A REPORT

ON

Up File Sharing Application

BY

SIDDHARTH SHAH

NIPUN SOOD

YASIF KHAN

AKASH SHEKHAWAT

LAV VIJAYVARGIYA

Prepared in partial fulfillment of the

Course No - ISF 341

Software Engineering

At

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

K.K. BIRLA GOA CAMPUS

Abstract

Our project for IS F341 Software Engineering Course is titled Up File Sharing. It aims to make file transfer as simple as the name is. Select the users you want to share the file with, and just press send. That's all it should take to send the file, and that's all it will take. Our project gives the file transfer process complete platform independence - across both devices as well as operating systems. Share class notes, collaborate on projects easily, transfer documents without needing to open mail, or upload and download from cloud manually or using cables. It follows the SaaS Model - providing file sharing capability as a service using an associated Web app and Android App.

Introduction

Up File Sharing is a multi platform file sharing application that facilitates seamless file transfer across heterogeneous devices. The idea for the was project by conceptualized by the need for an intuitive file sharing service that works across multiple platforms. There are a variety of ways consumers currently use to send files to each other - using cloud services(such as Google Drive) or by physically connecting the heterogeneous devices together through cables. This whole process is tedious, time consuming and not at all seamless. To remove all these hurdles, we decided to create a platform independent service that would cater to the file transfer needs of this generation seamlessly. Thus, the idea of Up File Transfer App was conceptualised.

Related Work

There are a lot of other file transfer applications available for use already. However, each one of them suffers from various shortcomings such as low file size limit, platform dependence, non intuitiveness and low proximity. For example Xender, a trending file transfer application only lets you transfer files across android devices and that too within the proximity range of that of a wifi hotspot.

Whatsapp (another popular offering) has a file size limited to 15MB. Up File sharing solves all these issues by introducing platform independence, higher file size limits, seamless and no proximity requirements i.e. you can send from one corner of the world to another. Thus our project is an improvement over other similar projects that are currently in use. Below is a table mapping the various shortcomings observed in current services and how our attempts to remove each of them -

Service	Drawback	Improvement by Up
Xender	File transfer limited to the range of a wifi hotspot	File transfer occurs over the internet - no proximity requirements between users
WhatsApp	File size limited to 16 Mb	Allows transfer for file sizes upto 50 Mb
Google Drive	No pairing capability among users, No seamless transfer, Users have to manually upload and download files	Pairing implemented, Seamless and easy for users.

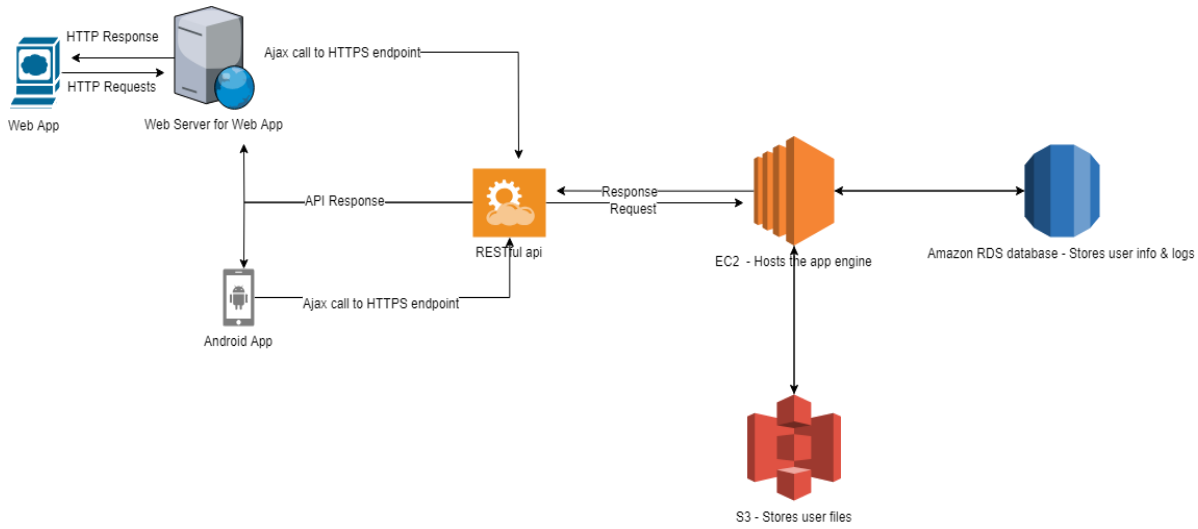
Project Design

Development model

The entire development cycle followed the **agile development model** with periodic **code sprints** of 2 weeks each designed to add new features or update existing ones. Division of labour was completely **decentralised** - with sub-teams of 1 or 2 responsible for the development of each major component - the app engine, the web app and the android app. Each sub-team had **complete ownership** of its product and was responsible for unit testing and integration testing within its own product.

