



CS 30700
Design Document

Team 2:

Noah Alderton

Joseph Barr

Derek Nobbe

Nicholas Raphael

Melissa Ruiz Ruiz

Emma Wrege

Index

• Purpose	3
○ Functional Requirements	
○ Non-functional requirements	
• Design Outline	7
○ High Level Overview	
○ Detailed Overview	
○ Sequence of Events Overview	
• Design Issues	9
○ Functional Issues	
○ Non-functional Issues	
• Design Details	12
○ Class Design	
○ Description of Classes and Interaction between Classes	
• Sequence of Events Overview	16
• Navigation Flow Map	19
• UI Mockup	20
○ Login Page	
○ Jobs Page	
○ New Job Wizard	
○ User Profile Page	

Purpose

Searching for jobs and internships provides a unique set of problems for an individual. Struggling to find work can be a very stressful situation purely based on the need of income to survive. Furthermore, the job search involves keeping track of innumerable variables about the various companies where one might want to work, which only piles onto existing anxiety. Thus, workers require a tool to do some of the legwork of job applications and to keep the sea of information organized so it can alleviate some of the monumental stress.

The purpose of this product is to develop a web application useful for keeping track of job/internship applications. Similar applications, such as JobTrack, already provide a good amount of the functionality Officium will provide: inputting, storing, and sorting various information about the companies to which one applies. However, our application will extend these capabilities to perform various calculations and retrievals of information to give the user vastly more insight into which companies could be right for them, all while keeping their applications organized for them.

Functional Requirements

1. Users can register for Officium with any current account or make a new one
 - As a user, I would like to be able to register for Officium with my email, Google, Facebook, Twitter, or GitHub
 - As a user, I would like to be able to logout.
 - As a user, I would like to be able to change my password.
 - As a user, I would like to receive a confirmation email when an account was created.
 - As a user, I would like to be able to recover my account after forgetting my password.
2. Users can enter in the specifics of the internships/jobs they applied for
 - As a user, I would like to be able to store the company name.
 - As a user, I would like to be able to store the job title of the application.
 - As a user, I would like to be able to store the location of the job/internship.
 - As a user, I would like to be able to store recruiter contact information.
 - As a user, I would like to be able to store how I applied to the job/internship.
 - As a user, I would like to be able to store the status of my application.
 - As a user, I would like to be able to store if I interviewed for the position.
 - As a user, I would like to be able to store any references used for a position.

- As a user, I would like to be able to store if I received an offer.
- As a user, I would like to be able to store the status of my applications.
- As a user, I would like to be able to store the link to the employers' portal.
- As a user, I would like to be able to delete a particular application.
- As a user, I would like to be able to store my credentials for the application portal.
- As a user, I would like to be able to modify company information.

3. Users can sort their applications in several ways

- As a user, I would like to be able to sort companies alphabetically.
- As a user, I would like to be able to sort companies by my level of optimism.
- As a user, I would like to be able to sort companies by my level of enthusiasm.
- As a user, I would like to be able to sort by due date.
- As a user, I would like to be able to sort by geographic proximity.
- As a user, I would like to be able to sort by cost of living.
- As a user, I would like to be able to rank companies by preference.
- As a user, I would like to be able to sort by estimated pay rate from the Glassdoor API.

4. Users can see and store personal specifications about their applications

- As a user, I would like to be able to rate my optimism for a particular application.
- As a user, I would like to be able to rate my enthusiasm for a particular application.
- As a user, I would like to be able to see the physical distance to a particular location.
- As a user, I would like to be able to see the cost of living for a particular location.
- As a user, I would like to be able to weigh my priorities for how to select a job.
- As a user, I would like to be able to insert notes for particular companies.
- As a user, I would like to be able to modify notes.
- As a user, I would like to be able to view all my contacts.

5. Users can store documents in their Officium account

- As a user, I would like to be able to upload my resume.
- As a user, I would like to be able to update my resume on file.
- As a user, I would like to be able to remove my resume on file.
- As a user, I would like to be able to upload my cover letter.
- As a user, I would like to be able to update my cover letter on file.
- As a user, I would like to be able to remove my cover letter on file.
- As a user, I would like to be able to store pictures of business cards.

6. Users can report any issues with the web application
 - As a user, I would like to be guided through the user interface.
 - As a user, I would like to be able to get help for the user interface.
 - As a user, I would like to be able to report bugs with my experience.
7. Users will be able to utilize various mediums to see different information about their applications
 - As a user, I would like to be able to generate a Sankey diagram for my applications.
 - As a user, I would like to be able to see Glassdoor reviews of a particular employer.
 - As a user, I would like to be able to see estimated pay rates for a particular employer from Glassdoor.
 - As a user, I would like to be able to see a map of the company locations using Google Maps.
 - As a user, I would like to be able to export due dates into a CSV file.
 - As a user, I would like to be able to export due dates into Google Calendar.
 - As a user, I would like to be able to export contacts as a vCard.
8. Users will receive email updates about their applications
 - As a user, I would like to receive timely emails when an application deadline is approaching.
 - As a user, I would like to receive emails when I have not heard from a company in a long time.

