**How to Use this Template**

1. Make a copy [ File → Make a copy... ]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

**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**"

---

**GitHub Username**: hbaxamoosa

# HelpWanted

## Description

The app allows business owners to post “Help Wanted” ads, and allows job seekers to view these ads in the app.

Business Owners can post details about their job offering, including name of the business, location, wage rate, picture of the business.
Job seekers can view jobs (based on location), share jobs and send an email to the business owner to express interest in the job.

## Intended User

This app is intended for use by business owners that are looking to hire for open positions, and for use by job seekers that are looking for work.

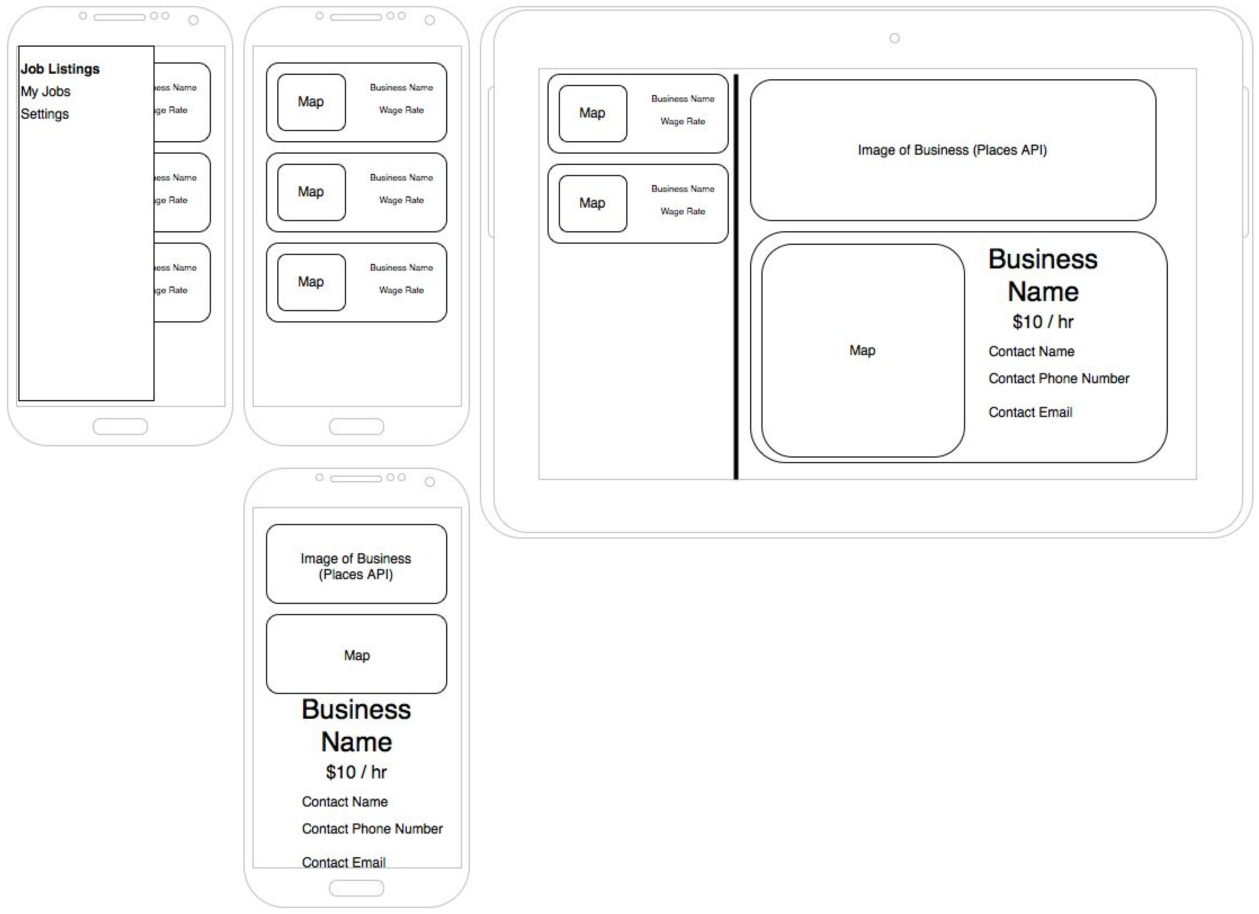
