THENDER: A Multiplatform and Dual Mode Data Transfer System

Taiwo Adigun[1], Tolulope Fakoya[2], Oluwamayowa Odetunde[3], Chukwuemeka Duruimo[4]

[1,2,3,4]Department of Software Engineering, Babcock University, Ilishan-Remo, Nigeria.

Abstract

The need for an efficient and secure transmission of data between devices and networks is becoming increasingly vital. Messaging Apps and systems sometimes encounter data transfer limitations such as restrictions on data size and security risks associated with using an intermediary server. With the ever-growing risk of data breaches and cyber-attacks, it is essential to have a system that can transfer information safely and without limitations. Secure and efficient transfer of data can be improved by developing a multi-platform data transmission system without the need for an intermediary server. The system makes use of socket technology for secure and efficient data transfer. It utilizes React Native and Electron JS for mobile and desktop frontends respectively, while the backend is built with Django Rest Framework and the PostgreSQL database for secure data storage. The system employs web sockets technology for real-time communication across the internet and utilizes Render for deployment, providing automatic scaling, high availability, and SSL encryption. The development of the proposed system eliminates the need for cloud storage in data transfer, and also requires both users to be online for successful data transmission. Additionally, data is temporarily stored at regular intervals on the receiver's device to avoid data loss due to the volatile nature of networks. In conclusion, this system provides a reliable and secure solution for efficient data transfer, which is essential for today's data-driven world.

Keyword: *Data Transfer, Multi-platform Data Transmission, Web Socket Technology, Render, Postman*

1.Background

In computing, data refers to information that has been converted into a form that is effective for transmission or processing (Kowalczyk, 2011). The act or process of conveying something from one person, place, or item to another is known as transmission. Data transmission is the transfer of information stored in electronic form from one host to another (Stroud, 1990). Samuel Morse's 1837 demonstration of a telegraph system is when data transfer first began (History.com Editors, 2022). The Great Western Railway endorsed telegraph service by 1843, allowing the service to spread across the country (Nonnenmacher, 2001). Improving the telegraph, according to the History of Computing organization, Alexander Graham Bell introduced the telephone in 1876 (Coe, 1995). Though standard telephone lines did not carry data traffic until nearly a hundred years later, the development of early telecommunications coupled with an 1895 invention by Guglielmo Marconi, the radio laid the groundwork for numerous subsequent developments in communication technology.

In 1947, Bell Labs introduced the transistor, a device that found integration in myriad electronic products. The US government expanded on these technologies in 1958 with its launch of a communications-oriented satellite, and the first facsimile transmission over standard telephone lines occurred four years later. After the first fax transmission in 1962, data modulation into sound for transmission across telephone lines spread in popularity for several years. The 1969 Internet Protocol (IP) development marked a significant milestone in data transmission history.

Sockets have a long history. Their use began with ARPANET in 1971 and eventually evolved into Berkeley sockets, an Application Programming Interface (API) included in the Berkeley Software Distribution (BSD) operating system launched in 1983. (IBM, 2021) Network programming became famous as the Internet and the World Wide Web developed in the 1990s. Web servers and browsers were using sockets to take advantage of freshly linked networks. Applications using client-server technology of all shapes and sizes have become commonplace. The low-level API has kept the same even if the underlying protocols utilized by the socket API have changed throughout time and new ones have emerged. (RealPython, 2022).

Currently, significant amounts of data are transferred between clients in a local area network (LAN) (Wantao et. al, 2010). However, an intermediary server is introduced when significant amounts of data must be transferred between clients over the internet (clients that cannot be on the same network). This is because clients cannot easily access themselves over the internet so they would send the data to a publicly accessible server. This method limits the amount of data a client can transfer to another via the internet; the system's goal is to enable client data transfers over the internet without being constrained by the quantity of data that can be transferred.

2.Related Works

Although many more data transfer systems exist, the descriptions below represent the characteristics of distinct data transfer systems. Our developed system allows seamless communication amongst devices, either offline or online, using the socket technology.

Napster

Napster is one of the first widely used file-sharing systems that became popular. It was founded in early 1999 as an online service to share audio files. During its prime time in 2001, approximately 1.6 million users were online simultaneously. It is estimated that over 2 billion audio files were downloaded up until that point. (Matthew Green, 2002)

Despite its success, Napster had severe problems with copyright issues and subsequent lawsuits. (Richard H. Stern, 2000) Based on a court decision in 2001, an injunction was passed ordering Napster to shut down. The service had a restart in 2003. It now uses a pay-per-song charging model to avoid additional copyright lawsuits (Stefan Saroiu, 2003). Classifies Napster as an unstructured, centralized P2P system. Unstructured systems are characterized by data placement in the network without any basis of knowledge on its topology, as in structured P2P systems. The main component in the system architecture is the cluster of dedicated central servers. These are responsible for bootstrapping as well as providing the lookup service (Sherman, 2000). The cluster maintains an index with all information on file locations. This includes a list of currently connected users and their files. Each time a peer starts Napster, it establishes a connection to the central server cluster (Carlsson, 2001). When looking for a file, the peer queries the index servers. The query is processed by checking each connected user on the availability of the file. A list of possible trade partners is returned. Then, the requesting peer can choose a user to download from. A direct Hypertext Transfer Protocol (HTTP) connection is established between peers, and the file can be downloaded. After the file exchange, the HTTP connection is closed.

Orion

(Klemm et al, 2003) offers a P2P file-sharing method specifically designed for the MANET, or called the Optimised Routing Independent Overlay Network (ORION). ORION consists of an algorithm for building and maintaining an application-layer overlay network that permits routing of all types of messages necessary to operate a P2P file sharing system, i.e., inquiries, responses, and file transmissions. In order to closely mimic the present topology of the underlying network, overlay connections are created on demand and maintained only as long as necessary. When compared to an off-the-shelf solution using a P2P system for the wireline Internet, TCP, and a cutting-edge MANET routing protocol, ORION combines application-layer query processing with the network layer process of route discovery, significantly reducing control overhead and improving search accuracy. In addition, compared to the off-the-shelf method, the overlay network offers low overhead file transfers and increases the likelihood of successful file transfers. Simulation analysis shows performance benefits compared to the off-the-shelf approach.

Hollyshare

Unlike conventional file-sharing systems, which allow anyone to make use of them without prior interaction with any of its users, HollyShare is a peer-to-peer decentralised application that was made for use by a group of individuals who are familiar with one another prior to their interaction on the system. Hollyshare was designed to be a catalog system rather than a query system, as are many of the more well-known peer-to-peer file-sharing applications and architectures. Because the catalog exists on each node in the system, searching for files is unnecessary. This is useful for a relatively small number of shared files when users are unsure what they want to download and would select something from an existing list and then search for something that is most likely not in their system. (Han, 2021)

4shared

4shared is a file storage and sharing service developed in Kyiv, Ukraine. It was founded in 2005. 4shared is one of the world's largest file-hosting sites, with 317 terabytes of file transfer daily and handling up to eleven million users simultaneously. The service allows users to upload files of up to 500 MB each for free and 5GB GB for paid subscribers. After activating the account, the account size gets 15GB of free space. The user is then supplied with a unique URL, which locates the file and enables anyone who knows it to download it. Uploaded files will be stored as long as the user logs in to his/her account at least once every 30 days or at least one of the files is downloaded every 30 days. While no end-to-end encryption is available, 4Shared supports FTP and Secure FTP connections to securely transfer many files. (Vangie Beal, 2021)

Xender

Xender is a file transfer and sharing application that shares files, photos, music, videos, contacts, and applic ations (Khandelwa, et. al, 2016). It allows users to transfer files of various sorts. It ranges across Android and iOS-based mobile devices without cables, Wi-Fi, or cellular internet connections and with no mobile data (Sharma et. al, 2019). The app allows for group sharing between up to 4 users, and there is no need to install any clients on your PC to share content between your phone and computer.

3.Proposed Method

This study design and implement a data transmission system that enables data transmission between individuals on the same network as well as across the internet without the need for storing the data on either server or remote storage. It is a cross-platform data transmission system that enables data to be transferred between clients on a LAN and over the internet without keeping the information on any servers. The scope includes:

i. Data transfer is done between two users only.
ii. Only users with an account on the system will be able to perform online data transfers.
iii. Online transmission requires both parties to be online in order to be able to transfer data with each other across the internet.
iv. Offline transmission necessitates that the receiver is on the same network as the transmitter.

Several data transmission techniques that utilized both local and online methodology were revised in order to build a methodology that allows data to be streamed between users without requiring a central server for file storage.

3.1. System Architecture

The proposed system architecture as describe by Figure 1 depicts different components of the system and the flow of information within the system. Socket technology and a Model-View-Controller (MVC) architectural pattern were used for the implementation of this architecture. MVC is a pattern widely used in software development to separate the presentation logic from the application logic.

