File Sharing Application Using P2P Protocol

CS492 Project

MDL18CS046 22 Emmanuel Antony
MDL18CS069 32 Malavika S Menon
MDL18CS098 48 Rose Joseph
MDL18CS123 61 Varun Krishna S

B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

June 2022

Model Engineering College Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***File Sharing Application Using P2P Protocol*** is a bonafide record of the work done by

MDL18CS046 22 Emmanuel Antony
MDL18CS069 32 Malavika S Menon
MDL18CS098 48 Rose Joseph
MDL18CS123 61 Varun Krishna S

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS492 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **APJ Abdul Kalam Technological University**.

Guide

Lekshmi Subha M S
Assistant Professor
Computer Engineering

Coordinator

Dr. Preetha Theresa Joy
Professor
Computer Engineering

Head of the Department

Dr. Preetha Theresa Joy
Professor
Computer Engineering

June 27, 2022

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere gratitude to all of them.

First and foremost, we would like to thank our esteemed Principal, Prof. (Dr.) Jacob Thomas V, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We are highly indebted to our Project Coordinator and Head of the Department, Prof. (Dr.) Preetha Theresa Joy for the guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

We would like to thank our Project Guide, Mrs. Lekshmi Subha M S for her support and valuable insights and also for helping us out in correcting any mistakes that were made during the course of the work.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped us get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding us through and enabling us to complete the work within the specified time.

Emmanuel Antony
Malavika S Menon
Rose Joseph
Varun Krishna S

Abstract

Majority of applications that implement file sharing, make use of the client-server model, whose efficiency decreases as number of users increase. However, Peer-To-Peer(P2P) based file sharing applications enable faster transfer of files simultaneously to multiple users. In a P2P network, instead of taking the whole file from a single node, the nodes take smaller pieces from hundreds of others, resulting in better utilisation of bandwidth. At present, no major application provides a smooth user experience for transferring files using peer-to-peer protocols. In this paper, we intend to develop a web application, with enhanced user experience for easier transfer of files based on P2P protocol, which would help enhance speed and divide the payload borne by the sender for sharing the file to a large number of users

Contents

List of Figures

List of Tables

Chapter 1

Introduction

File sharing is a key feature of several popular applications today, including messaging & social media platforms. In fact, there are several applications that focus solely on file sharing. With the development of storage facilities in our mobile devices, the scale and scope of file sharing has also increased considerably. Most applications available presently, depend on the traditional client-server model to transfer files. While this works efficiently in small groups with one or two users, the efficiency decreases as more and more users are involved. With the entire payload being borne by the sender, the bandwidth utilisation becomes poorer as the number of receivers increase. Hence, peer-to-peer networks can be considered as an alternative to the client-server model, using which a file-sharing application can be implemented. This application would allow for a smoother, faster and more efficient file transfer mechanism among a large number of users.

1.1 Proposed Project

1.1.1 Problem Statement

At present, no major application provides a smooth user experience or a straightforward method for a layman to transfer files using peer-to-peer protocol.

1.1.2 Proposed Solution

The proposed solution aims to build a web application which enables an easier alternative to transfer files to large number of users with enhanced speed and user experience using P2P protocol.

1.1.3 Objectives

- To develop a torrent-based file sharing application for faster transfer of files simultaneously to multiple users.
- To develop a Web application, with enhanced user experience for easier transfer of files via the P2P protocol.
- To facilitate users in sending text messages and creating group chats, to enhance the user experience of the application by providing an interface framework they are already familiar with.

- To help enhance the speed and divide the payload borne by the sender for sharing the file to a large number of users.
- To help optimise the bandwidth usage across users as the number of users increase.
- To build an application that is platform agnostic.

Chapter 2

System Study Report

2.1 Literature Survey

In this literature survey, we seek to understand P2P networks in depth, starting from the analysis of a generalised P2P network & then proceed to explore different P2P protocols including Bit-Torrent, Direct Connect(DC) and Multi-Torrent. The study also looks into the various algorithms used to address the critical lookup problem & considers a distributed data sharing system like PeerDB along with the advantages and disadvantages it offers. This literature survey presents:

1. M. Parameswaran, A. Susarla and A. B. Whinston [1], present P2P networking as an alternative for information sharing. This paper treats P2P networking as an alternative to the traditional client-server models, and investigates the various advantages and disadvantages a P2P system offers. In a P2P network, all nodes are of the same level of importance and each node can distribute content to all other nodes in the network. Owing to the various techniques available for monitoring traffic and redistributing the content to ease the load on individual nodes, P2P networks have enhanced load balancing capabilities. Also, since each individual node can distribute content, this helps in easier replication of the content, thereby leading to better redundancy and fault tolerance. In P2P, the information available at any user node is indexed. Having a central server with the full index may lead to the failure of the entire network in case of a server downtime. To tackle this, a few nodes may maintain a partial index. The decentralisation leads to increased fault tolerance. However, if the entire network isn't efficiently organised, this could result in poorer search retrieval time. Another major disadvantage of P2P networks, earning it the name 'The Wild Web', is that since each user can distribute content freely, this may lead to copyright infringement as well as the circulation of undesirable content.

2. Ng, W.S. and Ooi, B.C. and Tan, K.-L. and Aoying Zhou [2], discuss the limitations of various P2P systems and how a distributed data sharing system such as PeerDB (the prototype developed), will be useful. The paper presents three major limitations of common P2P systems. Firstly, the systems offer only file-level sharing and lack object/data management capabilities. Next, they are limited in their flexibility and extensibility. Finally, the node's peers are typically statically defined. In contrast, a system like PeerDB offers the following features. Here,

each node is a full-fledged object management system that supports content based search. The users may share data without using a global schema. The system makes use of a mobile agent, which helps in query processing and leads to better network utilisation. The paper also presents the basic architecture of the PeerDB system. The PeerDB application is built on top of BestPeer, which is a generic P2P system. It consists of two entities - large number of computers and relatively fewer number of location independent global name lookup (LIGLO) servers. BestPeer integrates two powerful technologies - P2P and mobile agent. It shares not only data, but also computational power. The architecture of a PeerDB node involves - Data Management System like MySQL, Database Agent system called DBAgent, cache manager and user interface.

3. J.A. Pouwelse, P. Garbacki, D.H.J. Epema and H.J. Sips [3] discuss the BitTorrent P2P File Sharing System and provide the measurement of the system based on four parameters - availability, integrity, flashcrowd handling, and download performance. BitTorrent is a popular file-sharing protocol, which has attracted millions of users. It is estimated to make up around 53% of all P2P traffic. It is only a file-downloading protocol and relies on other global components such as suprnova.org. The paper discusses the experiment performed to monitor various parameters by monitoring the global BitTorrent/Supernova components, using a Mirror script which measures the availability and response time of suprnova mirrors. The actual peers are monitored, using three more scripts, the Hunt script to follow and initiate a measurement of all the peers downloading a particular file, the Getpeer script to get the IP addresses of all the peers and the Peerping script to contact numerous peers in parallel and measure their download progress and uptime.

4. Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris and Ion Stoica disuss a critical common problem in P2P systems, lookup problem, and present a simple and general interface, a distributed hash table (DHT), for solving it. The lookup problem is stated as: Given a data item X stored at some dynamic set of nodes in the system, find it. The traditional approach to achieving scalability is to use hierarchy eg: DNS. A hash-table interface is an attractive foundation for a distributed lookup algorithm because it places few constraints on the structure of keys or the values they name.The main requirements are that data be identified using unique numeric keys, and that nodes be willing to store keys for each other. The values could be actual data items (file blocks), or could be pointers to where the data items are currently stored. To publish a file under a particular unique name, the publisher would convert the name to a numeric key using an ordinary hash function such as SHA-1, then call lookup(key). The publisher would then send the file to be stored at the node(s) responsible for the key. A consumer wishing to read that file would later obtain its name, convert it to a key, call lookup(key), and ask the resulting node for a copy of the file.

5. Y. Yang, A. L. H. Chow and L. Golubchik present a study on multi-torrent. It focuses on how incentivizing users to stay in a multi-torrent environment increases the performance, what incentives could be provided for nodes to contribute resources as seeds in a multi-torrent environment and what are the resulting performance consequences of such behavior, both on the nodes which are willing to be seeds and on the overall system. In order to emphasize

the importance of incentivizing users, a simulation based study is performed. Depending on the experiment, the nodes does not stay around, forced to stay around or given incentives to stay around. Results show that performance increases when incentives are given for staying around. They adopted a scheme known as Cross Tit-for-Tat(CTFT). Instead of choosing peers with the fastest downloading rates in a particular torrent, it looks at the peers' aggregate downloading rate in all torrents in which they participate. To illustrate the performance consequences of the approach, the effects of different parameters using simple scenarios are explored. The experiment is performed for different number of torrents, different node arrival rates, different fraction of nodes staying around, different weight of CTFT.

6. P. Gurvich, N. Koenigstein and Y. Shavitt investigate the DC file sharing network. Direct Connect (DC) is a peer-to-peer file sharing protocol. Direct Connect clients connect to a central hub and can download files directly from one another. The paper makes two main contributions: it is the first measurement study of the DC network; query duplication problem that stems the protocol's scalability potential. Once a day, the agent downloads a DC hub list from a public hub-list server. It then initiate connection to the top 150 largest hubs. Upon connection the agent records the hub's user count and the total amount of shared content, track the number of connected users in a hub and monitors and records search queries received from the clients. The number of users in a hub and its query frequency are highly correlated and reflect its activity. Recently, the content of P2P queries had been the focus of several studies in the field of data-mining and marketing. The average shared folder size was 52.7 GB, which is considerably higher than users on other file sharing networks. Inevitably users receive duplicated queries belonging to users with whom they share more than one hub. Identified that about 40% of the intercepted text queries were duplicated.

2.2 Proposed System

The proposed system involves the creation of a P2P based file sharing application using web technologies to facilitate the transfer of files across multiple participants while providing a smooth user experience. The application is intended to be faster than traditional client-server models and aims to optimise the bandwidth usage as the number of users increase. The application, built on the web, is platform agnostic and doesn't lock into any ecosystem. Further, it allows any device to transfer files seamlessly through the web browser. In addition, the application allows users to send text messages and create group chats. These features generally observed in a chat application, adds on to the user experience by providing the users with a familiarised pattern.

Chapter 3

Software Requirement Specification

3.1 Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, abbreviations, references and a brief outline of the SRS. The aim of this document is to gather, analyze and give an in-depth insight of the complete software made, also defining the problem statement in detail. The detailed requirements of the software and hardware needed for the project are provided in this document.

3.1.1 Purpose

The purpose of the Software Requirements Specification document is to maintain all the functions and the specifications of File Sharing Application using P2P Protocol. Besides, it contains detailed descriptions of all the software, functional and non-functional requirements.

3.1.2 Document Conventions

This document follows MLA (Modern Language Association) Format. Bold-faced text has been used to emphasize section and subsection headings. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams.

3.1.3 Intended Audience and Reading Suggestions

This document contains software functionality, software requirements and user documentation.

- Developer: If a developer wishes to read, alter, amend, or add new requirements to an existing program, they should examine this document first and update the requirements appropriately so as not to affect the system's true purpose or render it insignificant.
- User: The user of this application examines the diagram and specification in the document to see if the software meets all of the appropriate criteria and if the software developer has implemented them all. In the case of any query, he or she may check the user handbook for clarification.
- Tester: This document is required for the tester to prepare his/her test cases in order to ensure that the project's initial requirements are implemented in the deliverable.

- Documentation Writer: If a writer wishes to alter the contents of the handbook, they must ensure that they have an overall understanding of the SRS and other documents pertaining to the project, and also have a hands on experience on using the system. This document need not be read sequentially; users are encouraged to jump to any section they find relevant.

3.1.4 Project Scope

To create an application with an enhanced and robust user experience in comparison to existing file sharing applications, by creating a web based application. This layer would also implement minimal logic to maintain groups and chats in order to make the file sharing experience seamless and more user-friendly. Users would be able to create groups, send text messages and share files amongst the members of the group. The file sharing is done via BitTorrent, a commonly used protocol for P2P file sharing.

3.1.5 Overview of Developer's Responsibilities

Responsibilities of the developers are:

1. Maintain appropriate coding standards and design.
2. Contribute to technical design documentation.
3. Contribution in accordance with the software development life cycle.

Features proposed to be implemented:

1. To create a system that facilitates file sharing across groups.
2. To create an interface that enables the bridging of the existing Web torrent protocol which in turn utilizes the BitTorrent protocol to make file sharing easier, without being concerned by the protocol implementation details.
3. To create an enhanced user experience, while sharing files across groups in a bandwidth efficient, intuitive manner.

3.2 Overall Description

This section includes product perspective, product functions, user classes and characteristics, operating environment, design and implementation constraints, user documentation, general constraints and assumptions and dependencies of the project.

3.2.1 Product Perspective

This application would serve as a replacement to existing file sharing applications based on various sharing techniques such as WiFi-Direct, Direct Connect, Cloud, Bluetooth, P2P. Its advantages over the existing systems are:

- Faster speeds than Bluetooth based approach.
- Smoother user experience in comparison to Direct Connect/P2P approaches.
- Better bandwidth utilisation than Cloud/Direct Connect based approaches.
- Not limited by geographic constraints unlike Direct Connect/Bluetooth/WiFi- Direct based approaches.

The application is primarily intended at groups who want to share medium-large files across to all participants in a group. It intends to make the entire flow of sharing a file with a group smoother, while making it faster in the process. The system contains 3 modules as follows:

1. User Presentation layer
2. Application Logic layer
3. Protocol Bridge

3.2.2 Product Functions

3.2.3 User Classes and Characteristics

As it is intended to provide a seamless experience while sharing files across groups, all the users are treated in the same manner. Also, as the proposed system is fully decentralised there is no need for any privileged super user. The user may utilise this application to share files across groups in a convenient and user friendly manner. A user can register on the platform, create groups, send text-based messages and can have files sent over with minimal effort.

