# Project: Red Pony

Daniel Kim
Daniel Olivera
Jared Coleman
Mario Lopez
Viet Le

# Problem

- Traditionally, access to items in a physical world is restricted by physical laws
    - Location
    - Time
- Items in the digital realm are not restricted in this way
- This is often an advantage, but can sometimes be a burden
    - Items that belong to a location
    - Items that should not be shared
    - Items that should expire

# Goal

- To create a framework for enforcing physical access restrictions for digital files
  - Tie digital files to the physical world
- To create an application that uses the framework and demonstrates its applicability
  - Location restrictions
  - Time restrictions

# Proposed Solution

The Astral Framework

- Server-side framework for storing and providing access to files with location and time restrictions
- Tools
    - MongoDB
    - SSL
    - Node.js RESTful API
    - JSON Web Token authentication

Phylo

- File Sharing
    - Share files to locations
    - Create rooms
        - File sharing
        - Discussion
- Tools
    - Ionic
    - AngularJS
    - HTML5

# The Astral Framework

# Requirements

- Purpose
  - Enforce location and time restrictions
- Responsibilities
  - Files
  - Users
- Scalability
  - Ability to support large applications
- Applicability
  - General purpose
  - Useful for a wide variety of applications

# Access Control

- Server-side access control
- Location
    - Query database for files within a threshold of location
- Time
    - Query database for files with expirations greater than current time
    - Use location to determine local time

# Responsibilities

Files

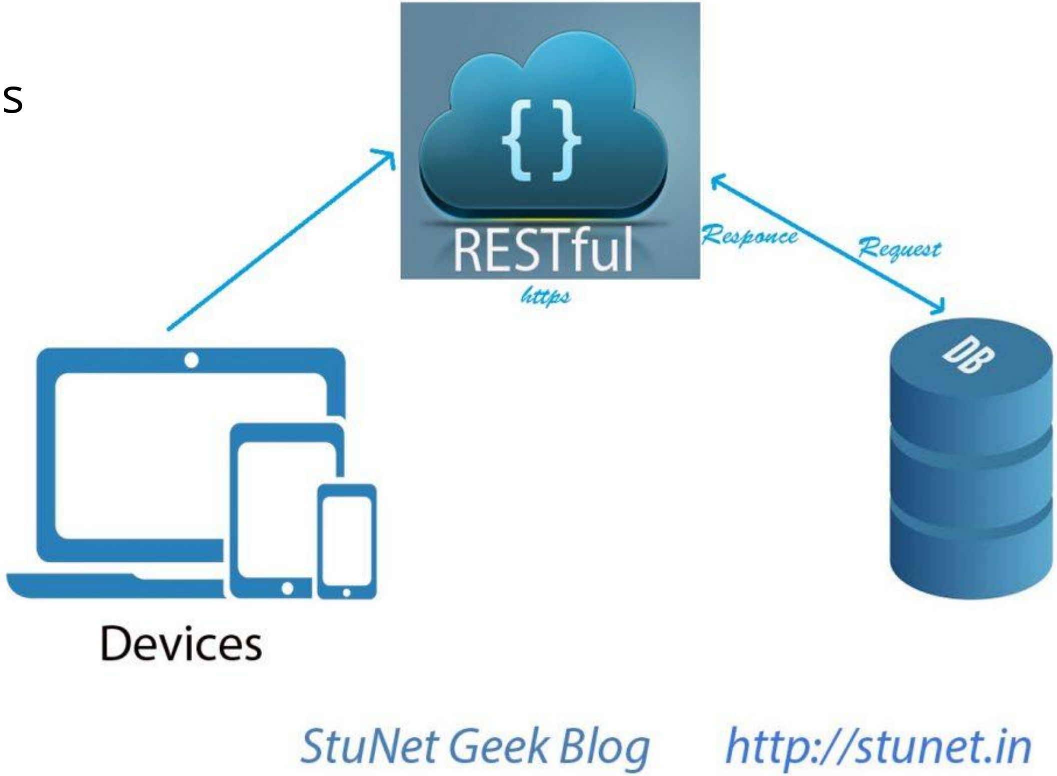
- Attributes
  - Name
  - Author
  - Location
  - Expiration
  - Data
- When user requests files, return those within radius of user location

Users

- Data
  - Phone
  - Username
  - Password
  - Data
- Allow users to create accounts
- Allow users to login

# Scalability

- Use popular, scalable tools
  - Node.js
  - MongoDB (Mongoose)
- RESTful API
  - <u>Create</u> new file/user
  - <u>Read</u> file/user
  - <u>Update</u> existing file/user
  - <u>Delete</u> file/user



# Applicability

- Scavenger hunt application
- Generating foot-traffic for businesses
- Local social networks
- Virtual Classroom
- Community applications

# Tools

MongoDB

- NoSQL database
- JSON Objects
- Roles: admin, user
- Advantages
  - Cross Platform
  - High performance
  - Complex queries
  - Scalable