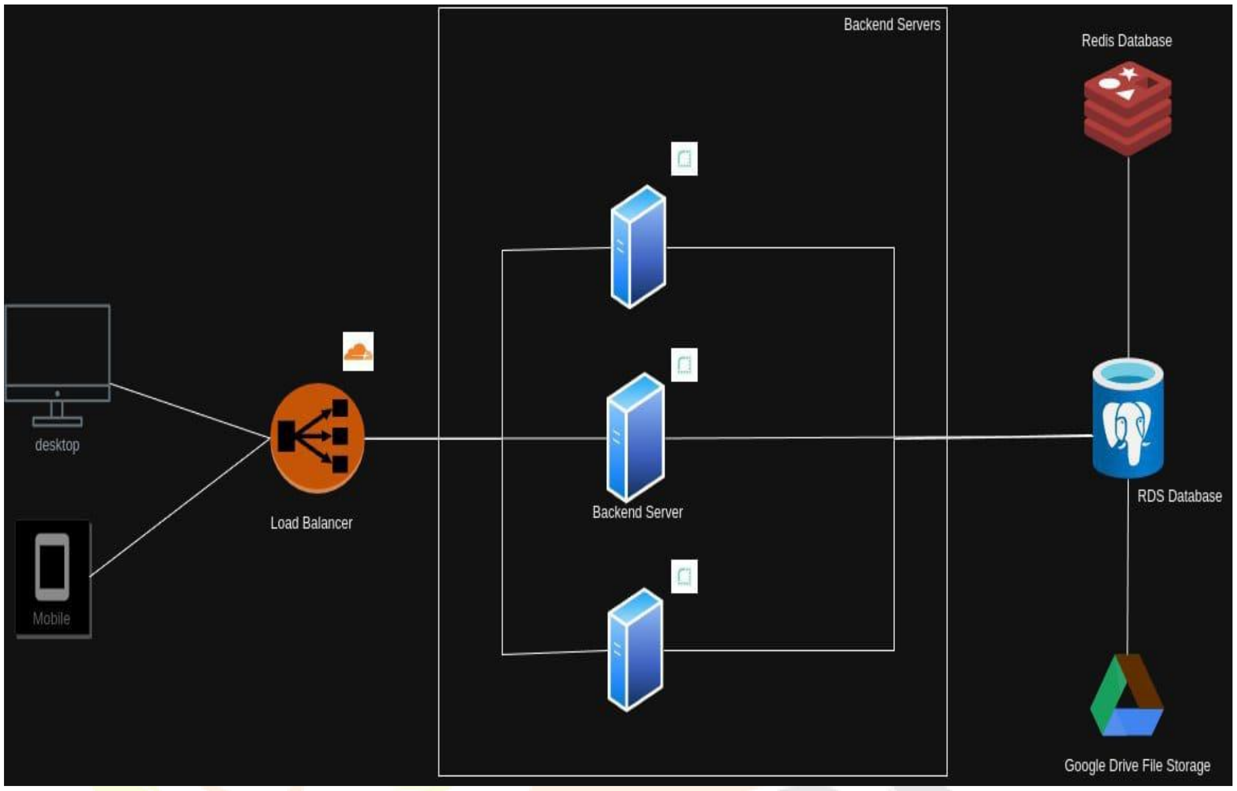


Figure 1: Architecture of the Proposed System

The model is responsible for managing the transfer process and storing data related to transmissions, such as the file being transferred, the sender and receiver information, and the transfer status. The proposed system makes use of two databases- PostgreSQL and Redis. PostgreSQL is utilized in the system to store metadata related to transmissions such as user accounts, files being transferred and data logs. Redis is an in-memory key-value store used for caching data related to the transfer process such as the data chunks being transmitted. Redis helps to effectively reduce the load on the database by caching frequently accessed data. The view represents the user interface of the application and is implemented in the proposed system using React Native and ElectronJS front-end components. The view would allow the users to interact with the system and monitor the progress of the transfers. The controller is responsible for handling user input and managing the communication between the model and the view. The controller is implemented using Django REST APIs.

3.2. System Design

This data transmission system is designed to have two categories of users: guests who can only share data locally and users who can transfer data online, which is depicted in Figure 2 below. Also, because the system does not require intermediary server few information components that are kept are described using Figure 3.

Moreover, the system is designed to capture the following functional non-functional requirements.

The functional requirements include the following:

i. The system shall allow users to register
ii. The system shall allow users to change their password
iii. The system shall allow users to change details

iv. The system shall allow users to choose an online or local transfer mode.
v. The system shall be able to send files.
vi. The system shall be able to receive files.
vii. The system shall be able to access files from the file system.

The non-functional requirements are as follows:

i. The system shall store data in a temp file of 100 MB
ii. The System shall be available 24/7
iii. The System shall encrypt the data being transferred
iv. The system shall have a minimum of 3 re-tries during transmission

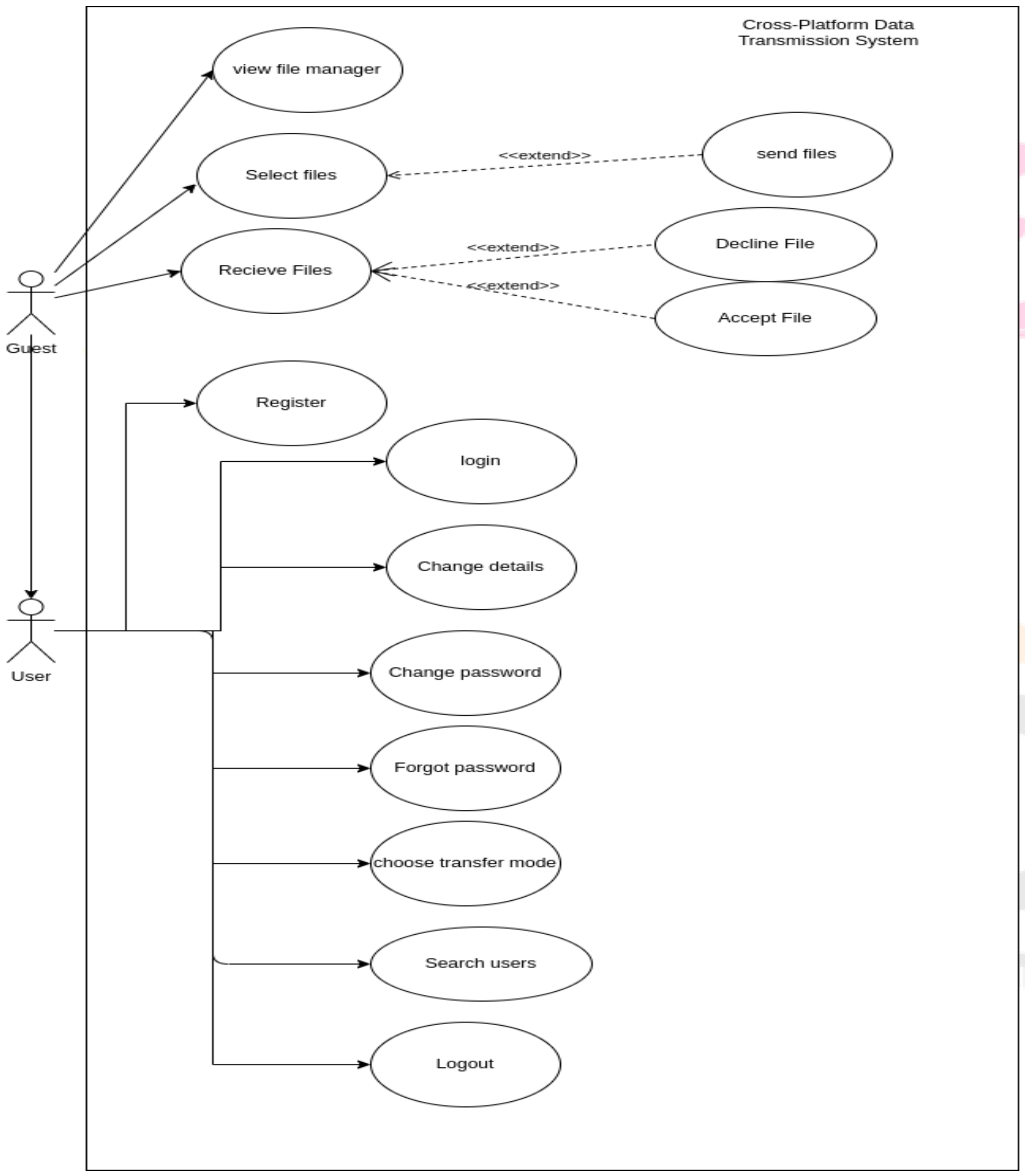


Figure 2: Use Case Diagram

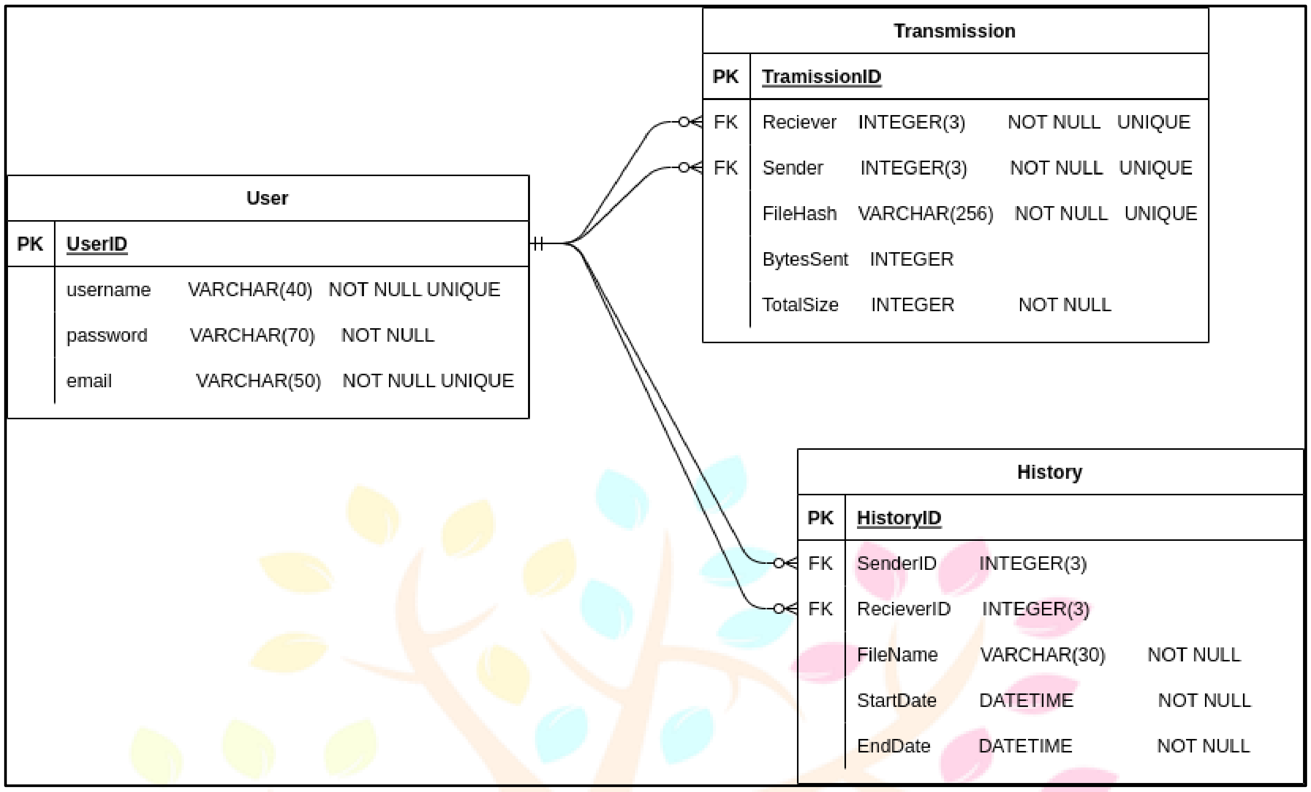

Transmission
PK TramissionID
FK Reciever INTEGER(3) NOT NULL UNIQUE
FK Sender INTEGER(3) NOT NULL UNIQUE
FileHash VARCHAR(256) NOT NULL UNIQUE
BytesSent INTEGER
TotalSize INTEGER NOT NULL
User
PK UserID
username VARCHAR(40) NOT NULL UNIQUE
password VARCHAR(70) NOT NULL
email VARCHAR(50) NOT NULL UNIQUE
History
PK HistoryID
FK SenderID INTEGER(3)
FK RecieverID INTEGER(3)
FileName VARCHAR(30) NOT NULL
StartDate DATETIME NOT NULL
EndDate DATETIME NOT NULL

Figure 3: Entity Relationship Diagram

3.3. Development Setup

The platform for the implementation of THENDEER is Visual Studio Code using Python programming language. The different tools for different specific tasks include Django, React Native, Electron Js, Postgresql and Redis. During the implementation of the system, Git helped keep track of all the changes made to the codebase and revert to previous versions if necessary, and collaboration amongst developers on the same codebase.