3.2.4 Operating Environment

The system is aimed to work on any modern web browser, across different operating systems. As the GNU/Linux is a free and open source operating system, it maybe be installed in the user's computer without any hassle.

3.2.5 Design and Implementation Constraints

1. Regulatory Policies: NA
2. Hardware Limitation: Devices must have a stable internet connection, Should have decent computational power
3. Interfaces to other application: NA
4. Technology Stack: JavaScript, React, Python, Django
5. Communication Protocols: Communication over TCP
6. Safety and Security Considerations: NA
7. Criticality of the Application: As the application is decentralised, its availability is constrained by the availability of the people participating in the actual process of file sharing.

3.2.6 User Documentation

All documentation will be made in accordance with requirements pertaining to open source software under the GNU General Purpose License. Additionally, user documentation will be published with the concerning paper to determine proper use.

3.2.7 General Constraints

Users who intend to use the application should have a decent internet connection, while also consenting to sharing of their bandwidth/CPU resources. They should have a device wherein they can afford sparing some amount of network/cpu resources for the application.

3.2.8 Assumptions and Dependencies

It is assumed that most of the participants using the application are willing to share their computing resources and have access to a network with decent bandwidth (10-50 Mbps). It is also assumed that the users are interested in participating in a peer driven model in sharing the files. The primary dependency of the given application would be the Web Torrent/ BitTorrent protocol itself, which is used to transfer the files. It is intended to locate a popular torrent tracker and to use the same for peer discovery.

3.3 External Interface Requirements

This section describes the user interfaces, hardware interfaces, software interfaces and communication interfaces.

3.3.1 User Interfaces

The user interface for the project is a web-based React application. It is similar to that of a group messaging application, wherein users can send messages and create groups. The users would then be able to share files with a given group. The other members would then be notified of the file being transferred.

3.3.2 Hardware Interfaces

Any device that is capable of having access to the internet and supporting a modern web browser can use the system.

3.3.3 Software Interfaces

The system works well on any modern operating system, equipped with a TCP Network Stack and a file system. The UI logic is almost entirely built in JavaScript and React.

3.3.4 Communication Interfaces

The torrent protocol would be used as the primary communication protocol which uses TCP under the hood.

3.4 Hardware and Software Requirements

This section includes the hardware and software requirements of the proposed system.

3.4.1 Hardware Requirements

The frontend being exposed via the browser, the hardware should be capable of running a browser smoothly. The hardware requirements include:

- 1.6 GHz or faster processor
- 4GB of Ram
- 128GB of Disk Space is recommended
- 10-50 Mbps Network Bandwidth

3.4.2 Software Requirements

This project requires the following software for functioning:

1. Operating System: Any modern Operating System.
2. JavaScript: JavaScript is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative styles.
3. React: React is a powerful frontend library used to manage the view. It uses a virtual DOM making the process of updating the DOM quite fast on the browser.
4. Bit Torrent Client: A bit torrent client implementation is used at the library level to create a protocol bridge in order transfer data using bit torrent protocol.
5. Python: Python is a powerful programming language with efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. Python is also suitable as an extension language for customizable applications.
6. Django: Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so that the developers can focus on writing their app without needing to reinvent the wheel. It's free and open source.

3.5 Functional Requirements

This section gives the details of the features and functions this system provides for various users and additional information on how each module works.

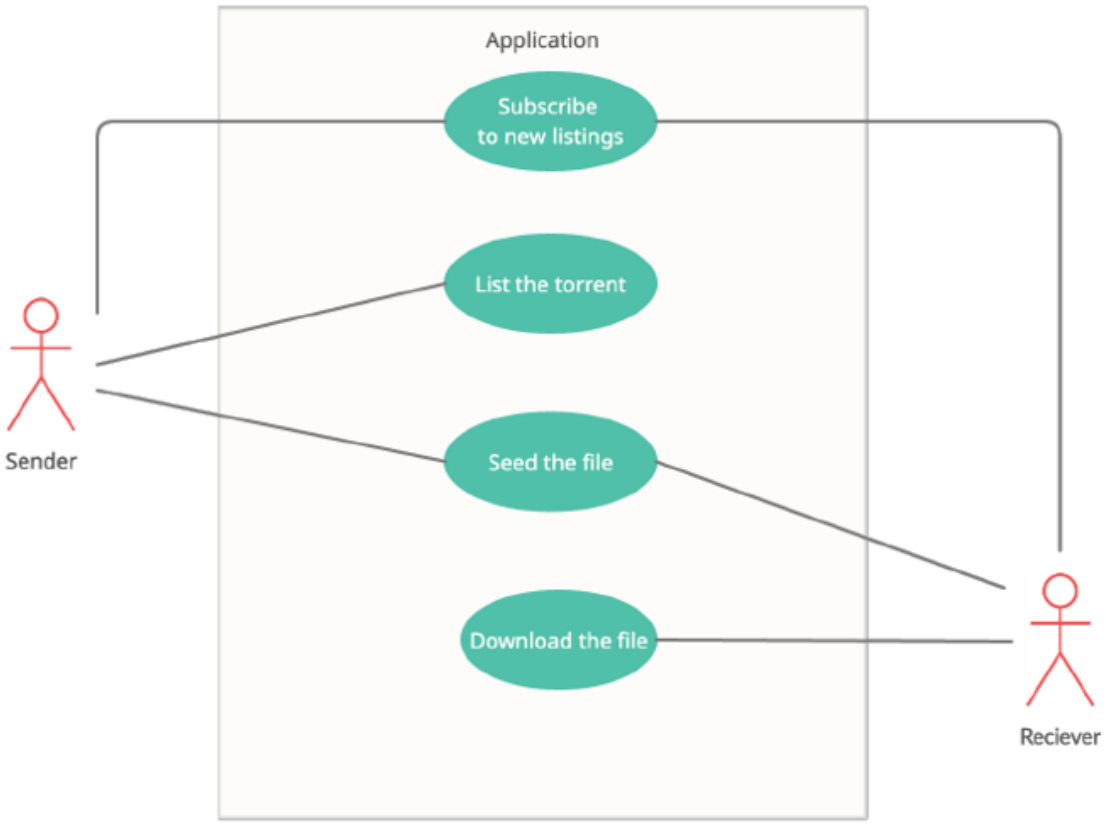

Figure 3.1: Use Case Diagram of the Application

3.5.1 User Registration

- Description: Registers the users with the required information, and gets the user's contact information.
- Actors: The actors include User and the application logic layer.
- Precondition: NA
- Main Flow of Events: The user registers with their details successfully on the application.
- Alternative Flow of Events: The registration is unsuccessful and yields in an error.
- Post Condition: The user details is stored in the application logic layer.

3.5.2 Creating Groups

- Description: Creating groups to make the user experience while sharing files seamless.
- Actors: The actors include User and the application logic layer.
- Precondition: The users should be registered successfully.
- Main Flow of Events: The user adds other members on the platform to create groups successfully.
- Alternative Flow of Events: Group creation is unsuccessful which yields in an error.
- Post Condition: Groups have now been created and users can share files in these groups.

3.5.3 File Transfer

- Description: Transferring files across groups.
- Actors: The actors include User, the application logic layer and the protocol bridge.
- Precondition: The groups should be created successfully.
- Main Flow of Events: A user who intends to share a file with the group, does so by broadcasting that it is listing this torrent. The other users in the group have subscribed to this event and on receiving this start downloading the file from the user via the WebTorrent/BitTorrent protocol thought the protocol bridge.
- Alternative Flow of Events: The file doesn't get sent over due to peers not getting discovered.
- Post Condition: The file has been successfully sent over to the users of the group via P2P.

3.6 Non-functional Requirements

3.6.1 Performance Requirements

Major performance requirements are:

- The system has to be interactive, with minimal delays.
- It should be able to handle large number of users in a group.
- It should be able to transfer files at around 10 Mbps.

3.6.2 Safety Requirements

As this application can be used to transfer files, including those which are malicious in nature, usage should be restricted to adults, or children under parental supervision.

3.6.3 Security Requirements

To be able to use torrent protocol, the IP of the seeder is required. However, exposing this would expose the location of the seeder and is to be avoided by broadcasting the IP to all the group participants. The packets transported are to be also encrypted to avoid packet sniffing.

3.6.4 Software Quality Attributes

- Reliability: The files should be transferred at reasonable speed and its integrity is to be verified on arrival
- Robustness: The system should be able to handle a lot of users in a group and should be able scale with the size of the files as well.
- Testability: The system must be modularized and have a proper structure in order to conduct tests.
- Functionality: It should serve as a seamless file sharing application across groups.
- Usability: The system is easy to use and handle.
- Correctness: The integrity of the file transferred is verified upon arrival.
- Maintainability: Different versions of the system should be easy to maintain.
- Flexibility: The system should be flexible enough to modify.

3.7 Other Requirements

Besides the functional and non-functional requirements, there are also other requirements like ease of use and error tolerance. Ease of use describes the ease with which users can interact with the system and error tolerance speaks about the extent to which error is tolerated if it occurs in the system.

- Ease of Use: The proposed system should have an enhanced user experience enabling users to navigate and share files with minimal effort.
- Error Tolerance: Corruption of data in the files sent over is to be minimized and avoided.

Chapter 4

System Design

4.1 System Architecture

System architecture shows the various modules of the system and the interaction between them. Users can log into the system and can upload a file. A user is signed-up via the application's front-end. The local application houses a local database to store the details of the user and other associated metadata. The application logic abstracts the data from the database/P2P file sharing bridge to show meaningful data to User presentation layer. The file sharing bridge is responsible for connecting the application logic to the file sharing protocol in itself, in this case the bit-torrent protocol.

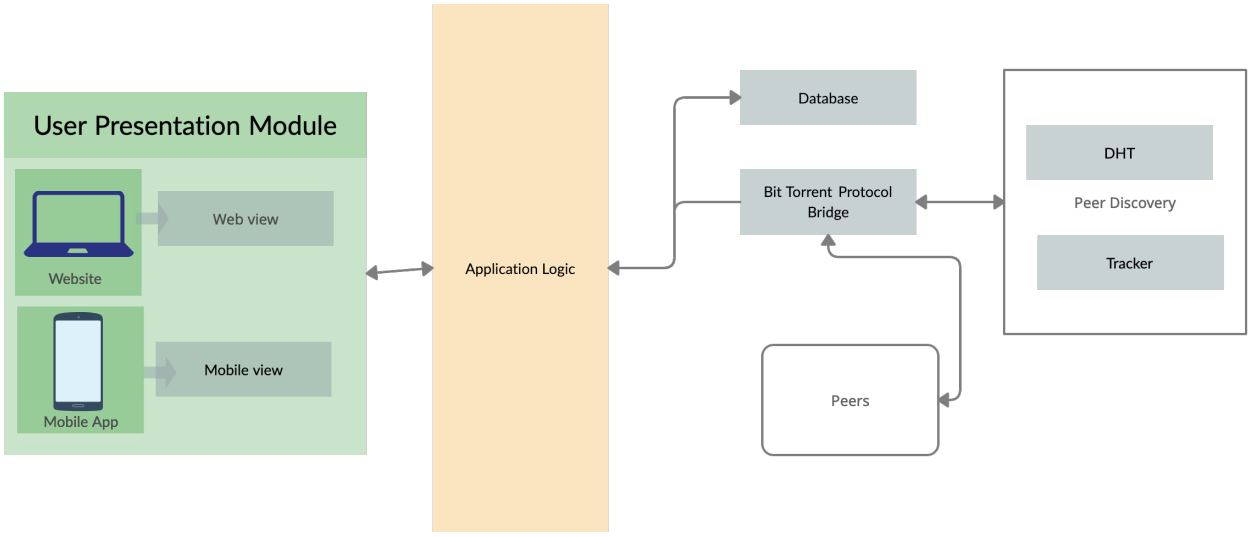

Figure 4.1: System Architecture

4.2 Block Diagram

The block diagram describes the overall flow of data and control between the major modules of the system. After a user uploads a file through the user presentation layer, it is then passed on to the application logic layer and finally to the Bit-Torrent Protocol through the protocol bridge. The Bit-Torrent Protocol handles the transfer of files between user. Once the file reaches the destination, it travels through the protocol bridge to the application logic layer and finally is viewed by the user with the help of the user presentation layer. The block diagram describing this flow can be found below.