Node.js

- Asynchronous event driven JavaScript runtime
- Advantages
  - Javascript
  - Lightweight
  - Efficient
  - Scalable

# Security

- Secure Socket Layer (SSL)
  - Establish an encrypted layer between server and client
- Authentication
  - Two-factor
    - Phone-number verification code
    - JSON Web Tokens
    - Password for recoverability
  - Login with google
- Passwords
  - User passwords are hashed with salt **then** stored
  - Logins will compare hashes

# Competition

- MongoDB, MySQL
  - **Not complete** - developers need to implement location and time access restrictions
- Google, Facebook, etc
  - **Does** provide user management
  - **Does not** provide file management
  - **Does not** enforce location or time access restrictions

# Phylo

# Overview

File sharing over the Physical world

- Access Restrictions
  - Location
  - Time
  - User ID
  - User demographic
- Virtual rooms
  - Reflect physical locations
  - File sharing
  - Discussion

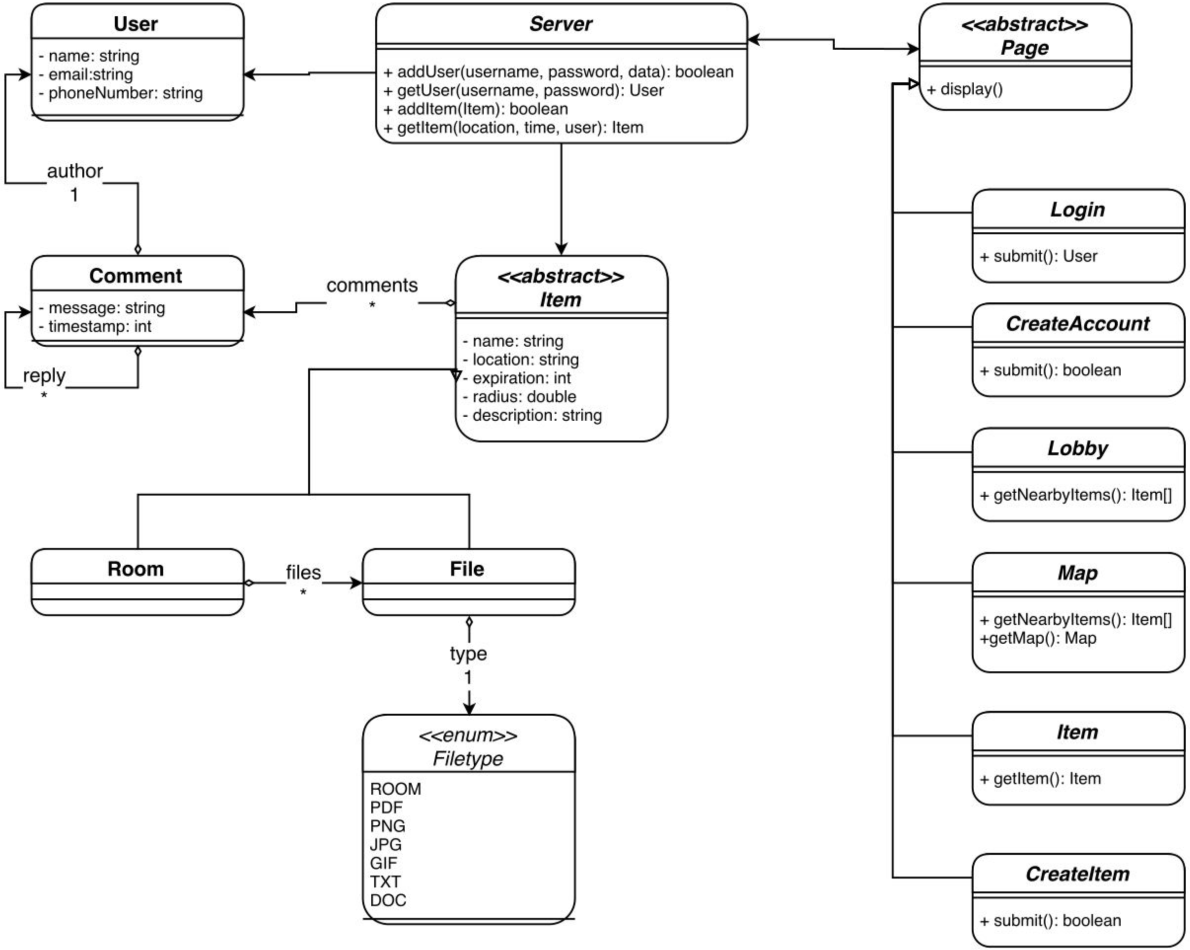
# Sharing Files

- Users can drop files to their location
- Users can access files, if the meet requirements
    - File location must fall within user radius
    - File expiration must be in the future
    - For some files, users must have a valid user ID (have been invited)
    - For some files, users must belong to a certain demographic
- Users can a heatmat of files that exist outside the radius
    - No content
    - Only otherwise available files

