



Imperium

Design Document

Team 15

Jeremy Potratz
CJ Enright
Anthony Popilov
Peter Reckamp
Jacob Szarzynski

Table of Contents

Purpose	4
Functional Requirements	4
Non Functional Requirements	5
Design Outline	6
View Controller Interaction	7
Controller Model Interaction	8
Design Issues	9
Functional Issues	9
Will users need to sign up for this service?	9
Options	9
Justification	9
How will job employers be matched to job-seekers?	9
Options	9
Justification	10
How will user profiles be displayed when searching for matches?	10
Options	10
Justification	10
Will users have a way to filter their job searches?	10
Options	10
Justification	10
Will users be able to link data from 3rd party applications?	11
Options	11
Justification	11
Nonfunctional Issues	11
What language should the backend be written in?	11
Options	11
Justification	11
What database should we use?	12
Options	12
Justification	12
What platform should our product release on?	12
Options	12
Justification	12
What hosting provider should we use?	13
Options	13
Justification	13

Design Details	14
Class Design	14
Description of Classes	14
Models	14
Views	15
Controllers	15
Sequence Diagrams	16
Initial sequence	16
Login attempt	16
Student/Employer “likes” another user	17
Student/Employer runs out of locally stored profiles	17
Navigation Flow	17
UI Mockups	19

Purpose

We are creating a tool to assist students who are searching for jobs or internships, as well as recruiters searching for students to fill those positions. We will be doing this by developing a web app that allows for students and recruiters to connect with each other and establish a relationship. In order to support our app we will have a backend running Node.js that will serve our static assets as well as service API requests. That server will be backed by Firebase, which will act as our database to store all user and application information.

Functional Requirements

1. User Account

As a user,

- a. I would like to be able to create an account for Imperium so that I can find a job or so that I can recruit
- b. I would like to be able to login and manage my Imperium account so that I can use Imperium as effectively as possible
- c. I would like to be able to reset my password if I forget it so that I won't be locked out of my Imperium account
- d. I would like to upload a photo of myself/my company so that companies can see more about me
- e. I would like to logout of my account so that others cannot potentially hijack my account
- f. I would like to delete my account so that I can no longer be associated with the service
- g. I would like to have the option to remember my password for easier login

2. Student Account

As a student,

- a. I would like to upload my resume so that companies can learn more about me
- b. I would like to create a bio about myself so that companies can quickly understand who I am
- c. I would like to create job specific preferences about what I want so that companies can more effectively find me
- d. I would like to filter companies based on job listings
- e. I would like to specify a radius of where I am interested in having a job
- f. I would like to specify whether I want a job, internship, or co-op
- g. I would like to specify that I am interested in jobs for more than 1 major

3. Employer Account

As an employer

- a. I would like to have a main page for my company if time allows

- b. I would like to create sub groups for specific majors that I am interested in so that I can more effectively find potential employees if time allows
 - c. I would like to create a bio for the specific job/internship
- 4. Student Actions
 - As a student,
 - a. I would like to be able to view employer profiles so that I can see what companies are hiring
 - b. I would like to be able to like employers that I'm interested in so that I can show interest in those employers
 - c. I would like to chat with employers who I have matched with so that I can talk to companies where there is a mutual interest
 - d. I would like to 'favorite' up to 3 companies to show that I am extra interested
 - e. I would like to unmatched with an employer so that I can adjust my interests over time
 - f. I would like to view a history of past matches of employers
 - g. I would like to "match" with an employer who has viewed my profile
 - h. I would like to send a calendar that has all my availability windows in the chat
 - i. I would like to be able to see when an employer has favorited me
- 5. Employer Actions
 - As an employer,
 - a. I would only want to give certain people access to certain sub groups that they are in charge of if time allows
 - b. I would like to be able to navigate between subgroups if time allows
 - c. I would like to search through students in specified subgroups if time allows
 - d. I would like to "match" with a student who has viewed my profile
 - e. I would like to view student profiles so that I can get a better understanding of a student
 - f. I would like to be able to like students that I'm interested so that we can begin to build a professional relationship
 - g. I would like to chat with students I've matched with so that I can find employees to hire
 - h. I would like a history of past students I matched with
 - i. I would like to easily schedule interviews with matched students so that I can continue the relationship past our initial meeting
 - j. I would like to unmatched with a student so that I can better keep track of students I'm interested in
 - k. I would like to hide my profile from students for when positions are full
 - l. I would like to be able to see when a student has favorited me
 - m. I would like to remove job listings if the position has been filled
- 6. Notifications
 - As a user,
 - a. I would like to get desktop notifications when i get matched or have new messages if time allows

- b. I would like to receive email notifications that can give me updates about my profile if time allows
- c. I would like to receive a notification when another user favorites me
- d. I would like to disable certain notifications as I please if time allows

Non Functional Requirements

1. Performance

As a developer

- a. I would like the website to support thousands of users
- b. I would like the website to act fast between pages and have communication be instant
- c. I would like the website to be bug free so that users are not inconvenienced with crashes

2. Server

As a developer

- a. I would like the application to support real time server client communication
- b. I would like the application to store X students and X employers in the database
- c. I would like the application to interface swiftly between Firebase and Google Cloud
- d. I would like the application to be able to store both employer and student data.