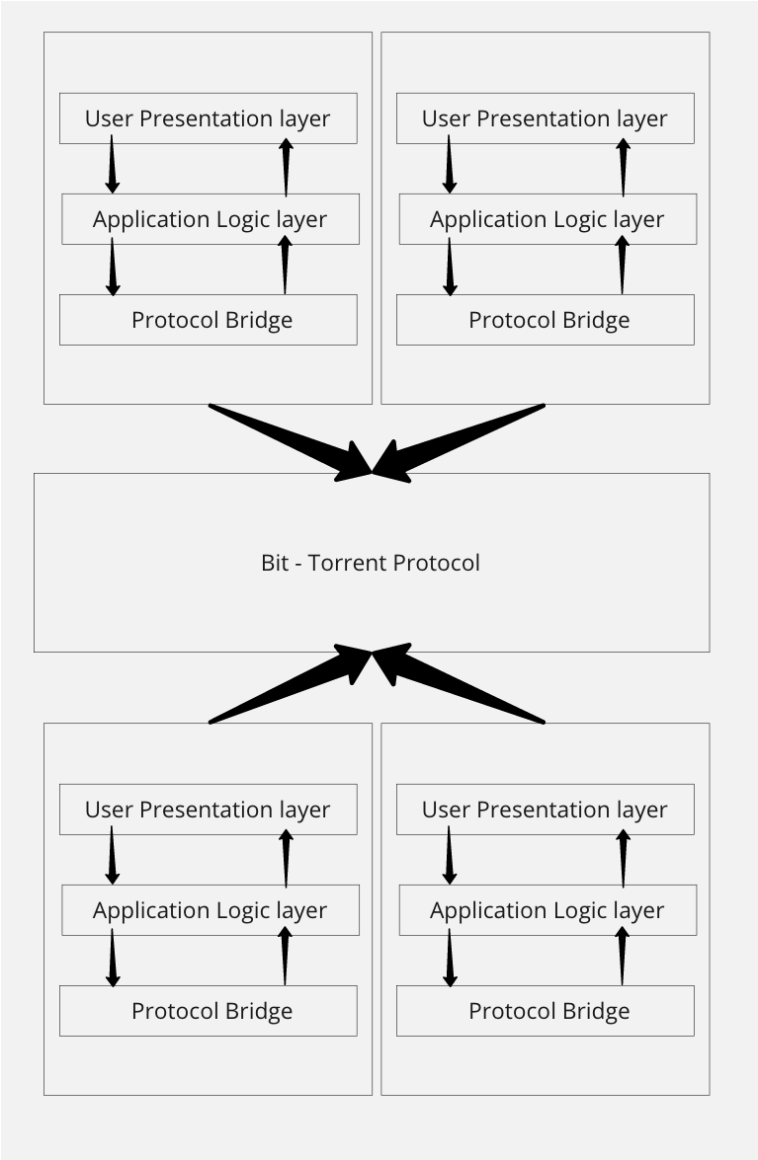

Figure 4.2: Block Diagram

4.3 Use Case Diagram

Figure 4.3: Detailed Use Case Diagram of the Application

4.4 Activity Diagram

A flowchart or activity diagram explaining the user activity flow.

4.4.1 Activity Flow for User Sign Up

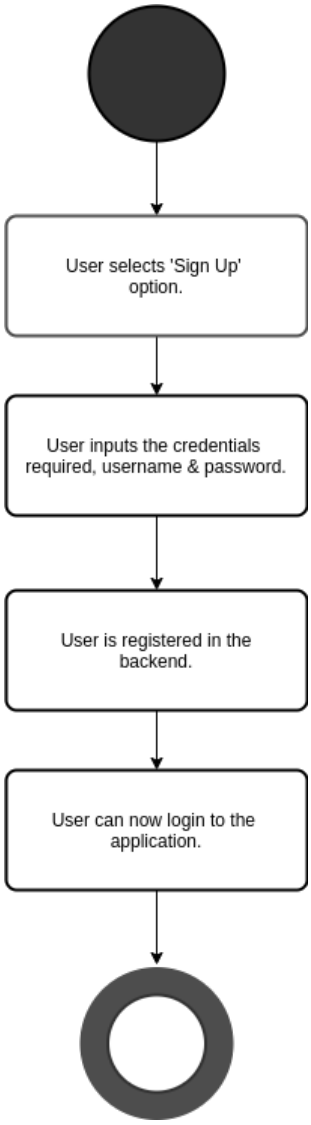

Figure 4.4: Activity Diagram for User Sign Up

The User Sign Up activity flow starts with the user selecting the Sign Up option. The user inputs a username and password and is registered into the system. It auto logs in the user after that.

4.4.2 Activity Flow for User Sign In

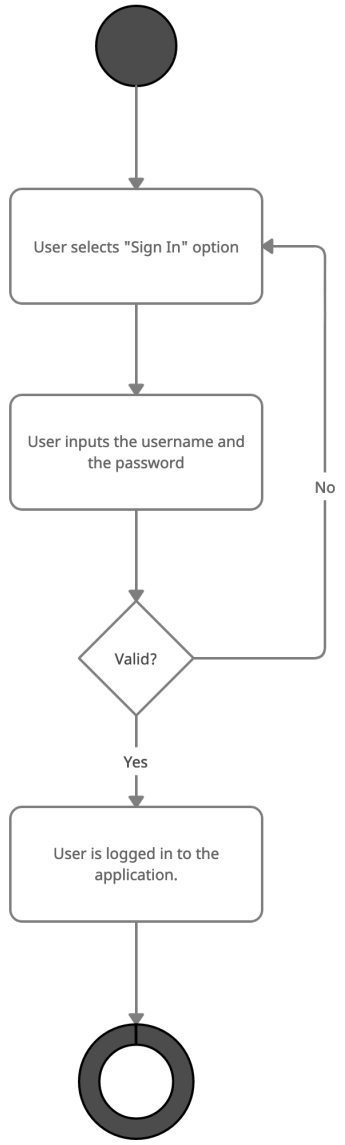

Figure 4.5: Activity Diagram for User Sign In

The User Sign In activity flow starts with the user selecting the Sign In option. The user then inputs their username and password and the system checks whether it is valid or not. If it is valid then the user is logged in, or else the user has to try again.

4.4.3 Activity Flow for Creating a Group

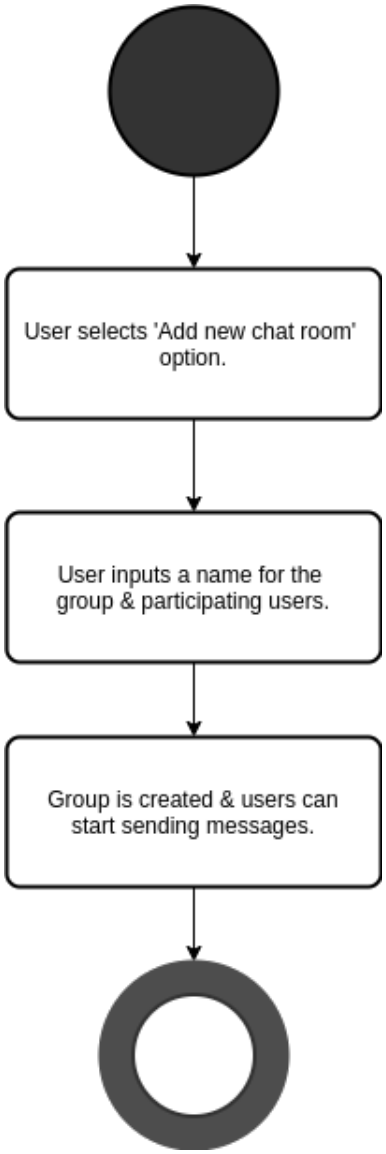

Figure 4.6: Activity flow for creating a Group

The group creation activity flow starts with the user selecting the New Group option and then inviting other participants for the group using their usernames. The user can then send files and messages in the group.

4.4.4 Activity Flow for Attaching a File

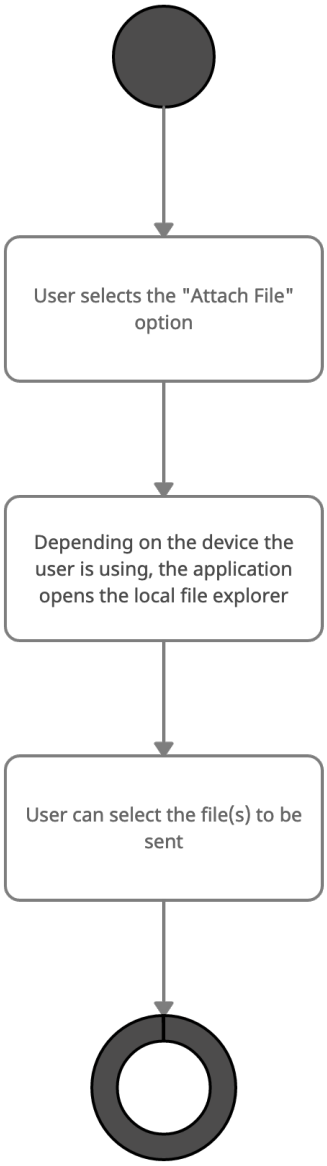

Figure 4.7: Activity Flow for Attaching a File

The attaching a file activity flow starts with the user selecting the Attach File option. Then the user can proceed to select the file(s) using the local file explorer.

4.5 Input Design

The inputs first name, last name, username, email and password are required to register to the application. For authenticating via the login page, the inputs required are username and password.

Message inputs are text messages and files. The inputs for creating a group include the name of the group as well as the participants.

4.6 Database Design

This section details the various tables used in the design of the system, namely User Table, Chat Table and Message Table.

The User table is used to store the details of the user, which includes the User ID, the username, the groups the user belongs to.

The Chat table contains the specifications of the chat room including the Room ID, the group name and the users belonging to the group.

The Message table describes the various attributes of the message, which includes the Message ID, the text message, the Magnet URI of the torrent file uploaded, the user sending the message, the group to which the message is sent and the timestamp of the message.

A detailed diagram describing the various tables and their attributes, along with the primary keys and foreign keys of each table is given below.

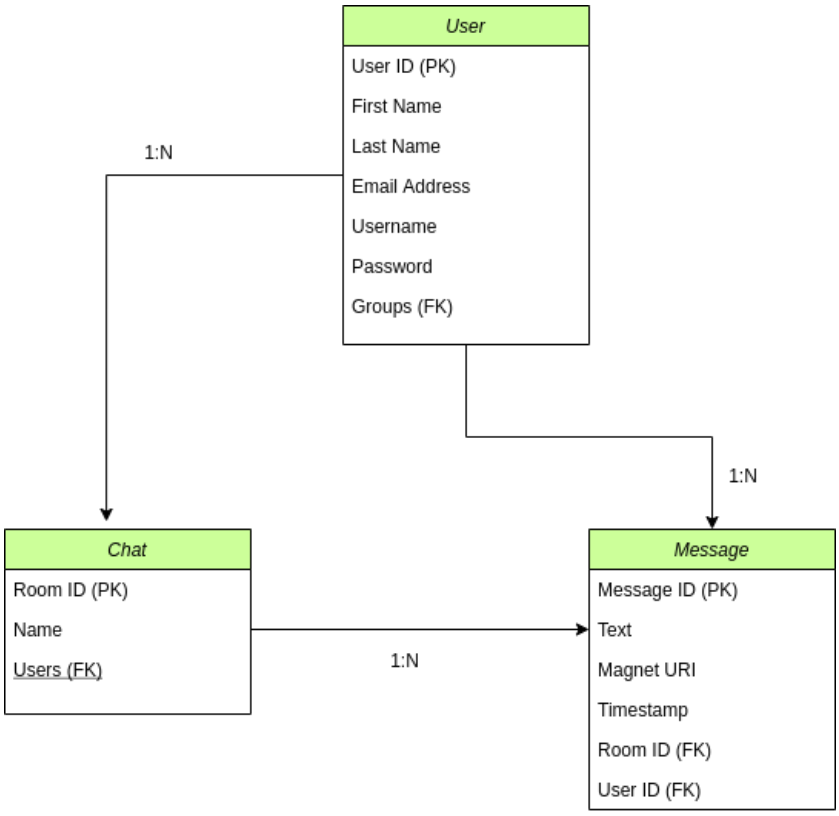

Figure 4.8: Database Design

4.7 Libraries and Packages Used

- Axios
- WebTorrent
- React
- React Multiple Select Dropdown Lite
- @mui/icons-material

Axios is a simple promise based HTTP client for the browser and node.js. Axios provides a simple to use library in a small package with a very extensible interface. On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

WebTorrent is a streaming torrent client for the web browser and the desktop. WebTorrent is written completely in JavaScript and uses WebRTC for peer-to-peer transport whenever possible. No browser plugins, extensions, or installation is required to use WebTorrent in your browser.

React is a JavaScript library for creating user interfaces. The react package contains only the functionality necessary to define React components. It is typically used together with a React renderer like react-dom for the web, or react-native for the native environments.

React Multiple Select Dropdown Lite is a lightweight Multiple/Single Select Pure functional component for react using React-Hooks. It is a React component providing multiselect functionality with various features like selection limit, CSS customization, checkbox, search option, disable preselected values, flat array options, keyboard navigation and grouping features.

@mui/icons-material package provides the Google Material Icons converted to SVG Icon components. Google has created over 2,000 official Material icons, each in five different "themes". For each SVG icon, we export the respective React component from the @mui/icons-material package.

4.8 Module Description

4.8.1 User Presentation Layer

This layer is responsible for the user interface/view of the application. It takes user input at various stages and passes it onto the application logic layer. On receiving the response from the application logic layer, it updates the view appropriately.

4.8.2 Application Logic Layer

The layer responsible for abstracting the interactions with the file sharing bridge module and the database.