# Virtual Rooms

- Users can create virtual rooms at their location
- Virtual rooms only show up to invited members
  - Room must fall within user radius
  - Expiration must be in the future
  - Restricted by user ID (invitation)

# UML



**User**
- name: string
- email:string
- phoneNumber: string

***Server***
+ addUser(username, password, data): boolean
+ getUser(username, password): User
+ addItem(Item): boolean
+ getItem(location, time, user): Item

***<<abstract>> Page***
+ display()

**Comment**
- message: string
- timestamp: int

***<<abstract>> Item***
- name: string
- location: string
- expiration: int
- radius: double
- description: string

***Login***
+ submit(): User

***CreateAccount***
+ submit(): boolean

***Lobby***
+ getNearbyItems(): Item[]

***Map***
+ getNearbyItems(): Item[]
+getMap(): Map

***Item***
+ getItem(): Item

***CreateItem***
+ submit(): boolean

**Room**

**File**

*<<enum>>*
*Filetype*
ROOM
PDF
PNG
JPG
GIF
TXT
DOC

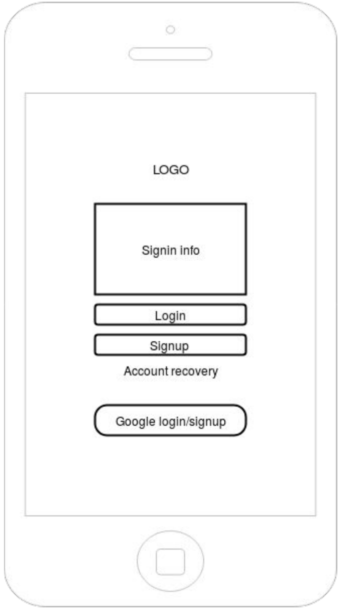
# Ionic

- Framework for mobile application development with web technologies
  - Javascript
  - AngularJS
  - HTML5
- Free and open source
- Fully cross-platform
- Command Line Interface (CLI)
  - Developing
  - Testing on different platforms (IOS, Android, etc.)

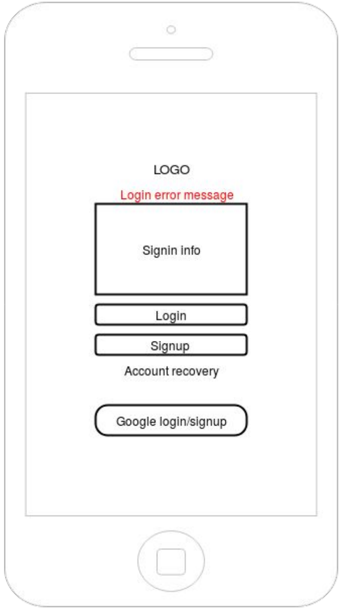
# Competition

- Snapchat, Instagram, etc
  - **Do** enforce time restrictions (disappearing files)
  - **Do not** have file-sharing features (only photos)
- Dropbox, Google Drive, etc
  - **Do** allow file sharing
  - **Do not** enforce location or time restrictions for shared files

