **Offline Mode:** Allows for pre-setup of file transfer sessions that initiate automatically when both parties are online.
- **Notification System:** Provides real-time notifications for session status updates, completion, or errors.
- **Customizable Settings:** Users can configure preferences such as transfer speed limits, default file save locations, and notification settings.
- **Session Expiry Management:** Automatically terminates idle sessions to enhance security and resource management.
- **Integration with System Storage:** Seamlessly integrates with the system's file explorer for easy file selection and management.

3.4 Features of Admin:

Administrators play a crucial role in maintaining the functionality, security, and efficiency of the ShareTo application. Here are the detailed features available to administrators:

- **User Management:** Provides the ability to view, query, and manage user accounts. Administrators can deactivate inactive accounts, delete unauthorized accounts, or update user permissions to ensure a secure platform.
- **Session Monitoring:** Enables real-time monitoring of active file-sharing sessions. This feature allows administrators to detect unusual activity, ensure compliance with system policies, and provide technical support if required.
- **System Logs:** Records and displays logs of all system activities, including login attempts, file transfers, and errors. Administrators can review these logs to identify security issues, optimize system performance, and maintain an audit trail.

- **Settings Configuration:** Allows administrators to configure system-level settings, such as encryption protocols, bandwidth limits, and default user permissions, ensuring the system aligns with organizational or user requirements.
- **Application Health Monitoring:** Provides tools to track server health, storage capacity, and system uptime. Alerts are generated for potential issues like server overloads or connectivity problems.
- **Access Control Management:** Implements role-based access controls (RBAC), enabling the assignment of specific permissions to different user categories and ensuring sensitive functions are restricted to authorized personnel.
- **Troubleshooting Tools:** Includes diagnostic utilities for identifying and resolving common technical issues within the application, ensuring uninterrupted operation.
- **Data Encryption Management:** Enables oversight of encryption standards and protocols used for file transfers, ensuring compliance with the latest security practices.
- **System Updates and Maintenance:** Administrators can schedule and implement updates to keep the application secure and incorporate new features. Maintenance windows can be configured to minimize downtime.
- **Reporting and Analytics:** Generates detailed reports on system usage, user activity, file transfer statistics, and overall performance. These insights help improve system efficiency and support strategic decisions.
- **Admin Profile Management:** Allows administrators to update their profiles, including login credentials and contact information, ensuring secure and personalized access to admin functions.

These features empower administrators to maintain the ShareTo application as a secure, efficient, and user-friendly platform for peer-to-peer file sharing.

3.5 Features of User:

The user functionality in the ShareTo application is designed to ensure a smooth and secure experience for individuals who initiate and receive file transfers. Here are the key features available to users:

- **User Registration and Login:**
 - **Account Creation:** Users can register and create an account with their unique credentials (username, email, and password) for secure access.
 - **Authentication:** Users log in using JWT-based authentication to ensure secure access to the application and maintain privacy.

- **Session Setup and File Transfer:**
 - **Initiating File Transfers:** Users can initiate peer-to-peer file transfers directly between two devices using WebRTC, ensuring security and privacy.
 - **File Selection:** Users can select files for transfer using an intuitive drag-and-drop interface or system file explorer integration.
 - **Multiple File Support:** Share multiple files simultaneously without delay, making large transfers more efficient.
 - **Real-Time Updates:** Users receive real-time notifications on transfer progress, including file size, time remaining, and status updates.
- **Session Controls:**
 - **Pause, Resume, and Terminate:** Users can pause, resume, or terminate file-sharing sessions as needed, giving them full control over the process.
 - **Offline Mode:** Users can pre-configure transfers, which will initiate once both devices are online, ensuring flexibility in managing transfers.
- **Session Encryption:**
 - **End-to-End Encryption:** All file transfers are encrypted to maintain privacy and prevent unauthorized access to sensitive data.
- **File Transfer History and Logs:**
 - **Access Transfer History:** Users can view the history of previous file transfers, including transfer dates, status, and file details, for reference.
 - **Error Handling:** In case of an issue during the transfer, users receive error reports and suggestions for resolution.
- **Customizable Settings:**
 - **Personalization:** Users can adjust settings such as transfer speed limits, file save locations, and notification preferences to suit their needs.
 - **Bandwidth Optimization:** The application dynamically adjusts transfer speeds based on the network's conditions, ensuring optimal performance.
- **Session Expiry and Resource Management:**
 - **Automatic Session Expiry:** Inactive sessions are automatically detected and terminated to save system resources and ensure security.
- **Cross-Platform Support:**

- **Operating System Compatibility:** The application works seamlessly across multiple operating systems, including Windows, macOS, and Linux, ensuring broad accessibility.
- **Notification System:**
 - **Real-Time Notifications:** Users are alerted when a transfer is completed, paused, or encounters an error, keeping them informed at all stages.
- **File Preview and Metadata Display:**
 - **Preview Files:** Users can preview files and view metadata (e.g., file size, type, and date) before initiating transfers, ensuring accuracy.
- **Auto-Update Mechanism:**
 - **Keep Software Updated:** The application automatically checks for updates and notifies users when new features or security patches are available.
- **Cross-Device Compatibility:**
 - **Device Interoperability:** The system supports file transfers between different devices and operating systems, ensuring a seamless experience for all users.

These features are designed to make ShareTo a comprehensive, secure, and user-friendly file-sharing solution, providing an easy way for users to manage and execute file transfers between their devices.

CHAPTER 4

SYSTEM REQUIREMENTS

System requirements define the functional and non-functional aspects essential for the platform's successful development and deployment. This section provides a detailed explanation of these requirements, focusing on how they support the platform's goals of usability, reliability, and scalability.

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements outline the core functionalities that the system must deliver to meet user needs effectively.