Non-functional Requirements

1. Performance

- As a developer, I would like Officium to be responsive so I can use it on various devices.
- As a developer, I would like the site to run smoothly with no bugs.
- As a developer, I would like the server to be able to handle at least 100 clients at a time.
- As a developer, I want the site to be able to handle errors gracefully and not crash

2. Server

- As a developer, I want the server to support the client and carry information to and from the interface in a timely manner.
- As a developer, I want the server to be able to handle multiple client requests with ease.
- As a developer, I would like the server to store information in an organized and easy-to-access way.
- As a developer, I would like the server to handle requests to an API in an efficient manner.

3. Appearance

- As a developer, I would like the appearance of the site to reflect a business aesthetic and be pleasing to the eye.

4. Security

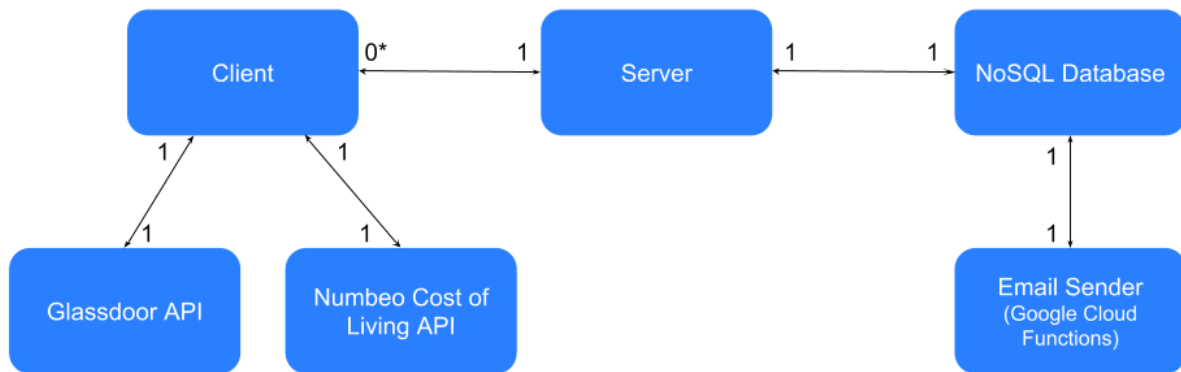
- As a developer, I want any data collected by the user to be secure.
- As a developer, I would like the store user credentials safely in the server.

5. Ease of Use

- As a developer, I want the site the be user friendly and self-explanatory.
- As a developer, I want the site to be able to be run on a web-browser.

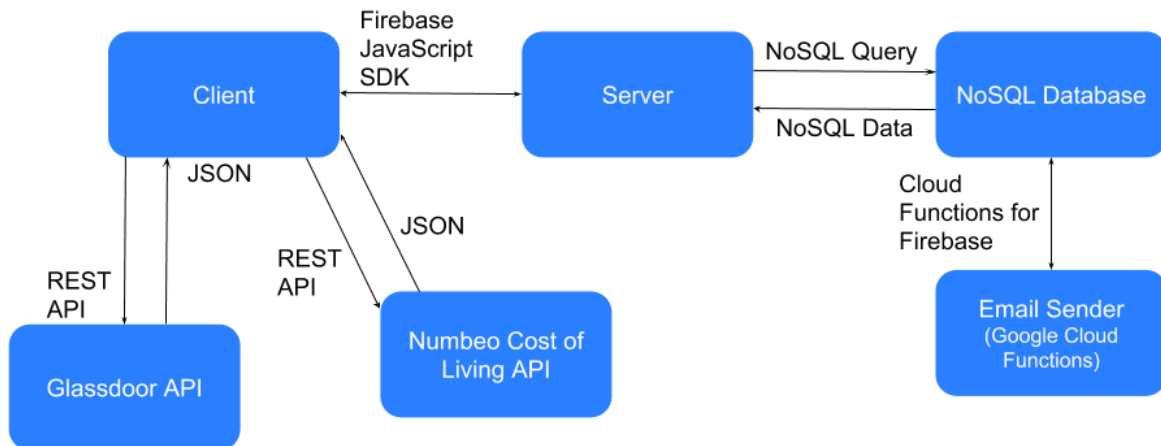
Design Outline

High Level Overview



Officium will have many simultaneous clients that will use the same single Firebase for the backend. The clients will request existing data about the client and the client will send new data or update existing data. At that point, Firebase will reference its NoSQL Database that stores all user information. When the client needs company reviews or typical salaries, it will request data from the Glassdoor API. Additionally, when data on a location's cost of living is necessary, a Numbeo API call will be completed. Finally, the email sender Google Cloud Function will reference the database on a daily basis, and send out emails when users have not updated an application in a while, or if a response date has passed. Additionally, emails will be sent out whenever a new account is created on the database to welcome the new users, and when passwords need to be reset.