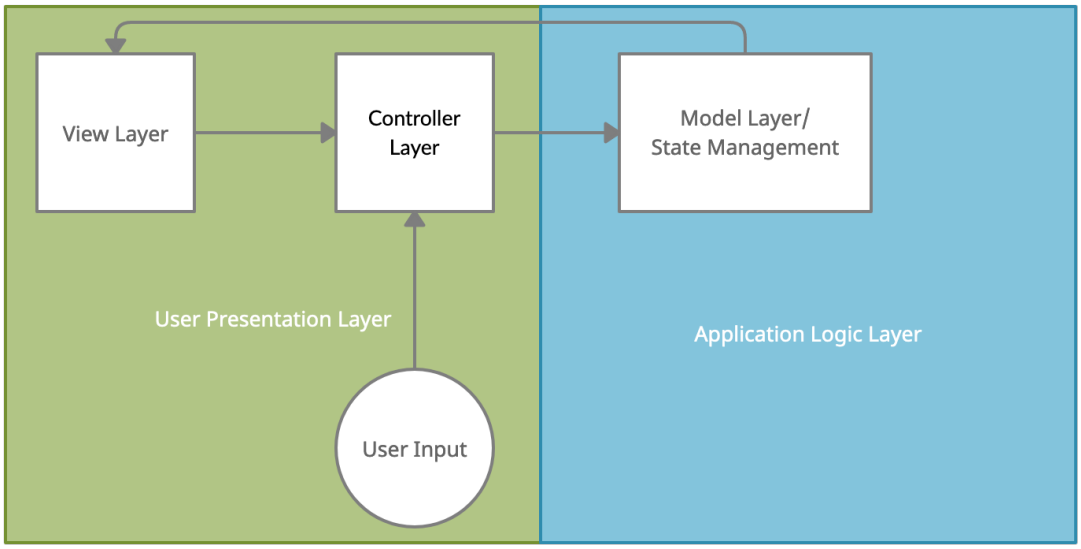

Figure 4.9: User Presentation Layer

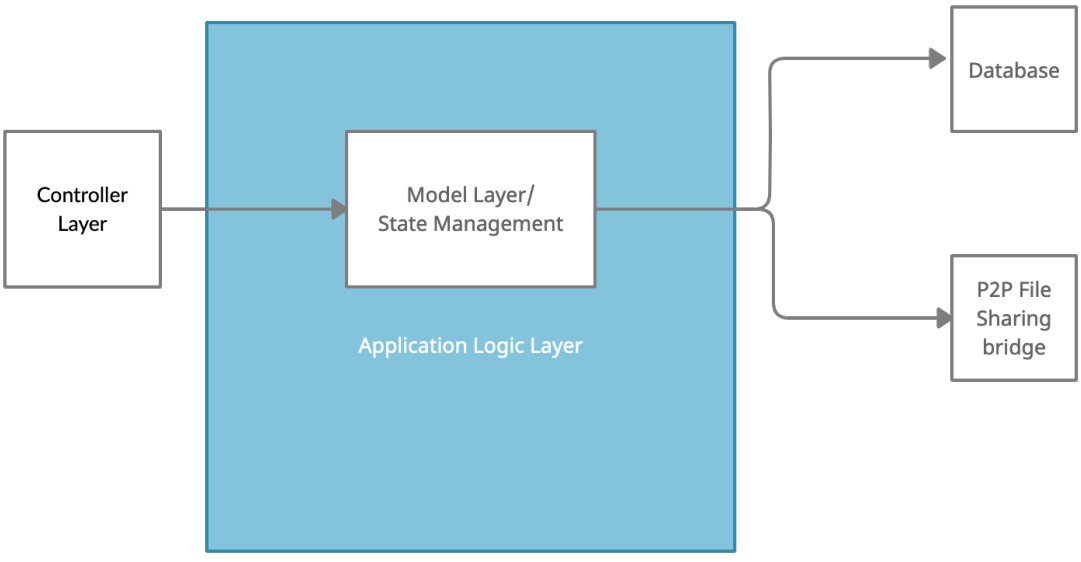

Figure 4.10: Application Layer

4.8.3 P2P File Sharing Bridge Layer

The layer responsible for initiating actions on the P2P protocol itself, i.e. seeding a new file, resuming a file download, pausing the current download and so on. Figure 4.11 details the P2P file sharing layer.

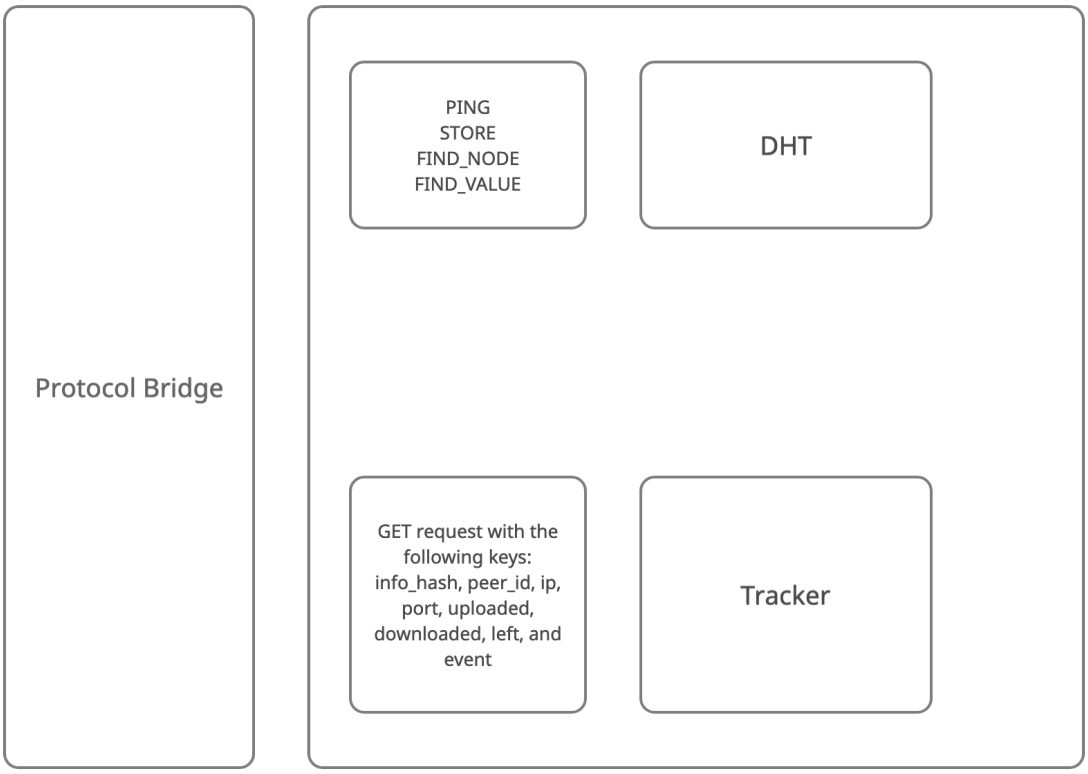

Figure 4.11: P2P Bridge

4.9 Data Flow Diagram

The data flow diagram is a graphical representation of the flow of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system, which can later be evaluated. Data Flow Diagrams can also be used for the visualization of data processing.

4.9.1 Level 0 DFD

Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all underlying details. The data flow in level 0 DFD takes place in the following way: A user can login/register to the p2p file sharing system and can share files using the P2P protocol in groups. Figure 4.12 depicts the Level 0 DFD.

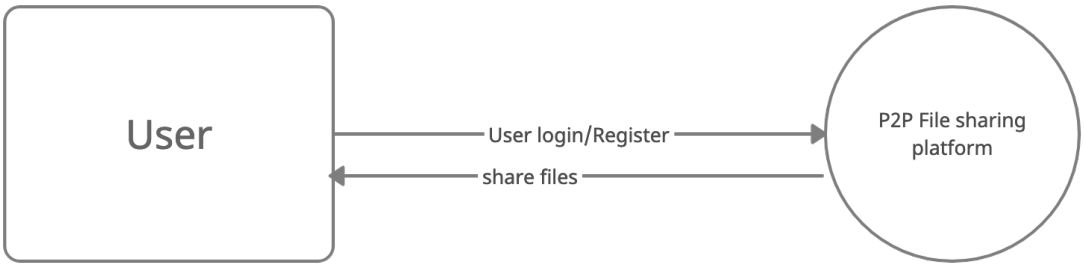

Figure 4.12: Level 0 DFD

4.9.2 Level 1 DFD

Level 0 DFD is broken down into more specific Level 1 DFD. Level 1 DFD depicts basic modules in the system and the flow of data among various modules. Level 1 DFD also mentions basic source of information. The data flow in Level 1 DFD takes place in the following way: The user creates a group, initiates file transfer, the file is transferred through the p2p network

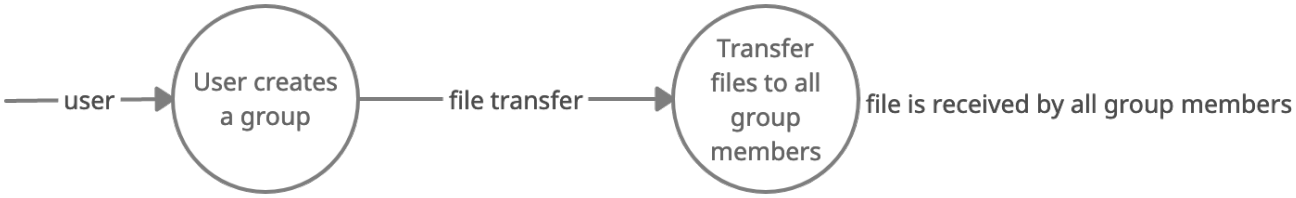

Figure 4.13: Level 1 DFD

4.9.3 Level 1.1 DFD

When a user wants to initiate a file transfer, first it updates the tracker by making a GET call to the tracker or by updating the DHT. The other members in the group would be broadcasted after this step, then the other users can search the DHT/ get the seeder information from the tracker and start downloading the file.

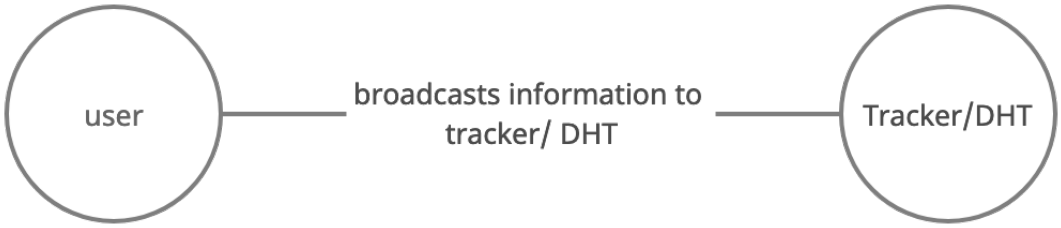

Figure 4.14: Level 1.1 DFD

Chapter 5

Implementation

5.1 Overall Working

The file sharing application is implemented as a web application where the users can register and login with their credentials. The user's data is registered via a POST request and saved as a user record in the database. User can create a chat room or select an existing chat room. To create a chat room, the user provides the room name and names of participants from the list of users. This information is sent through a POST request to the Chat API and the chat room is created and recorded in the chat table. Messages to share with participants in the room can be either a text message or a file. Each message is associated with a particular chat room and a user and it is recorded in the messages table. When a user selects a particular chat room, all the messages in that chat room are retrieved from the server via long polling. Messages retrieved are displayed in the corresponding chat room, sorted in the order of the time at which they were sent. To send a file, the user uploading the file selects a file from the local system. The torrent file of the selected file is generated and the magnet URI of this torrent file is uploaded to the server. The participants receiving this magnet URI start downloading the torrent or seed them, thus enabling P2P transfer.

5.1.1 Backend

The backend is used as an intermediary in our application to facilitate the storing and transfer of messages. It stores user information (which if required can be anatomised) to identify the author of each message/file transfer. It primarily exposes 3 API endpoints - authentication, message, chatroom. Authentication endpoints are used to register a user, to create a user session by logging in. Chat room endpoint is exposed to create/update details of chat room/group. Message endpoint is related to all the messages sent and received in each chatroom. The backend uses an ASGI server, which Django provides. This is interfaced with the Django application itself. It was written using the Django rest framework, which aids in creation of well structured RESTful APIs. The type of authentication being used here is session auth, with details related to a session stored in the database. Postgres is used here, with the Django ORM being used as an Object Relational Mapping to ease the interaction with the database.

5.2 Algorithms

Algorithms are step by step instructions to carry out a process. The various algorithms used in the proposed system are mentioned in the following sections

5.2.1 Overall Algorithm

1. Start.
2. Register as a user of the application.
3. Login with the user credentials.
4. Once authenticated, create a chat room by providing the room name and the names of participants from the list of users.
5. After creating the group, input a text message or upload a file and click the send button.
6. The file uploaded by the sender will be downloaded for other members in the group.
7. End.

5.2.2 Algorithm for Chat Display

1. Start.
2. Each message is associated with a particular chat room in the user interface.
3. When a user selects a particular chat room, all the messages in that chat room are retrieved from the server via long polling.
4. The retrieved chat messages are displayed, sorted in the order of the time at which they were sent.
5. End.

5.2.3 Algorithm for File Upload

1. Start.
2. The file is selected from the local system.
3. The torrent file is generated for the selected file.
4. The magnet uri of this torrent file is uploaded to the server.
5. End.

5.2.4 Algorithm for P2P Transfer

1. Start.
2. The torrent magnet uri is uploaded to the server.
3. The other users in a chat room receive this torrent magnet uri.
4. All the users in the chat room start downloading the torrent/seeding them enabling P2P transfer of the file across the chat room.
5. End.

5.2.5 Updating the DHT (Included in the WebTorrent Library)

1. Start.
2. For the given torrent file, the hash is computed.
3. The hash with the IP address is stored on the DHT via STORE(hash,ip addr) function of the DHT.
4. The DHT is then updated.
5. End.

5.2.6 Calculating Distance while Traversing the DHT (Included in the WebTorrent Library)

1. Start.
2. Every node is assigned a node id.
3. The XOR between any 2 node id's is calculated.
4. This is treated as the distance between any two given nodes.
5. End.

5.3 Development Tools

A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or support other programs and applications. The term usually refers to relatively simple programs, that can be combined together to accomplish a task. The most basic tools are a source code editor and a compiler or interpreter. Other tools are used depending on the language, development methodology, and individual engineer, and are often used for a discrete task, like a debugger.

5.3.1 Visual Studio Code

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and MacOS. Visual Studio Code is a source code editor that can be used with a variety of programming languages. Instead of a project system, it allows users to open one or more directories, which can then be saved in work spaces for future reuse. The source code is free and open source. Some of its features are:

- It includes support for debugging, embedded Git control, syntax highlighting.
- Provides intelligent code completion, snippets, and code refactoring.
- It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.
- Visual Studio Code can be extended via plug-ins, available through a central repository. This includes additions to the editor and language support.
- Visual Studio Code has out-of-the-box support for almost every major programming language.
- Cross-platform (Windows, Linux, MacOS).