3.4. Deployment

The process of deploying the proposed system into a production environment is ensuring that the system runs smoothly in the production environment. Docker engine, a containerization platform is used to deliver the product. This allows THENDER and its dependencies to be packaged into portable container that can easily be deployed and run on any platform. Docker provides portability, scalability and consistency across different environments hence reducing the risk of compatibility issues. Render, a cloud hosting platform is used for the deployment of the system by creating a service and making necessary configurations by specifying the environment variables, such as the database connection and authentication secrets. The performance settings of the service were also configured to aid system reliability. After the application code and all its assets are uploaded, render automatically builds and deploys the application, and provides a URL where the application can be accessed.

3.5. Testing

This is a critical part of ensuring the reliability, functionality, and security of the transfer system. This is done using Postman, a popular API testing tool used to test and debug RESTful APIs. It provides an easy-to-use interface for making HTTP requests, viewing responses, and testing various scenarios. A Postman collection was created for THENDER with the necessary environment variables and headers set for testing the system APIs. As described in Figure 4, steps carried out while testing with postman includes making requests to the REST endpoints, inspecting the response data, and verifying the results. Different testing scenarios were also performed using Postman, such as testing for invalid input data and error handling. Postman helps ensure the reliability, scalability, and performance of the APIs, and provides a robust testing framework for the system.

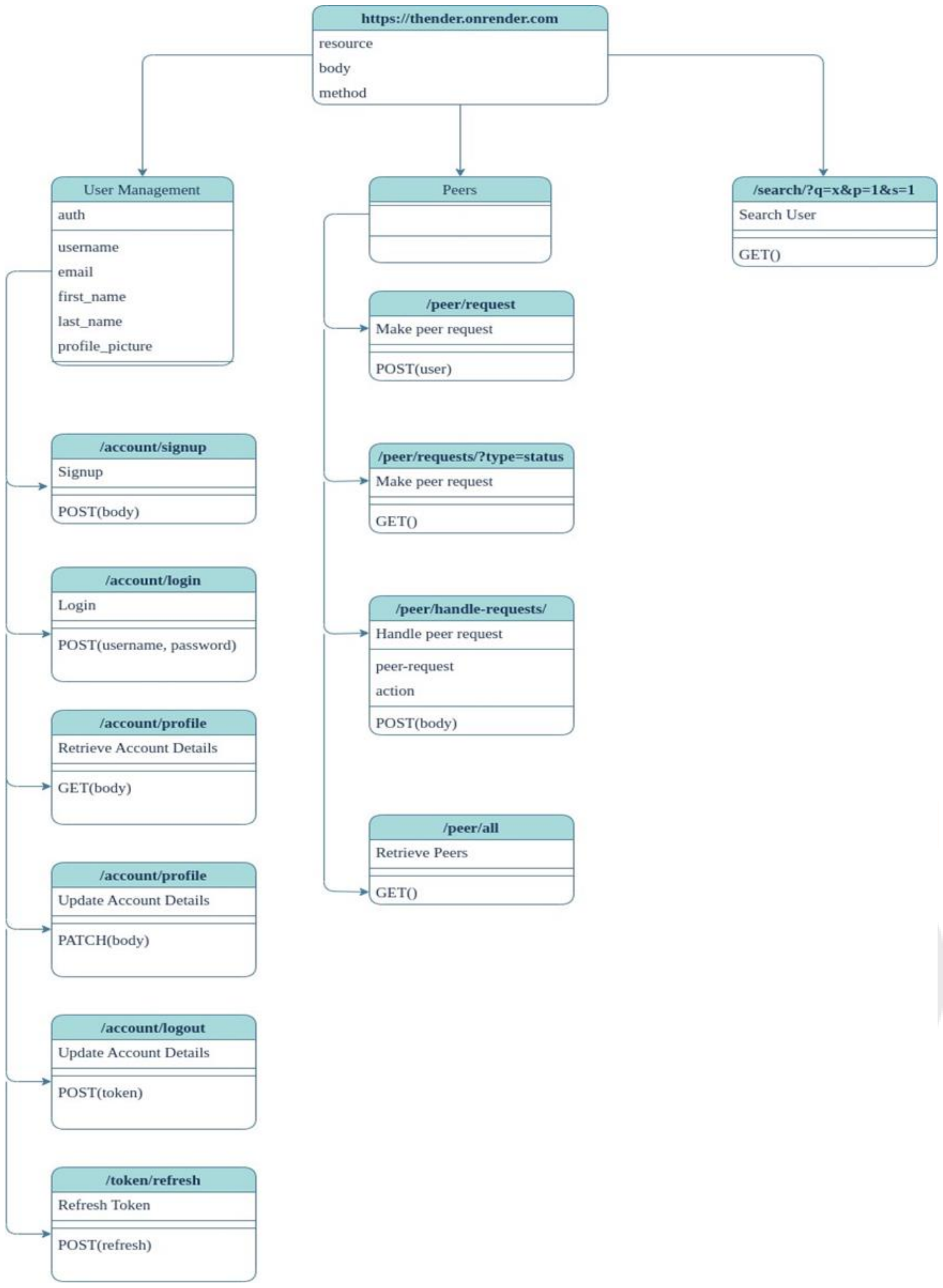


Figure 4: API Endpoints

4.Results and Discussion

The various user interfaces for both mobile and desktop devices with effective user experiences relating to the system requirements are described below.

4.1 Splash Screen

The user is immediately welcomed with the branding of the application as the system is launched. The purpose of the splash screen is to create a positive user experience by providing an aesthetically pleasing and informative welcome screen. It loads quickly and does not delay the user from accessing the application. It is very much consistent with the rest of the application's design and functionality.

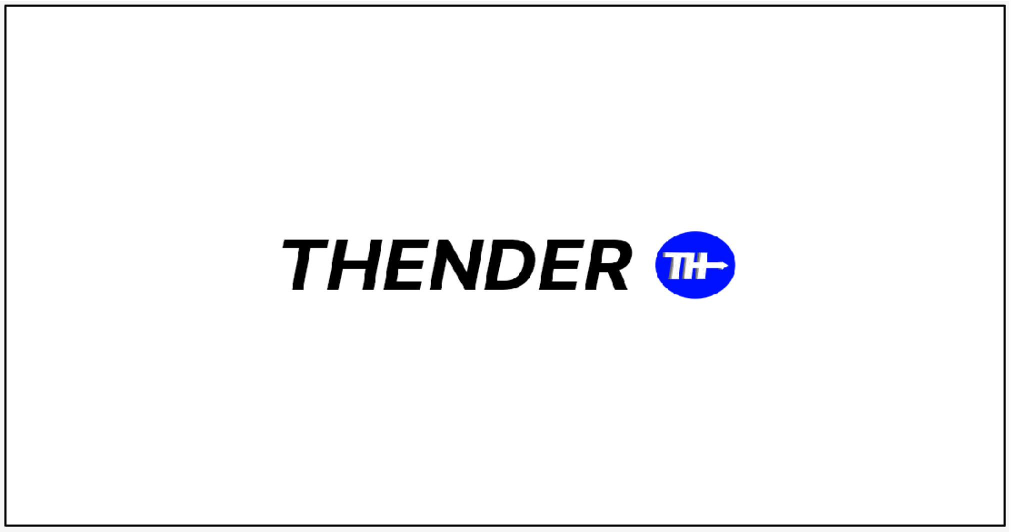


Figure 5: Desktop view of Splash Screen

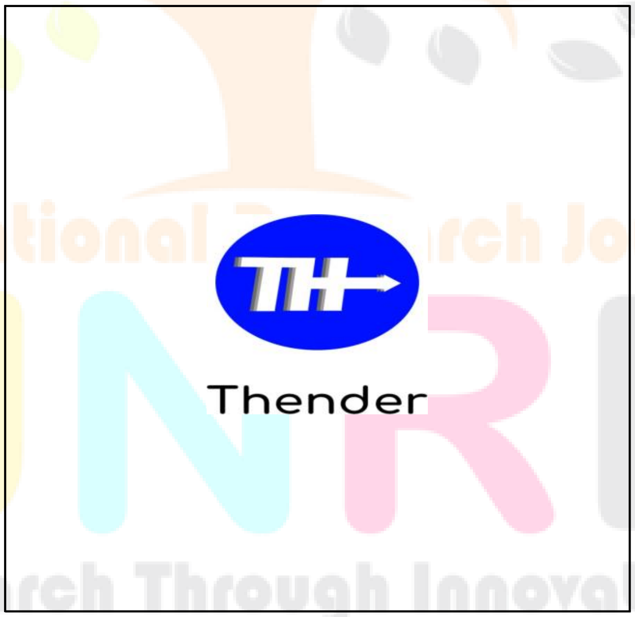


Figure 6: Mobile view of Splash Screen

4.2 Sign Up Page

The register page of the software application is designed to make it easy for new users to create an account and start using the system. The page features a clean and simple layout, with clear instructions and input fields for users to enter their information.