Detailed Overview



The React client communicates the the backend using the Firebase JavaScript SDK. Much of the backend itself is abstracted from the developer, as Firebase is known as a “backend as a service.” Firebase uses NoSQL as its database. Furthermore, the NoSQL database communicated with Google Cloud Functions, which are various functions that can interact with the database. These functions can be set to run periodically or during certain circumstances like creating a new database entry. The Glassdoor API is used to retrieve company reviews and expected salaries. Numbeo’s API retrieves the cost of living of for cities across the globe. It is going to be used to compare the various workplace locations. Both APIs utilize REST API to request data, and return JSON.

Design Issues

Functional Issues:

1. What information is collected upon account setup?

- Options
 1. Name, Email, Password
 2. Name, Username, Email, Password
 3. Name, Username, Email, Password, Phone Number
- Choice: Name, Email, Password
- Justification

We decided that it would be unnecessary to implement a username system, as the users aren't interacting with one another. Usernames are also more difficult for users to remember compared to their email address. Additionally, phone numbers are unnecessary, as Officium doesn't send text message, make phone calls, or implement 2-factor authentication.

2. What external accounts should also allow for Officium login?

- Options
 1. None
 2. Facebook, GitHub, Google, and Twitter
 3. LinkedIn, Facebook, GitHub, Google, and Twitter
- Choice: Facebook, GitHub, Google, and Twitter
- Justification

It is important to allow users to log into services using preexisting accounts, as it limits the number of passwords that they need to remember. Users tend to be more likely to use a service if they can use an account they already have, instead of making a new one. Firebase allows for authentication with Facebook, GitHub, Google, and Twitter. While it would be nice to implement LinkedIn integration, it would be difficult and unnecessary, especially as most users that have a LinkedIn account likely also have an account with one of the other available services.

3. Should employer accounts be implemented?

- Options
 1. Accounts for individuals seeking employment
 2. Accounts for individuals seeking employment and employers
- Choice: Accounts for individuals seeking employment
- Justification

During the brainstorming phase for Officium, we discussed allowing companies to make accounts and interact with users. However, we decided that this would be outside the scope of this project. Officium is primarily designed for individuals looking to better manage their job search. It isn't designed to help match potential employees with available jobs. There already exist other services that allow potential employees to interact with employers like LinkedIn and AngelList.

4. How should salary information be retrieved?

- Options
 1. User Inputted
 2. Glassdoor API
- Choice: Glassdoor API
- Justification

Another concept that was brought up during brainstorming was how to compare salaries for potential jobs. Most of the information about the job application is collected from the user, however salary isn't often mentioned in job applications themselves. Therefore, we decided to find another way to collect data about the expected salary for each applied job. Glassdoor has that data available for lots of positions, so we decided to use their API to match that data with the user's applied positions.

Non-functional Issues:

1. What frontend framework should be used?

- Options
 1. React
 2. Angular
 3. Vue
- Choice: React
- Justification

The decision to use React was primarily motivated based upon past experience. Half of our group members have experience with React for web app development, so we decided it was best to stick with a framework that we were familiar with to limit time spent learning a new framework.

2. What backend-as-a-service should be utilized?

- Options
 1. Firebase
 2. CloudBoost
 3. Back4App
- Choice: Firebase
- Justification

We immediately eliminated CloudBoost, because they did not offer a free tier. We decided to go with Firebase over Back4App, due to the easier authentication offerings provided by Firebase, higher free tier limits, and allowing multiple cloud code jobs. While we face vendor lock-in by using Firebase opposed to Parse with Back4App, for the scope of this project, Firebase will suit Officium very well.

3. What language should be used for Google Cloud Functions?

- Options
 1. Node.js
 2. Python
 3. Go
- Choice: Node.js
- Justification

As of 30 January 2019, Google Cloud Functions allows for projects to be written in one of three languages: Node.js, Python, and Go. However, both Python and Go are in beta. Node.js is the only one considered “stable,” so we decided it was best to choose it to hopefully prevent any issues. Additionally, it will be nice to have both the frontend and backend written in JavaScript to prevent any syntax confusion when making changes to the codebase.