1. **Login/Signup Modules with Secure Authentication:**

- The platform incorporates secure login and signup processes, utilizing **JSON Web Tokens (JWT)** for authentication.
- User credentials are encrypted before being stored, ensuring protection against unauthorized access.
- The signup process allows users to register easily, while login ensures that only verified users access the platform.

2. **File-Sharing Capabilities with History Tracking:**

- Users can upload files securely and share them with others through unique links or direct sharing options.
- A **history tracking module** logs all file-sharing activities, including uploads, downloads, and shared links, enabling users to review past actions.
- File metadata (e.g., size, type, and sharing date) is stored to enhance organization and usability.

3. **API Integration for Real-Time Updates:**

- The system integrates APIs that enable real-time updates using technologies like **Socket.IO**.

- Notifications inform users of file-sharing events, ensuring seamless collaboration.
- Real-time communication supports file-sharing statuses (e.g., “file uploaded” or “download completed”) and user activity updates.

These functional requirements ensure the platform meets essential user needs, offering a secure and efficient environment for file sharing.

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the system's quality attributes, ensuring performance, security, and reliability.

1. Performance:

- The platform is designed to handle multiple concurrent users without delays.
- Load balancing mechanisms are implemented to distribute user requests efficiently across servers, ensuring quick response times even during peak usage.
- The backend is optimized to process requests in milliseconds, enhancing user satisfaction.

2. Security:

- All file transfers are encrypted using **end-to-end encryption protocols (e.g., HTTPS and TLS)**, ensuring data confidentiality.
- User sessions are protected with JWT, preventing unauthorized access to sensitive areas of the platform.
- Regular security audits are conducted to identify and mitigate vulnerabilities.

3. Reliability:

- The system aims for **99.9% uptime**, ensuring it remains operational for users at all times.
- Redundant server architectures prevent downtime due to hardware or network failures.
- Automatic data backups protect against accidental data loss and provide recovery options.

These non-functional requirements ensure the platform is robust and dependable, maintaining trust and reliability for users.

4.3 DESIGN GOALS

Design goals focus on creating an efficient and scalable system that prioritizes user experience and security.

1. Usability:

- The user interface is designed with simplicity in mind, using **Material UI** and **Tailwind CSS** to create clean and responsive layouts.
- Navigation is intuitive, allowing users to quickly access essential features like file uploads and activity history without confusion.
- Accessibility is a priority, ensuring the platform is usable by individuals with diverse needs, including those with disabilities.

2. Scalability:

- The system is designed to handle an increasing number of users and files as demand grows.
- **MongoDB** serves as a scalable database, capable of managing expanding datasets without performance degradation.
- Modular architecture ensures that new features can be added easily without disrupting existing functionalities.

3. Security:

- The platform employs advanced encryption methods for securing both stored data and data in transit.
- Role-based access control ensures that only authorized users can perform specific actions, such as sharing files or accessing administrative tools.
- Regular updates to dependencies and libraries help maintain a secure environment, reducing risks from outdated software.

These design goals align with the project's overall vision of creating a secure, scalable, and user-friendly file-sharing solution that can grow with user needs.