3. Appearance

As a developer

- a. I would like the website to look visually appealing and aesthetically pleasing
- b. I would like the user to be able to switch over to a night theme if time allows

4. Security

As a developer

- a. I would like to verify that they applicant is a valid employer to avoid fake people from creating employer accounts
- b. I would like to limit one account per email to help avoid one person creating multiple accounts if time allows
- c. I would like the password to be hidden when typing it in for login
- d. I would like passwords to be at least longer than 7 characters and contain a number to ensure that user accounts are secure

5. Usability

As a developer

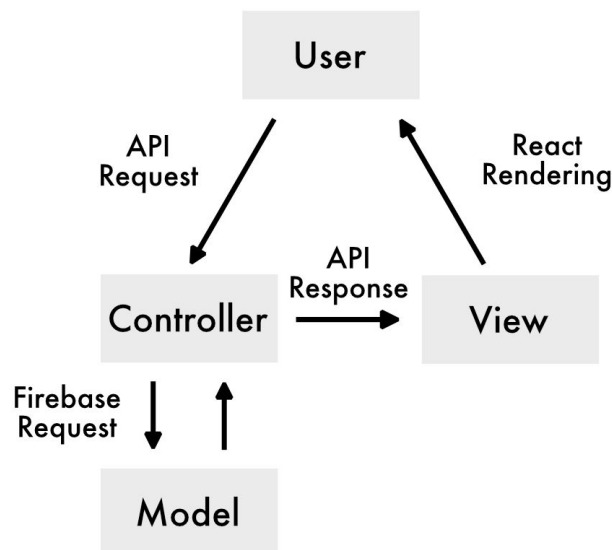
- a. I would like to access Imperium on any device regardless of size or resolution so that I don't need a desktop to use our product
- b. I would like to be able to easily navigate through the website so that I can efficiently find jobs or potential employees

Design Outline

We will be following the Model View Controller (MVC) architecture for this project in order to ensure maintainability and ease of development. Users will make API requests to controllers running on our backend which will in turn query our models stored in our Firebase database to either update or read information. That information will then be sent back to the user as a response, with which the view may be updated depending on the response.

Often in MVC architectures, views are HTML templates that are rendered server side and served to the user. For our product all rendering of data into UI will take place client side using React. This will allow us to speed up transfer times and reduce total load on the server which will help us achieve our performance goals set in our product backlog (more detail on this will be in the design issues section).

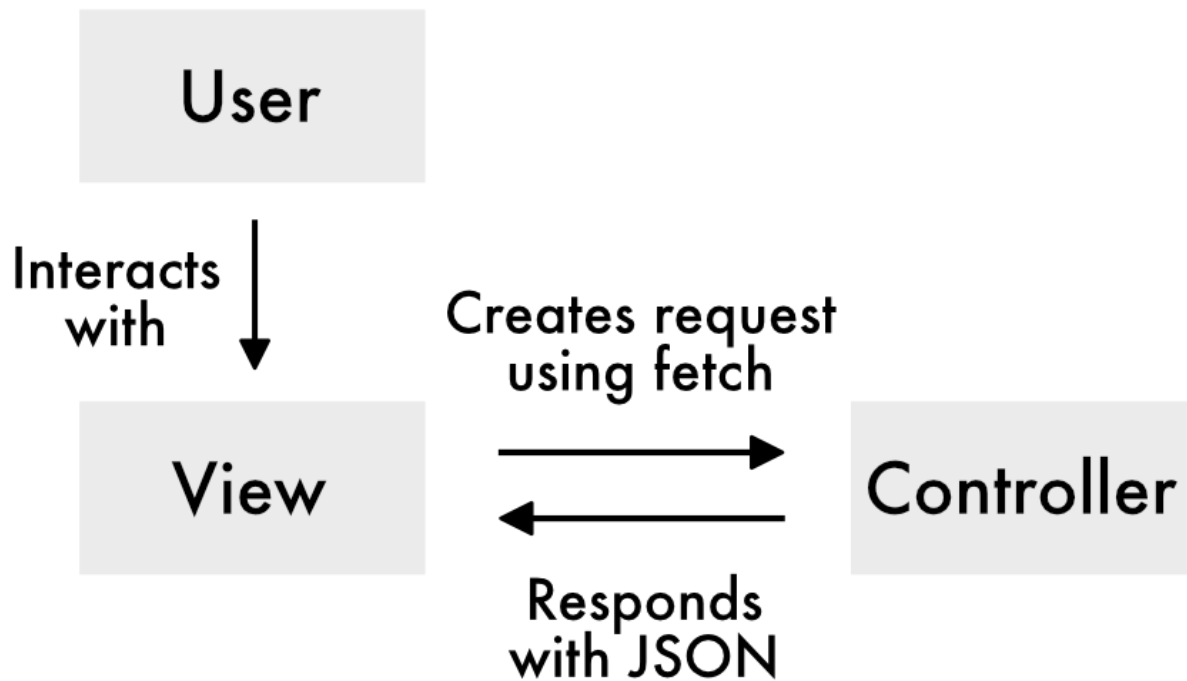
At a high level, the general structure of our application will look like this:



View Controller Interaction

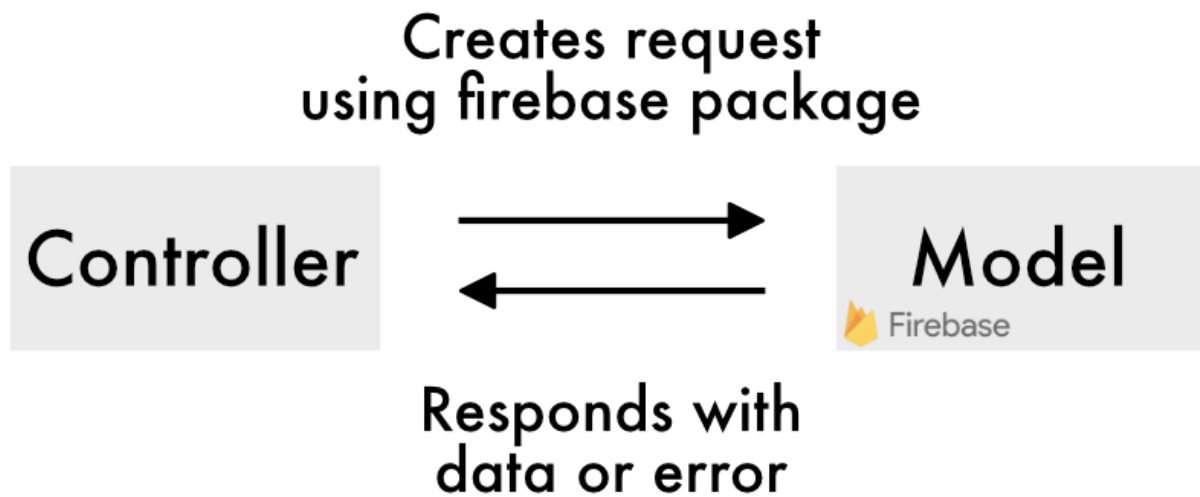
Requests will be generated by the user interacting with the view. Those interactions will begin HTTP requests to our backend server using React's `fetch` function. Requests sent from the view will use JavaScript Object Notation (JSON) to convey any state data that the backend might need to service a request. For example, the view might send its current page number (if viewing a list) so that the backend will know how much to offset its query by. After the controller queries the database (detailed in the next section), it will send a JSON response to the view.

Based on that JSON response, the view may update itself to reflect that new information. A slightly more concrete overview of that looks like this:



Controller Model Interaction

Controllers interacting with models will generally have two cases: reading data and writing data. Both of these operations will execute SQL queries on our Firebase database in order to achieve whatever the desired effect may be. In cases where the controller will return a response to the view, the data received from the database may need to be cleaned up or trimmed and will always need to be converted into JSON. To simplify our interactions with Firebase, we will be using the firebase package from the Node Package Manager (npm) and will interact with the API that it exposes. Again, a slightly more concrete overview of that looks like this:



Design Issues

Functional Issues

Will users need to sign up for this service?

Options

- Login using a 3rd party source such as Facebook or Google
- No login necessary
- Signup with email and username directly through Imperium

Justification

We chose to have users sign up directly through Imperium. Users will have to sign up using their email and by creating a password for their Imperium account. Following this, the user will have to specify whether they are a recruiter seeking students to hire, or vice versa; a student seeking for a potential job opportunity. Having users sign in provides a convenient means for storing data and keeps everything locally accountable. The user will further have to fill in more criteria in order to configure their account to their liking, but logging in directly through Imperium will be crucial.

How will job employers be matched to job-seekers?

Options

- Implement a swipe method
- Implement the usage of multiple types of buttons
- Implement a 'drag and drop' method

- Implement a 'double-click' method

Justification

We chose to implement the multiple button system, as it is the easiest and most convenient for the user. As far as UX, buttons are easily accessible, straightforward, and can be satisfying to click on. A rudimentary design would include "Like" and "Pass" options, as job seekers search amongst job employers and vice versa. The click of a button is the perfect means for connecting two groups of users searching amongst each other.

How will user profiles be displayed when searching for matches?

Options

- A name, image, and a link to a LinkedIn account
- A name, image, and a bio dictating exactly what the user is searching for
- A link to the user's LinkedIn account
- A name, image, bio, and an optional link to LinkedIn

Justification

We chose to implement a name, image, bio, and an optional link to LinkedIn. LinkedIn, although not necessary, would be a crucial means in order to ensure user credibility. Otherwise, a user can simply sign up as whoever they please and write down any arbitrary skills they may or may not have. An image is necessary and should likely be a professional headshot of the user. A bio can be more informal and state the user's mission statement or something about themselves. This app should not be based on LinkedIn, but should allow for the use of LinkedIn as a useful resource. A user's profile should be a quick summary of their necessary information, of which this method captures perfectly.