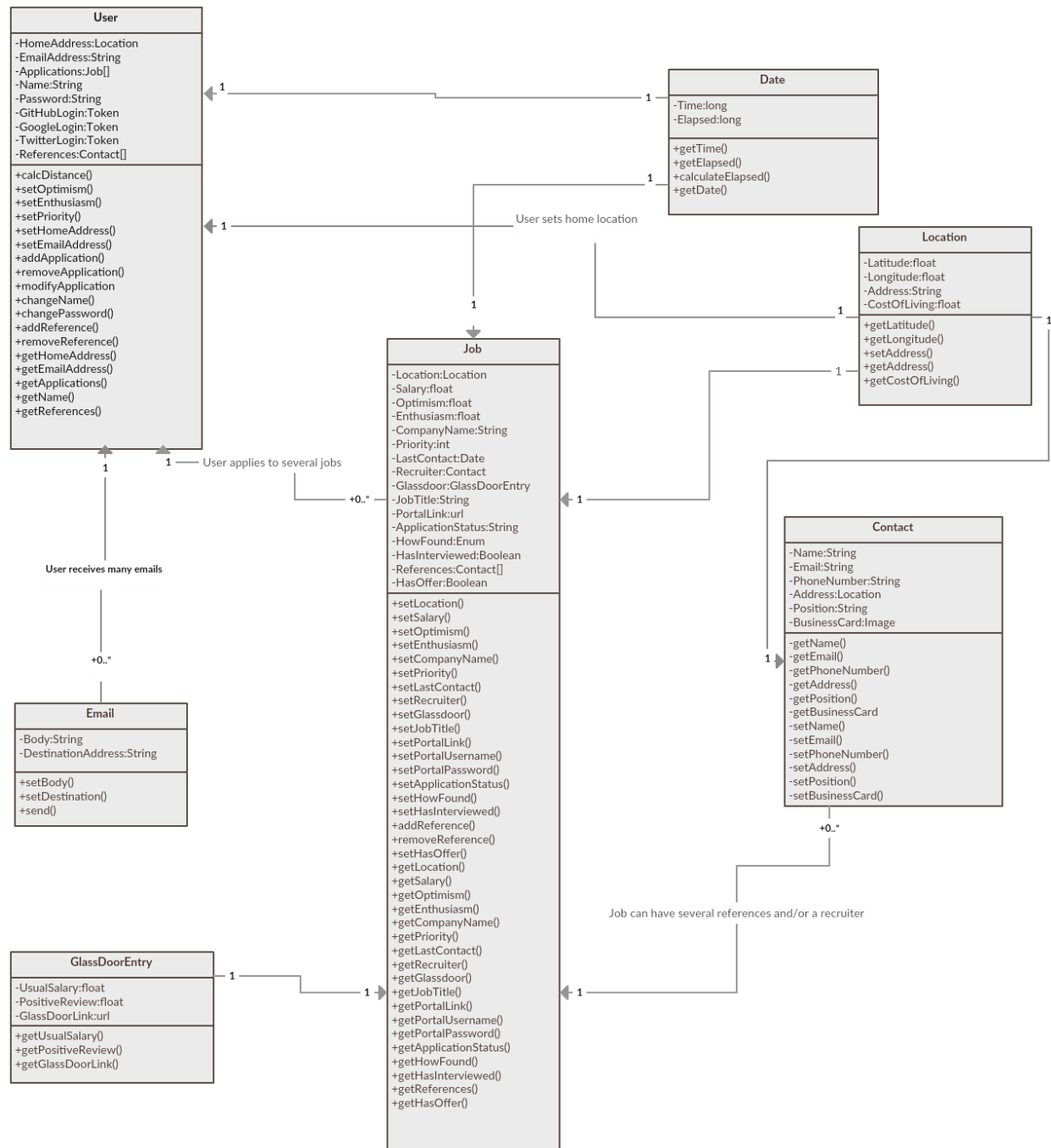
4. What service should be used to send emails?

- Options
 1. Gmail
 2. Sendgrid
- Choice: Gmail
- Justification

The decision to use Gmail was relatively easy. We are using the free “Spark” plan for Firebase. One of the caveats of this tier is that Google Cloud Functions are limited to only Google services for outbound networking. This means that we are unable to send requests to Sendgrid to send emails using the “Spark” plan. Luckily, Google offers Gmail as a free service to send emails. The only flaw of Gmail is that it prohibits sending more than 500 emails a day. Using Gmail wouldn’t be scalable, however the scope of CS 307, 500 emails a day should be sufficient.

Design Details

Class Design



Description of Classes and Interaction between Classes

We will use the classes Job, User, Email, GlassDoor Entry, Contact, Location, and Date to manage information in the backend of Officium.

1. User

- a. A User object is created when someone signs up with their email or with a Google, Github, or Twitter account.
- b. The user will select a password if they chose to sign up with their email.
- c. The user will input their preferred name.
- d. The user will fill in their Home Address and preferred Email Address.
- e. At any time, the user will be able to modify their name, password, or email.
- f. Once signed up, the user will be taken to their dashboard, the screen at which their Applications will be listed.
- g. The dashboard will also have a link to the user's References, a separate page.
- h. The reference page will list all the Contacts the user has created of references they may use for a job application.
- i. On the main dashboard, the user will be able to sort their Applications by the various criteria of the Jobs, as well as access the relevant links provided by each job.

2. Job

- a. A Job is created when the User chooses to add an Application.
- b. For the Job, the user will input:
 - i. The location, as an address
 - ii. Expected salary
 - iii. Level of optimism
 - iv. Level of enthusiasm
 - v. Company name
 - vi. Priority ranking, relative to all other applications
 - vii. Date of last contact
 - viii. Recruiter contact
 - ix. Job title
 - x. Portal link
 - xi. Application status
 - xii. How the user found the job
 - xiii. Whether the user has interviewed
 - xiv. References for this job
 - xv. Whether the user has received an offer
- c. When the user submits the Application, it will be added to the list.

- d. At any time, the User will be able to edit any of the information in the Application or delete it.