4.4 System Sequence Diagram

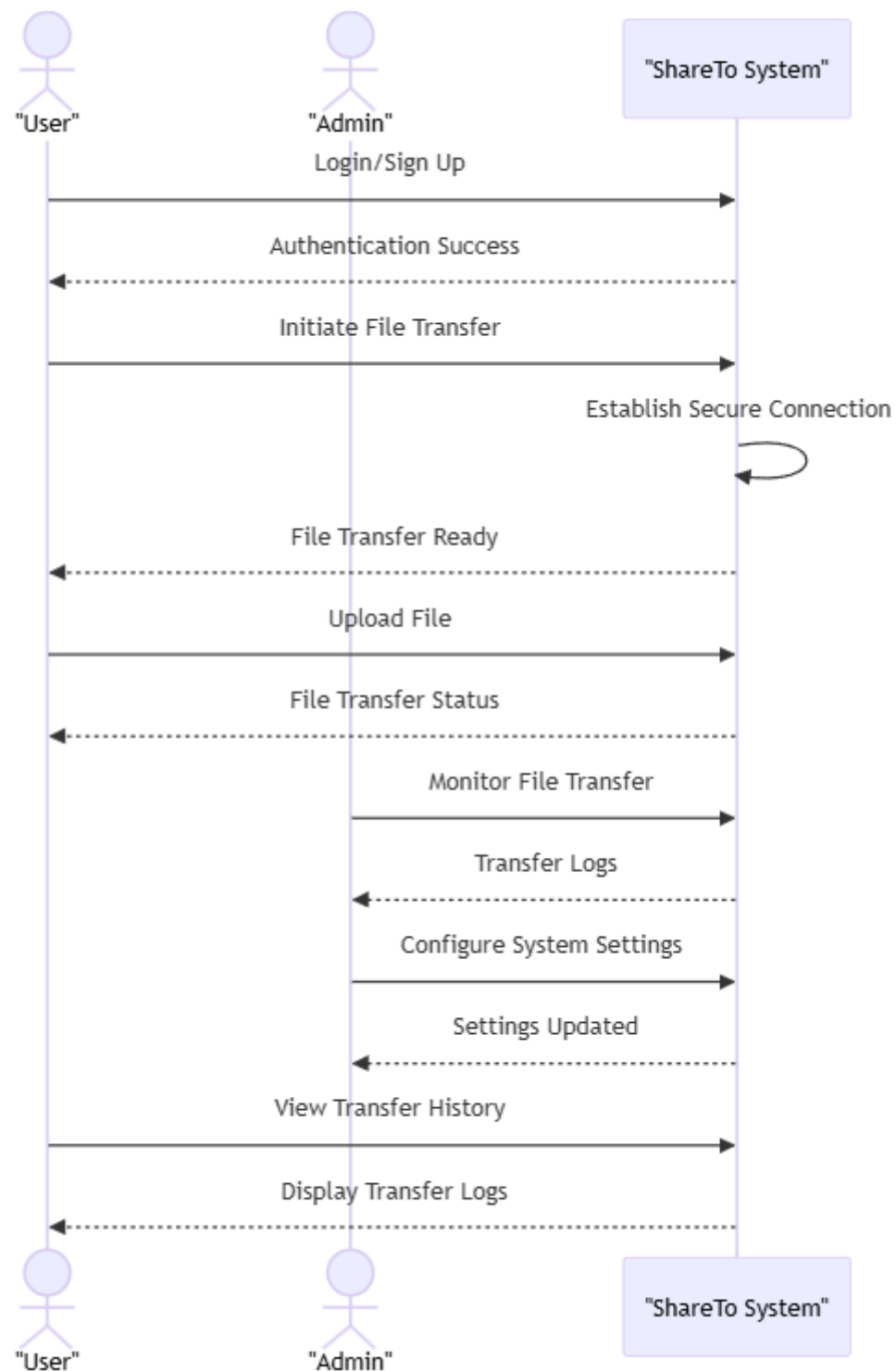


Fig 4.1 System Sequence

Key Interactions:

1. **Login/Sign Up:** The "User" and "Admin" roles initiate the process by logging in or signing up to the "ShareTo System."
2. **Authentication Success:** Upon successful authentication, the "User" and "Admin" are granted access to the system.
3. **Initiate File Transfer:** The "User" initiates a file transfer request to the "ShareTo System."
4. **Establish Secure Connection:** The "ShareTo System" establishes a secure connection to facilitate the file transfer process.
5. **File Transfer Ready:** The "ShareTo System" signals that it is ready to receive the file.
6. **Upload File:** The "User" uploads the file to the "ShareTo System."
7. **File Transfer Status:** The "ShareTo System" provides the "User" with the status of the file transfer process.
8. **Monitor File Transfer:** The "Admin" can monitor the file transfer process and view transfer logs.
9. **Configure System Settings:** The "Admin" has the ability to configure various system settings.
10. **Settings Updated:** The "ShareTo System" updates its settings based on the configurations made by the "Admin."
11. **View Transfer History:** The "Admin" can view the history of previous file transfers.
12. **Display Transfer Logs:** The "Admin" can display detailed transfer logs for specific files.

4.5 ER Diagram for a File Transfer System

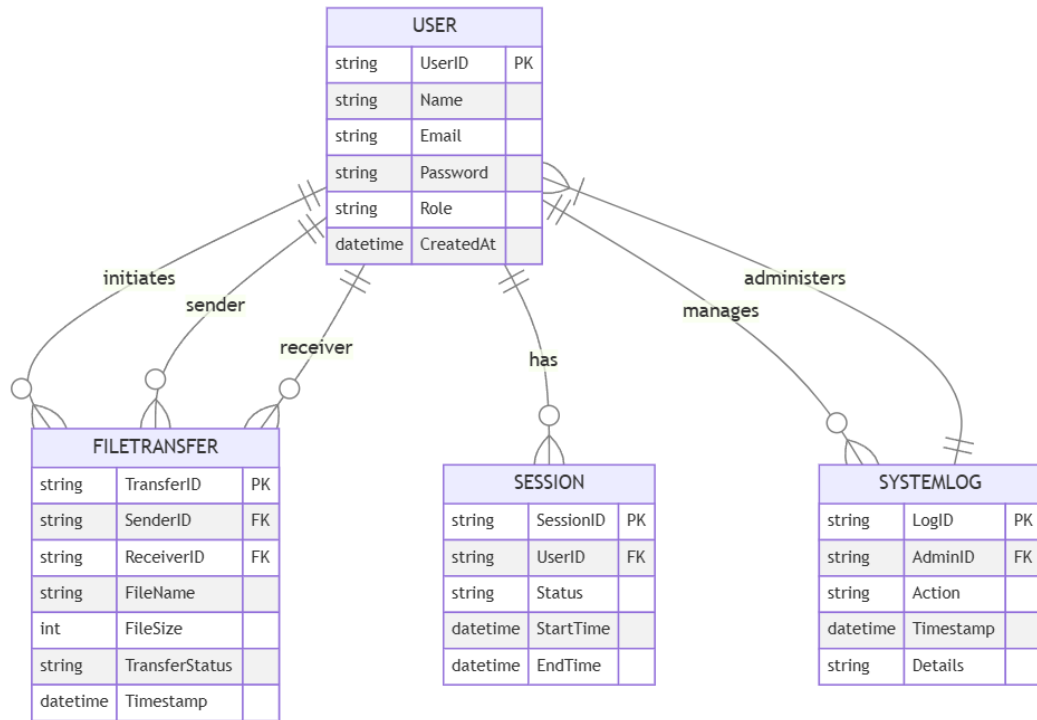


Fig 4.2 E-R Diagram

This ER diagram illustrates the entities and relationships involved in a file transfer system.

Entities:

- **USER:** Represents a user of the system.
 - **Attributes:** UserID (PK), Name, Email, Password, Role, CreatedAt
- **FILETRANSFER:** Represents a file transfer record.
 - **Attributes:** TransferID (PK), SenderID (FK), ReceiverID (FK), FileName, FileSize, TransferStatus, Timestamp
- **SESSION:** Represents a user session.
 - **Attributes:** SessionID (PK), UserID (FK), Status, StartTime, EndTime
- **SYSTEMLOG:** Represents a system log entry.