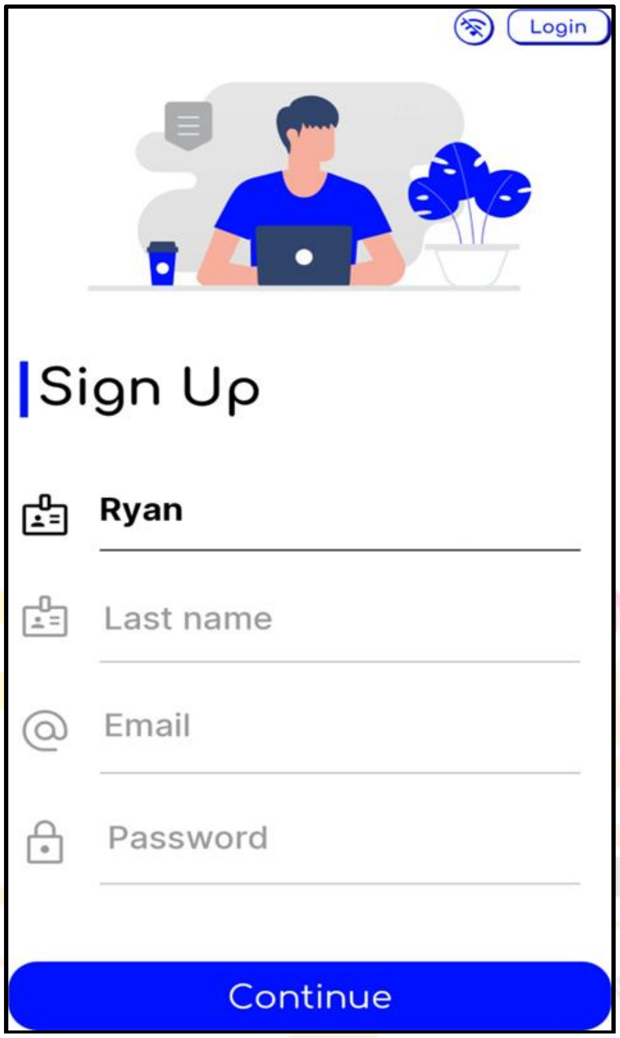


Figure 7: Mobile view of Sign Up Page

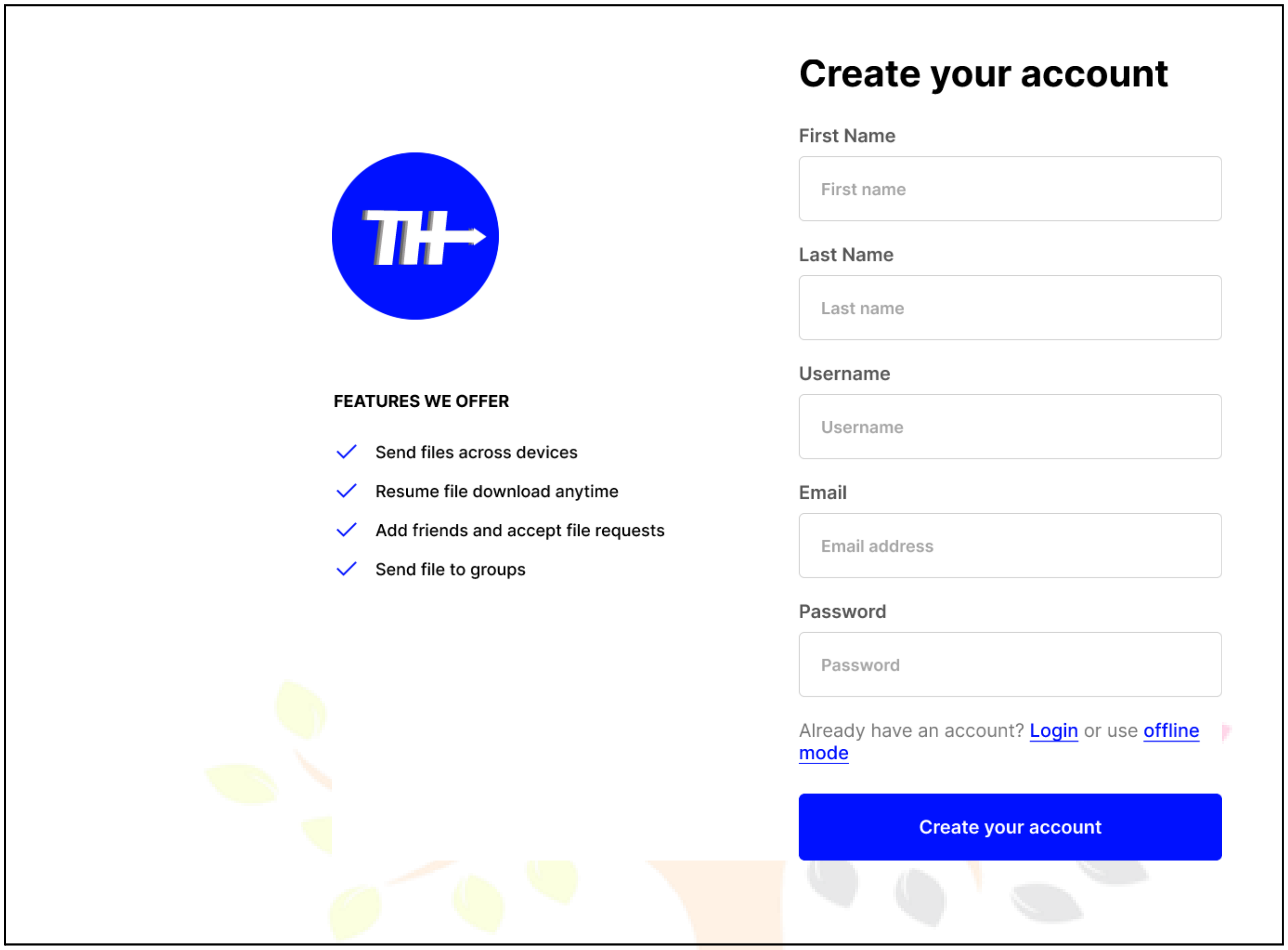


Figure 8: Desktop view of Sign Up Page

4.3 Login Page

The login page of the software application is designed to be intuitive and user-friendly, with a simple layout that makes it easy for users to access their accounts. The page features a clean, modern design with a colour scheme that is consistent within the application. There is a login form where users can enter their email and password to access their account. The wifi icon at the top right indicates the online status of the user.