5.3.2 GitHub

GitHub is a web-based hosting service for version control using Git. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management etc. for every project. Main features are:

- Seamless code review.
- Project and team management.
- Provides quality documentation.
- Supports large amounts of code hosting.

5.3.3 Replit

Replit is a coding platform that is used write code and host apps. It also has many educational features built-in, making it great for teachers and learners too. For software developers, Replit is used as cloud-based IDE. Having a cloud IDE offers several advantages. There's no setup – one can access your environment from any device, including phone or tablet, and everything will work efficiently. One can manage things like dependencies, build scripts, and environment variables in a single place and always be in sync. Replit can be used in different ways:

- Code editors (e.g. VS Code, Sublime Text, IntelliJ)
- Development environments (e.g. your operating system, and build tools like npm or pip)
- Cloud providers (e.g. AWS, Netlify)

- Team collaboration tools (e.g. Google Docs, GitHub)
- Teaching tools (e.g. Canva, Moodle, Blackboard)
- Learning tools (e.g. Codecademy, Coursera, Udemy, Udacity)

Chapter 6

Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs, and to verify that the software product is it for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development
- responds correctly to all kinds of inputs
- performs its functions within an acceptable time is sufficiently usable
- can be installed and run in its intended environments, and
- achieves the general result its stakeholders desire.

6.1 Testing Methodologies

Software testing methodologies are for making sure that the software products/systems developed have been successfully tested to meet their specified requirements and can successfully operate in all the anticipated environments with required usability and security. Software testing methods are traditionally divided into white and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White-Box Testing: After accessing the source code, tests for internal structures or workings of a program are done, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level.

Black-Box Testing: Treats the software as a black-box , examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Here the black-box testing is used for the system. The testing methods applied were:

- Unit Testing: Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
- Integration Testing: Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing.
- System Testing: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

6.2 Unit Testing

Unit testing refers to the testing of individual software modules or components that make up an application or system. It validates that each module of the software performs as designed. In the following section, we have included the various test cases and the corresponding images for applicable test cases.

6.2.1 User presentation module

Test1: Register

Test CaseID	**Steps to execute**	**Expected results**	**Actual Results**	**Status**
TC_01	Enter valid values in the required fields. Click the Register button	User should be registered successfully. User redirected to Login page.	User is registered and redirected to Login page.	Pass
TC_02	Enter valid values in the required fields. Click the Register button	User should be registered successfully. A successful registration message should be shown	User not registered.	Fail
TC_03	Enter invalid value in any of the fields	An error message should be shown	Error message shown	Pass
TC_04	Do not enter value in any required field	It should show a mandatory symbol on mandatory fields	Mandatory symbol shown on mandatory fields	Pass

Table 6.1: Test cases of register feature

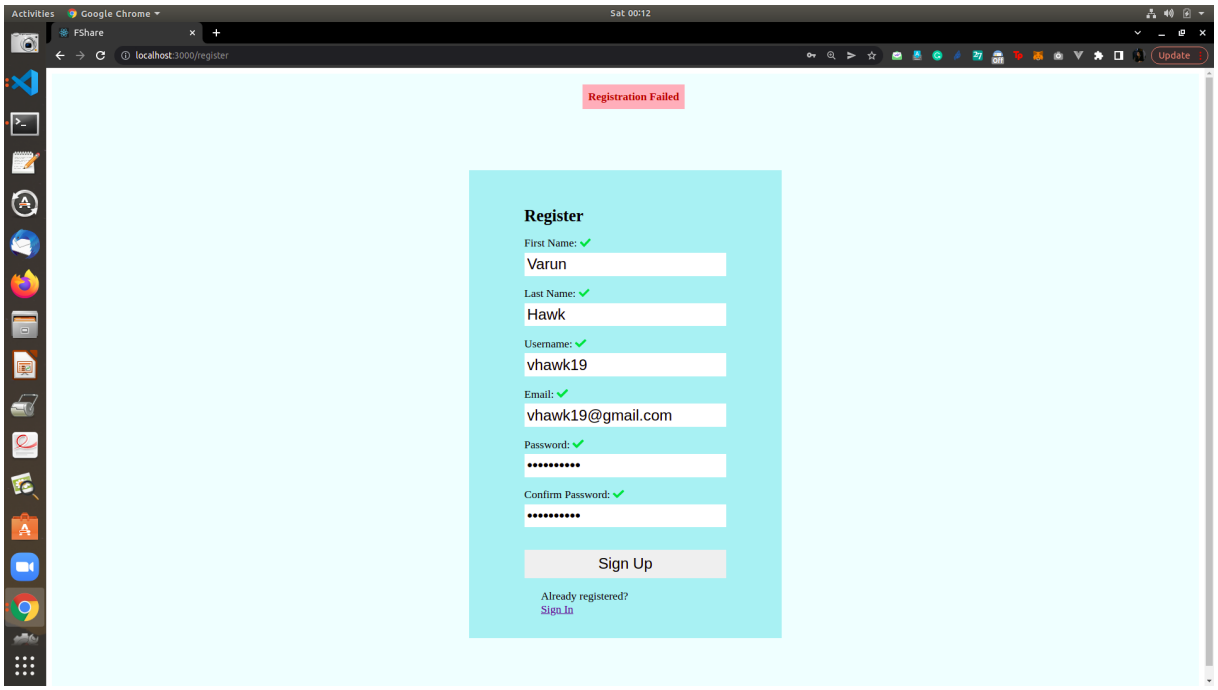

Figure 6.1: TC_02: Registration Failed

Test2: Login

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_05	Enter valid username and password. Click the Login button	User should log in	User is logged in. Rooms are displayed	Pass
TC_06	Enter valid username and password. Click the Login button	User should log in	Login Failed	Fail
TC_07	Enter invalid username and/or password	User should not log in. Error message should be displayed	User not logged in. 'Login failed' message is displayed	Pass

Table 6.2: Test cases of login feature

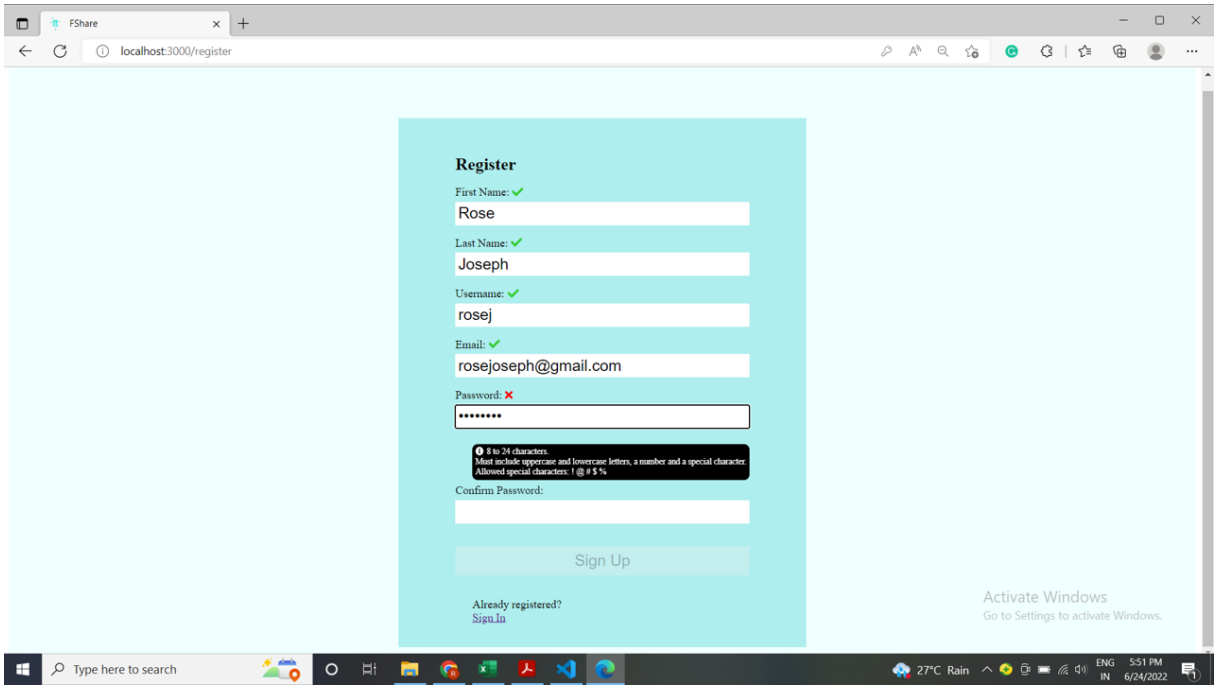

Figure 6.2: TC_03: Invalid password

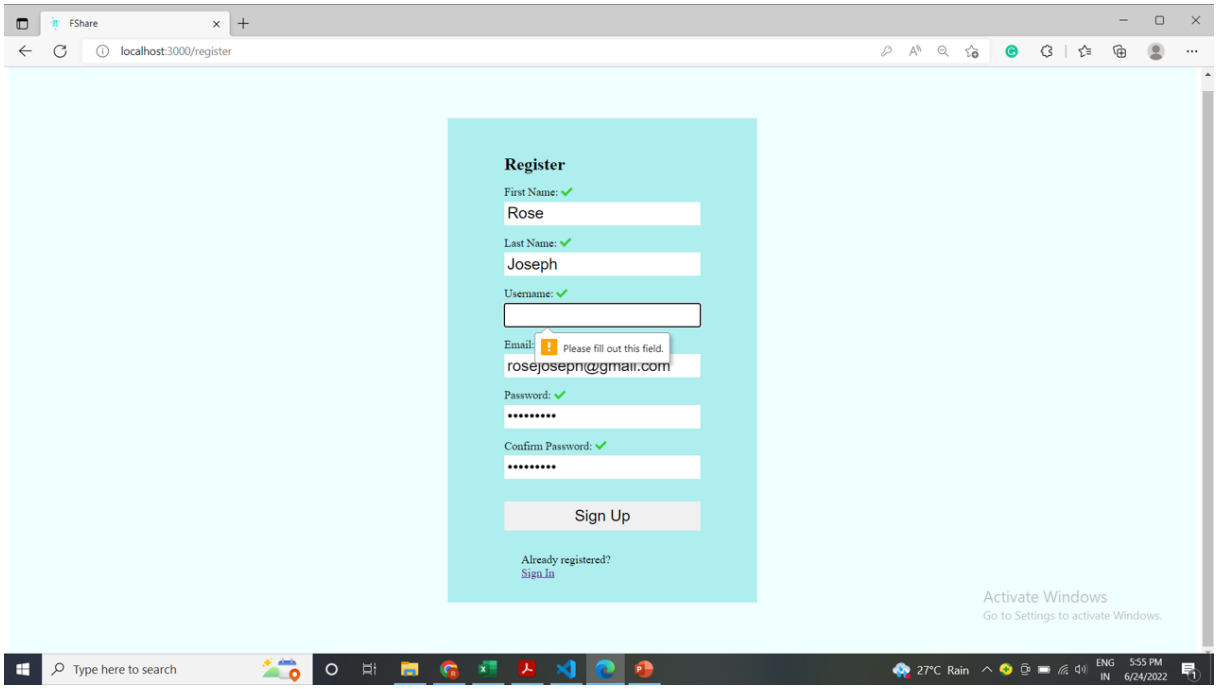

Figure 6.3: TC_04: Username not filled

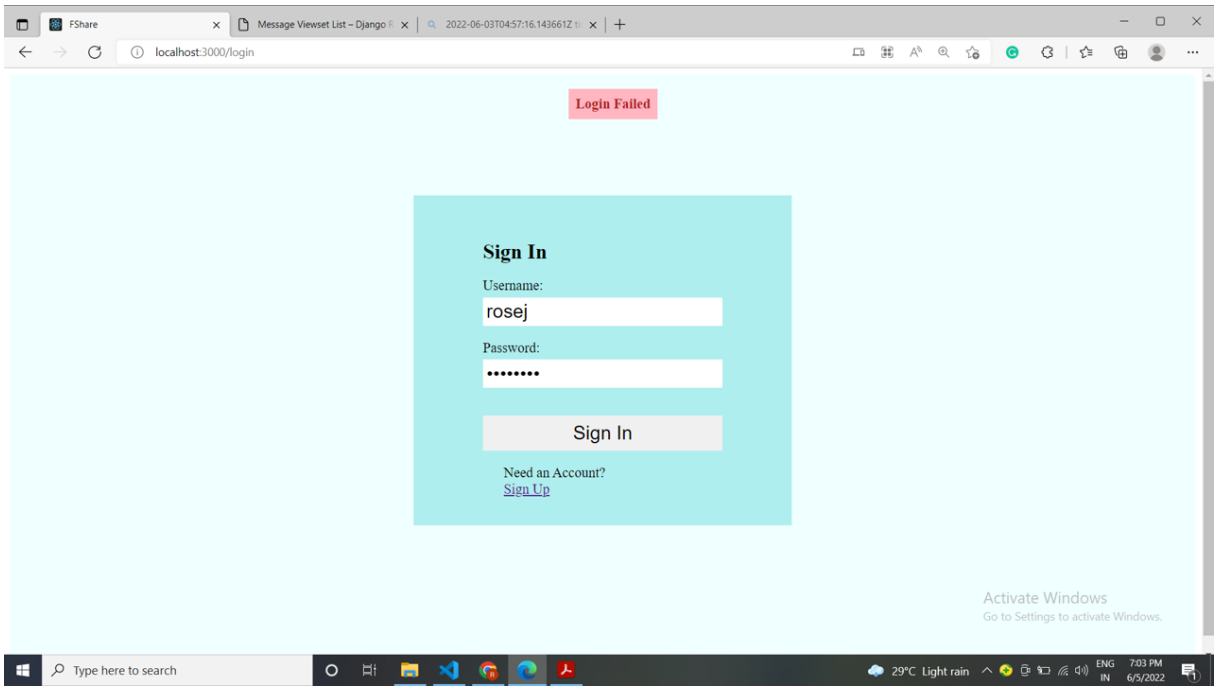

Figure 6.4: TC_07: Invalid password

Test3: Chat Room Creation

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_08	Enter room name and add participants. Click the create room button	New room should be created and it should be shown in the sidebar. 'Chat room added' message is shown	New room is added and the message is displayed	Pass
TC_09	Enter room name and add participants. Click the create room button	New room should be created and it should be shown in the sidebar. 'Chat room added' message is shown	New room not created	Fail