3. Email

- a. An Email will have the body and the destination address to which the Body will be sent, with a subject determined relevant to why it was sent.
- b. An Email will be sent to a User according to their settings and certain circumstances involving timing and context.

4. Date

- a. A Date will contain a time, which is a number of seconds in epoch time, and the elapsed time from the present to when the date was.
- b. Dates will be used to store when certain events occurred or are expected to occur, such as when an application was submitted or when the user expects to hear back from a certain company.

5. Location

- a. A Location will have a street address and will calculate and store, based on the address, the relevant latitude and longitude.
- b. A Location will also leverage an API to determine the expected cost of living in this particular location.

6. Contact

- a. A Contact will contain all the relevant information for the User to reach an individual.
- b. A Contact will be created when the user initiates the relevant flow and inputs the necessary information.
- c. A Contact will have:
 - i. Name
 - ii. Email Address
 - iii. Phone Number
 - iv. Address
 - v. Position Name
 - vi. Image of Business Card
- d. A Contact will either be stored as a recruiter or in a list of references.

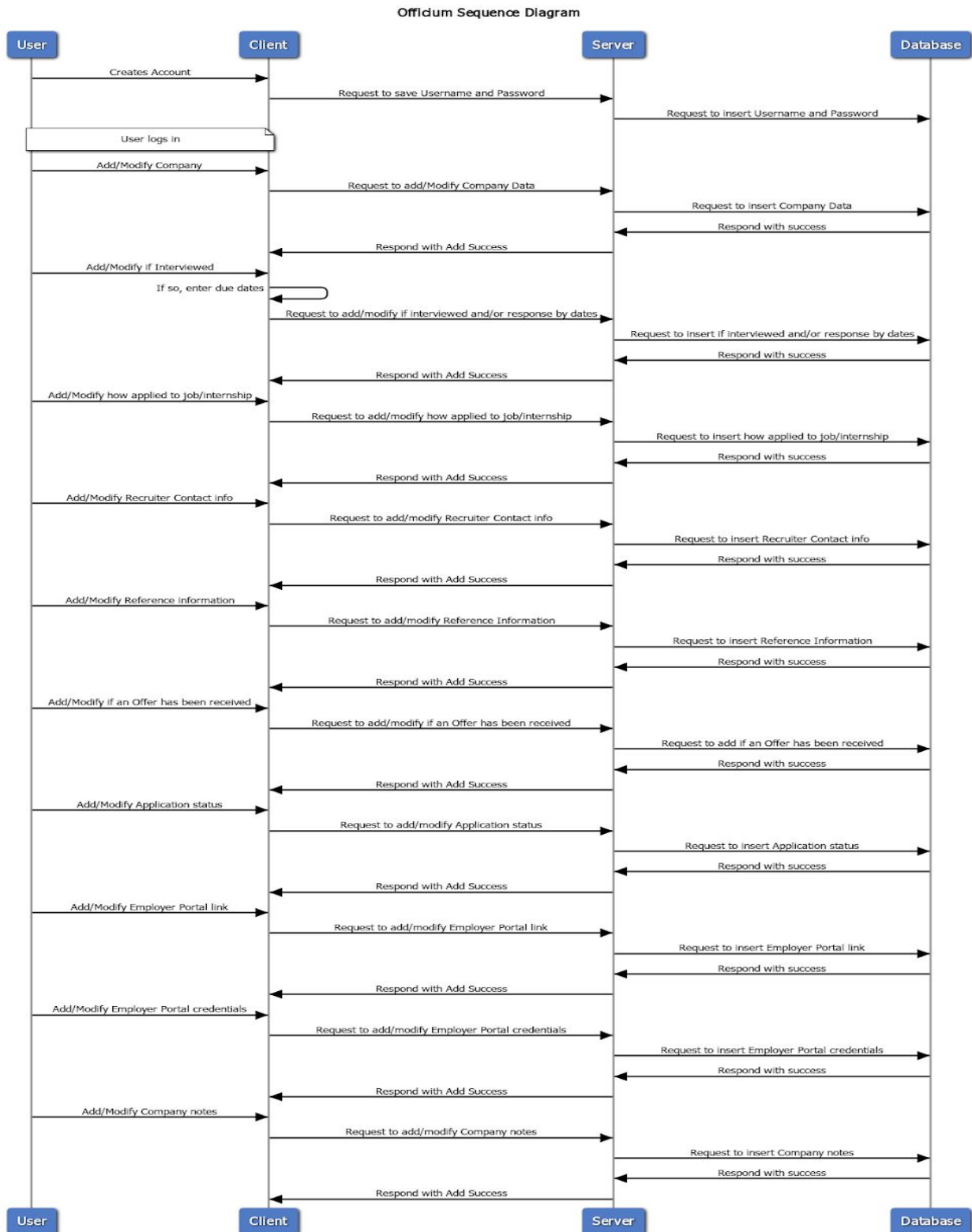
7. GlassDoorEntry

- a. A GlassDoorEntry is created automatically when the User creates a job.
- b. The application will access the Glassdoor API and use it to collect information about the relevant Job.
- c. The GlassDoorEntry will retrieve and store:
 - i. Usual salary
 - ii. Proportion of positive reviews to negative
 - iii. Link to the Job's page on Glassdoor
- d. A GlassDoorEntry will provide information that will be shown alongside the rest on the Applications page.