System Architecture

Our application is completely **cloud native** with the entire IT infrastructure deployed on the Amazon Web Services platform. The cloud based approach was used as it provides on demand IT services with a pay as you go pricing model. This removes the capital expenditure of allocating & maintaining the required hardware ourselves - enabling us to focus on the business logic of the service itself. The app engine that is hosted on an **EC2 instance** - providing us with the low-cost, low-latency and scalable on demand compute capacity.



To host the users files, we used a dedicated **S3 bucket**. S3 provides a secure storage service where each bucket can store upto 5 terabytes of data. To store the logs and other user information critical to the business logic, we used the **Amazon RDS** service with a dedicated **MySql database**. Another EC2 instance is used to host the web server for WebApp. The file sharing service is offered to clients in the form of a browser based Web App as well as an Android App. The apps communicate with the app engine using **RESTful apis**.

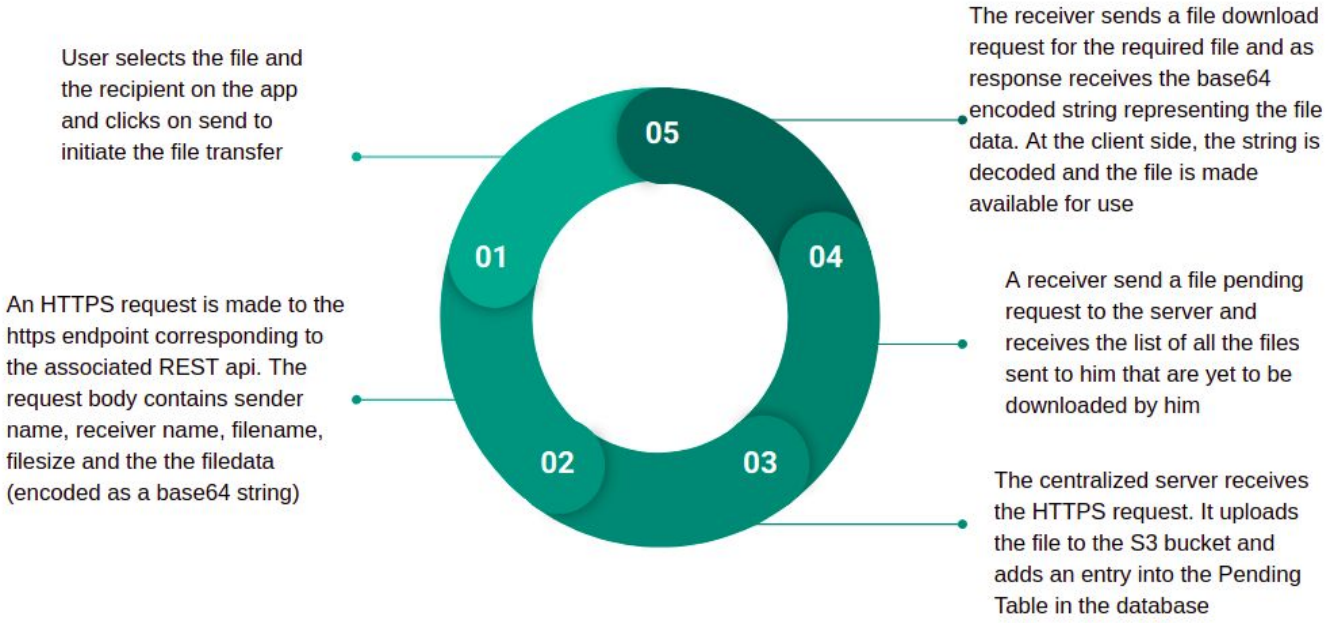
Software Architecture

The project makes use of the popular Client-Server architecture to implement the file transfer service. Since the use cases and functionality of the service are currently small in terms of scope, we opted for a monolith over a microservices architecture. An EC2 instance acts as the centralized server and hosts the Flask app that responds to HTTP requests made by the clients (Web App or Mobile App).

The communication takes place using REST apis created using the Flask microframework for Python. The app engine interacts with the MySql database to store information such as user details, pairing data, pending file transfers, etc. It also interacts with an Amazon S3 bucket which acts as a file storage container for all the files uploaded by the users.