## Features

- Allows Business Owners to post one or more jobs. A job posting will include the business name, wages, location, contact name, contact email, contact phone.
- Allows Job Seekers to view job postings, by location.
- Allows Job Seekers to share a job posting.
- Allows Job Seekers to mark a job as a favorite.
- Allows Job Seekers to email the Business Owner.
- Allows Job Seekers to phone the Business Owner.
- Job postings will expire after 7 days. App will notify the Business Owner that posted the job that their job posting has expired. App will notify the Job Seekers that their favorite job posting has expired.
- App will allow Business Owners to create an Account (Google Account) to keep track of jobs they have posted.
- App will allow Job Seekers to create an Account (Google Account) to keep track of jobs they have favorited.
- App will display ads.

## User Interface Mocks

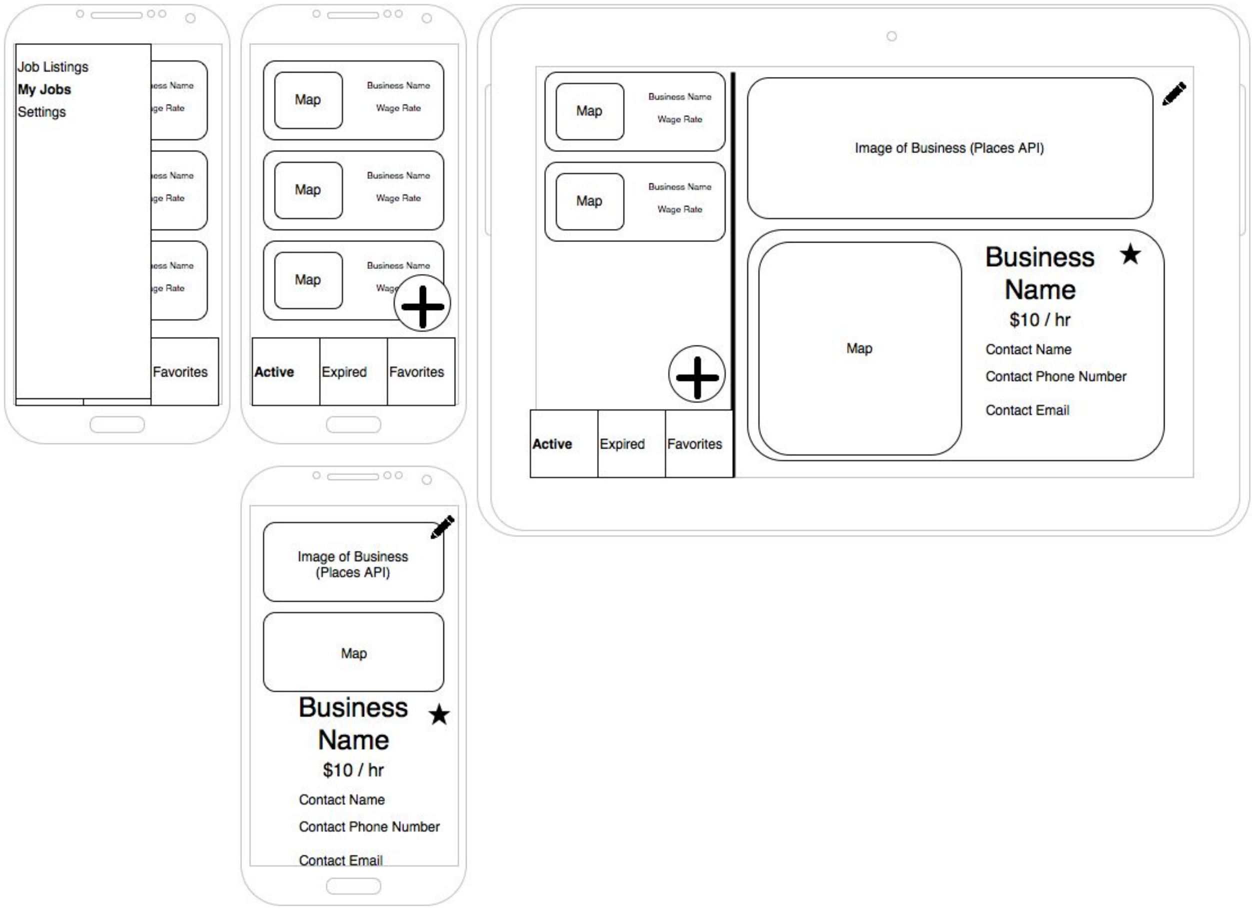
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