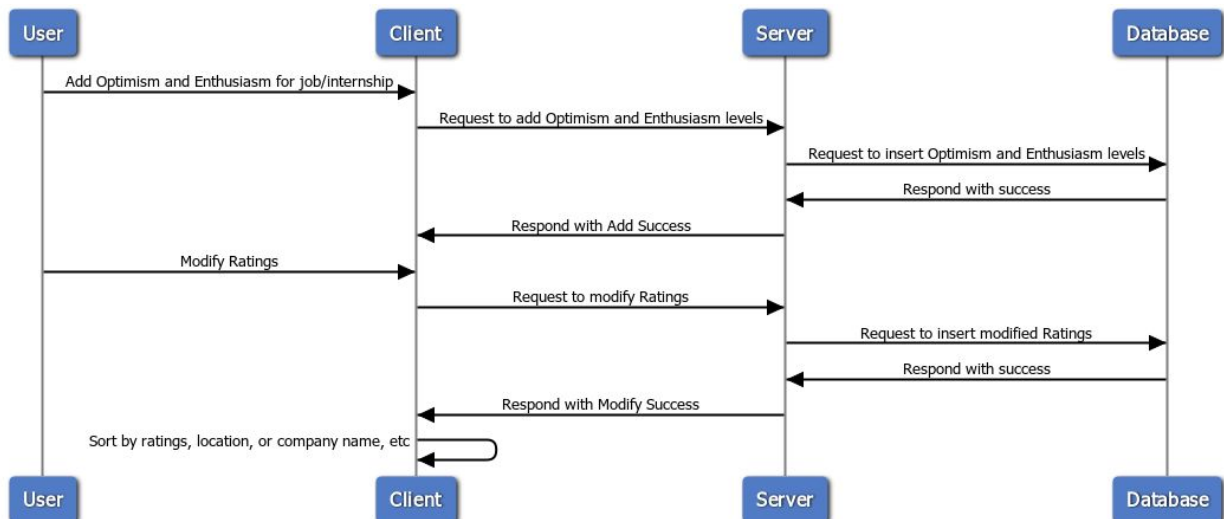
Sequence of Events Overview

This sequence diagram shows the interaction between user, client, server and database. The sequence commences with the user creating an account on the web app which will be given to the client and then handed to the server so it can create a query for the database to store the user's information. Once the account has been created, the user can choose to add or modify information such as company name, company location, personal ratings, etc. Each addition or modification will start at the client and then will be sent as a request to the server which will create a query that requests to store or change information in the database. Once the database has received its new data, a success message will be displayed on the client side. Whenever the client needs to retrieve information, the request goes through server which subsequently queries the database and returns the resulting data.

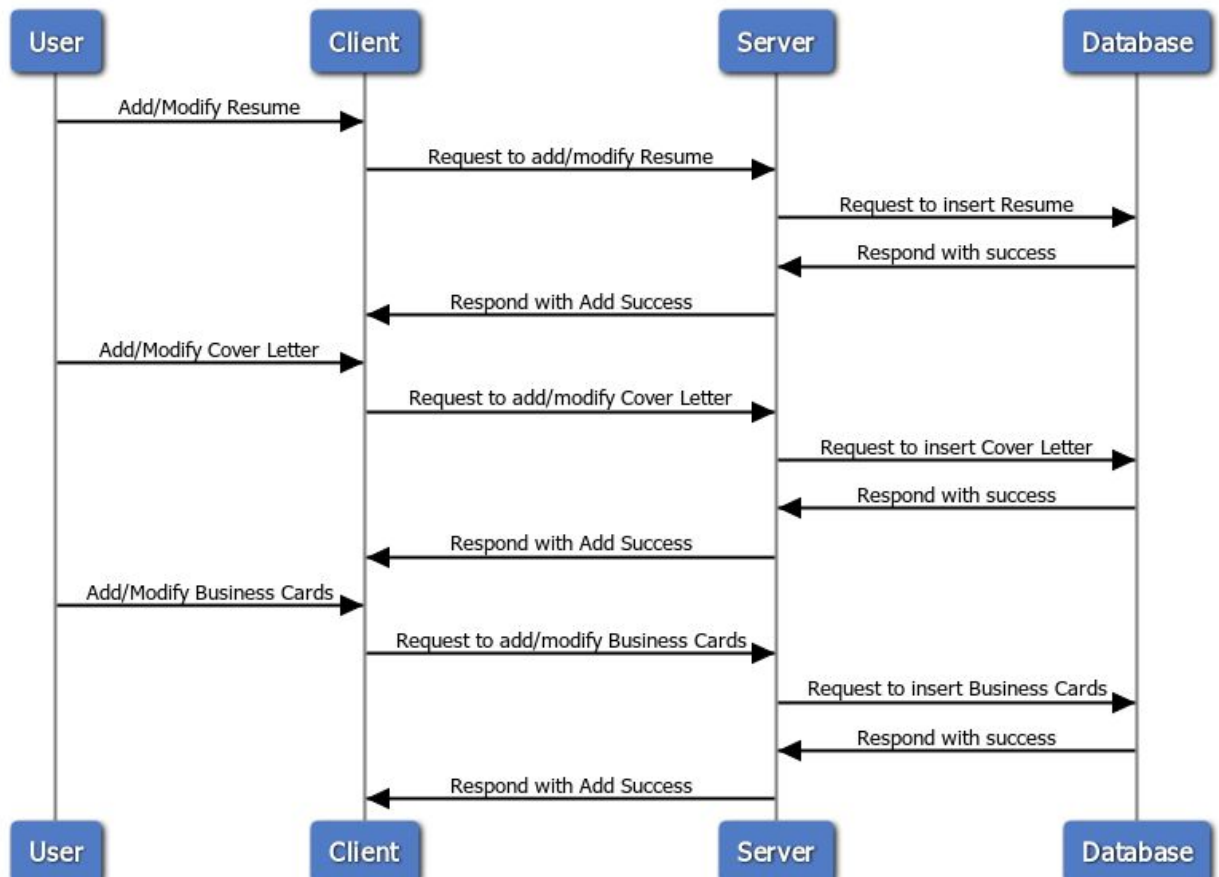
Sequence of how specifics of the internships/jobs they applied for will be stored and retrieved.