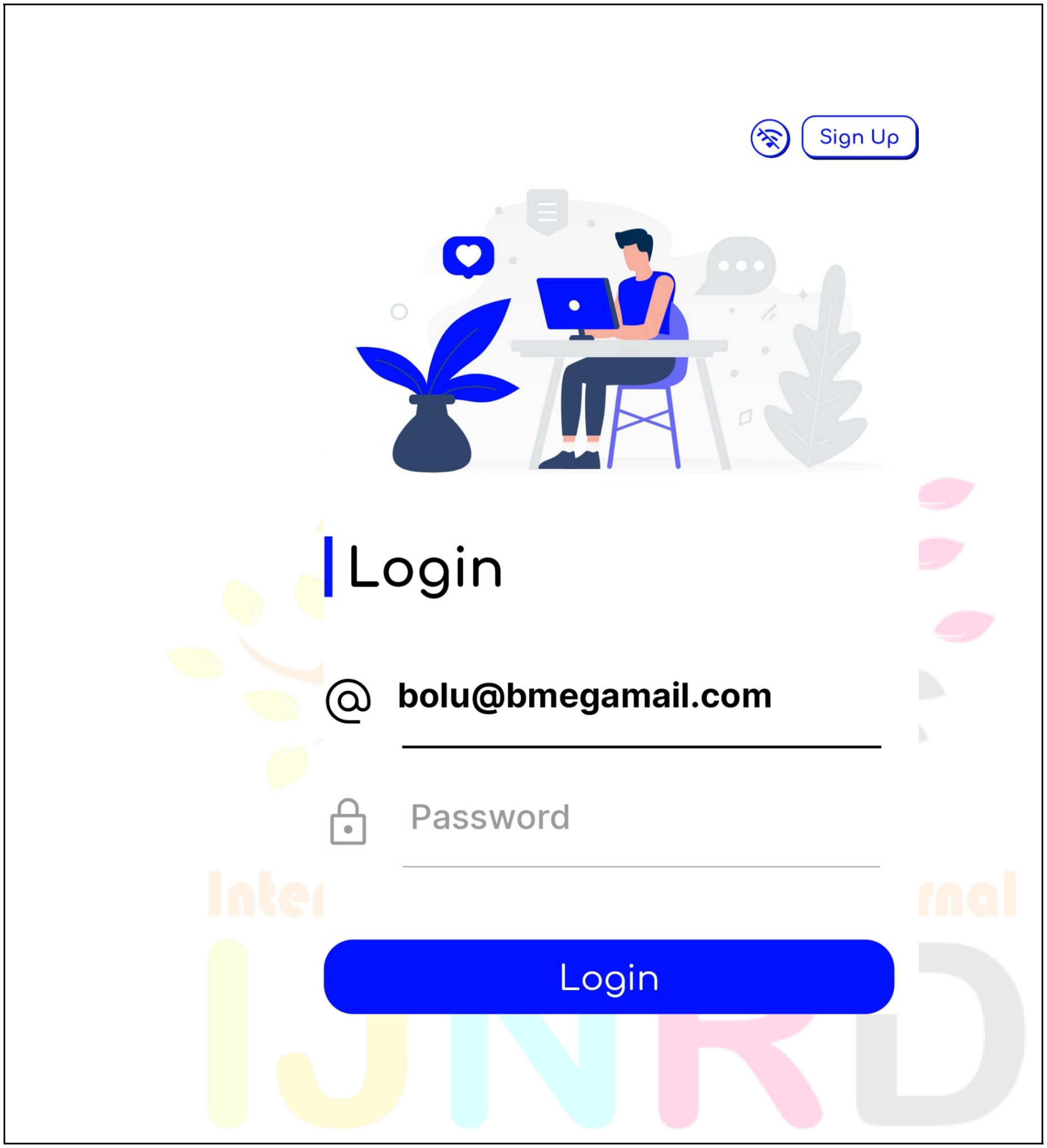


Figure 9: Mobile view of Login Page

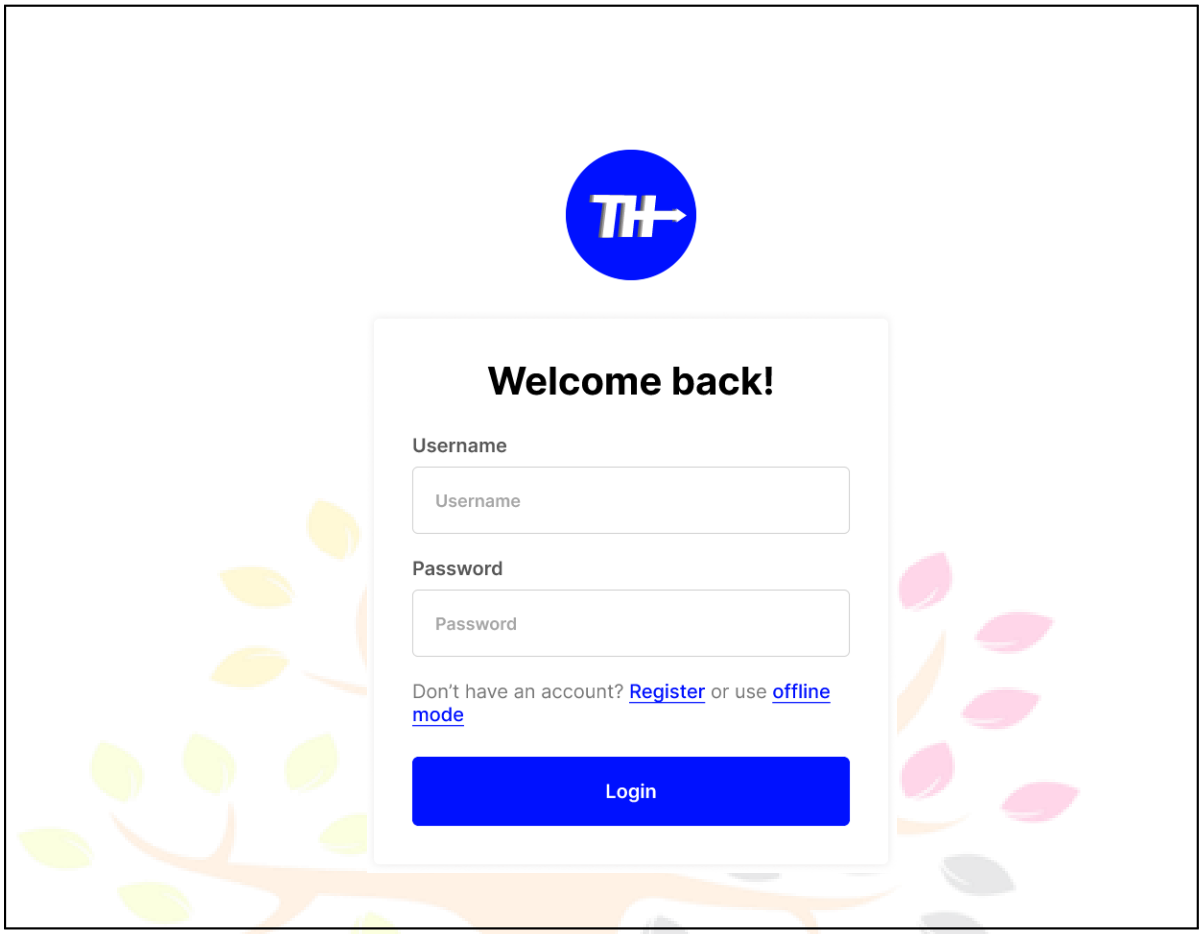


Figure 10: Desktop view of Login Page

4.4 Peers Page

This page provides an interface for a logged in user to search for peers on their network to create a connection with. Peers are other devices on the same network as the user to which 'send' requests can be made. The user is also able to search for peers and an error page is displayed if no peers are currently on the user's network.

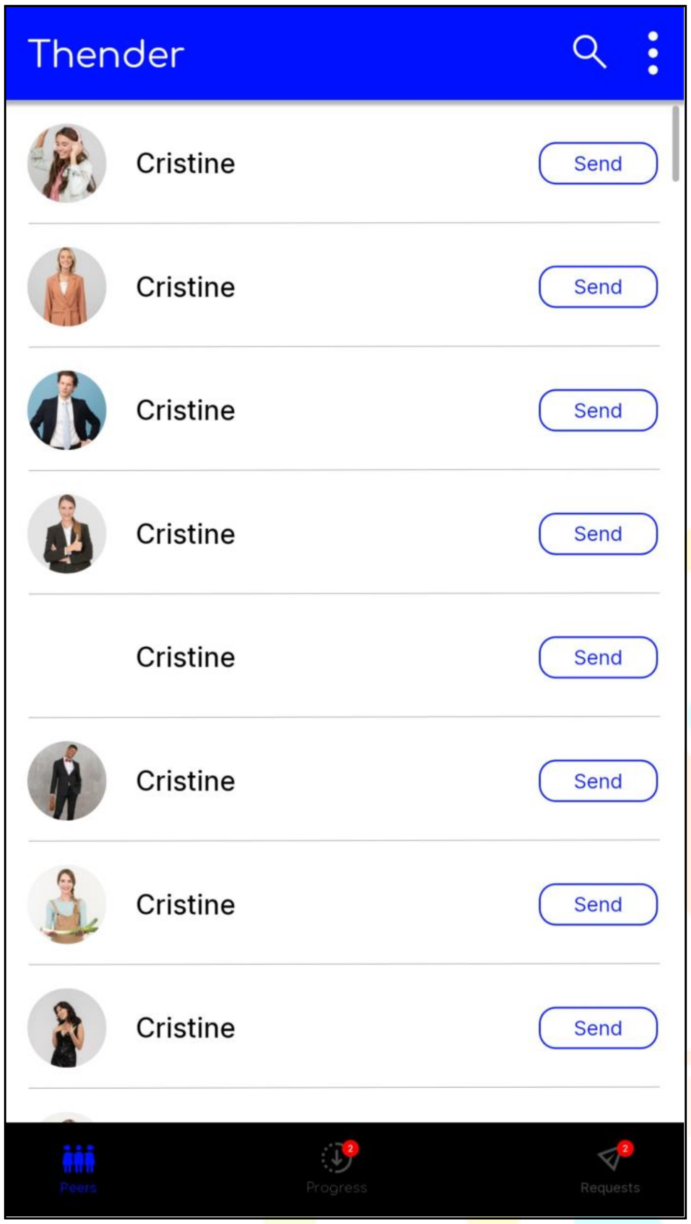
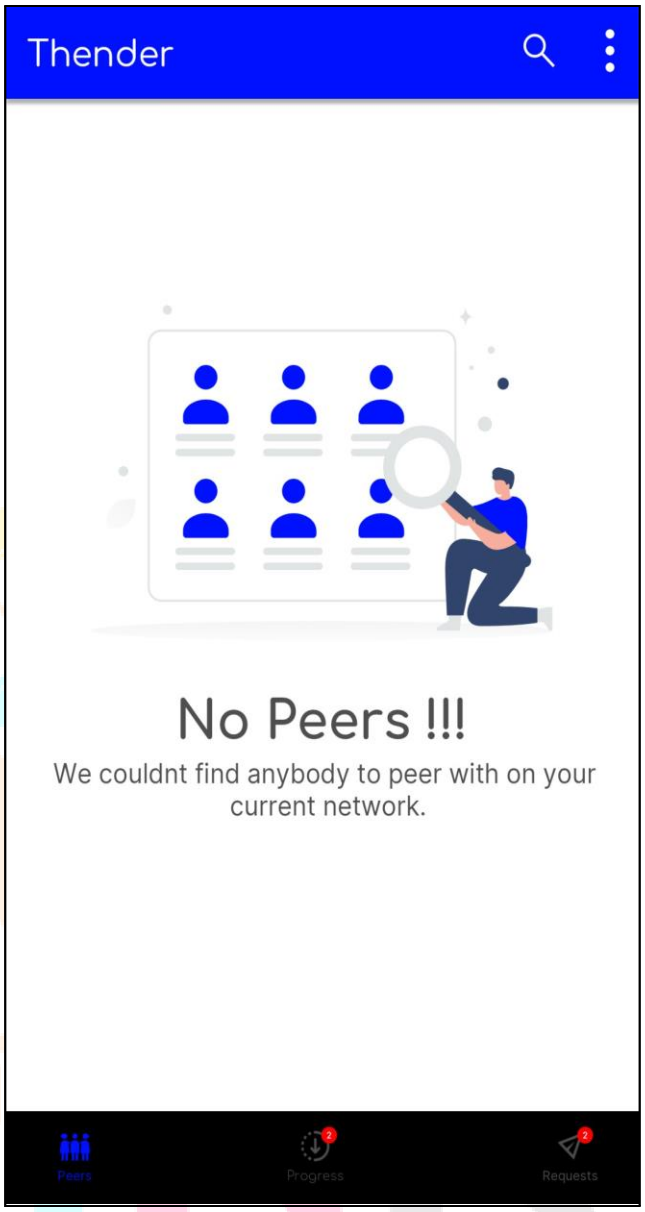


Figure 11: Mobile view displaying peers



No Peers !!!

We couldnt find anybody to peer with on your current network.