## Job Listings



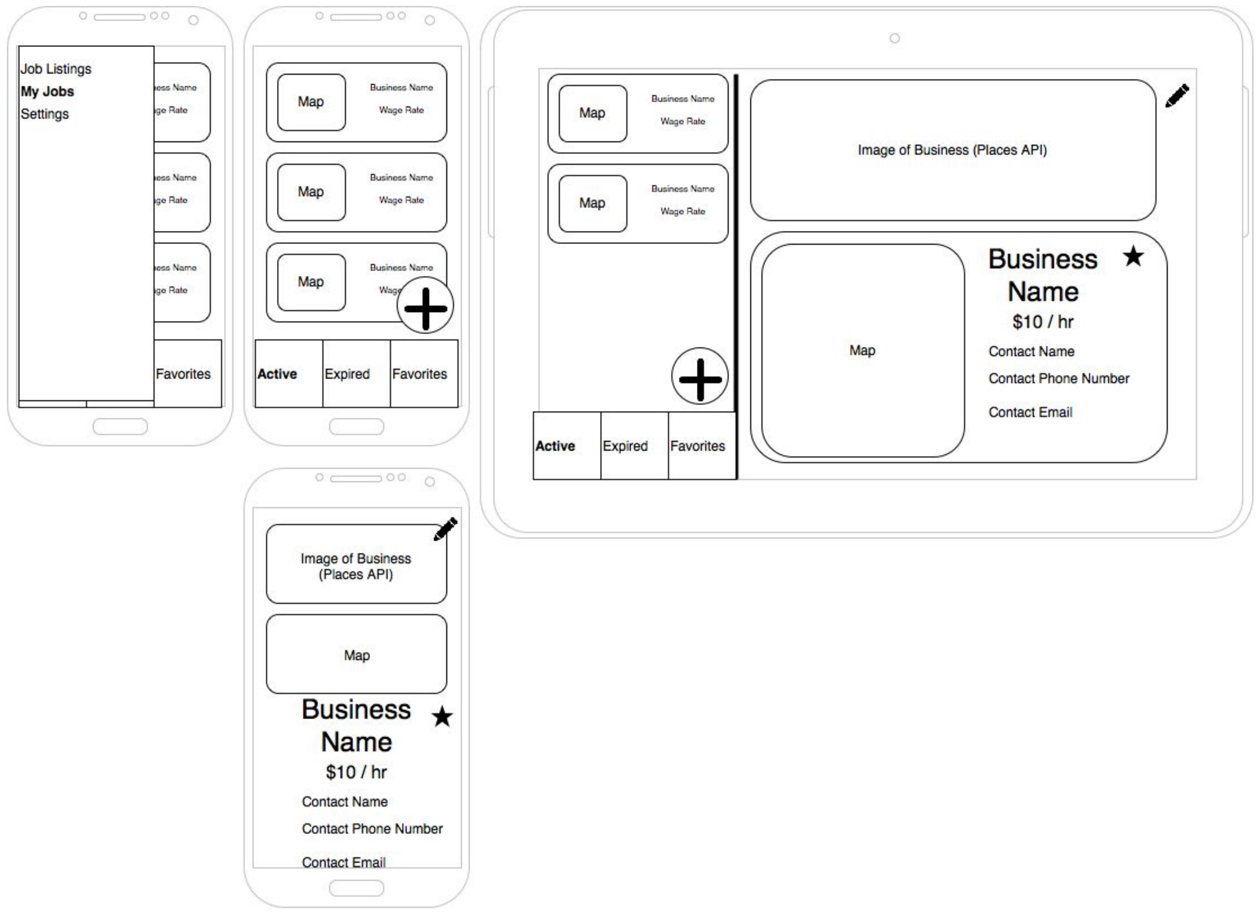
This will be the MainActivity view for both Business Owners and Job Seekers. This will show a list of active job postings for the given geolocation.