Sequence of how personal specifications will be stored and retrieved.

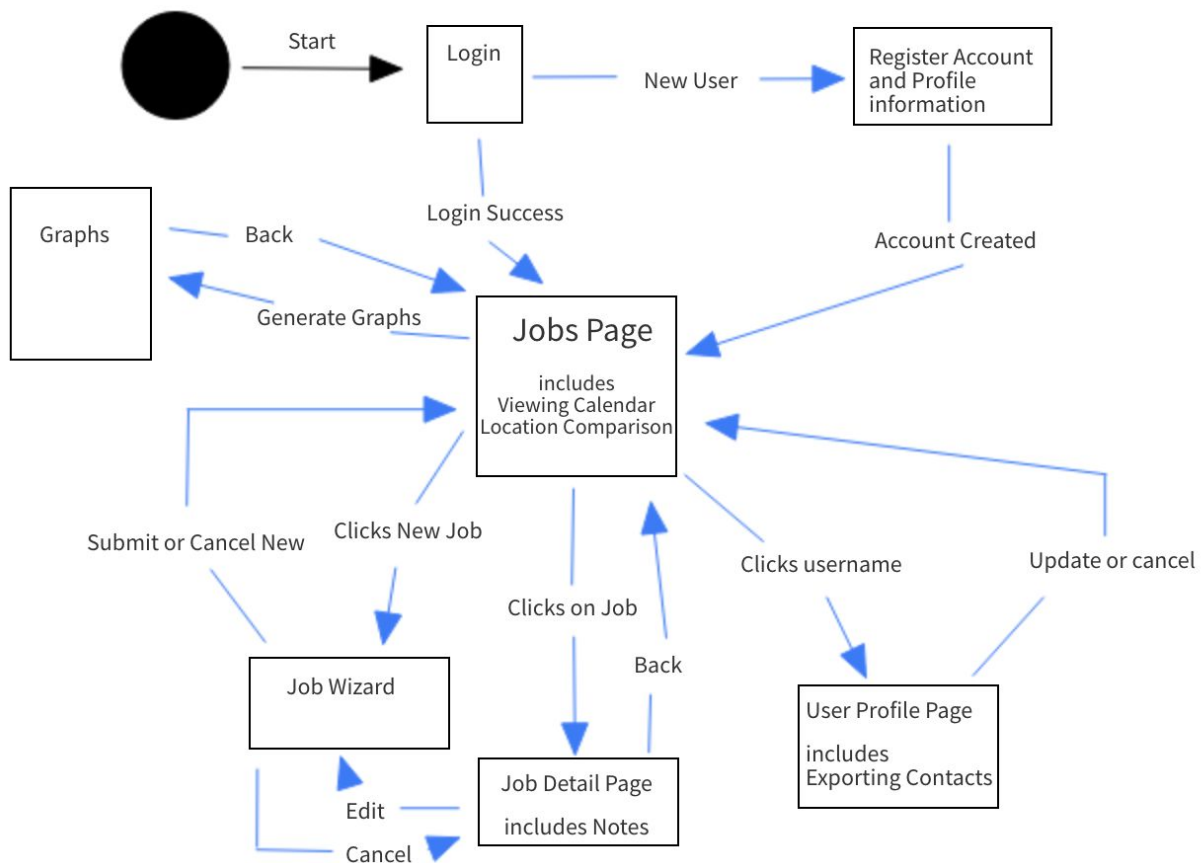


Sequence of how documents such as resume, cover letter, and business cards will be stored.



