
GitHub Username: Your GitHub username here

Lifeguard Hiring App

Description

The hirling website is slow and requires repeated login. This app will take your login credentials once, and login periodically so that you always have a reasonably recent view of who is up next for hiring. The app also gives notifications as to when you are close to being hired.
App also includes a calendar view so that you can see when you have worked this month and total hours. This makes filling out your timecard trivial.

Not sure how to write a good description? Search 5-star apps on the Play Store for inspiration.

Intended User

LA County has about 600 recurrent lifeguards. Maybe half of them use iOS phones, so they would need a similar app written in swift. So really, there are hopefully about 400 recurrent lifeguards with android phones that could potentially use this app.

Features

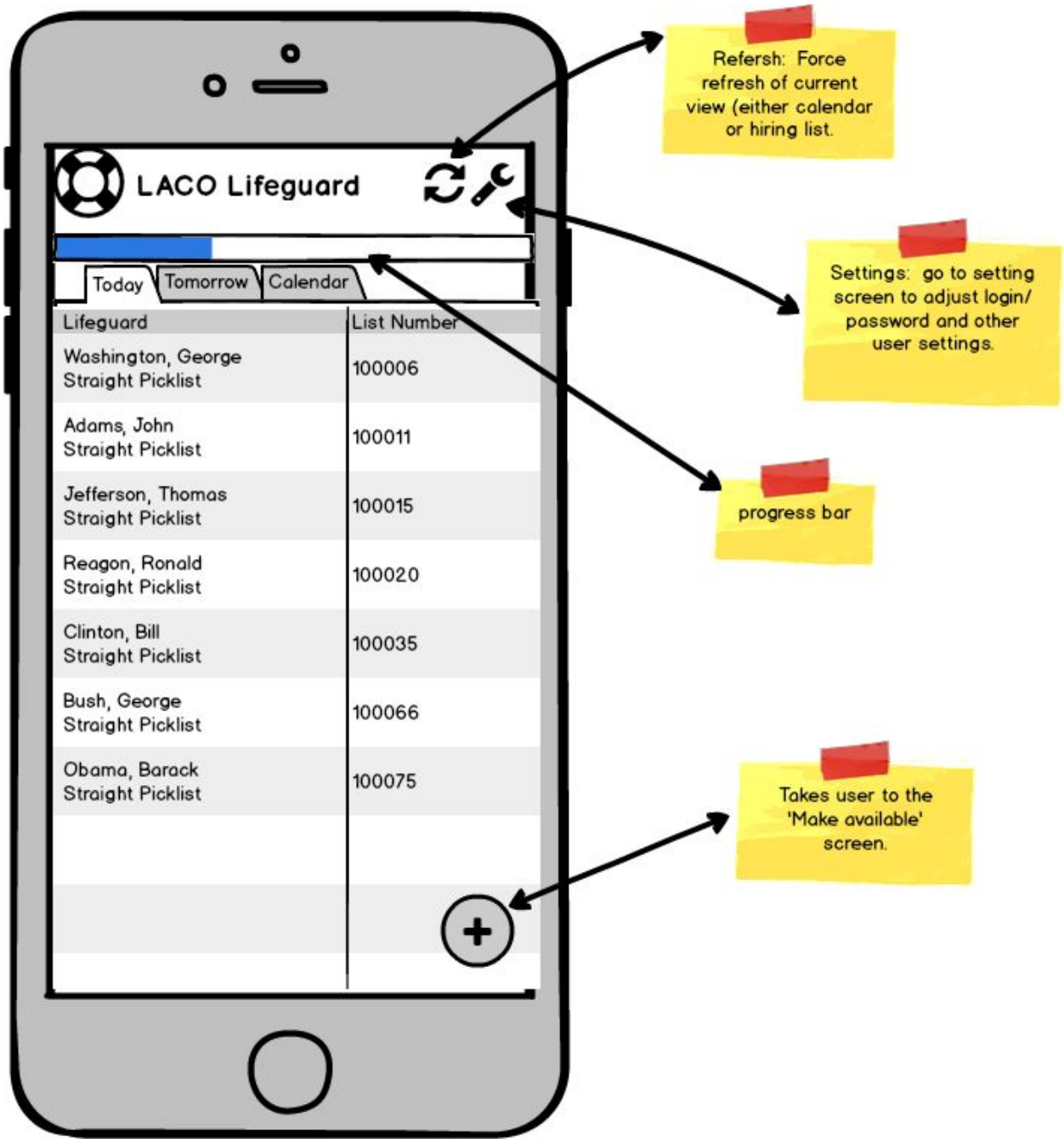
Main features:

- Monitor their place on the hiring list
- Review previous shifts worked or assigned
- Make user available and unavailable for work
- Notify user when user has been made available
- Notify user when user is within 5 people of being hired
- Notify user when user has been assigned a shift