Figure 12: Indicating peer search status

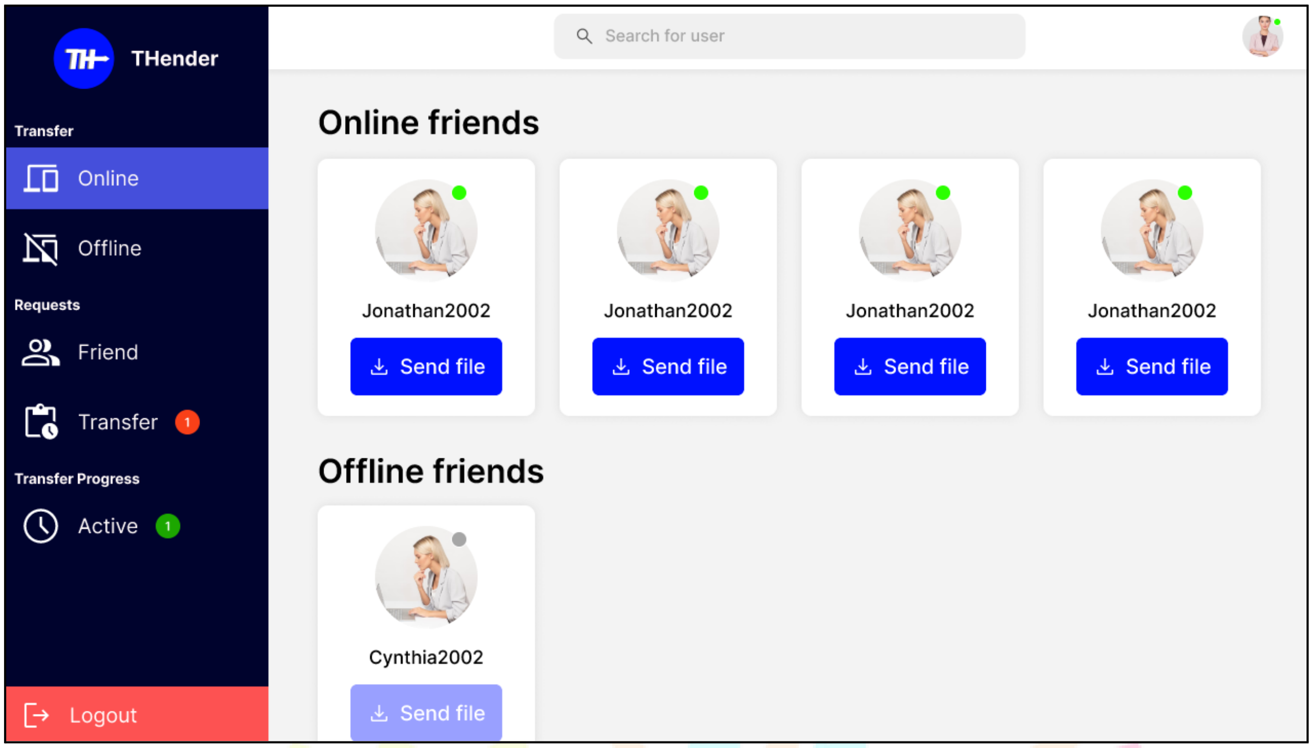
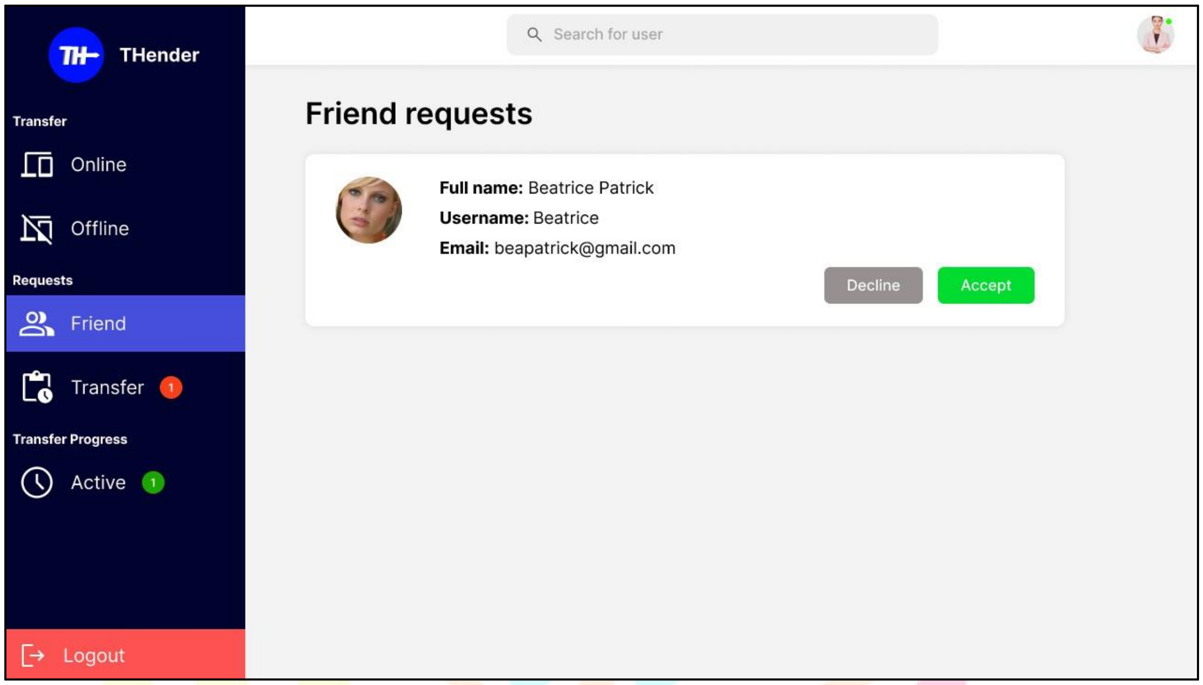


Figure 13: Desktop view displaying peers



Friend requests

Full name: Beatrice Patrick
Username: Beatrice
Email: beapatrick@gmail.com

Figure 14: Desktop view displaying peer requests

4.5 Transmission Section

A user is able to send and receive transmission requests to and from peers. The user is required to select the recipient from the "peers" section and is also able to view transmission requests received. A user has the option to accept or reject a transmission request. If accepted, the progress of the data transmission can be monitored.