Table 6.3: Test cases of chat room creation feature

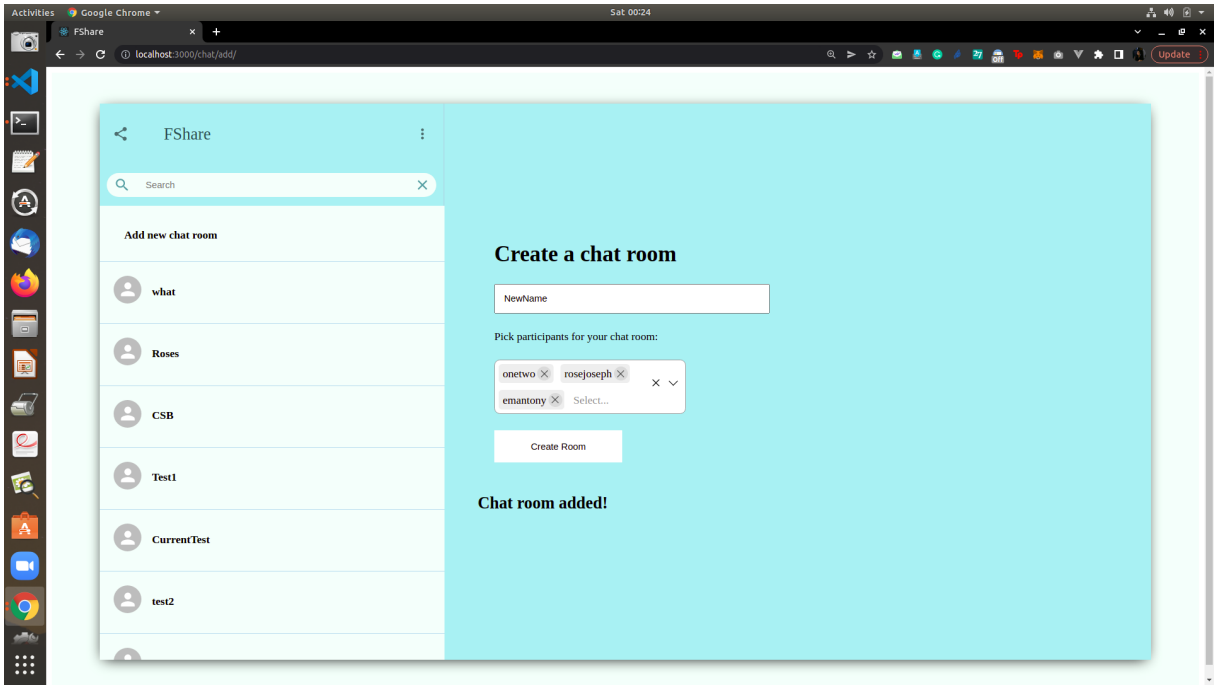

Figure 6.5: TC_08: Chat room added successfully

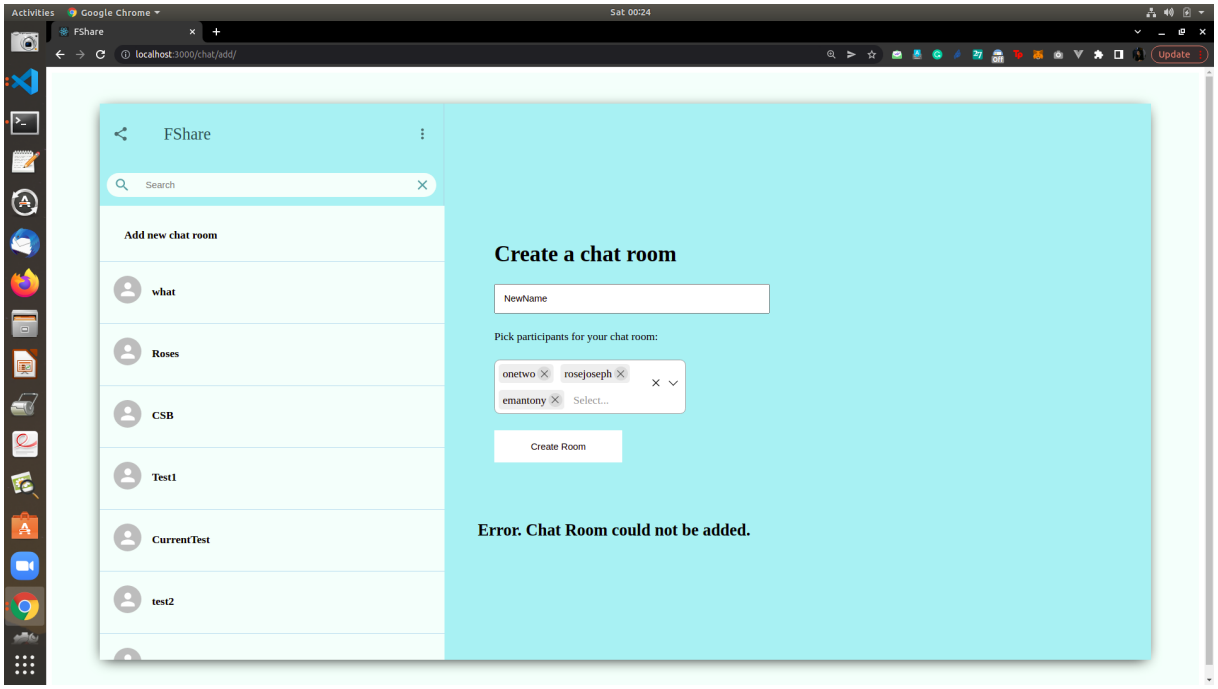

Figure 6.6: TC_09: Chat room could not be added

Test4: Send Text Message

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_10	Enter message in the text input area of a chat room. Click the send button	Message should appear in message display area of all participants at corresponding positions	Message is displayed in the message display area	Pass
TC_11	Enter message in the text input area of a chat room. Click the send button	Message should appear in message display area of all participants at corresponding positions	Message not sent/displayed on the message display area	Fail

Table 6.4: Test cases of text message feature

Test5: Upload/Download file

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_12	Attach file and click the send button	The file should be sent across to the group and is available for download	The file should was sent across to the group and was available for download	Pass
TC_13	Attach file and click the send button	The file should be sent across to the group and is available for download	The file was not sent across to the group and was not available for download	Fail

Table 6.5: Test cases of upload/download file feature

6.2.2 Integration Testing

Integration testing refers to the phase of testing in which tests are performed after integrating the modules with each other. This phase of testing is primarily used to test integration failures between different modules.

Test1: Chat

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_01	Send message from the frontend client	The message should be stored in the database, with the right timestamp and data	The message was stored in the database with the right details	Pass
TC_02	Send message from the frontend client	The message should be stored in the database, with the right timestamp and data	The message did not get stored in the database with the right details	Fail

Table 6.6: Testcases of chat

Test2: Torrent File

Test CaseID	Steps to execute	Expected results	Actual Results	Status
TC_03	Upload a file and click on the send button	The torrent magnet uri of the uploaded file should be stored in the database	The magnet uri was stored in the database	Pass
TC_04	Upload a file and click on the send button	The torrent magnet uri of the uploaded file should be stored in the database	The magnet uri was not the stored in the database	Fail

Table 6.7: Testcases of seeding a file

6.3 System Testing

In system testing, all modules are integrated in the order of execution. Testing is performed over the system as a whole. For the system test to pass successfully, a user must be able to register, authenticate themselves, send messages/files to a chat room and other users in the chat room must be able to download this file.

- Conditions
    1. Decent network speed for all users in a room.
    2. The tracker is up and running.
    3. Browser has access to local file system.

- Steps

    1. Register and login to the application.
    2. Create a chat room or select an existing chat room.
    3. Send a file to the chat room.
    4. Other users download the file.
    5. Stop

Figures 8.1, 8.2, 8.3, 8.4, 8.5 depict the flow of the application when the system testing is successfully completed.

Chapter 7

Graphical User Interface

The graphical user interface (GUI), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces which require commands to be typed on a computer keyboard. Designing the visual composition and temporal behavior of a GUI is an important part of software application programming in the area of human-computer interaction. Its goal is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline named usability. Methods of user-centered design are used to ensure that the visual language introduced in the design is well-tailored to the tasks. For several decades, GUIs were controlled exclusively by a mouse and a keyboard. While these types of input devices are sufficient for desktop computers, they do not work as well for mobile devices, such as smart-phones and tablets. Therefore, mobile operating systems are designed to use a touchscreen interface. Many mobile devices can now be controlled by spoken commands as well. Because there are now many types of digital devices available, GUIs must be designed for the appropriate type of input. For example, a desktop operating system, such as OS X, includes a menu bar and windows with small icons that can be easily navigated using a mouse. A mobile OS, like iOS, includes larger icons and supports touch commands like swiping and pinching to zoom in or zoom out. Automotive interfaces are often designed to be controlled with knobs and buttons, and TV interfaces are built to work with a remote control. Regardless of the type of input, each of these interfaces are considered GUIs since they include graphical elements.

7.1 GUI Overview

Developing a user-friendly chat application which allows users to transfer files via P2P protocol was one of the primary objectives of the project. Hence, the GUI had a major role to play in the design and development of the application. To provide the users with a seamless experience in transferring files, a web-based chat application was developed. The chat interface allows users to register, create new chat rooms and send text messages along with files in the existing chat rooms. The web interface for the project was developed using React and styled with CSS. React is a popular JavaScript Library for building user interfaces, which efficiently updates and renders just the right components when the data changes. CSS is the styling language used for the web. The Material Icons or '@mui' package has been used for including various icons for styling.

7.2 Main GUI Components

The major components include:

- User Registration: The form consists of 5 fields - First Name, Last Name, Username, Email and Password. Form validation for email and password is included, along with a 'Confirm Password' prompt. The submit button, sends a POST request with the input from the user and registers the user record in the database. The registration form can be found at the '/register' route.
- Login: The form consists of two fields - Username and Password, for authentication. The form can be found at the '/login' route.
- 'Create Chat Room': The form contains two fields - Name and Participants, for the chat room. The participants for the chat room can be selected from the list of registered users.
- Chat Sidebar: The sidebar displays the list of available chat rooms. Once the user clicks on a chat room, the corresponding chat window opens to the right of the sidebar. The sidebar also contains a 'Search Box' to search through the available chat rooms.
- Chat Window: The chat window displays the list of messages in the selected chat room. The message, the creator of the message and the timestamp of the same are displayed in the window. It also allows users to input text messages and files to be sent to the chat room. 'Send' button sends a POST request with the message and the file to be added to the database. The 'Upload File Icon' allows users to upload files on click. The respective chat window can be opened through the route '/chat/room:id'.

The working of GUI can be explained through the following sequence of events.

- An unregistered user can register by providing their details through the '/register' page. Once successfully registered, they are redirected to the '/login' page.
- A registered user can login at the '/login' page and is redirected to the home-screen of the Chat on successfully logging in.
- In the chat home-screen, a list of available chat rooms are displayed. Either the user can choose one of these rooms or create a new one.
- To create a new chat room, the user clicks on the 'Add New Chat Room' option displayed on the top of the sidebar. After filling in the name and the participants for the chatroom, the user clicks on the 'Create Room' button to submit. On successful creation, a message 'Chat room added!' is displayed on the window. The newly created chat room can then be accessed through the sidebar.
- Once the user selects a chat room from the sidebar, the chat window is displayed to its right.
- The user can then input a text message or upload a file and hit the 'send' button, which adds the message to the chat room.
- Once the other participants of the chat room log in, the files are downloaded to their local device.