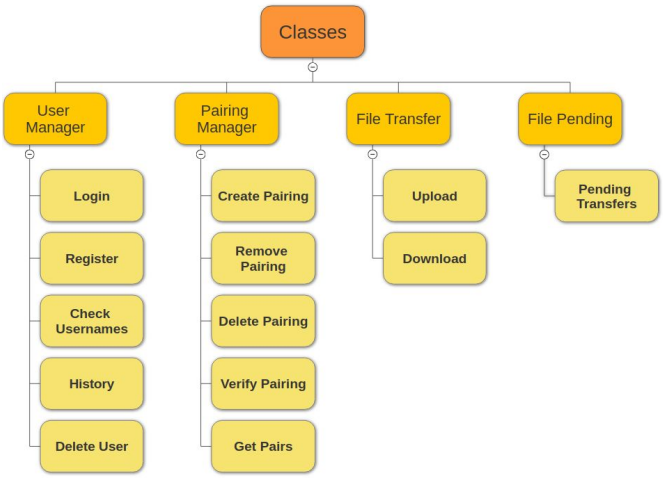
The bucket acts as a necessary backup if incorrect/failed file transfers occur on either end. The entire file transfer process takes place in a secure manner as the Web App supports HTTPS - Ensuring that the communication between the clients and the server in end to end encrypted.

File Transfer Workflow



Logical Classification of REST API's

The diagram below depicts the REST Api's that form the core business logic of the app engine. An object oriented approach was used wherein each api is implemented as a function within its associated class. The various classes are depicted as the second level of the diagram whereas the api's form the third level. The classes are highly cohesive in nature with minimal coupling between them.



Software tools and technologies

Component	Tools
App Engine	Python, Flask, Nginx, MySQL, gunicorn, Boto, Postman
Android App	Java, Android Studio, XML, MySQL
Web App	Nodejs, Expressjs, JavaScript, HTML, CSS

Instructions to run the Apps

Instructions For Android app :-

- Install APK in the phone.
- Register using the Register button.
- Login using your credentials.
- Click on Pairing to Pair with your friends.
- Use the Up Button for Sending files and Down button for files downloading received files.
- Menu Tab can be clicked to go to History, Pairing and Logout options.
- You can select more than one recipient for sending by sending comma separated names(i.e. “yasif,akash,nipun”).
- Downloaded files will be saved in Internal Memory in a folder called "Up File Sharing".
- Click on History to check your upload and download log history.

Instructions for Web App :-

- Open the link - https://up-karoon.ga/
- Register by clicking on the “Sign up” button and fill details.
- Login using your credentials.
- Click on Pairing to Pair with your friends.
- Use Up button for Sending files and Down button for Downloading received files.
- You can directly add paired users to the recipient list while sending by clicking on them.
- You can select more than one recipient for sending by sending comma separated names(i.e. “yasif,lav,nipun”).
- Downloaded files will be saved in the default Downloads folder of the browser.
- Click on History to check your upload and download log history.