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Today Hiring List View

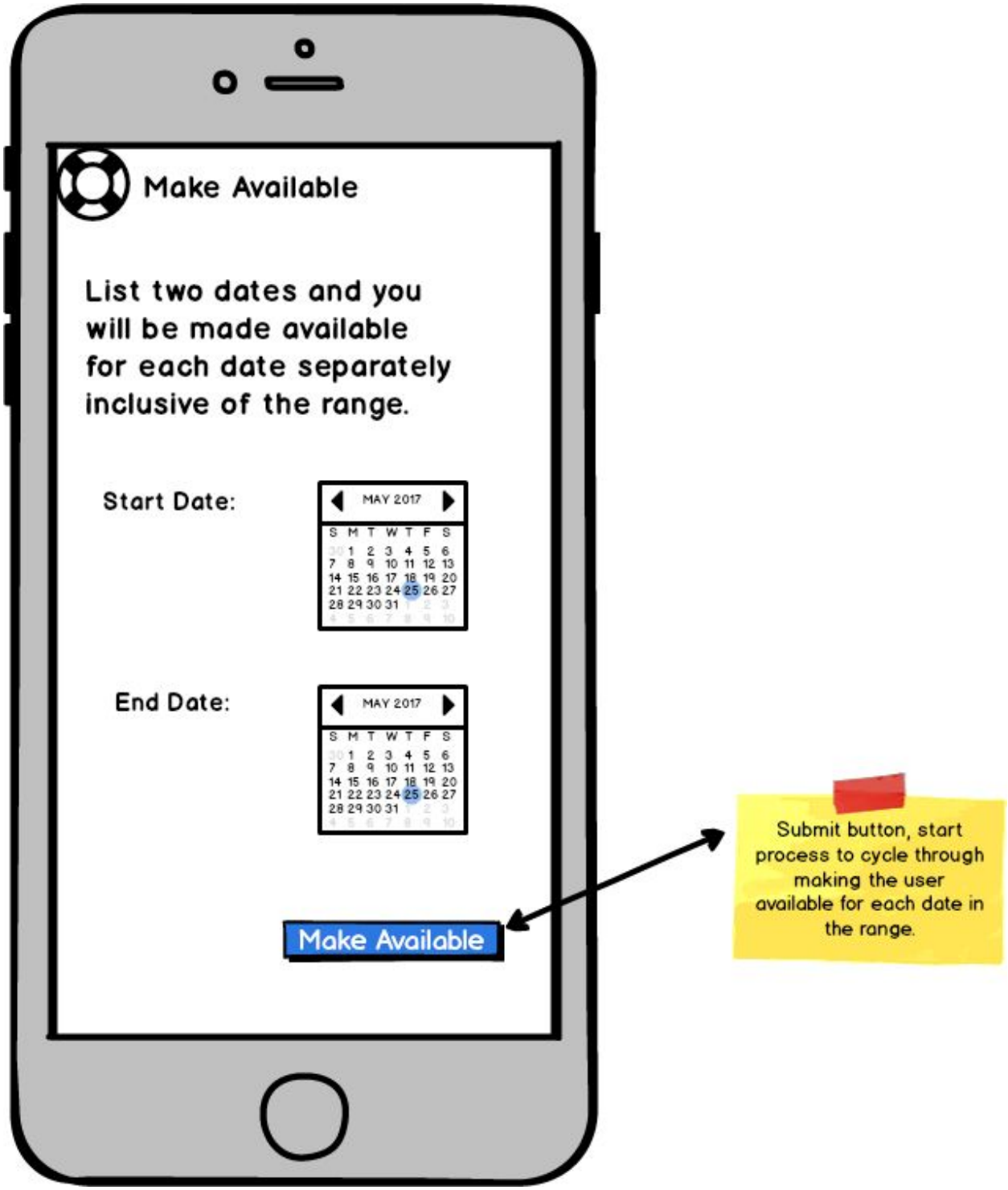


Shows the list of available lifeguards in order of seniority.