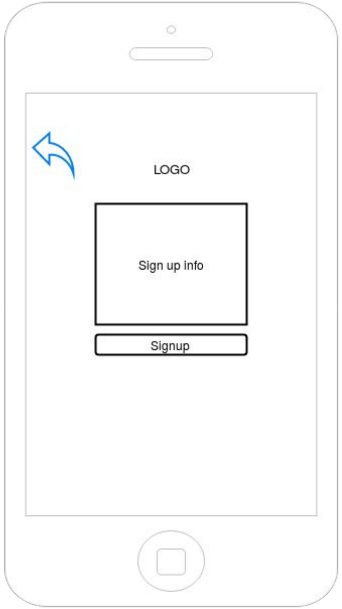
# UI - Login/Signup



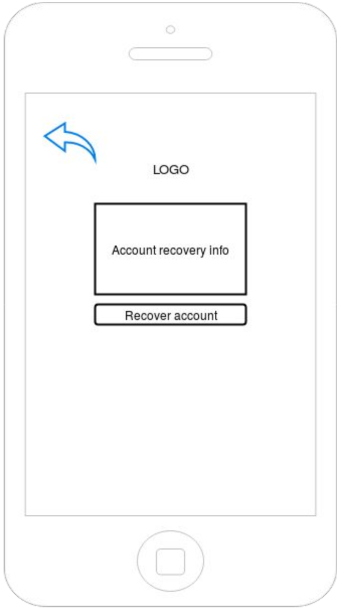
**Main login**



**Login error**

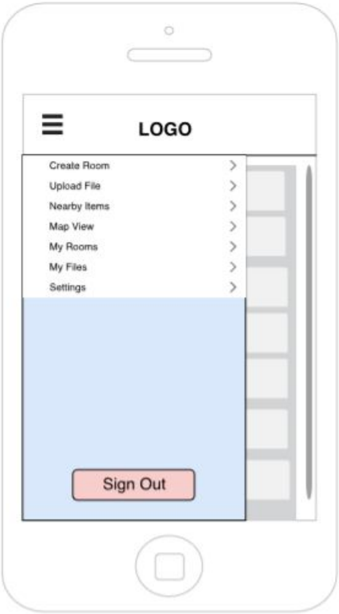


**Sign up**

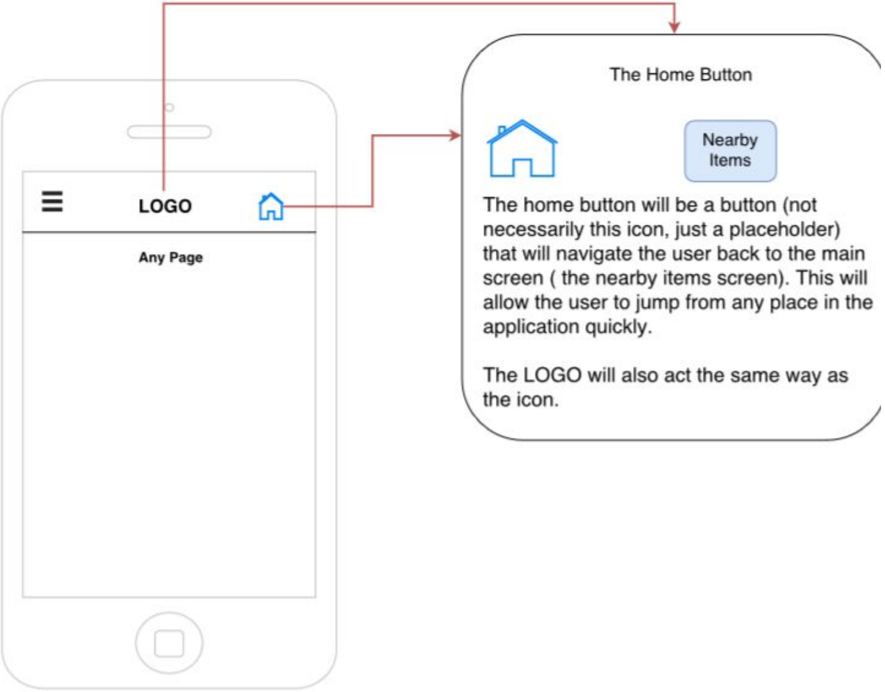


**Account recovery**

# UI - Navigation



**Side Menu**

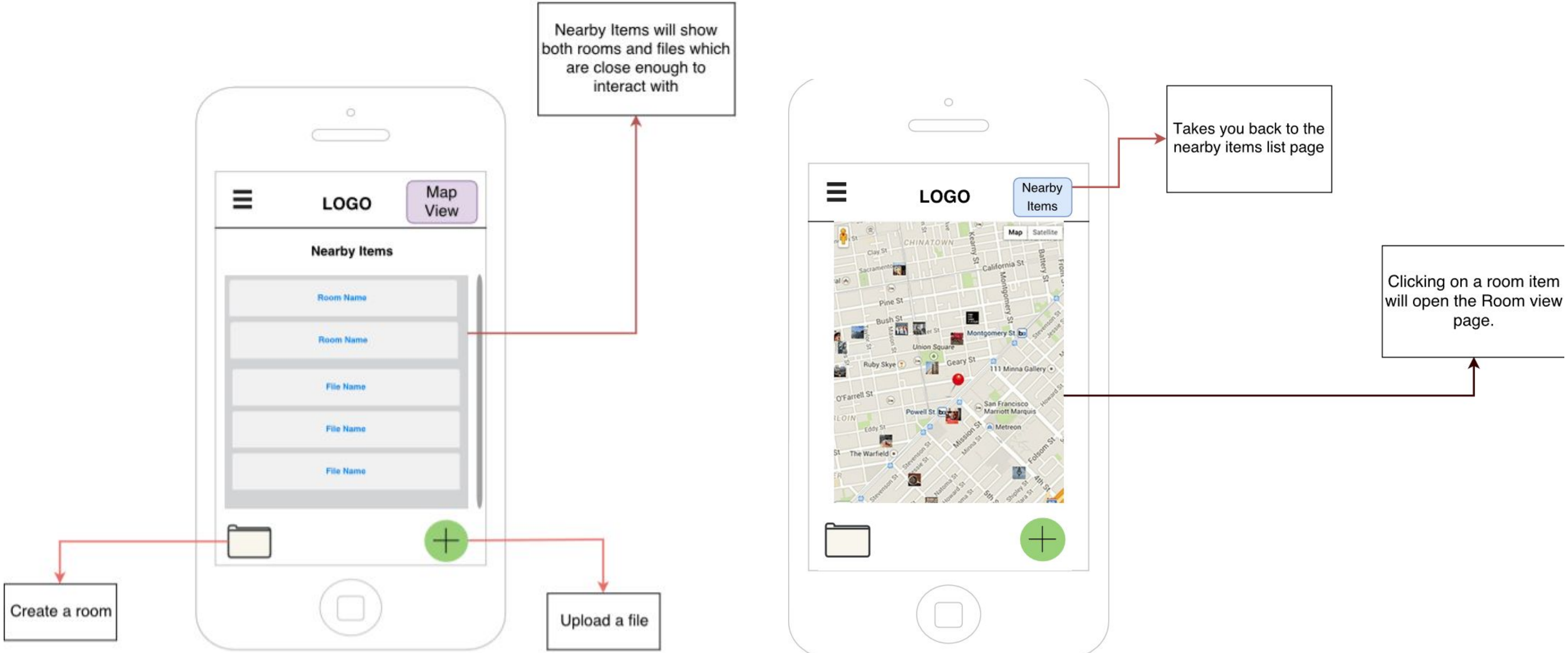


The home button will be a button (not necessarily this icon, just a placeholder) that will navigate the user back to the main screen ( the nearby items screen). This will allow the user to jump from any place in the application quickly.

The LOGO will also act the same way as the icon.

**Home Button**

# UI - Nearby Files/ Rooms



Nearby Items will show both rooms and files which are close enough to interact with

Create a room

Upload a file

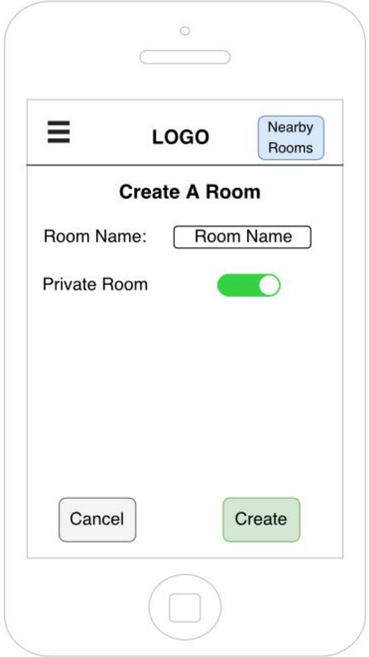
Takes you back to the nearby items list page

Clicking on a room item will open the Room view page.