Will users have a way to filter their job searches?

Options

- Based on location
- Based on university degree
- Based on specific company
- Based on skills they have/are looking for

Justification

We chose to filter job searches using a combination of all the above methods. There are many factors to decide when looking for a job or looking for someone to hire for a job. Location is crucial, as most companies want employees to come into the office daily. Degree is obviously necessary, too, as this will be a key method for employers to configure what kind of employees they are searching for. Furthermore, students should be able to search for specific companies

they'd like to apply for as well. On both sides, users will also convey which skills they have (if they are a student) and which skills they are searching for (if they are a job recruiter)

Will users be able to link data from 3rd party applications?

Options

- Link data from Facebook
- Link data from LinkedIn
- Link data from Twitter
- No link to any other 3rd party applications

Justification

We chose to implement linking data from LinkedIn, as this is a professional software and should not use data from social media. Although Facebook and Twitter can have viable information for job recruiting/searching, it makes the most sense to use LinkedIn, which is already in wide-use and commonly accepted in the industry. Users will be suggested to have some sort of LinkedIn profile as a reference in order to verify information put into Imperium. Otherwise, it would be easier for random users to sign up on Imperium and lie about their experience and the company they are associated with.

Nonfunctional Issues

What language should the backend be written in?

Options

- Ruby
- Python
- Java
- JavaScript
- Go

Justification

We decided to write our backend in JavaScript mainly for ease of development. It has syntax that is similar to languages we have been exposed to in CS classes at Purdue and it allows us to use the same language for our frontend and backend so everyone can be involved with both sides. Languages like Ruby and Python have pretty wildly different syntaxes and idioms, so while those language are very popular they are not the best choice for a time constrained project like ours. Java and Go look pretty familiar to use, but they are usually considered heavy weight languages that can become unwieldy in the smaller scale we're dealing with. Java and Go are also usually more performant than JavaScript, however, JavaScripts performance will already be more than enough for the scale we're dealing with and will allow us to iterate quicker than any other option.

What database should we use?

Options

- PostgreSQL
- MySQL
- AWS RDS
- MongoDB
- Firebase

Justification

We ended up choosing Firebase mainly due to the features it had over the other options. There were two main categories from that list above, hosted databases and databases we would have to host ourselves. By choosing a hosted database we have a much greater uptime guarantee as professionals are taking extreme care to minimize down time. We also chose Firebase because it allows for some abstractions away from the raw database level which will make it easier for us to conceptualize what's happening in our models. Finally, we chose Firebase because it has a free tier that should be able to service us even after this project has come to an end.

What platform should our product release on?

Options

- iOS
- Android
- Windows
- MacOS
- Web

Justification

Our main reason for choosing the web as our platform was because it has the widest reach of all platforms listed above. That is, all platforms listed above can use a web app. Compare that to, for example, how an iOS app can only be used on iOS and none of the other platforms. What also helped us come to this conclusion was the total experience our group had with each of these platforms. We had the most overlap of skills within web, so it was the natural choice. Finally, we had to consider our own access to the platforms. In order to develop for iOS you have to have a Mac and an iPhone. For windows you have to have a windows computer, and same for MacOS. We determined that, regardless of personal technology owned, we could all work on a web application without any barriers to entry.

What hosting provider should we use?