- **Attributes:** LogID (PK), AdminID (FK), Action, Timestamp, Details

Relationships:

- **User - FileTransfer:**
 - initiates: A user initiates a file transfer.
 - sender: A user is the sender of a file transfer.
 - receiver: A user is the receiver of a file transfer.
- **User - Session:**
 - has: A user has one or more sessions.
- **User - SystemLog:**
 - administers: An admin user manages the system and generates system logs.
- **FileTransfer - Session:**
 - is part of: A file transfer is part of a session.

CONCLUSION

The system requirements provide a comprehensive framework for the platform's functionality and performance. By focusing on secure authentication, efficient file-sharing, and real-time updates, the functional requirements ensure the system meets user needs. Non-functional requirements emphasize reliability, performance, and security, while the design goals aim to create an accessible, scalable, and secure solution. Together, these requirements form the foundation for a robust and future-proof file-sharing platform.

CHAPTER 5

SYSTEM DESIGN

System design focuses on creating the structural framework for the platform, ensuring it is efficient, scalable, and user-friendly. This section elaborates on the primary and secondary design phases and the user interface, detailing how they work together to meet project goals.

5.1 PRIMARY DESIGN PHASE

The primary design phase establishes the foundational structure of the system. It involves creating visual and functional representations of core modules to guide development.

1. Wireframes for Login, Dashboards, and File-Sharing Modules:

- **Login Module:** Wireframes outline the design of secure login and signup screens, featuring input fields for user credentials, error handling for invalid inputs, and options like "Forgot Password" for recovery.
- **Dashboard:** The dashboard wireframe displays an overview of recent activities, quick links to core functionalities, and a summary of file-sharing events.
- **File-Sharing Module:** Designs focus on intuitive interfaces for uploading, downloading, and sharing files. Wireframes include clear action buttons, progress indicators, and options for selecting sharing recipients.

2. Data Flow Diagrams (DFDs):

- **User Interactions:** DFDs visualize how users interact with the system, from login authentication to file-sharing workflows.
- **Backend Communication:** Diagrams map the flow of data between the frontend, backend, and database during operations such as login, file uploads, and history retrieval.
- **Real-Time Updates:** Illustrations show how technologies like Socket.IO manage notifications and updates, ensuring timely communication between users and the system.

5.2 SECONDARY DESIGN PHASE

The secondary design phase involves refining the system based on feedback and optimizing its performance for real-world use.

1. Improved User Interface Layouts Using Material UI:

- **Consistency and Aesthetics:** Feedback from initial designs informs UI refinements, resulting in clean, visually consistent layouts. Material UI components like buttons, sliders, and forms are used to enhance the overall look and feel.
- **Accessibility Enhancements:** Refinements include larger fonts, color contrast adjustments, and keyboard navigation support to make the system accessible to a broader audience.
- **Responsive Design:** Layouts are optimized for devices of all sizes, ensuring usability on desktops, tablets, and smartphones.

2. Backend Optimizations for Faster Response Times:

- **Database Queries:** Optimized MongoDB queries reduce latency during data retrieval, improving performance for file-sharing history and large datasets.
- **API Performance:** The backend code is streamlined to handle requests efficiently, minimizing response times even under heavy user loads.
- **Caching Mechanisms:** Temporary storage of frequently accessed data improves performance by reducing repetitive database calls.

The secondary design phase transforms initial designs into a polished, high-performing system ready for deployment.

5.3 USER INTERFACE

The user interface (UI) is the most visible aspect of the platform, designed to provide a seamless and intuitive experience for all users.

1. Responsive UI:

- The system adapts to various screen sizes and resolutions, ensuring consistency and usability across devices.
- Features like collapsible sidebars and touch-friendly buttons enhance navigation on smaller devices.

2. Key Components:

- **Dashboard:** A centralized hub for user activities, displaying file-sharing events, recent uploads, and notifications. The dashboard also includes shortcuts for accessing essential features like help/support.
- **File Management Modules:** Includes easy-to-use interfaces for uploading, downloading, and organizing files. Features like drag-and-drop functionality simplify file uploads, while filters and search bars enable efficient file retrieval.
- **History Tracking:** A dedicated module displays logs of file-sharing activities, helping users track their interactions and manage their files effectively.

3. Visual Appeal:

- The use of Tailwind CSS and Material UI ensures that the interface is both functional and aesthetically pleasing.
- Intuitive icons, color-coded elements, and tooltips improve navigation and understanding of the platform's features.

By focusing on responsiveness, key components, and visual appeal, the user interface ensures a positive experience that caters to diverse user needs.

Conclusion

The system design process ensures the platform is built on a strong foundation, with clearly defined wireframes, data flows, and a polished interface. The primary design phase establishes the basic structure, while the secondary phase refines it for performance and usability. A responsive, visually appealing user interface completes the design, ensuring that the system is easy to use and accessible to all.

CHAPTER 6

ARCHITECTURE

The architecture of the platform outlines the structural design and technologies used to ensure scalability, performance, and security. It is built on a layered model with real-time capabilities and robust security mechanisms to meet the project's functional and non-functional requirements.

6.1 LAYERED ARCHITECTURE

The system's architecture is divided into three key layers: frontend, backend, and database, each serving specific purposes.

1. Frontend Layer:

- **Technologies Used:** Built using **React JS** for its flexibility and efficiency in creating dynamic, component-based user interfaces.
- **Tailwind CSS:** Enhances the visual appeal and responsiveness of the UI, ensuring the platform is accessible on various devices like desktops, tablets, and smartphones.
- **Key Functions:**
 - Manages user interactions such as file uploads, downloads, and navigation through dashboards.
 - Communicates seamlessly with the backend to fetch or send data via REST APIs or WebSocket connections.
- **User Experience:** Provides a fast, responsive, and interactive environment, improving engagement and usability.