Main

Map View

# UI - Create A Room



LOGO

Nearby Rooms

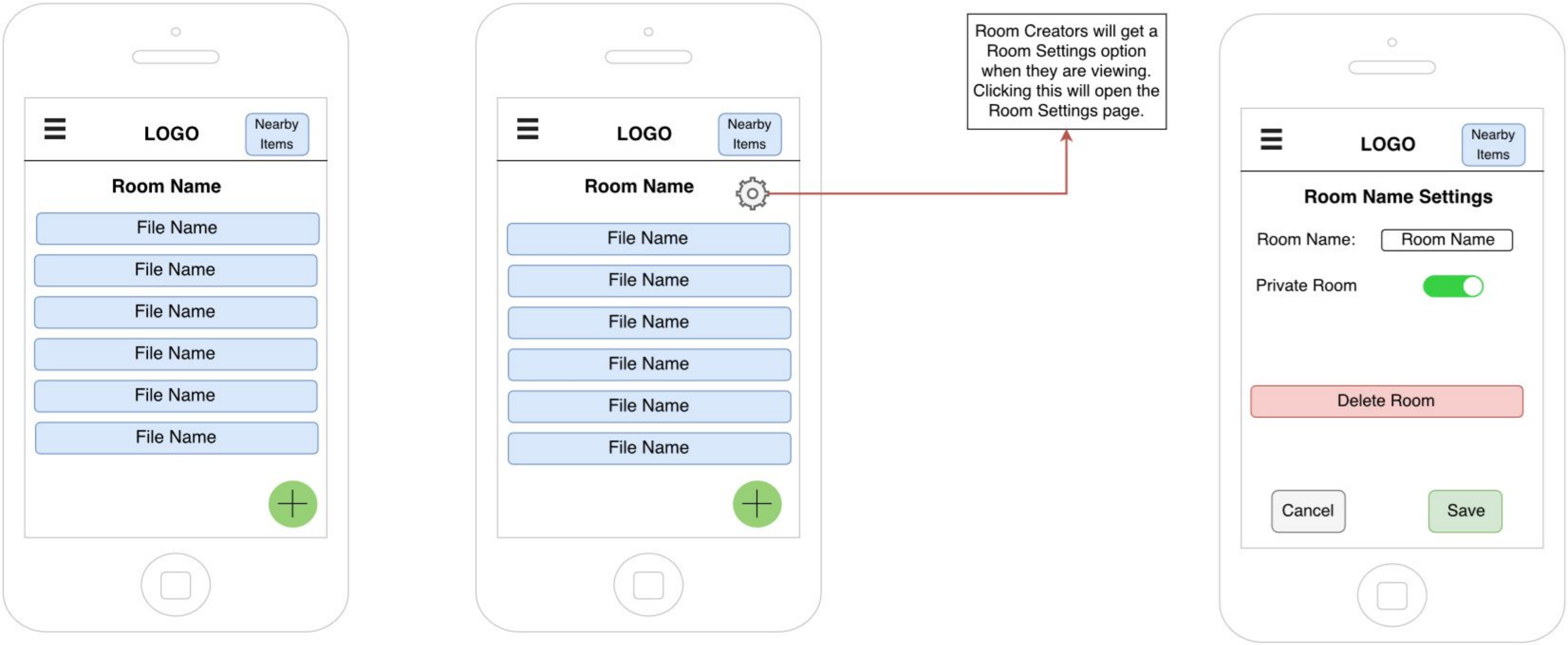
Create A Room

Room Name: Room Name

Private Room

Create Room

# UI - Room Views



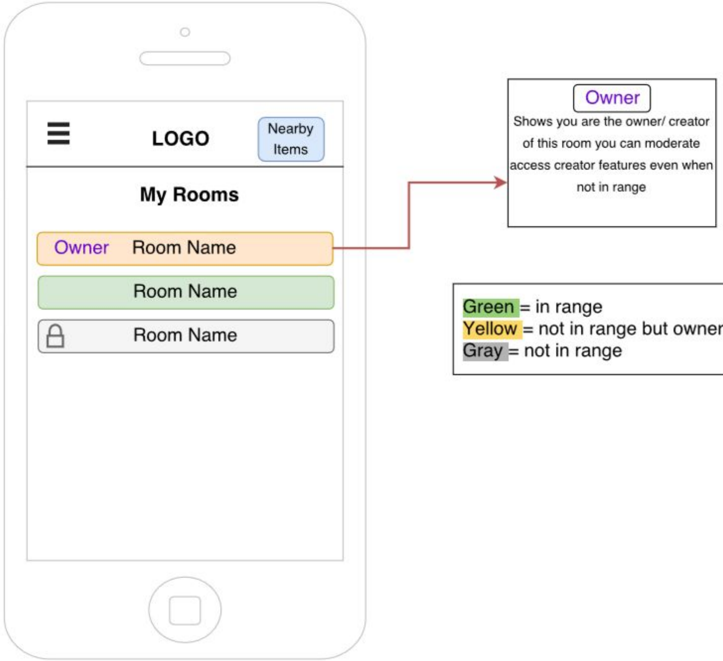
Room Creators will get a Room Settings option when they are viewing. Clicking this will open the Room Settings page.

**User Room View**

**Creator Room View**

**Room Settings View**

# UI - My Rooms

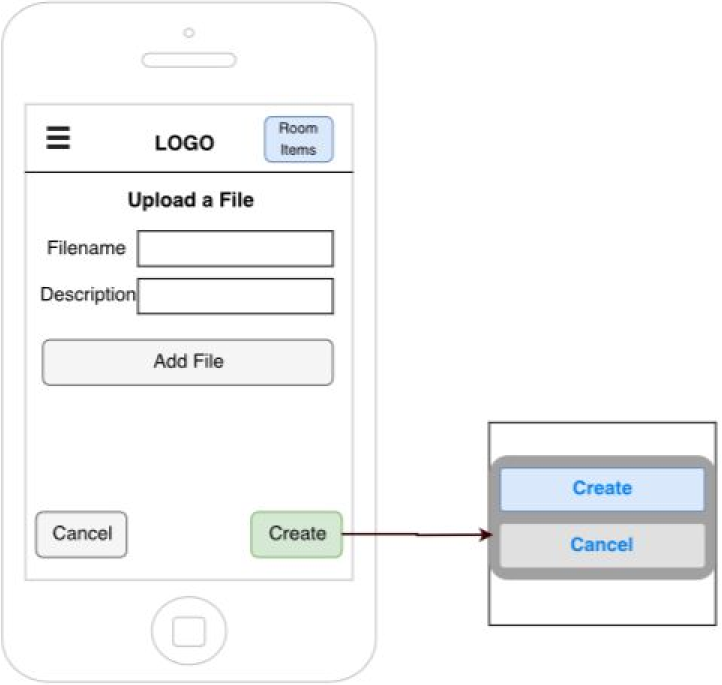


Shows you are the owner/ creator of this room you can moderate access creator features even when not in range