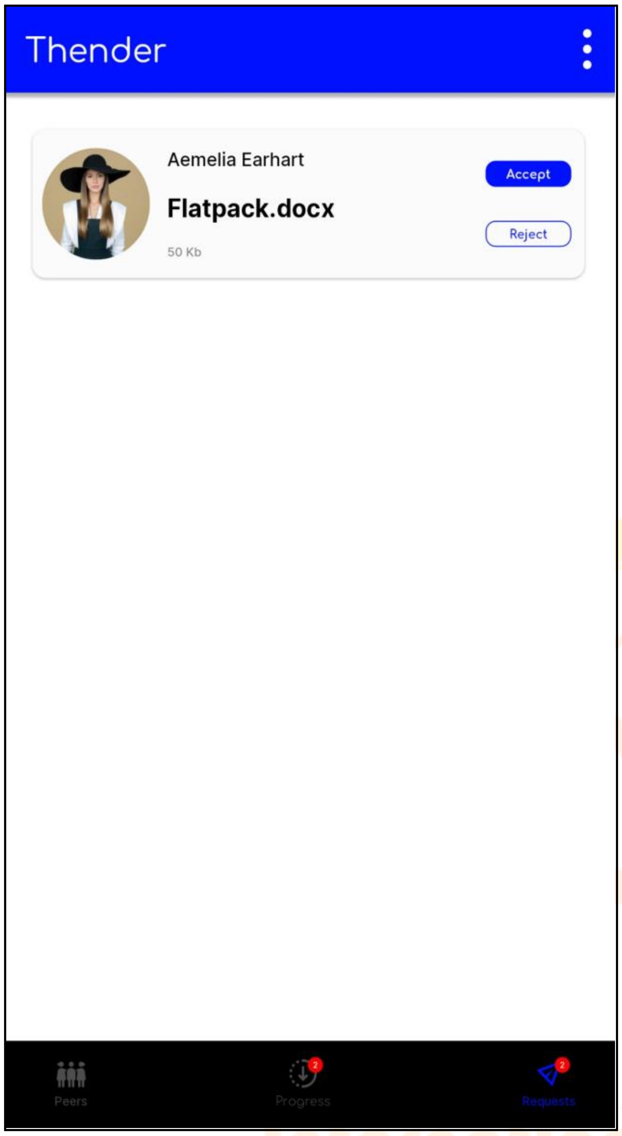


Figure 15: Mobile View Transmission Request

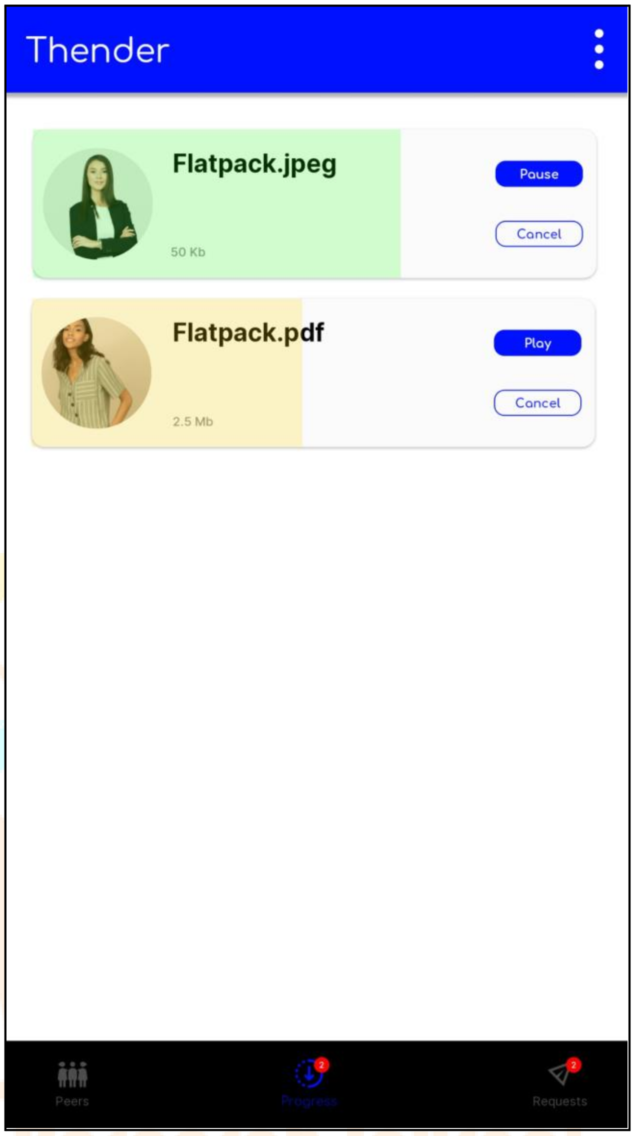


Figure 16: Mobile View Transmission Progress

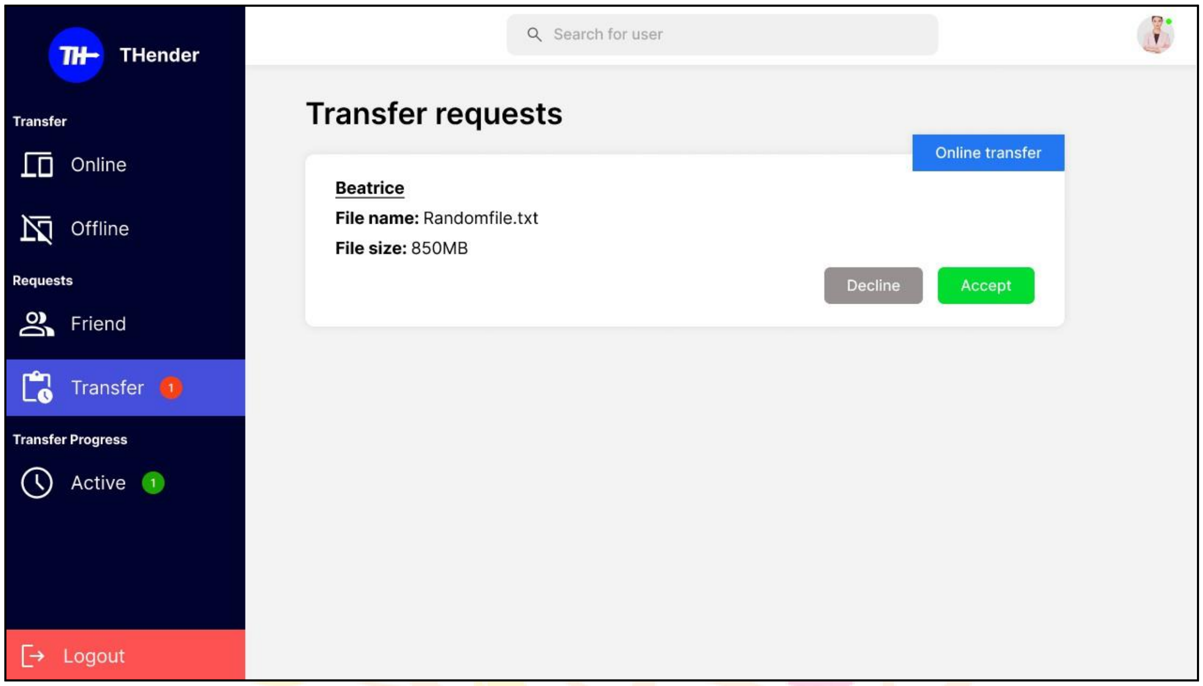


Figure 17: Desktop View- Transmission Request

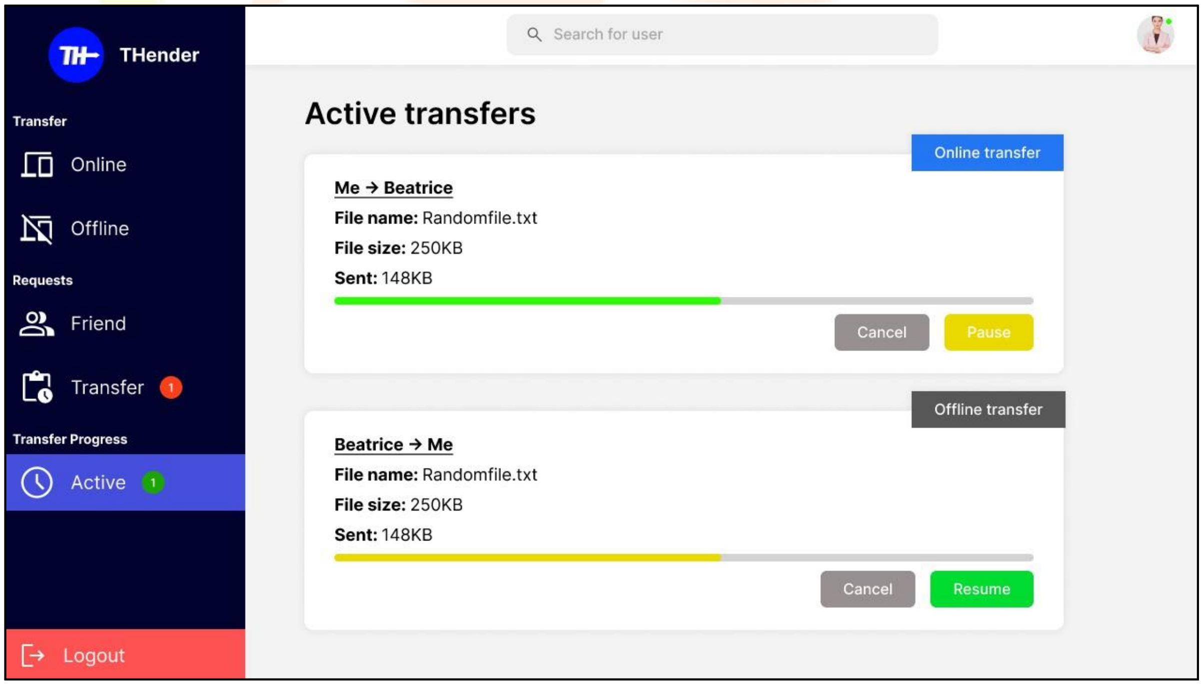


Figure 18: Desktop View- Transmission Progress

4.6 Offline Connection Page

This page offers an interface for offline users to connect with their peers and transfer files without the need for an internet connection. The interface is designed to be user-friendly and straightforward, presenting a simple form that prompts the user to enter the IP address of the intended recipient.

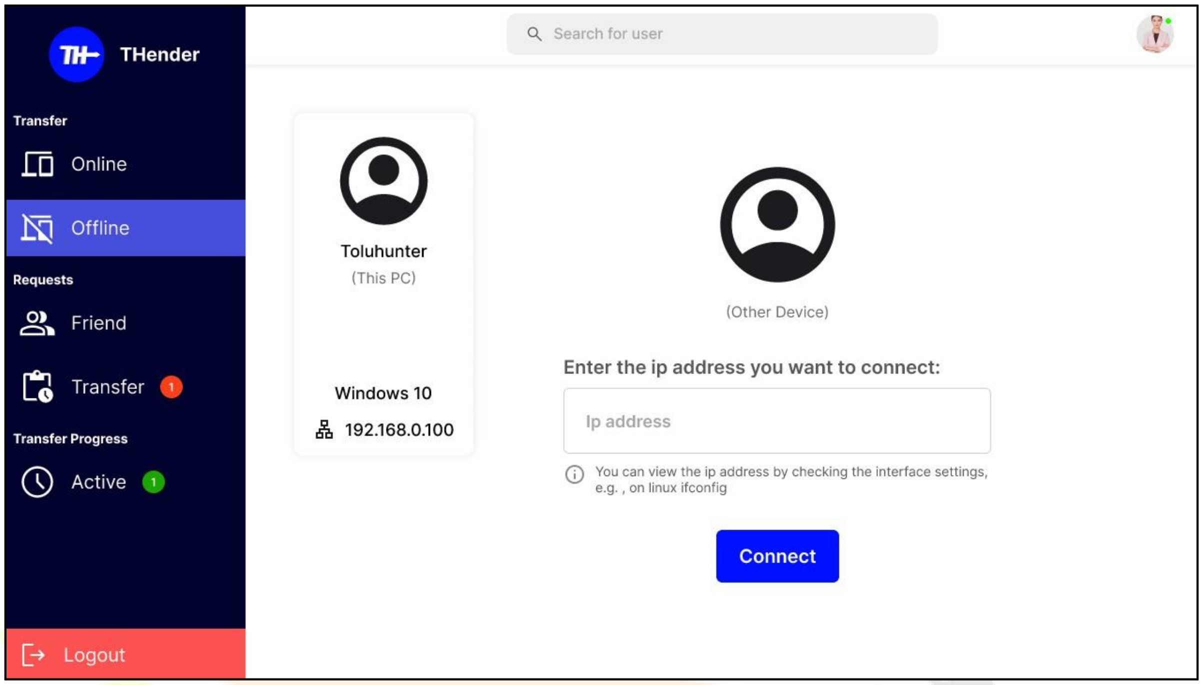


Figure 19: Desktop View- Offline Transmission

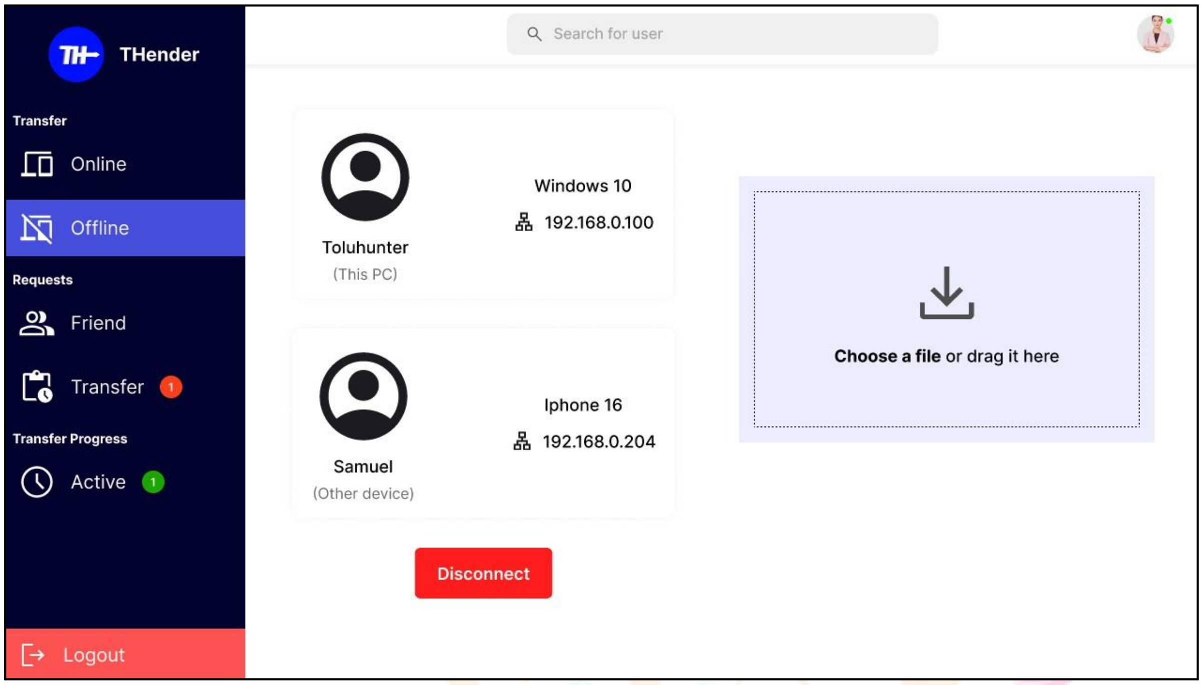


Figure 20: Desktop View- Offline Transmission Connection

4.7 Update User Profile

This page offers an interface for authenticated users to view and make changes to their user profile- firstname, last name, profile picture, email address and password.