## My Jobs



This activity will have a tabbed navigation: Active, Expired, Favorites.
This activity will have a FAB to allow Business Owners to post a new job.

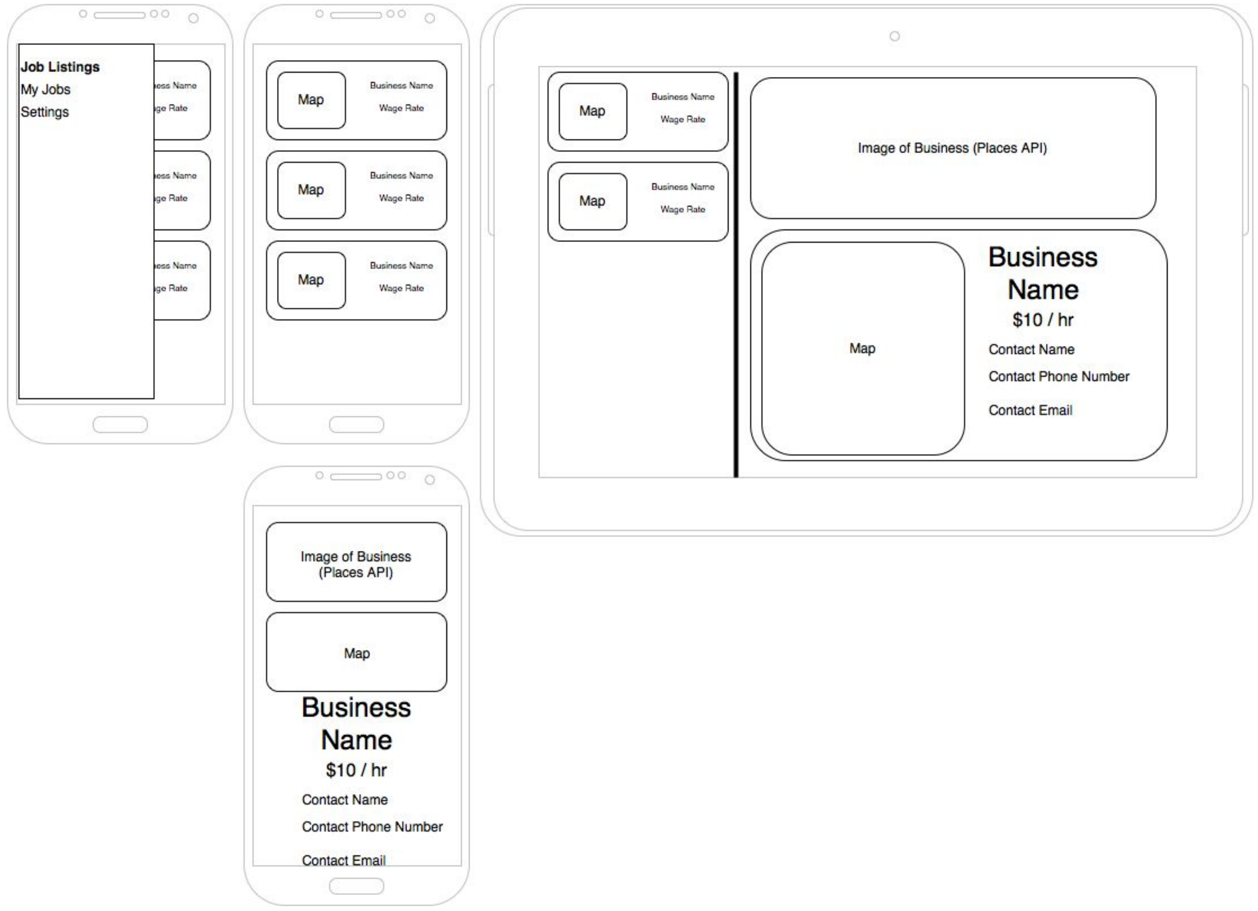
## Add/Edit Job



https://developers.google.com/places/android-api/placepicker#introduction

Allows the Business Owner to enter the following information about their job posting:
Name of Business
Address of Business (this should be clickable to view on a map)
Hourly Wage Rate
Contact Name
Contact Phone Number
Contact Email Address