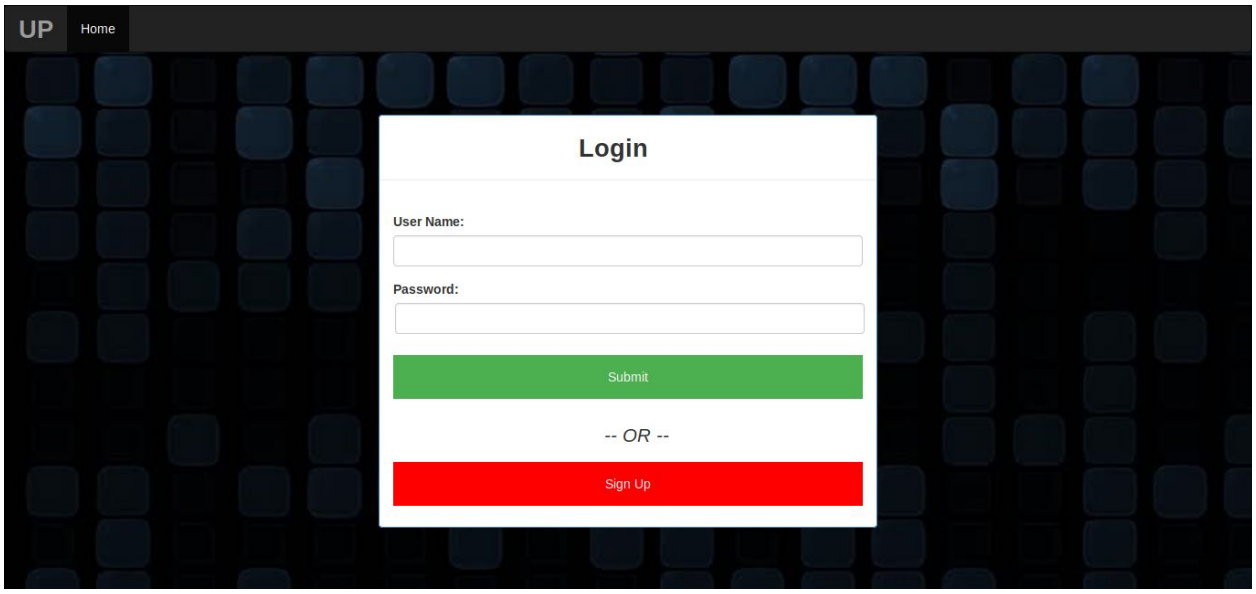
Usage References

1. Associated GitHub code repository - https://github.com/Speeeddy/up-file-sharing/
2. Link to Web App - https://up-karoon.ga/
3. Link to Android App Apk - https://tinyurl.com/ydfnw2zk/
4. Demo Video For Usage - https://tinyurl.com/yadj5fm7/