Chapter 8

Results

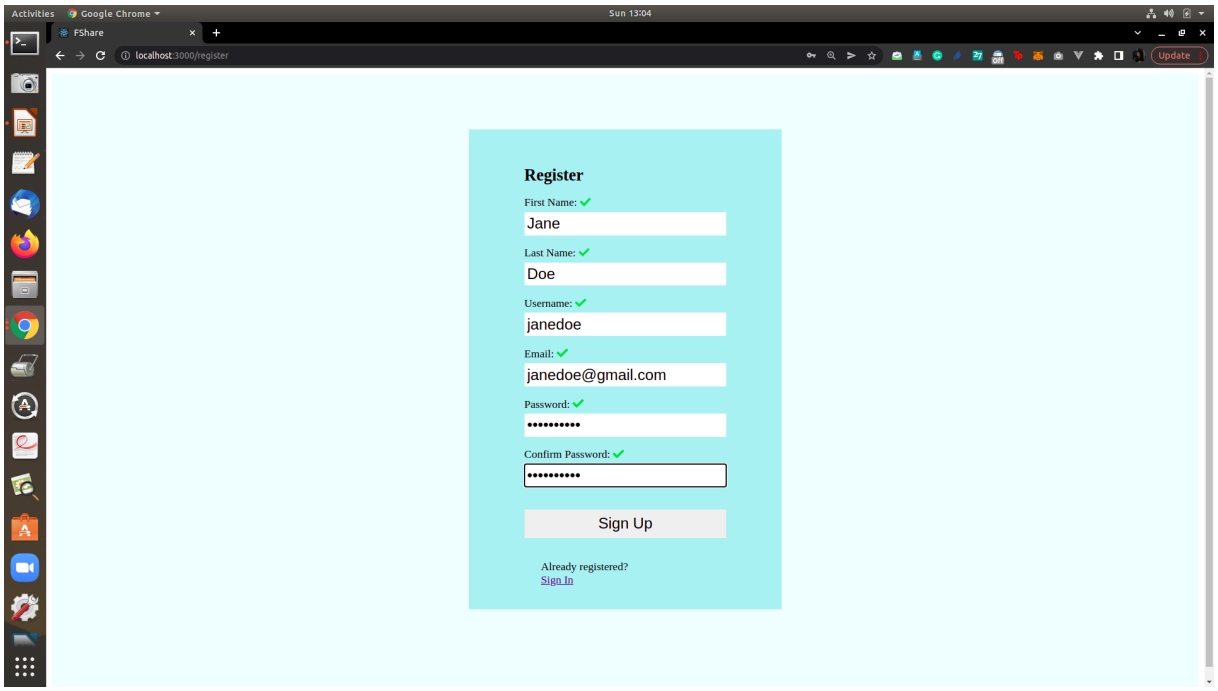

Figure 8.1: User Registration Page

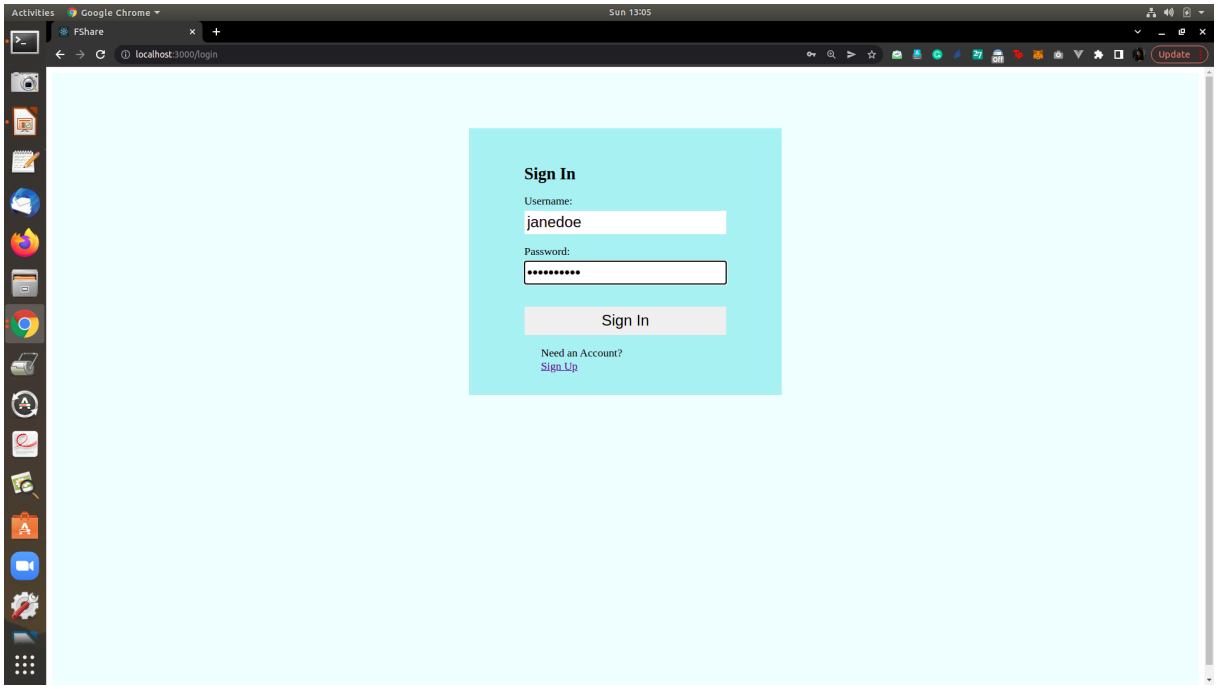

Figure 8.2: User Login Page

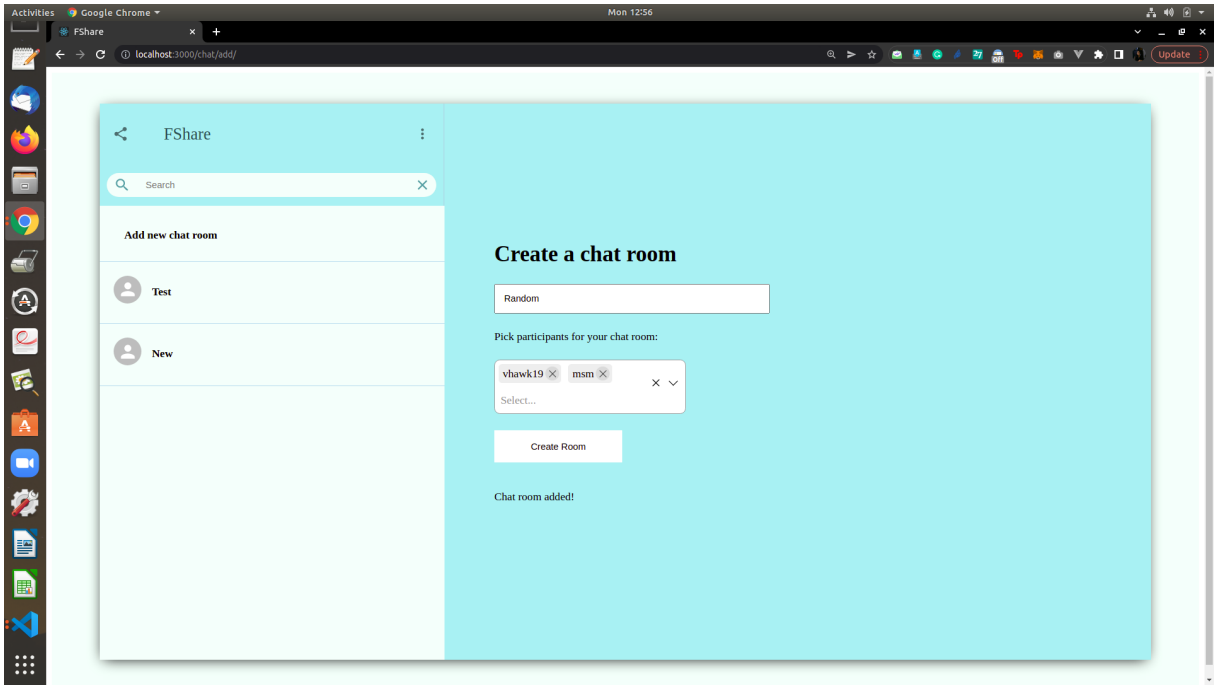

Figure 8.3: Adding a new chat room

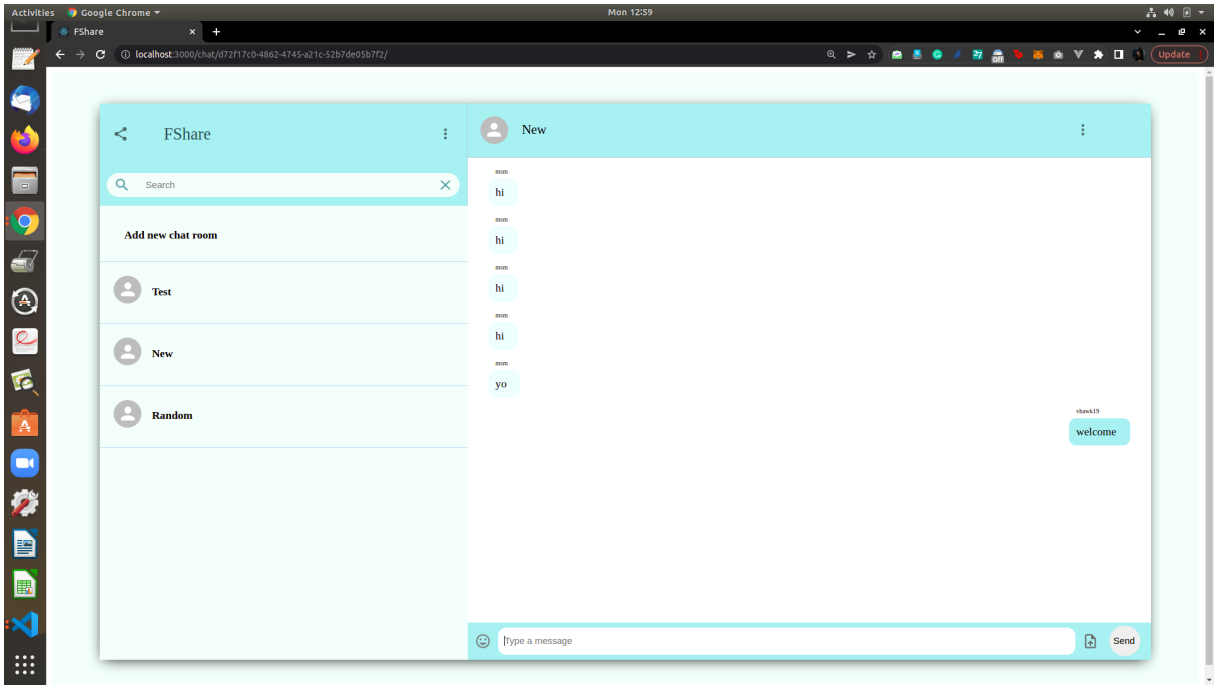

Figure 8.4: Chat Screen

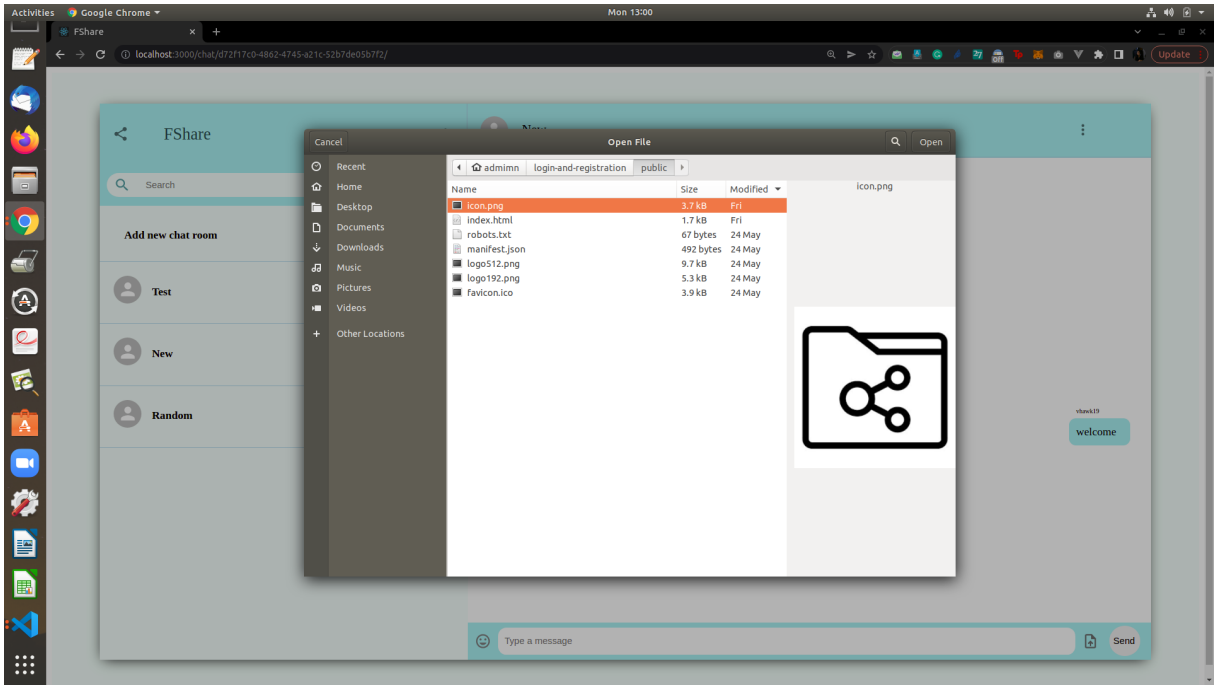

Figure 8.5: Uploading a file by clicking on the upload icon

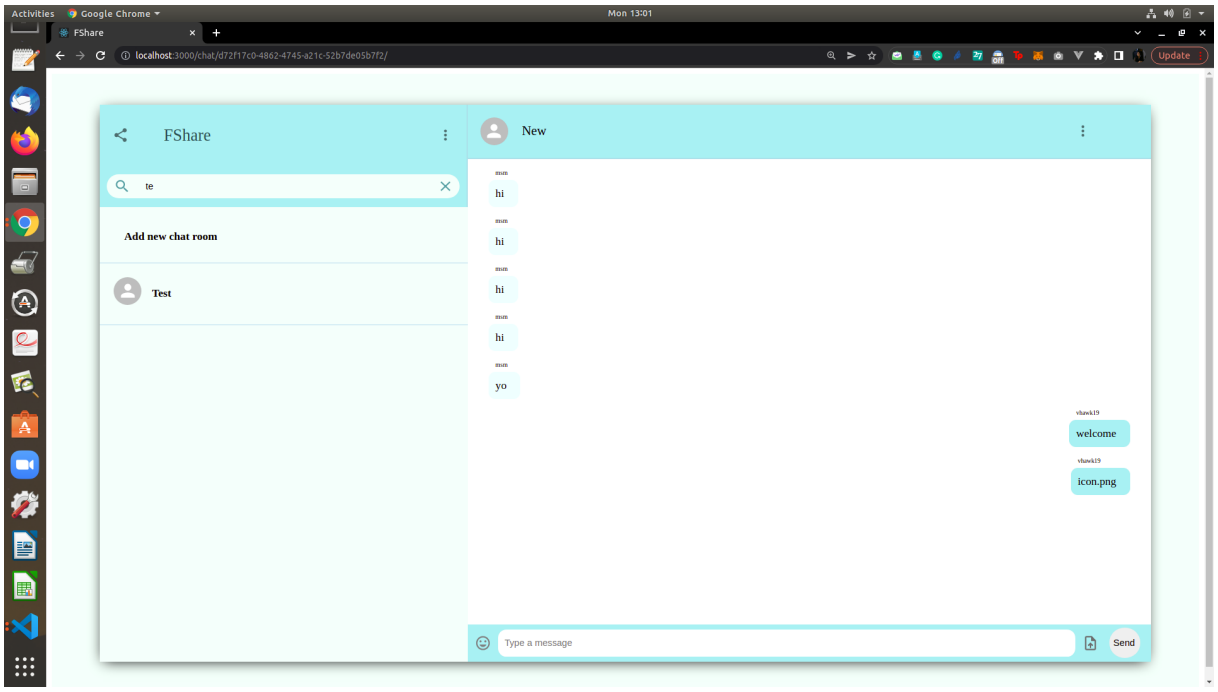

Figure 8.6: Search bar for chat rooms

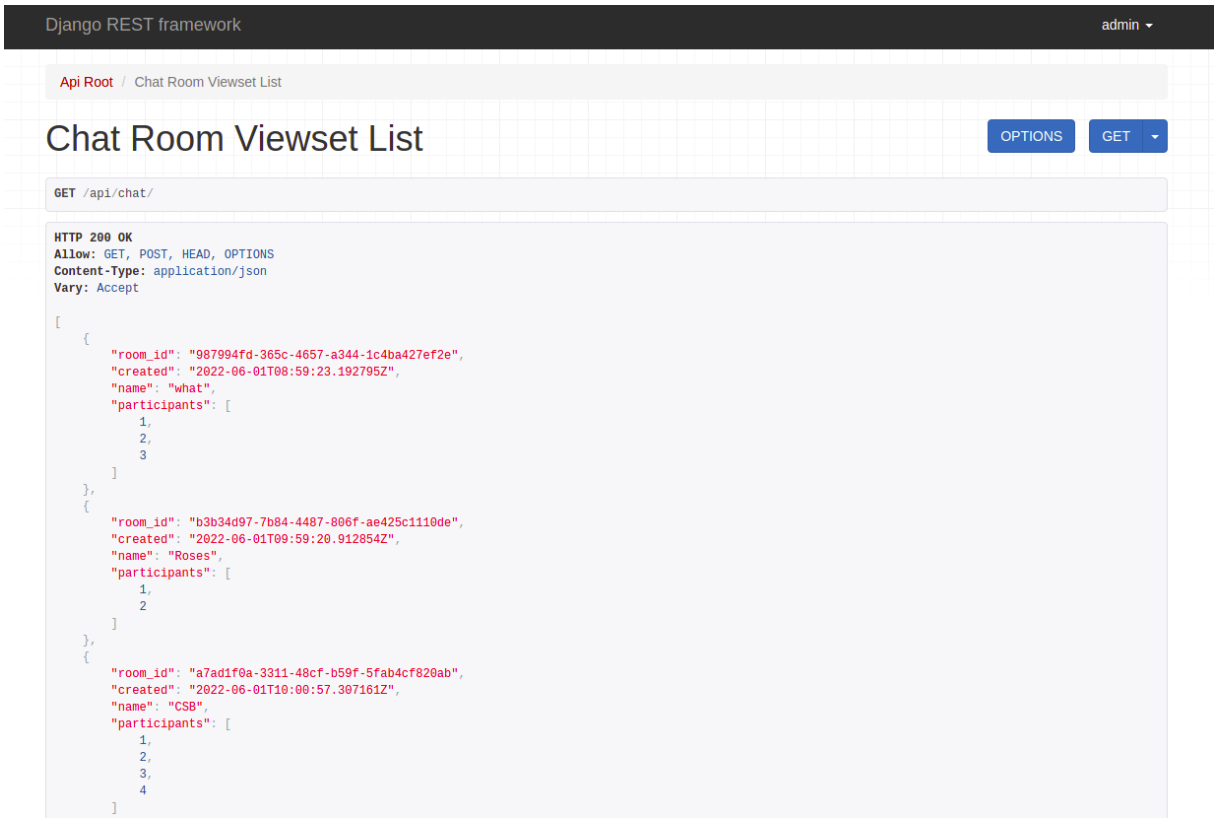

Figure 8.7: Django API for chat rooms

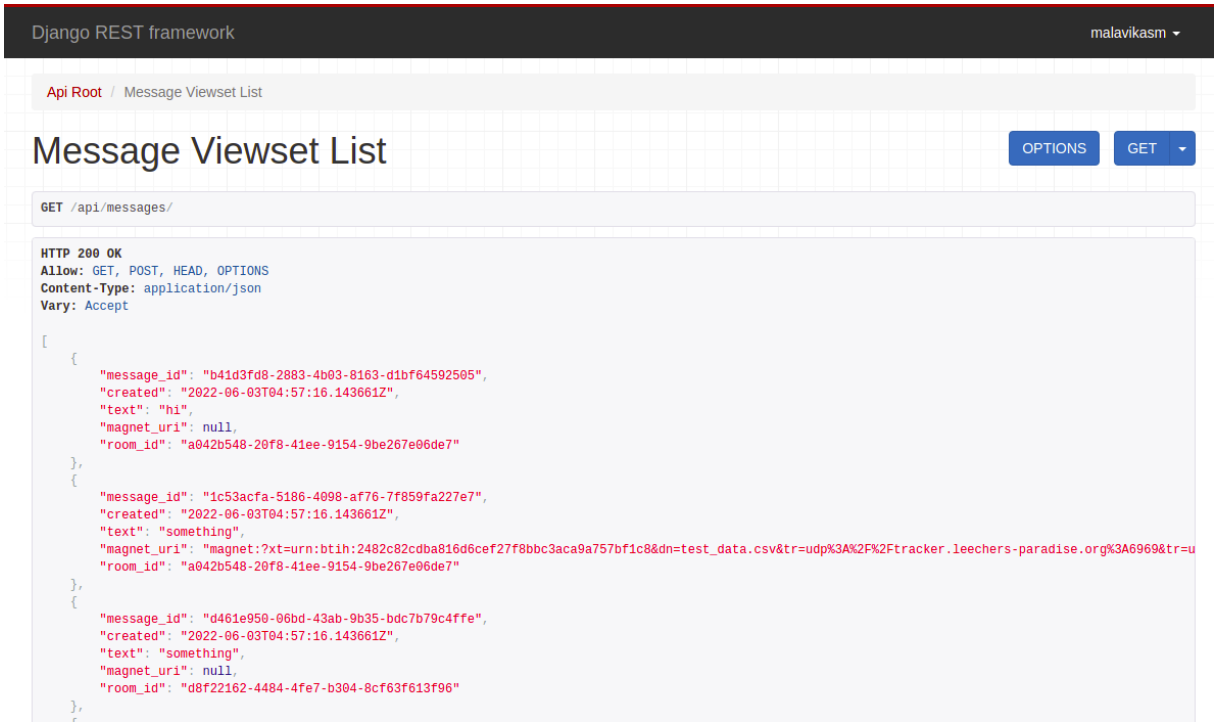

Figure 8.8: Django API for messages

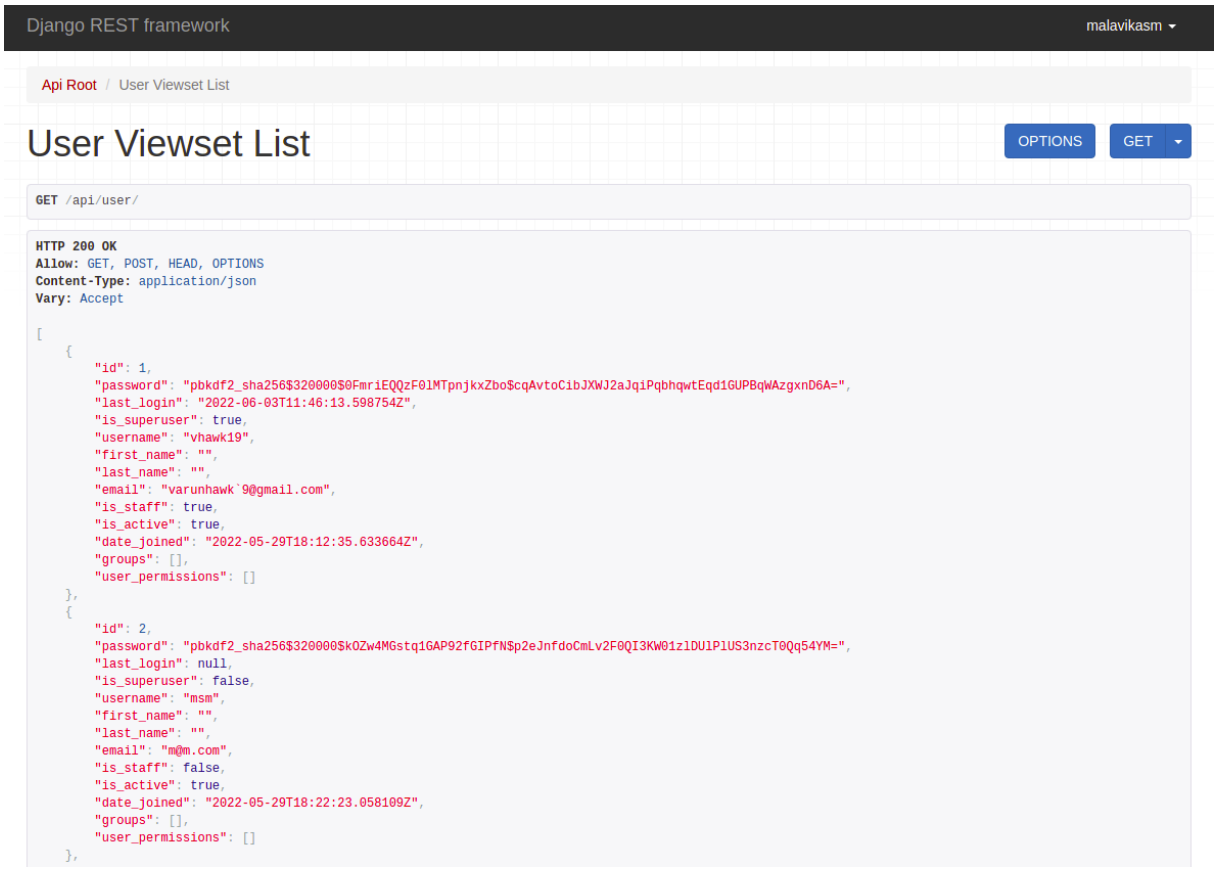

Figure 8.9: Django API for users

Figure 8.10: Tracker information - No connected peers

Figure 8.11: Tracker information - Active torrents and peers

Chapter 9

Conclusion

The project titled 'File Sharing Application using P2P Protocol' provides a platform for transferring files seamlessly, via the P2P protocol. The application is designed to be a web-based chat interface, through which users can transfer files to multiple other users through chat rooms. The chat-based interface allows laymen and users without sufficient knowledge of the P2P protocol or 'torrenting', to be able to transfer files, just as easily as they would with any popular messaging application. The users upload files through the chat application & the backend logic implemented through the WebTorrent library enables the torrenting of the corresponding file among multiple active users present in the respective chat room.

Chapter 10

Future Scope