2. Backend Layer:

- **Technologies Used:** Developed using **Node JS** and **Express** to handle server-side operations effectively.
- **Key Responsibilities:**
 - Manages application logic, such as authentication, file-sharing workflows, and activity tracking.
 - Processes incoming requests from the frontend and communicates with the database for data retrieval or storage.

- Provides API endpoints to integrate real-time updates using technologies like Socket.IO.
- **Scalability:** Optimized to handle multiple concurrent users efficiently, ensuring smooth performance even during peak loads.

2. Database Layer:

- **Database Used: MongoDB,** a NoSQL database, supports unstructured and semi-structured data formats, making it ideal for storing file metadata and user activity logs.
- **Key Functions:**
 - Stores user credentials, file-sharing history, and metadata (e.g., file names, sizes, and timestamps).
 - Ensures fast data retrieval through efficient indexing and query mechanisms.
- **Advantages:** Its scalability and flexibility allow the system to handle growing data volumes as the user base expands.

The layered architecture ensures modularity, making the system easier to maintain and scale in the future.

6.2 REAL-TIME UPDATES

Real-time capabilities are a crucial feature of the platform, enhancing user interaction and collaboration.

1. Socket.IO:

- Provides real-time notifications for activities such as file uploads, downloads, and sharing.
- Manages bi-directional communication between the server and clients, ensuring instant updates without requiring manual refreshes.
- Examples of Use:
 - Alerts users when a file is uploaded or shared with them.
 - Displays active users or session updates on the dashboard.

2. WebRTC:

- Enables quick peer-to-peer file transfers, bypassing server storage for temporary files, thus reducing server load.
- Ensures faster file-sharing speeds, especially for large files, by creating direct connections between users.

- Enhances the platform's usability for teams requiring real-time collaboration.

By integrating these technologies, the system ensures that users experience seamless interactions and immediate updates, fostering a highly interactive environment.

6.3 SECURITY ARCHITECTURE

Security is a cornerstone of the platform, protecting user data from unauthorized access and ensuring safe transactions.

1. Token-Based Authentication (JWT):

- **How It Works:**
 - Upon successful login, users receive a **JSON Web Token (JWT)**, which serves as proof of authentication.
 - Tokens are included in requests to secure API endpoints, verifying user identity and access rights.
- **Benefits:**
 - Prevents unauthorized access by ensuring only authenticated users can interact with the system.
 - Eliminates the need for storing session data on the server, reducing overhead.

2. Data Encryption:

- **At Rest:** Sensitive information like user credentials and file-sharing logs are encrypted before being stored in the database.
- **In Transit:** All communications between the client and server use **HTTPS**, ensuring data is encrypted and protected from interception.
- **Key Advantages:**
 - Safeguards against data breaches and cyber-attacks.
 - Builds user trust by maintaining high-security standards.

3. Regular Security Audits:

- Periodic reviews of system components, including API endpoints and database configurations, help identify vulnerabilities.
- Ensures the platform remains compliant with security best practices, such as OWASP guidelines.

These security measures provide a robust foundation for safeguarding user data and ensuring secure communication across the platform.

CONCLUSION

The architecture integrates a layered design, real-time capabilities, and advanced security measures to create a powerful, efficient, and user-friendly platform. The frontend ensures seamless interactions, the backend processes logic effectively, and the database manages data reliably. Real-time technologies like Socket.IO and WebRTC enhance collaboration, while JWT and encryption safeguard user data. Together, these components make the architecture scalable, secure, and ready for future enhancements.

CHAPTER 7

PROJECT SCREENSHOT



Fig 7.1 Dashboard Page

This image is the homepage of the ShareTo project. It features a split design with options for users to act as either a sender or receiver, emphasizing effortless sharing and instant connections.

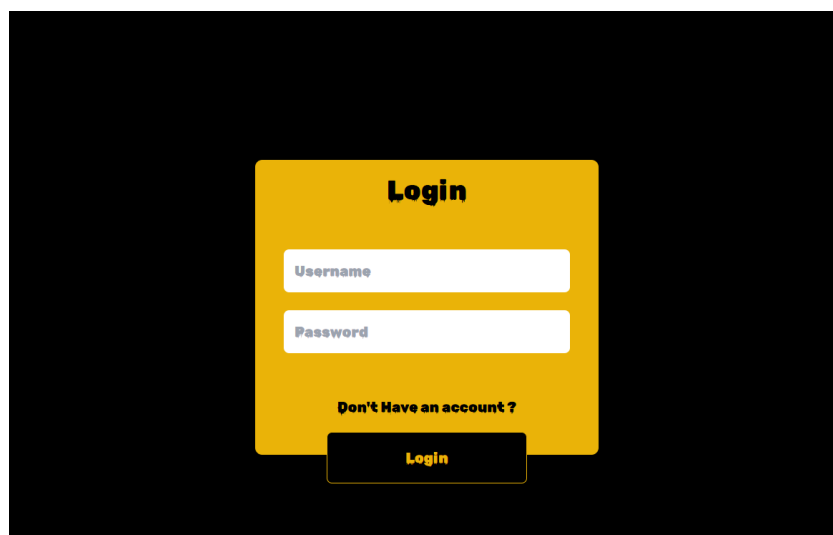
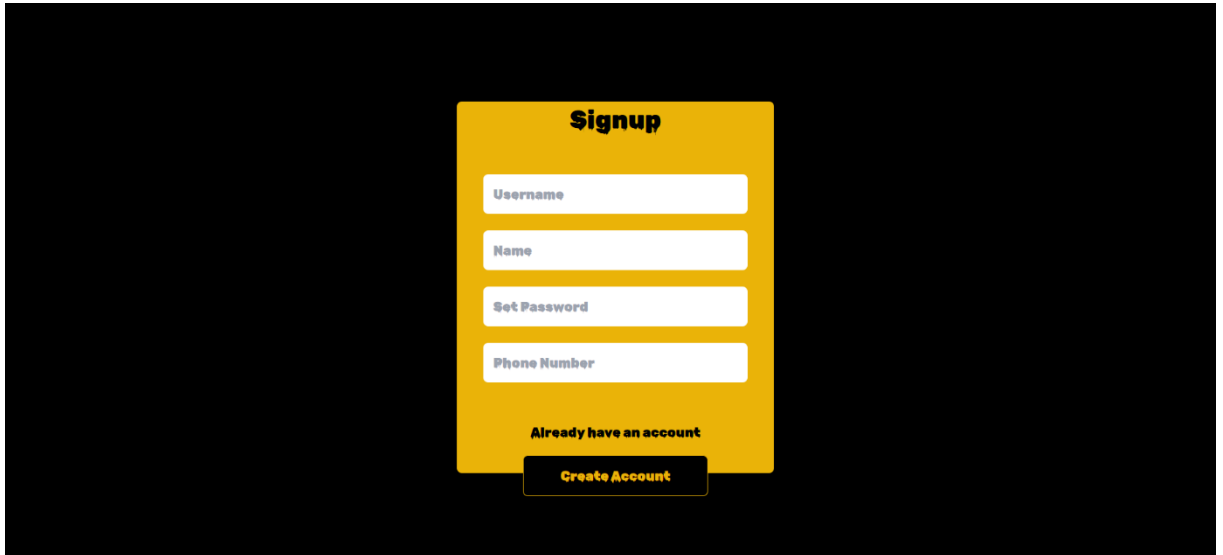