Options

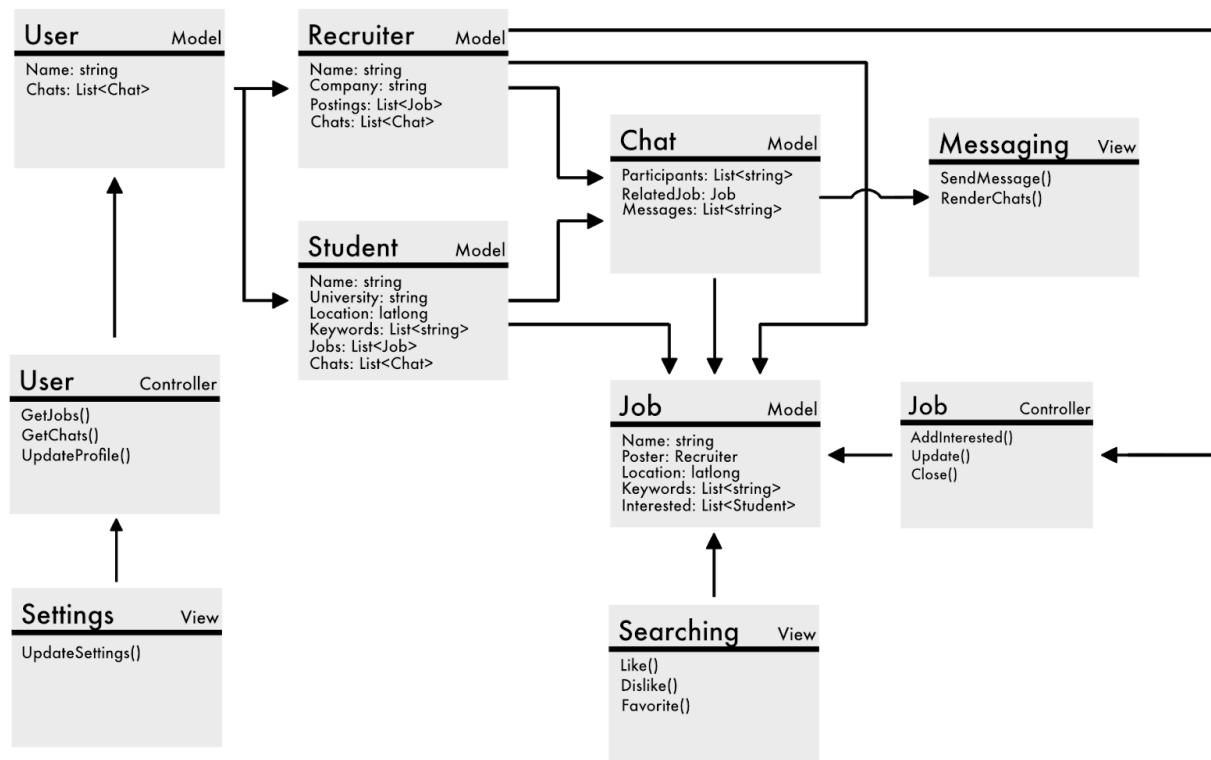
- DigitalOcean
- AWS
- Google Cloud

Justification

We chose Google Cloud for three reasons: cost, experience, and features. Google Cloud provides each user with \$300 of credit for free, so we should be able to host our project for quite a while without it costing us. Other platforms had free tiers as well, but none were as flexible or generous. We also had two group members who have deployed projects to google cloud before, but no one had deployed on to the other platforms. This led to us choosing the platform we knew the best. Finally, we chose Google Cloud because our database, Firebase, is also a Google product. This means that there are some features shared between the two that make development using both a bit easier.

Design Details

Class Design



Description of Classes

All of our models will be represented by tables in our Firebase database, with each individual item being a single row.

All of our Views will be written in JavaScript using React and will be as modular as possible to enable code reuse wherever possible.

All of our controllers will be written in JavaScript running on Node.js and will query our Firebase database upon receiving a request.

Models

- **User**: A user can either be a student looking for a job or a recruiter which will be denoted by a `Role` field. A student user will have a field holding the jobs that they've interacted with as well as personal info like a transcript, bio, and a résumé. Along with that students will have the ability to add various cosmetics including XXXXX (like a head shot). A

recruiter user will instead have a field of jobs they've posted and students they have interacted with. Other than that, the employers will be able to add a bio and facts about the company they are representing. Both will have a conversations field so that students and recruiters can communicate with each other.

- Job: Each job belongs to a recruiter user and will keep track of students interactions with it. It will also hold information relevant to the job like requirements, location, and company details.
- Message: All messages between recruiters and students will be kept track of in order to keep the message data persistent between the two parties.

Views

- Index: The index will be the first screen a logged in user sees when opening the app. From the index screen the user can either search for applicants or jobs (depending on if they're recruiters or students) or continue conversations they've started with other users. As a user, they will also be able to edit their personal information and other relevant settings.
- Searching: When a user wants to search for applicants or jobs they will use the searching view. The searching view will show jobs that are relevant to the user that it receives from the Job controller. The user's will also have the ability to reduce the search size by restricting search options.
- Messaging: When a user and an employee match. They will be brought to a messaging view that enables them to send and receive messages. The view will also give various options between the two users. These options will include deleting messages as well as sending pictures or link attachments.
- Settings: If a user would like to alter their personal settings. They can add or change their bio, personal picture, or résumé. Other than that it will give the option to the user to change the range of what their desired job is. This will include things like job type, distance, and experience needed.