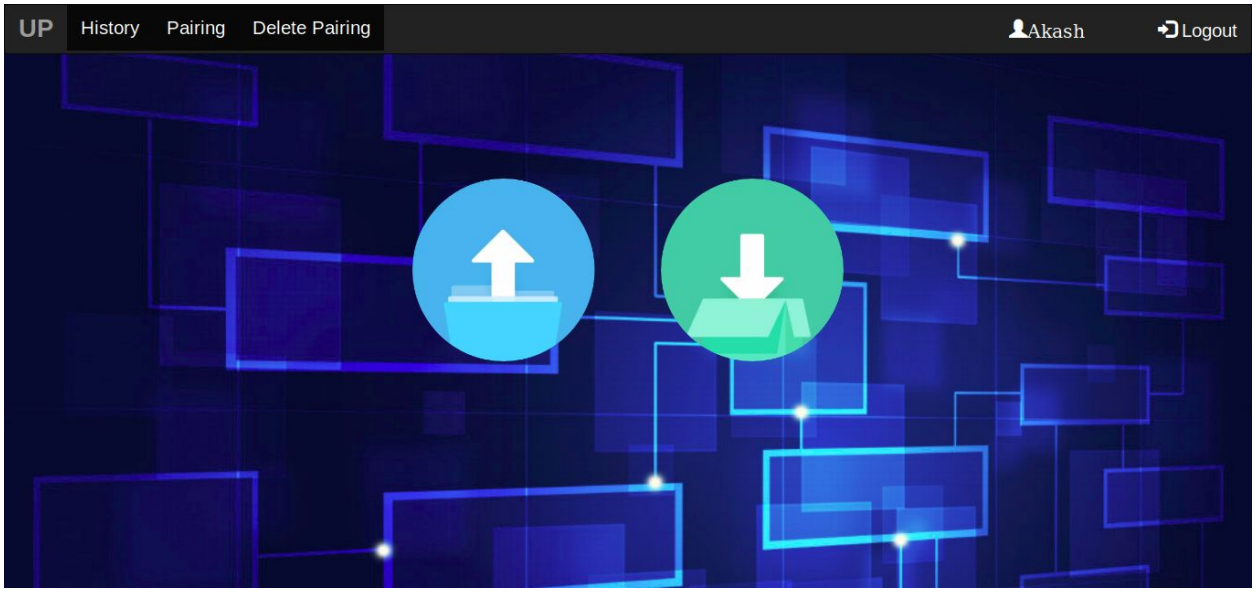
Project Snapshots

Below are a few snapshots of the project :-

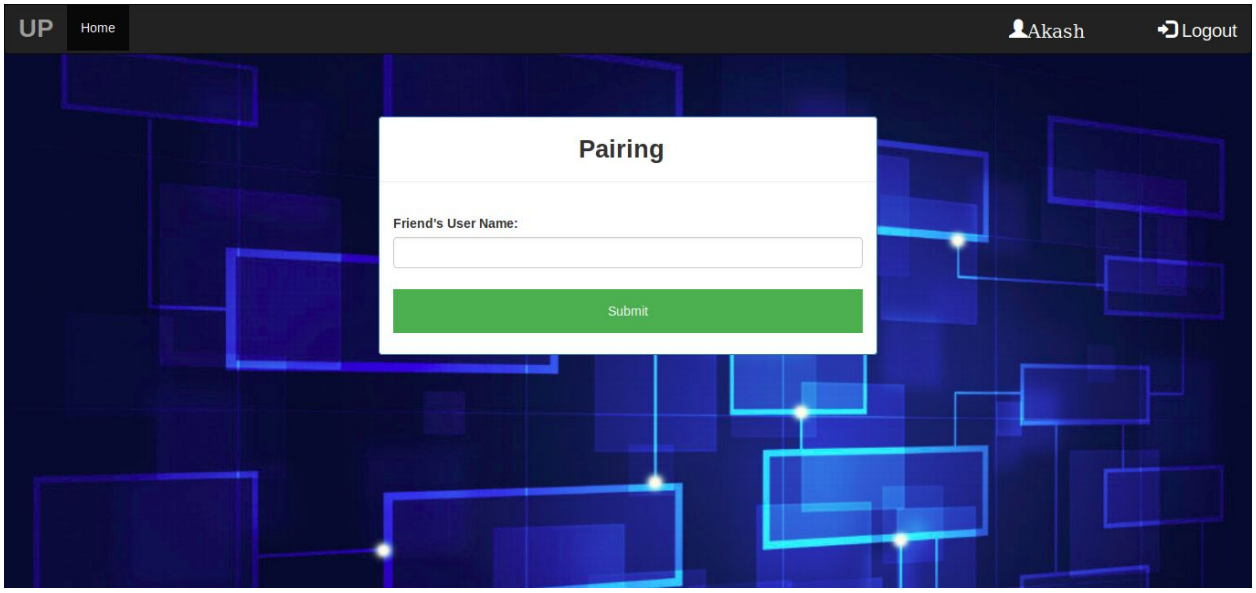
- Web App Login Screen -



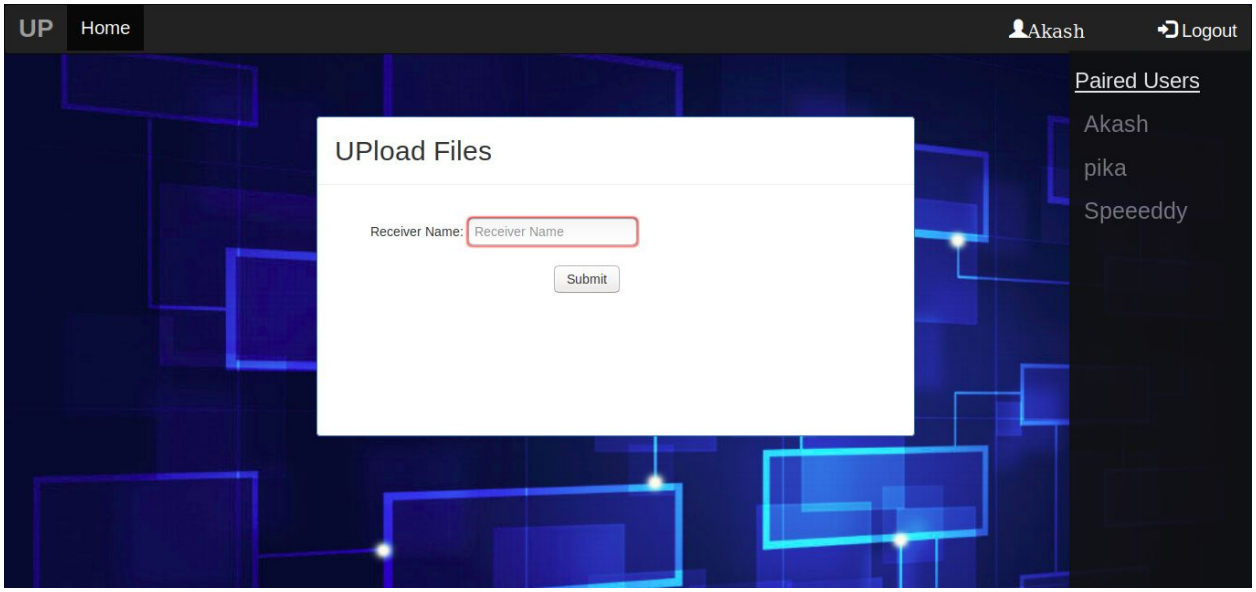
- WebApp Home -



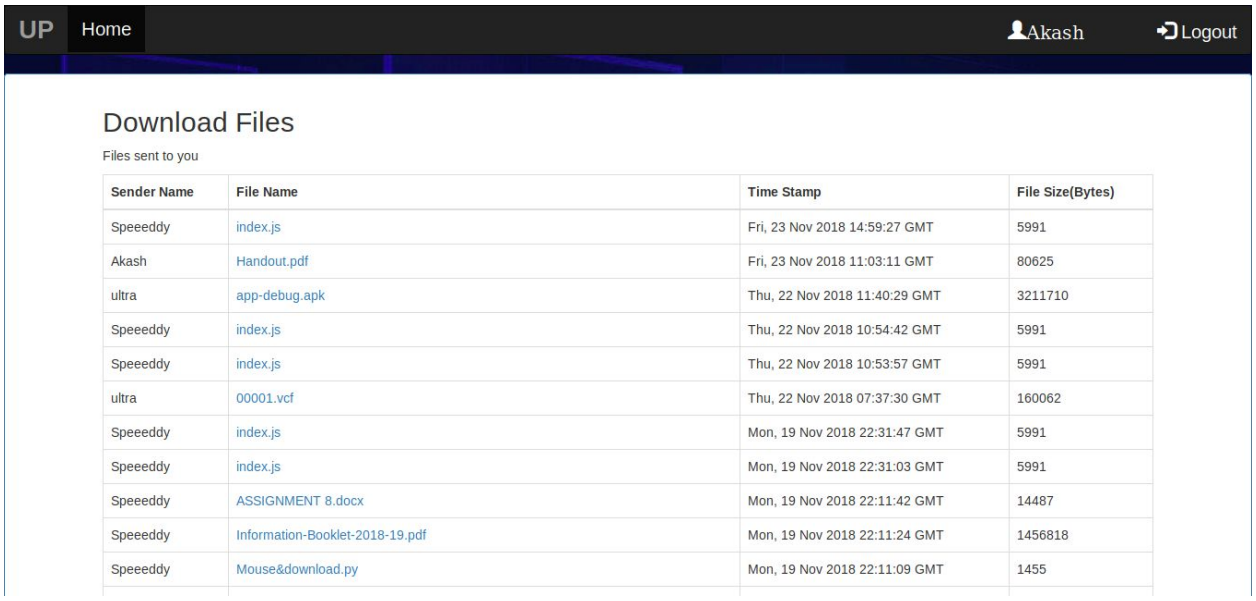
- Web App Pairing -



- Web App Upload -

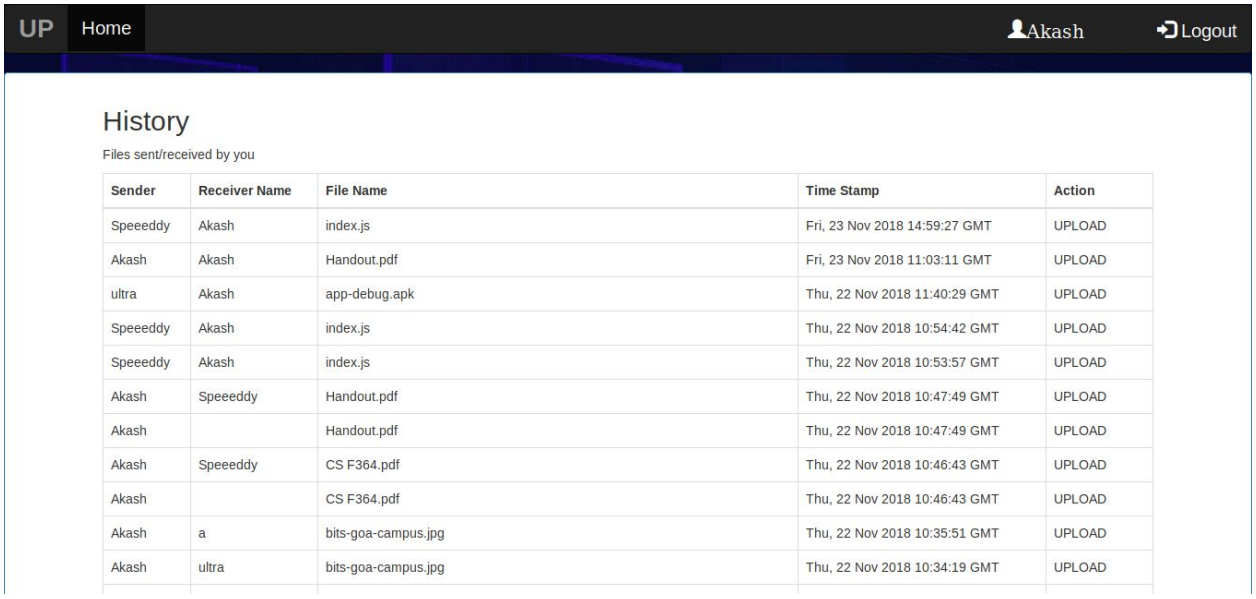


- Web App Download -



Sender Name	File Name	Time Stamp	File Size(Bytes)
Speeeddy	index.js	Fri, 23 Nov 2018 14:59:27 GMT	5991
Akash	Handout.pdf	Fri, 23 Nov 2018 11:03:11 GMT	80625
ultra	app-debug.apk	Thu, 22 Nov 2018 11:40:29 GMT	3211710
Speeeddy	index.js	Thu, 22 Nov 2018 10:54:42 GMT	5991