Fig 7.2 Login Page

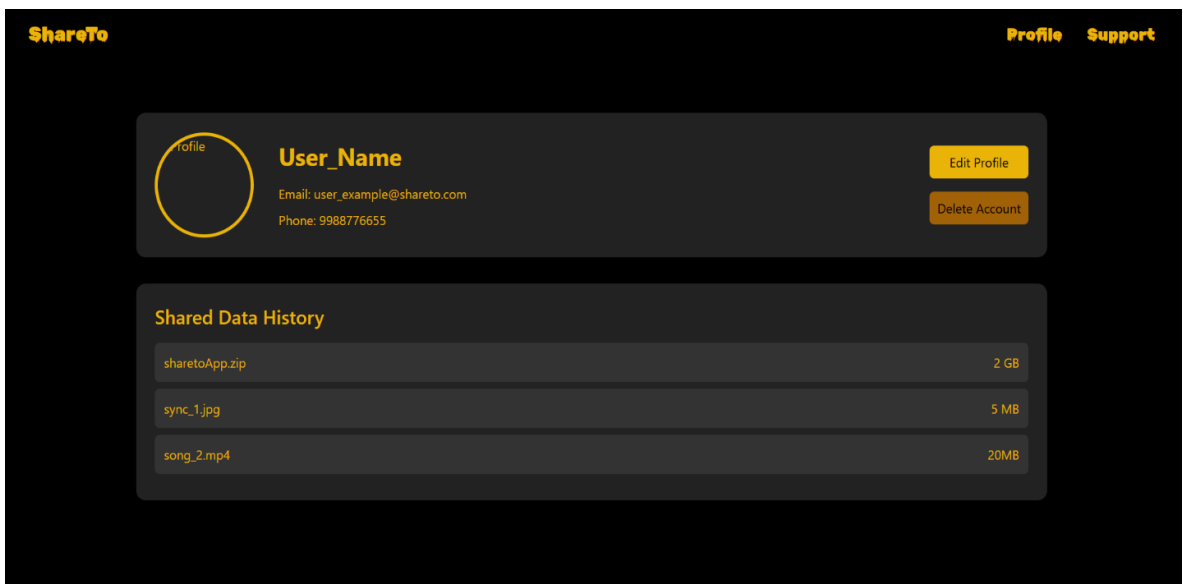
This image represents the login page of the ShareTo project. It features a minimalist yellow and black design with input fields for a username and password, along with a prompt for new users to sign up.



The image shows a 'Signup' form centered on a black background. The form is contained within a yellow rectangular box. At the top of the box, the word 'Signup' is written in bold black text. Below this, there are four white input fields stacked vertically, each with a label: 'Username', 'Name', 'Set Password', and 'Phone Number'. At the bottom of the yellow box, the text 'Already have an account' is displayed in a smaller font. Below the yellow box, there is a black button with the text 'Create Account' in yellow.

Fig 7.3 Signup Page

This is a basic signup form with fields for username, name, password, and phone number. The form is set against a black background with a yellow box containing the form elements. The button to create an account is at the bottom.



The image shows a user profile page with a dark theme. At the top left, the 'ShareTo' logo is visible. At the top right, there are links for 'Profile' and 'Support'. The main content area is divided into two sections. The first section is the user profile, which includes a circular profile picture placeholder labeled 'Profile', the user's name 'User_Name', their email 'Email: user_example@shareto.com', and their phone number 'Phone: 9988776655'. To the right of the profile information are two buttons: 'Edit Profile' and 'Delete Account'. The second section is titled 'Shared Data History' and contains a table of shared files.

File Name	Size
sharetoApp.zip	2 GB
sync_1.jpg	5 MB
song_2.mp4	20MB

Fig 7.4 User Page

This is a user profile page with a black background and yellow accents. It displays the user's name, email, and phone number. The page also shows a list of shared data history with file sizes and options to edit profile and delete account.