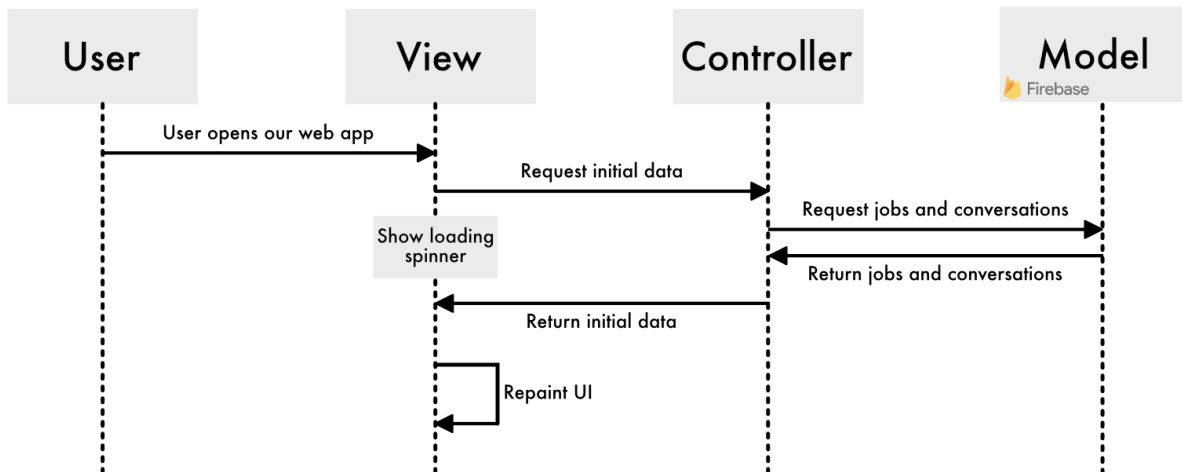
Controllers

- UserController: The user controller will handle all actions pertaining to users and will occasionally interact with jobs stored in the database. For example, when a user logs in their request will be routed to and handled by the user controller. The controller will also handle things like updating a user's profile and settings, as well as getting jobs for the user to search through. It will get these jobs by interacting with the database in a similar way to the job controller.
- JobController: The job controller handles the creation, reading, updating, and deletion of any given job. Each time one of those requests is made, the job controller will be checking it's poster field to make sure that no unauthorized edits are made. When a student likes a job, they will be added onto the "interested" list owned by the job. This allows our database to respond to a recruiters request to see interested students much

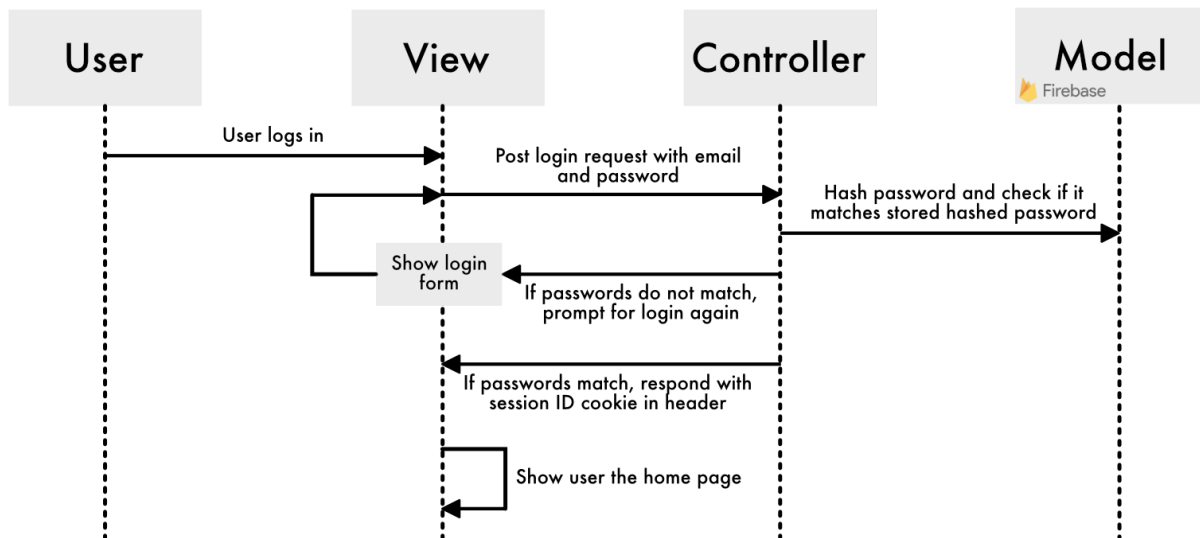
faster (we only have to look at that field in the Job, instead of going through every user and checking if they have that job in their like list).