Green = in range
Yellow = not in range but owner abilities
Gray = not in range

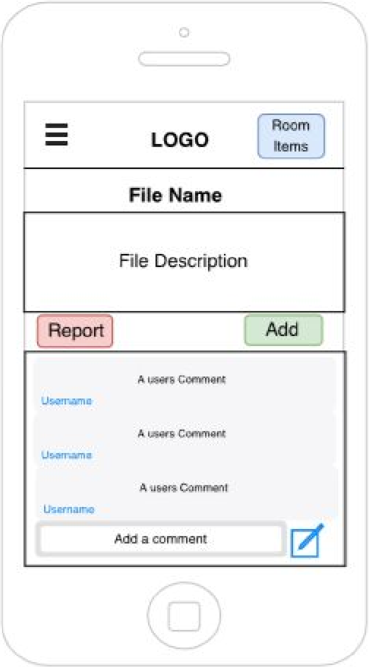
**My Rooms**

# UI - Create A File

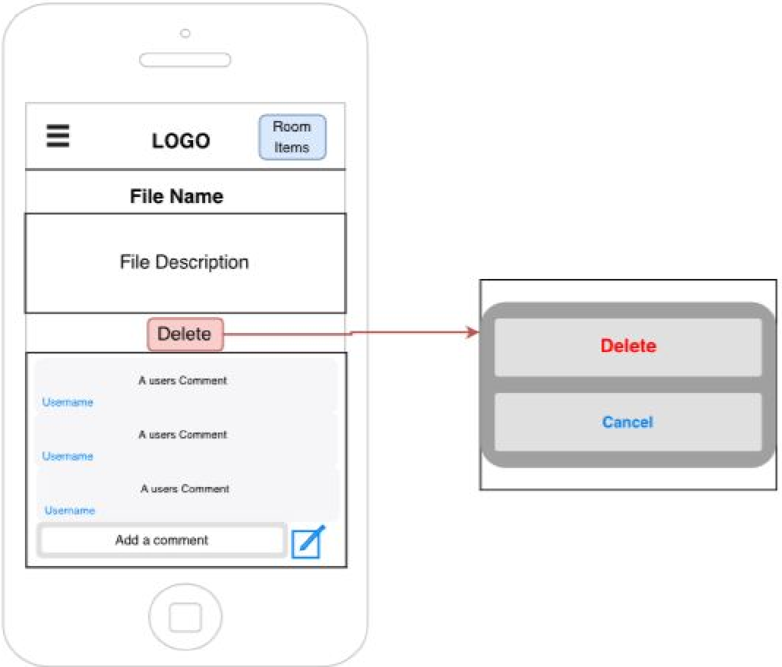


Create File View

# UI - File Views

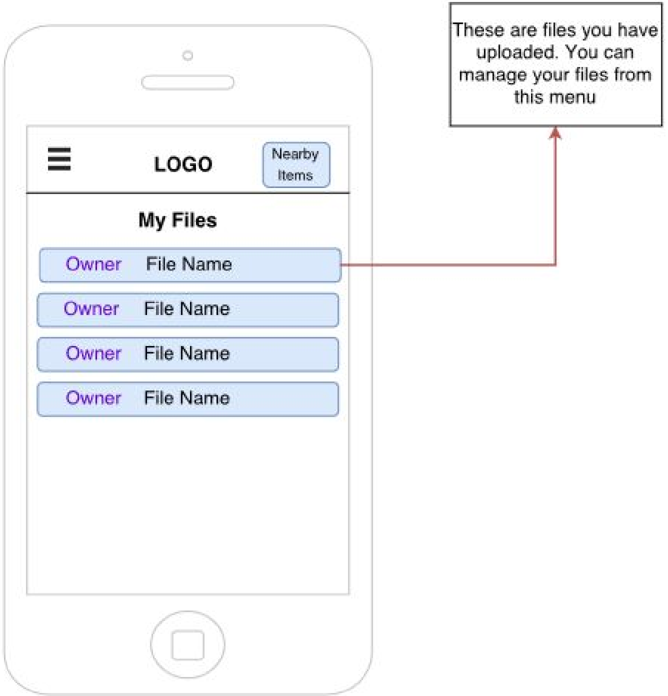


User File View



Creator File View

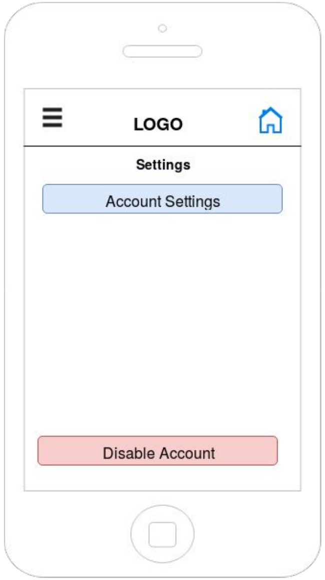
# UI - My Files



These are files you have uploaded. You can manage your files from this menu

My Files

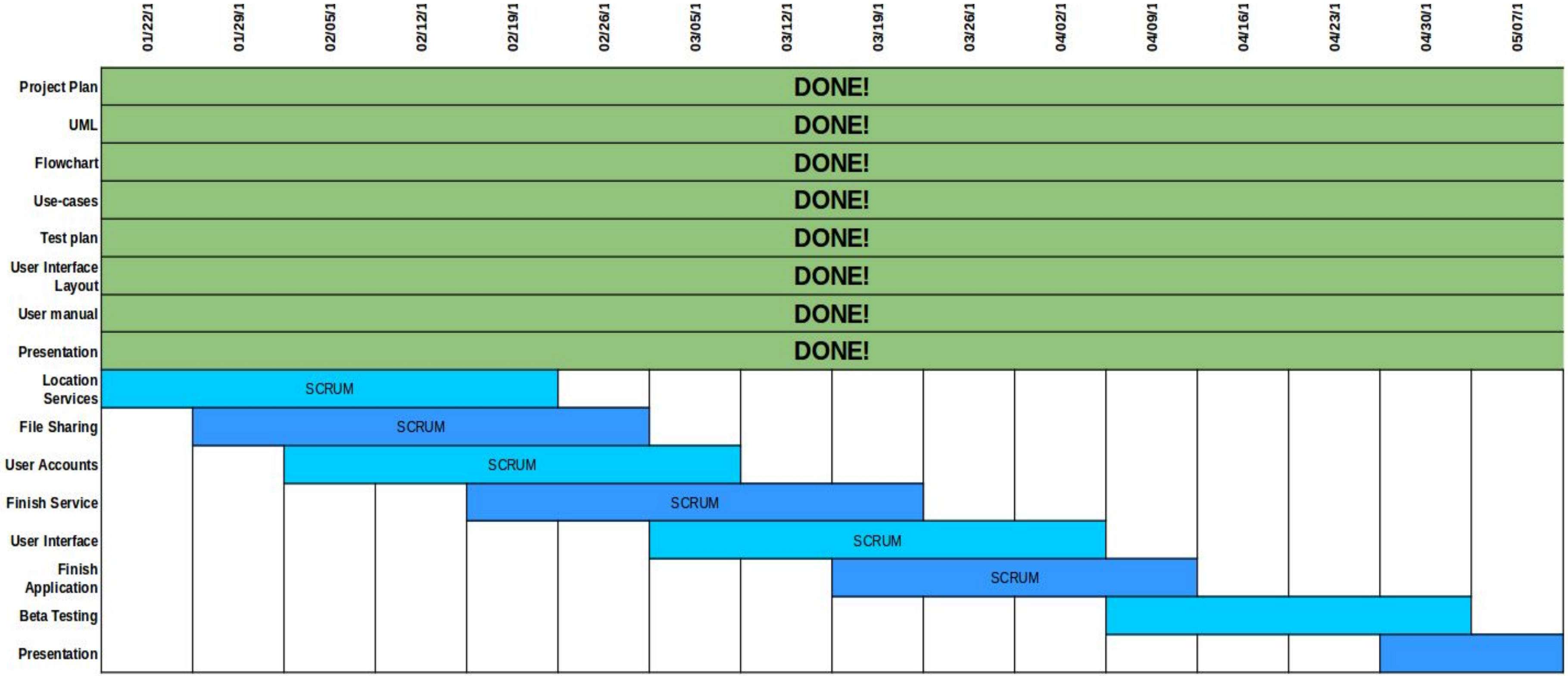
# UI - Account Settings



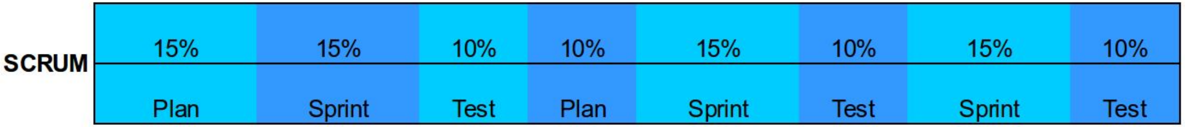
Settings

# Plan

# Timeline


01/22/1
01/29/1
02/05/1
02/12/1
02/19/1
02/26/1
03/05/1
03/12/1
03/19/1
03/26/1
04/02/1
04/09/1
04/16/1
04/23/1
04/30/1
05/07/1
Project Plan
DONE!
UML
DONE!
Flowchart
DONE!
Use-cases
DONE!
Test plan
DONE!
User Interface Layout
DONE!
User manual
DONE!
Presentation
DONE!
Location Services
SCRUM
File Sharing
SCRUM
User Accounts
SCRUM
Finish Service
SCRUM
User Interface
SCRUM
Finish Application
SCRUM
Beta Testing
Presentation

# SCRUM



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| SCRUM | 15% | 15% | 10% | 10% | 15% | 10% | 15% | 10% |
| | Plan | Sprint | Test | Plan | Sprint | Test | Sprint | Test |

- Plan
  - Goals for each sprint
  - Account for sprint duration
- Sprint
  - Develop to meet sprint goals
  - Working version should be complete by end of sprint
- Test
  - Test working version
  - Find and document bugs for next sprint

# Testing

- Use / Develop tools for testing when necessary
    - Ionic testing
    - Postman
- Test for all applicable test cases
- Document results in a detailed test report
    - ID, date, time, tester
    - Test cases
    - Results / Comments
- Evaluate need for new test cases

# References

RedPony github: https://github.com/Doliveraa/RedPony
Ionic documentation: http://ionicframework.com/docs/
MongoDB documentation: https://docs.mongodb.com/
Node.js documentation: https://nodejs.org/en/docs/
SCRUM: https://www.scrum.org/
Draw.io (used for all diagrams): https://www.draw.io/
Postman: https://www.getpostman.com/docs/