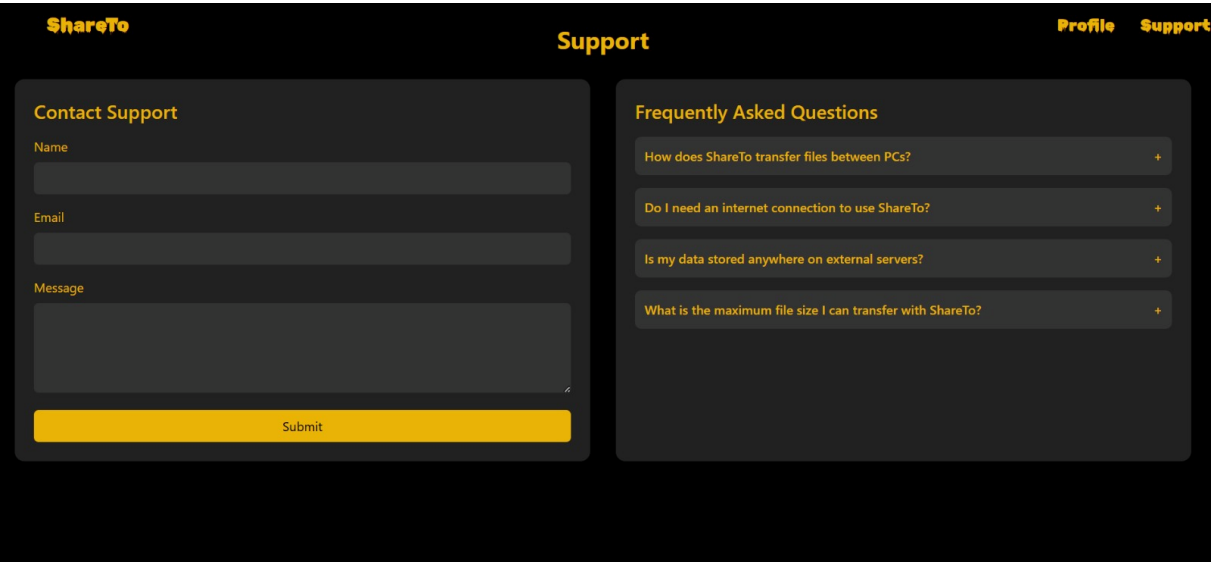


Fig 7.5 Help and Support

The Share-To Help and Support page provides a convenient way for users to get answers to their questions and resolve any issues they may have with the file transfer service. The page features a contact form for direct support requests and a list of frequently asked questions with detailed answers.

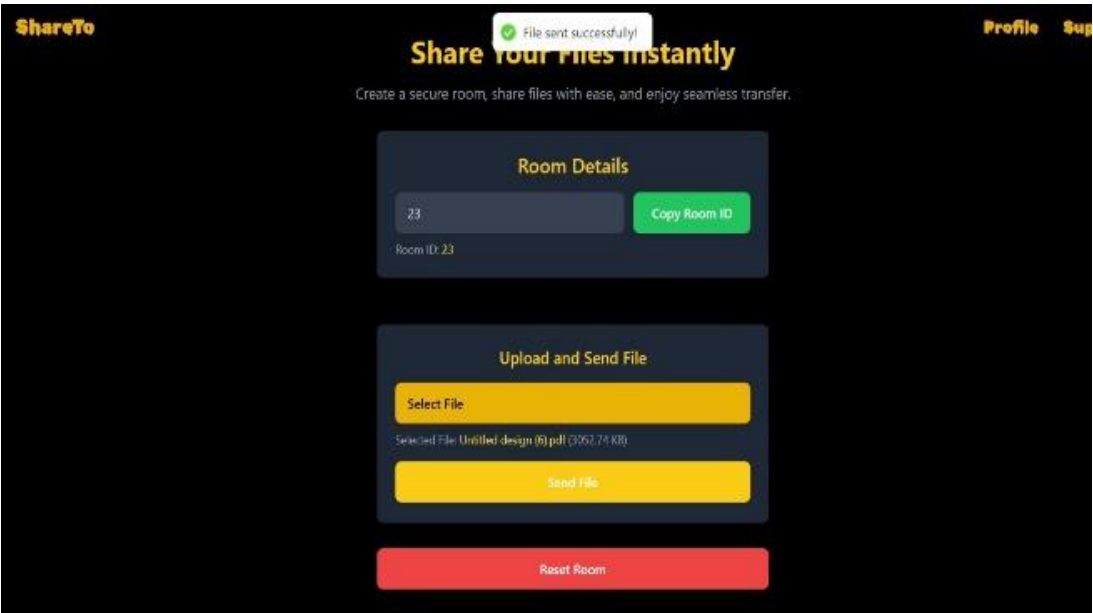


Fig 7.6 Sender File Page

ShareTo is a file-sharing platform that allows users to create secure rooms to share files easily. Users can upload files, share room IDs, and recipients can download files from the room.