Speeeddy	index.js	Thu, 22 Nov 2018 10:53:57 GMT	5991
ultra	00001.vcf	Thu, 22 Nov 2018 07:37:30 GMT	160062
Speeeddy	index.js	Mon, 19 Nov 2018 22:31:47 GMT	5991
Speeeddy	index.js	Mon, 19 Nov 2018 22:31:03 GMT	5991
Speeeddy	ASSIGNMENT 8.docx	Mon, 19 Nov 2018 22:11:42 GMT	14487
Speeeddy	Information-Booklet-2018-19.pdf	Mon, 19 Nov 2018 22:11:24 GMT	1456818
Speeeddy	Mouse&download.py	Mon, 19 Nov 2018 22:11:09 GMT	1455

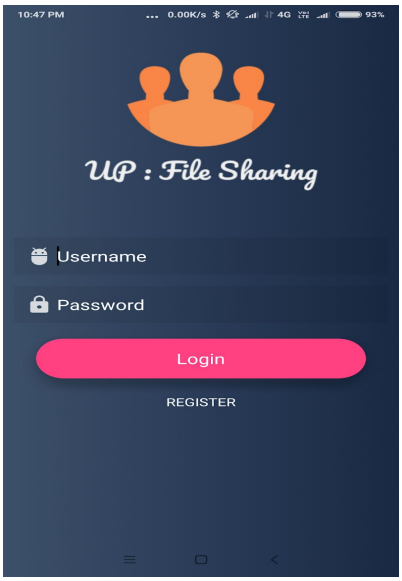
- Web App History Log -



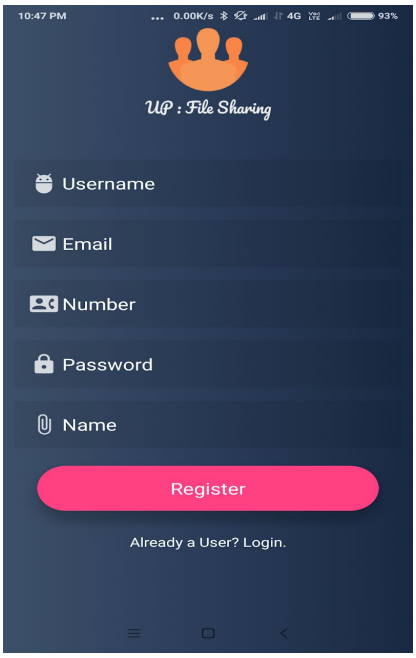
Sender	Receiver Name	File Name	Time Stamp	Action
Speeeddy	Akash	index.js	Fri, 23 Nov 2018 14:59:27 GMT	UPLOAD
Akash	Akash	Handout.pdf	Fri, 23 Nov 2018 11:03:11 GMT	UPLOAD
ultra	Akash	app-debug.apk	Thu, 22 Nov 2018 11:40:29 GMT	UPLOAD
Speeeddy	Akash	index.js	Thu, 22 Nov 2018 10:54:42 GMT	UPLOAD
Speeeddy	Akash	index.js	Thu, 22 Nov 2018 10:53:57 GMT	UPLOAD
Akash	Speeeddy	Handout.pdf	Thu, 22 Nov 2018 10:47:49 GMT	UPLOAD
Akash		Handout.pdf	Thu, 22 Nov 2018 10:47:49 GMT	UPLOAD
Akash	Speeeddy	CS F364.pdf	Thu, 22 Nov 2018 10:46:43 GMT	UPLOAD
Akash		CS F364.pdf	Thu, 22 Nov 2018 10:46:43 GMT	UPLOAD
Akash	a	bits-goa-campus.jpg	Thu, 22 Nov 2018 10:35:51 GMT	UPLOAD
Akash	ultra	bits-goa-campus.jpg	Thu, 22 Nov 2018 10:34:19 GMT	UPLOAD