Sequence Diagrams

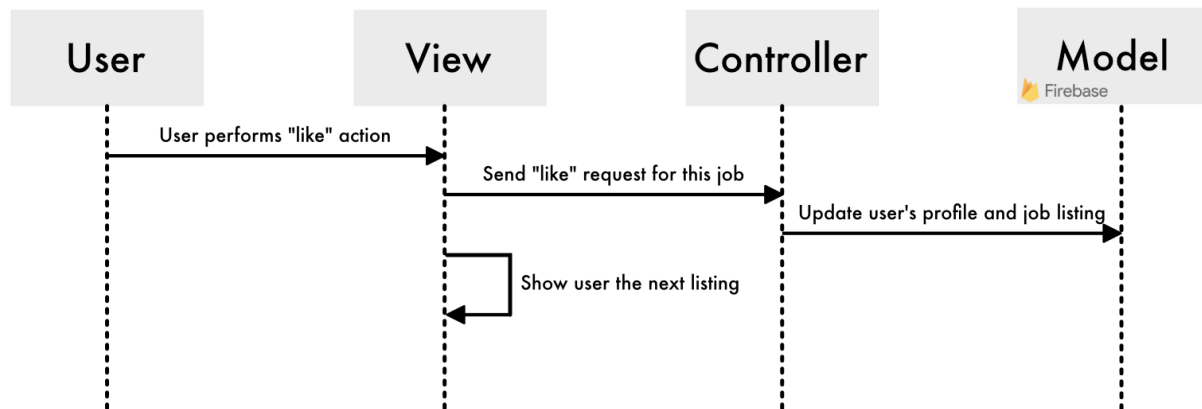
Initial sequence



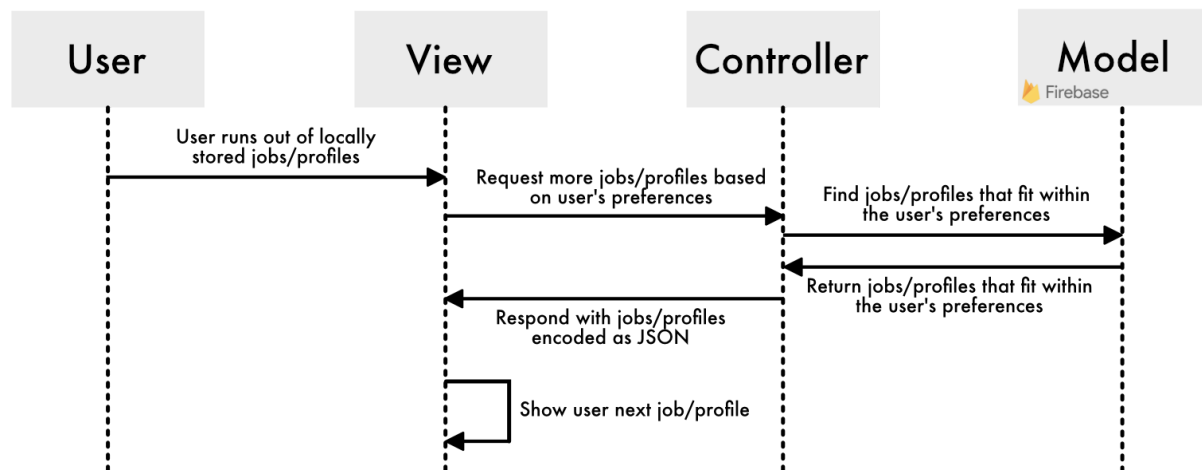
Login attempt



Student/Employer “likes” another user



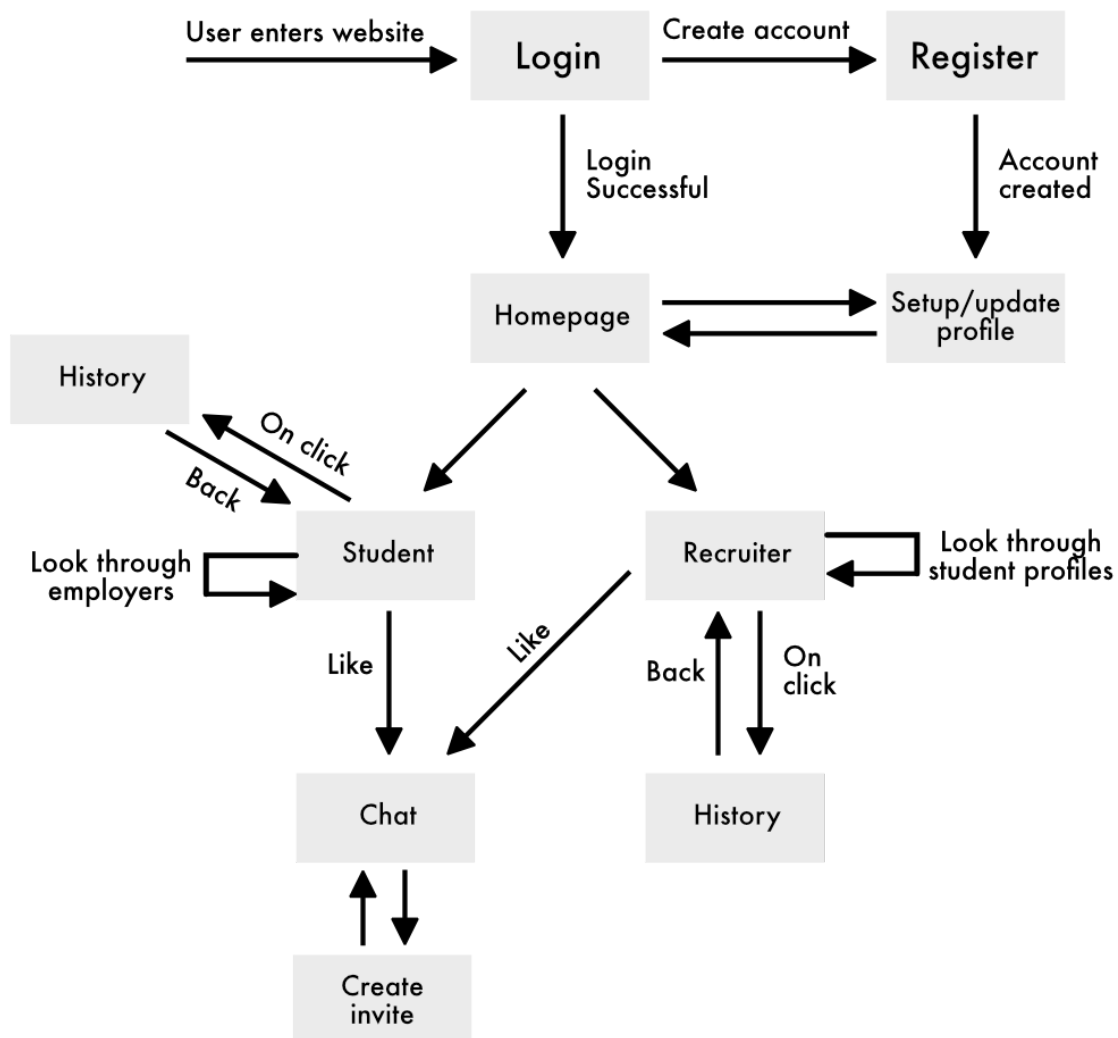
Student/Employer runs out of locally stored profiles