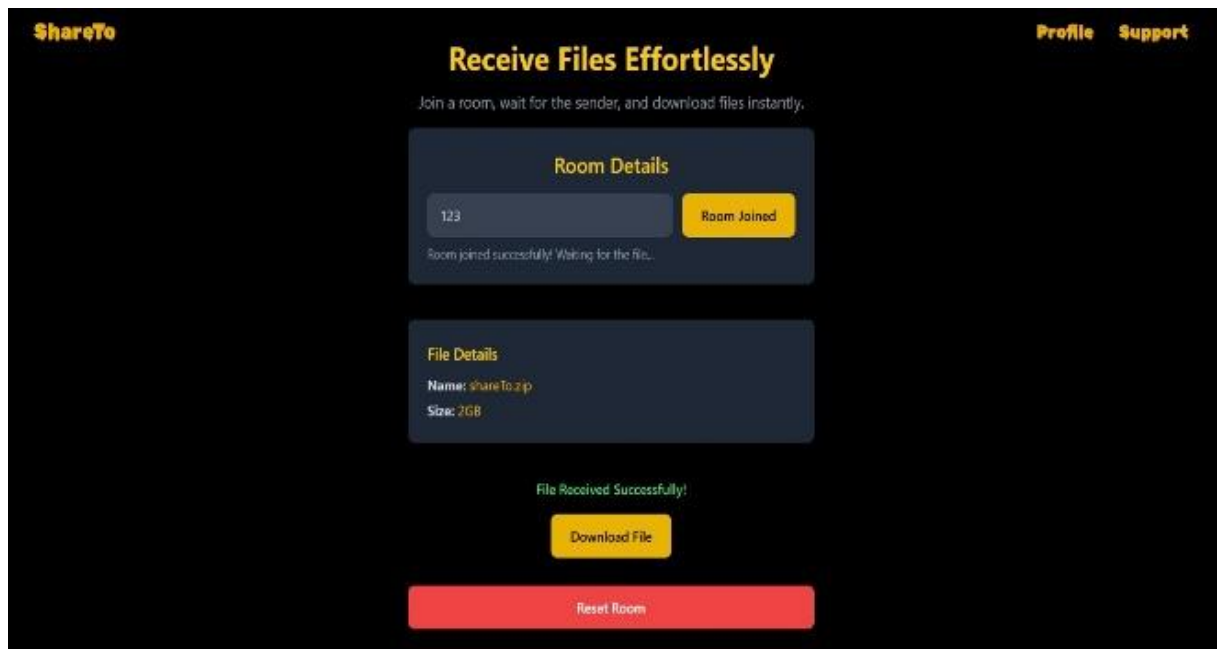


Fig 7.7 Receiver File Page

ShareTo lets you easily receive files. Join a room using its ID, wait for the sender to upload files, and then download them directly from the platform.

CHAPTER 8

CODE SCREENSHOTS

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  username : {
    type : String,
    required: true,
    unique: true
  },
  name : {
    type: String,
    required: true
  },
  password : {
    type: String,
    required: true
  },
  phoneNumber : {
    type: Number,
    required: true,
  }
});

module.exports = mongoose.model('user_model' , UserSchema);
```

Fig 8.1 User Schema

This code snippet defines a Mongoose schema for a user model. It includes fields for username, name, password, and phoneNumber, all marked as required. The username field is also set as unique.


```

const router = require("express").Router();
const {login , signup} = require('../controller/auth');
const {Getuser} = require("../controller/Usercontroller");

router.post('/login' , login);

router.post('/signup' , signup);

router.get('/GetUser' ,Getuser);

module.exports = router;

```

Fig 8.2 Router

This code defines Express routes for authentication and user retrieval. It maps HTTP POST requests to '/login' and '/signup' endpoints, handling them with functions from the 'auth' controller. It also maps an HTTP GET request to '/GetUser' endpoint, handled by the 'GetUser' function from the 'usercontroller'.

```

import { Navigate } from "react-router-dom";
Pieces: Comment | Pieces: Explain
const ProtectedAuth = ({children}) => {
  const token = localStorage.getItem('token');
  if(token){
    return children;
  }else{
    return <Navigate to='/Login'/>
  }
}

export default ProtectedAuth;

```

Fig 8.3 Protected Auth

The ProtectedAuth component in React uses a token-based authentication system. If a token is found in local storage, it allows access to protected routes. Otherwise, it redirects the user to the login page.

```
const user_model = require("../Models/UserSchema");
```

Pieces: Comment | Pieces: Explain

```
const Getuser = async(req,res) => {
  try{
    const id = req.query.id.toString();
    console.log("Id is : " , id);
    const user = await user_model.findById(id);
    console.log("user is : ", user)
    if(user){
      return res.status(200).json({
        status: 1,
        message: "User Found",
        User: user,
      })
    }else{
      console.log("User not is database");
      return res.status(400).json({
        status: 0,
        message: "User not found",
      })
    }
  }catch(e){
    console.log("Error occured in fetching user " , e);
    return res.status(500).json({
      status: 0,
      message: "User not found inside catch",
      error: e.message
    });
  }
}

module.exports = {Getuser};
```

Fig 8.4 User Controller

The code fetches a user from a database using the provided ID. It returns the user data if found, otherwise returns error messages with appropriate HTTP status codes.

CHAPTER 9

CONCLUSION

The "Share-To" project aims to revolutionize the way files are shared by providing a secure, efficient, and user-friendly platform. It overcomes traditional file-sharing challenges by leveraging modern technologies and ensuring a seamless user experience.

1.Modern Solution: "Share-To" offers an innovative peer-to-peer file-sharing application that caters to contemporary data-sharing needs.

2.Advanced Technology Stack: Built using React.js, Node.js, MongoDB, and WebRTC, it ensures reliability and high performance.

3.Real-Time Communication: Technologies like WebRTC and Socket.IO enable instantaneous file transfers.

4.Robust Security: Features end-to-end encryption and JWT-based authentication to safeguard user data and privacy.

5.Cross-Platform Compatibility: The application functions seamlessly across Windows, macOS, and Linux.

6.User-Friendly Design: With a drag-and-drop interface and intuitive navigation, the platform is accessible to users of all technical levels.

7.Scalability: Its architecture supports an expanding user base and growing data needs.

8.High Efficiency: Optimized bandwidth usage, session controls, and real-time updates ensure a smooth user experience.

9.Eliminates Dependencies: Removes the need for external storage or cloud-based platforms, redefining file-sharing practices.

10.Versatile Use Cases: Suitable for personal, academic, and professional scenarios, promoting better collaboration and productivity.

This project stands as a testament to innovative solutions in the digital era, providing a reliable and scalable platform for secure file-sharing.