Currently, the application utilises a centralised server for storing messages and metadata regarding the users' identity. Though the file transfer is entirely P2P and doesn't touch any centralised service, the identity of the user is stored in a server. This could be mitigated by writing the chat service entirely on the client side and exclusively using a DHT for discovering peers. However, this adds complexity to the chat layer, thereby making the entire application more complicated and less user-friendly. Another possible scope of improvement would be to use a hybrid approach based on the number of users in the chat room. For instance, if the number of people in a group/chat room is less, direct transfer(achieved via websockets/udp) may be preferred, else P2P could be used.

References

[1] M. Parameswaran, A. Susarla and A. B. Whinston, "P2P networking: an information sharing alternative," in Computer, vol. 34, no. 7, pp. 31-38, July 2001.

[2] W. S. Ng, B. C. Ooi, K. -. Tan and Aoying Zhou, "PeerDB: a P2P-based system for distributed data sharing," Proceedings 19th International Conference on Data Engineering (Cat.No.03CH37405), 2003.

[3] Pouwelse, J.A. & Garbacki, Pawel & Epema, D. & Sips, Henk. (2005). The Bittorrent P2P File-Sharing System: Measurements and Analysis. Peer-to-Peer Systems IV. 3640. 205-216.

[4] Balakrishnan, Hari & Kaashoek, Frans & Karger, David & Morris, Robert & Stoica, Ion. (2003). Looking up data in P2P systems. Communications of the ACM. 10.1145/606272.606299.

[5] Y. Yang, A. L. H. Chow and L. Golubchik, "Multi-Torrent: a Performance Study," 2008IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008.

[6] P. Gurvich, N. Koenigstein and Y. Shavitt, "Analyzing the DC File Sharing Network,"2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), 2010.

[7] Petar Maymounkov and David Mazières , "Kademila, a peer to peer Information system based on XOR Metric"