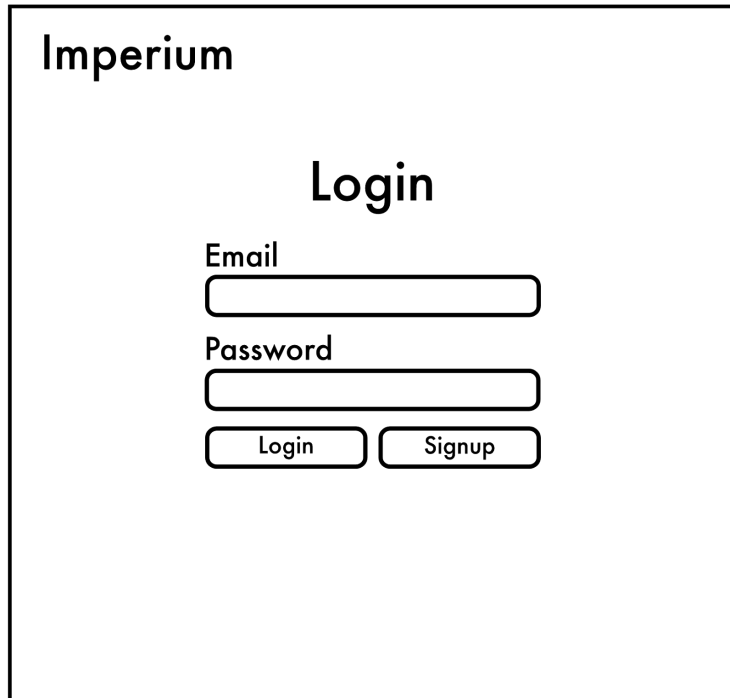
Navigation Flow

Our design view demonstrates the simplicity and ease of access of the app. All users start at the login screen. They either login in or create a new account. Upon logging in they will either go to a student homepage or an employer homepage. From the homepage they can then start looking for matches. Also from the homepage the user can go back to there settings and adjust them as they need to. The final thing the user can do on the homepage is look at their past

history. The user can then navigate over to there matches and start chatting with employers via a text chat. If both parties are interested they can send invites and availability windows to each other. The user at anypoint can logout of their account and navigate between the homepage and the matches/chat section of the app.



UI Mockups



The mockup shows a login page for 'Imperium'. It features a title 'Login' centered above two input fields labeled 'Email' and 'Password'. Below the password field are two buttons: 'Login' and 'Signup'.

Imperium

Login

Email

Password

Login Signup

This is the login page, simply enter your email and password to login. Password will be blocked out with dots. New users can click the signup button will take them to the registration page to create a new account.



Google

We're looking for motivated computer science students to join our Mountain View office this summer.

Field: Computer Science
Location: Mountain View, CA
Experience: None required
Salary: \$12

...



CJ Enright

I'm a student studying computer science at Purdue focusing on software engineering and systems programming. I'm looking for Work within 10 miles of here.

University: Purdue
Location: West Lafayette, IN
GPA: Good enough :)
Experience: Intern at Google

...



These are the main pages for students (left) and employers (right). The pages card actions behave similarly: click the check mark button to like someone's page, the X button to dislike someone's page, click the star in the middle to show special interest in a particular card (while simultaneously liking them). Users may only get 3 of these at a time. To navigate to the setting simply click the cog wheel button at the top right and that will take you over to the setting page. To Navigate to matches the user can use the hamburger bar on the top.

Matches

Google	→
XoomDat	→
StackOverflow	→
Microsoft	→
Company	→



Hey hi howdy

Howdy partner



Say something...



Once the user has navigated over to matches they can select a match and start chatting with whom they matched with. If the user ever want so unmatched they can click on the setting icon on the top right and go from there. In addition if a user would like to send a calendar invite in the chat this can be done through the settings. To get back to the main page at anytime they can simply go through the hamburger bar.