Mobile App Snapshots :-

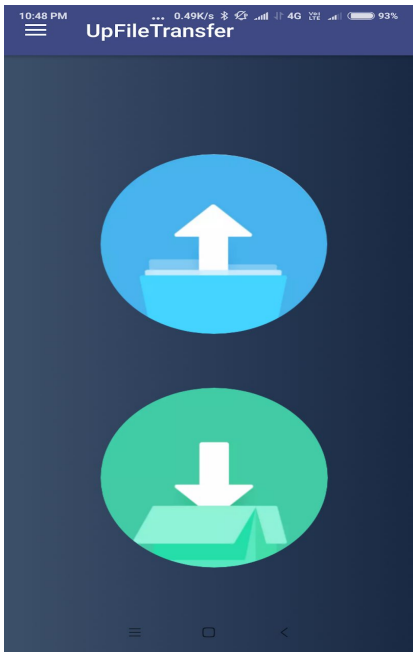
Login Screen



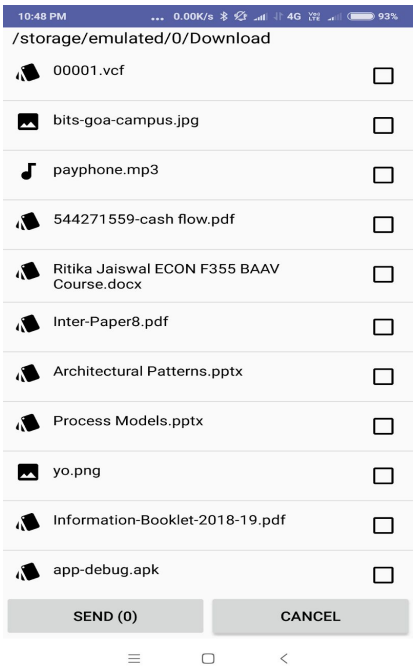
Register Screen



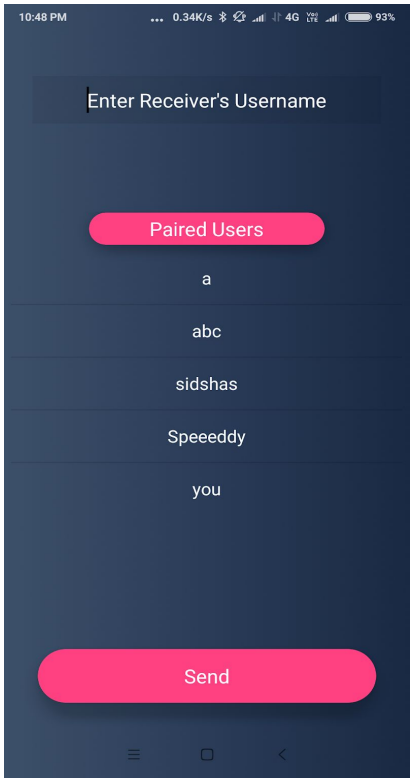
Send Receive Screen



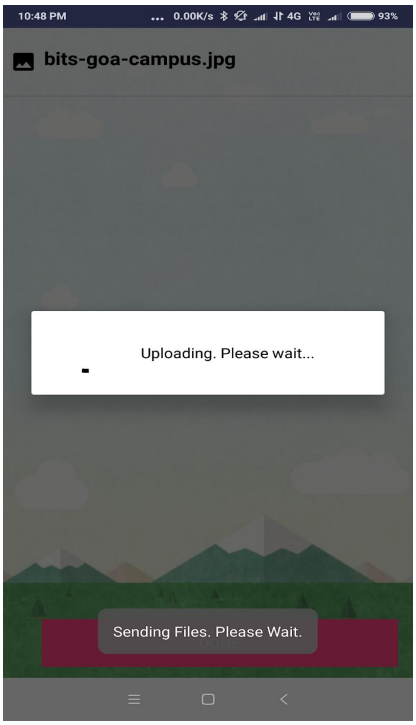
Select Files Activity



Peer Chooser Activity



File Sender Activity



Pending Downloads List