### Job Post Details



Allows the Job Seeker to view the following information about a job posting:
Name of Business
Address of Business (this should be clickable to view on a map)
Hourly Wage Rate
Contact Name
Contact Phone Number
Contact Email Address

It should also allow the Job Seeker to share the job posting, favorite the job posting, send an email to the Contact person, and call the Contact person.

## Key Considerations

#### How will your app handle data persistence?

Describe how your app with handle data. (For example, will you build a Content Provider or connect to an existing one?) The app will handle data persistence by storing the job postings into a SQLite DB on the device, accessed via a Content Provider. The app will also save/retrieve job posting from the Cloud using GAE.

If the User uninstalls the app, all their data should be restored from the server when they reinstall the app, and use the same User account.

### Describe any corner cases in the UX.

The app should have have any corner cases in the UI. However, the experience for the Job Seeker and the Business Owner are going to be the same, i.e. the app doesn't know the intent of the User. Therefore even the Job Seeker will be the '+' FAB on the "My Jobs" screens.

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

I will use the following:

- Square's Retrofit - make requests from the Cloud (GAE) to retrieve jobs from the server.
- Jake Wharton's Timber - for debugging.
- Facebook's Stetho - for debugging.
- Google Play Services Location - to determine the user's location and show job lists for the same zip code.
- Google Play Services Identity - to determine the user's identity and keep track of their job postings and favorites.
- Google Play Services Analytics - to track usage within the app.

## 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

- Create the project in Android Studio.
- Add a Google Cloud Endpoints Module.

## Task 2: Laying the Foundations - Implement UI for Each Activity and Fragment

- Create an Application class.
- Build UI for Job Listings
- Build UI for My Jobs
- Build UI for Add/Edit Jobs
- Build UI for Job Post Details

Each activity will utilize a Fragment (in preparation for a tablet layout).

## Task 3: Complete the Google Cloud Endpoints Module

- Create a Java Library that represents the Job Post
- Create an Android Library
- Create the GCE Module and Endpoints

## Task 4: Wire the Android app to post/retrieve Job Posts from GCE

- Create the content provider. App will locally store all Job Posts retrieved for the User's zip code, favorited Job Posts and expired Job Posts (for Job Post submitters).
- Update the Android app to utilize an AsyncTask to post/retrieve Job Posts from GCE

## Task 5: Create a Sync Adapter and Notification

- Sync Adapter should query GCE on a regular interval to determine if any new Job Posts have been added for the User's zip code
- Generate a Notification for the User to inform them of the added Job Posts

## Task 6: Create the Tablet UI

- Create the Tablet UI.
- Update the Phone/Tablet UIs to display correctly in Portrait & Landscape.

## Task 7: Add Functional Tests

- Add functional test cases to verify that the backend of working as expected.
    - Verify that a Job Posting can be made
    - Verify that a Job Posting can be retrieved
- Add test case to insert multiple Job Posting into Content Provider
    - Verify that each Job Posting is being synced to GCE via Sync Adapter
    - Edit one or more Job Postings and verify that changes are stored

## Task 8: Handle Error Cases

- Handle error cases for network connectivity, etc.

## Task 9: Create build variants

- Create a Free (with ads) and Paid (without ads) build variants.
- Add an interstitial ad in the Free variant. Display the ad when the user selects a Job Post in the Listing view to go to the Details view.
- Add a loading indicator in the Paid variant. Display the loading indicator when the user selects a Job Post in the Listing view to go to the Details view.

Add as many tasks as you need to complete your app.

---

**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**"