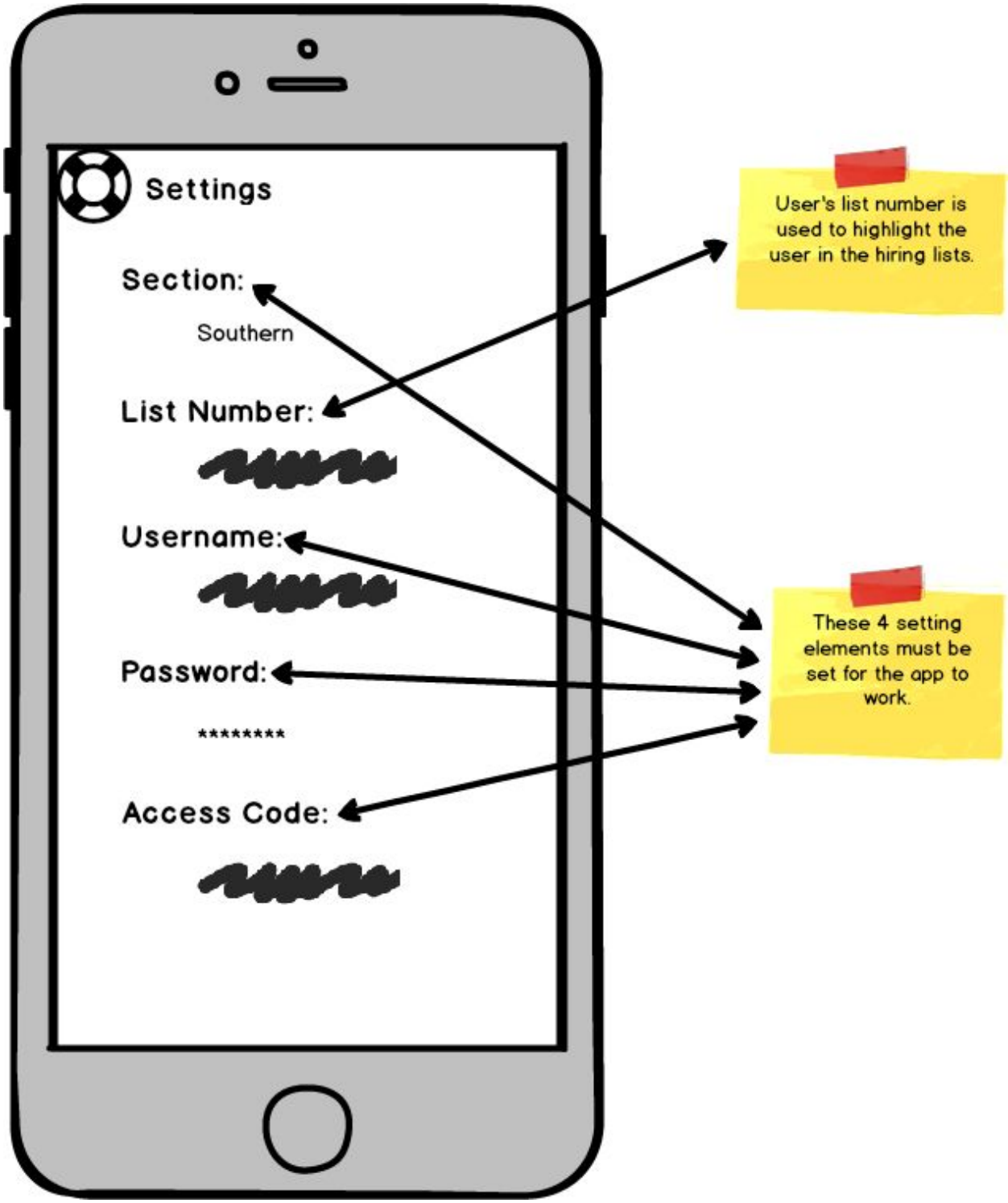
Calendar View

Replace the above image with your own mock [click on the above image, then navigate to Insert → Image…]
Provide descriptive text for each screen

Make Available View



Settings View



Key Considerations

How will your app handle data persistence?

I will build a Content Provider backed by a sqlite database. I will have one table to hold data for today and tomorrow's hiring lists. I will have a second table to hold the calendar data. Both will be loaded from the LA County website and saved to the database on a recurring basis.

App state and user preferences will be kept in a preferences file.

User preferences will be:

- Section preference (Southern, Central, Northern)
- List Number
- Username
- Password
- Access Code

App states will be:

- Available today
- HIred today, tomorrow
- Current date
- Place on today's hiring list

Describe any corner cases in the UX.

User can use the back button or the submit button to return to the MainActivity view if the user is currently on the Settings view or the Make Available view..

Describe any libraries you'll be using and share your reasoning for including them.

For example, Picasso or Glide to handle the loading and caching of images.

Describe how you will implement Google Play Services.

I will use either GcmTask or FirebaseJobDispatcher to schedule the loading of data from the Lifeguard website.
I will use identity to show the icon of the current user on the hiring list page.
I will use analytics to get statistics on app usage.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

You may want to list the subtasks. For example:

- Configure libraries
- Create database files to hold hiring list and calendar data
- Create a Task to load hiring list data for both today and tomorrow
- Create a parser to read in the hiring list data and load it into the database.
- Create a Task to load calendar data for the current month.
- Create a parser to read in the calendar data and load it into the database.

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for MainActivity
    - Build UI for Hiring List View
    - Build UI for Calendar View
- Build UI for Settings view
- Build UI for Make Available Form

Task 3: Calendar reload

Implement code reload the Calendar data when the hiring list data shows that we have been given a shift.

- Monitor updates to the Hiring list, if the user was available and now is not, then reload calendar data to reflect shift.

Task 4: Notifications

Notifications

- Create notification for successfully being made available
- Create notification for getting close to being hired (within 5 people)
- Create notification for being hired.

Task 5: Widget

Implement a widget to show today's hiring list.

- Build widget UI layout
- Build provider for widget
- Build service to update widget provider

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"