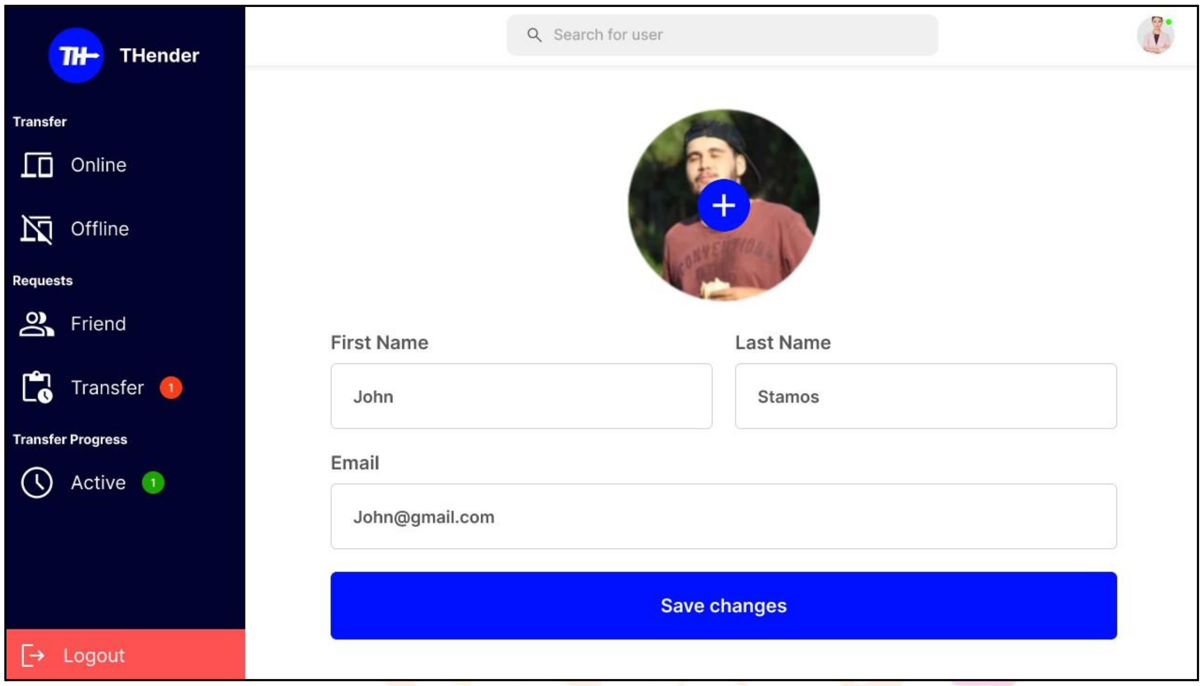


Figure 21: Desktop View- Update user Details

5.Conclusion

The increasing demand for efficient and secure transfer of information between devices, networks, and users leads to the need for a data transmission system. This study focuses on the design and implementation of a multi-platform data transfer system that places a lot of emphasis on reducing security risks related to the use of an intermediary server during transmission, and also eliminating the limitations of other systems that impose limits on the sizes of data being transferred. To combat the limitations of size and security associated with the use of intermediary servers to store data during transmission, this study was carried out to develop a multiplatform transfer system, available for both mobile and desktop devices. The system has users, referred to as 'peers', who are able to connect over the internet or on a network and make transmission requests and establish a connection to send and receive data. The application also provides functionality where users can pause and resume transmissions seamlessly. The application provides a straightforward and user-friendly user interface for simplicity and optimal performance of its features.

REFERENCES

Alhowaidi, M., Nadig, D., Hu, B., Ramamurthy, B., & Bockelman, B. (2021). Cache management for large data transfers and multipath forwarding strategies in named data networking. Computer Networks, 199, 108437.

Carlsson, B., & Gustavsson, R. (2001). The rise and fall of napster-an evolutionary approach. In Active Media Technology: 6th International Computer Science Conference, AMT 2001 Hong Kong, China, December 18–20, 2001 Proceedings 6 (pp. 347-354). Springer Berlin Heidelberg.

Coe, L. (1995). The telephone and its several inventors: A history. McFarland.

Coffield, D., & Shepherd, D. (1987). Tutorial guide to Unix sockets for network communications. Computer Communications, 10(1), 21–29. doi:10.1016/0140-3664(87)90311-2

Fette, I., & Melnikov, A. (2011). The websocket protocol (No. rfc6455).

Gupta, B., & Vani, M. P. (2018). An overview of web sockets: The future of real-time communication. Int. Res. J. Eng. Technol. IRJET, 5(12).

Han, S., Hore, B., Issenin, I., McCarthy, S., & Tauro, S. (2021). HollyShare: Peer-to-Peer File Sharing Application. https://www.ics.uci.edu/~bhore/papers/Hollyshare.pdf

History.com Editors. (2022, August 12). Morse Code & the Telegraph. Morse Code & Telegraph: Invention & Samuel Morse - HISTORY. Retrieved November 7, 2022, from https://www.history.com/topics/inventions/telegraph

IEEE Micro, Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and Analyz ing the Characteristics of Napster and Gnutella Hosts. Multimedia Syst., 9(2):170–184, August 2003.

Ingalls, R. (2014). Sockets tutorial. línea]. Available: http://www. cs. rpi. edu/~ moorthy/Courses/os98/Pgms/socket. Html. Solworth, J. A. (2004). Socket Programming.

Khandelwal, S., Sailor, I. R., Mistry, N., & Dahiya, M. S. (2016, December). An insight into file sharing artifacts using Xender application. In 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 386-389). IEEE.

Klemm, A., Lindemann, C., & Waldhorst, O. P. (2003, October). A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484) (Vol. 4, pp. 2758-2763). IEEE.

Kowalczyk, S., & Shankar, K. (2011). Data sharing in the sciences. Annual review of information science and technology, 45(1), 247-294.

Lee, J., Gunter, D., Tierney, B., Allcock, B., Bester, J., Bresnahan, J., & Tuecke, S. (2001).Applied techniques for high bandwidth data transfers across wide area networks.

Liu, Q., & Sun, X. (2012). Research of web real-time communication based on web socket. Matthew Green. Napster Opens Pandora's Box: Examining How File-Sharing Ser

Nilay, S. K. I. R. S., & Dahiya, R. M. D. M. S. An Insight into File Sharing Artifacts using Xender Application.

Nonnenmacher, Tomas. "History of the U.S. Telegraph Industry". EH.Net Encyclopedia, edited by Robert Whaples. August 14, 2001. URL http://eh.net/encyclopedia/history-of-the-u-s-telegraph-industry/

O'Grady, S. (2021, August 5). The RedMonk Programming Language Rankings: June 2021

–tecosystems. RedMonk. Retrieved November 7, 2022, from https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/

Rajendran, A., Mhashilkar, P., Kim, H., Dykstra, D., Garzoglio, G., & Raicu, I. (2013, May). Optimizing large data transfers over 100Gbps wide area networks. In 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (pp. 26-33). IEEE.

Reppy, J. H. (1996). A safe interface to sockets. Technical memorandum, AT&T Bell Laboratories.

Richard H. Stern. Napster: A Walking Copyright Infringement?

Rodrigues, S. H., Anderson, T. E., & Culler, D. E. (1997, January). High-performance local area communication with fast sockets. In USENIX 1997 Annual Technical Conference (pp. 257-274). Usenix.

Saroiu, S., Gummadi, K. P., & Gribble, S. D. (2003). Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia systems, 9(2), 170-184. Sharma, M., Singh, K., & Sharma, C. (2019, February). A Relative Study on Image Eminence while Sharing via Different Apps. In Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India

Sherman, C. (2000). NAPSTER. Online, 24(6), 16-23.

Stroud, J. (1990). CHAPTER SIX.1 Transmission of information. In D. Haynes (Ed.), Information Sources in Information Technology (pp. 96-121). Berlin, Boston: K. G. Saur. https://doi.org/10.1515/9783110976496.96vices Threaten the Enforcement of Copyright on the Internet. Ohio State Law Journal, 63(799), 2002.

Vobornik, P. (2015). Tool and mechanisms for efficient transfer of data in cloud client-server applications

Wang, V., Salim, F., Moskovits, P., Wang, V., Salim, F., & Moskovits, P. (2013). The websocket protocol. The Definitive Guide to HTML5 WebSocket, 33-60.

Wantao, L., Brian, T., Rajkumar K., and Ian Foster. (2010). A data transfer framework for large-scale science experiments. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10). Association for Computing Machinery, New York, NY, USA, 717–724. https://doi.org/10.1145/1851476.1851582

Xue, M., & Zhu, C. (2009, May). The socket programming and software design for communication based on client/server. In 2009 Pacific-Asia Conference on Circuits, Communications and Systems (pp. 775-777). IEEE.

Zwaenepoel, W. (1985). Protocols for large data transfers over local networks. ACM SIGCOMM Computer Communication Review, 15(4), 22-32.