Navigation Flow Map

At the start, the user is greeted with a welcome page with the ability to login or create an account. After doing so, they are immediately taken to the Jobs Page. The Jobs Page is the centralized “home” area of the application in which we provide the user easy access to a majority of the functionality we are providing. They can add a new job application, view details of an existing job application, access their profile, generate graphs, and more.



UI Mockup

Note: Every underlined button/text has functionality

Desktop UI

Login Page



The mockup shows a clean, minimalist login interface. At the top, the word 'Officiu' is written in a large, elegant blue script font, with a blue pen icon integrated into the end of the word. Below the logo, the tagline 'A Career Tool Designed For You' is centered in a small, dark blue sans-serif font. The login form consists of two white rectangular input fields with thin grey borders, labeled 'Username' and 'Password' in a light grey font. Below these fields is a solid blue rectangular button with the text 'Sign in' in white. At the bottom of the page, three links are displayed in a small, dark blue font: 'Sign in with Twitter', 'Sign in with Google', and 'Sign in with Github', each preceded by a small blue icon.

Officiu

A Career Tool Designed For You

Username

Password

Sign in

[Sign in with Twitter](#) [Sign in with Google](#) [Sign in with Github](#)

User Profile Page

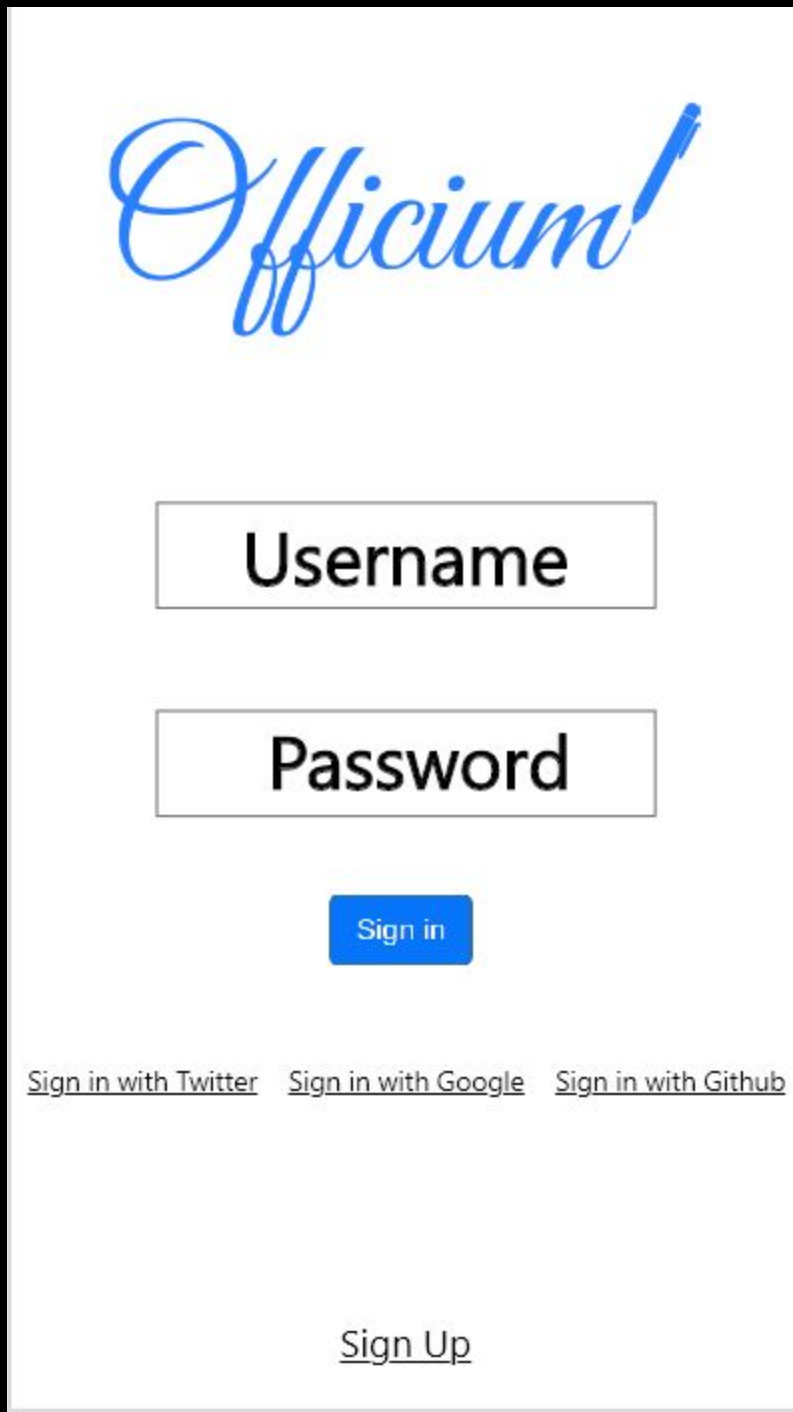
<div data-bbox="285 275 406 315">Resumes</div> <div data-bbox="212 373 310 487">Resume1.pdf Resume1.pdf Resume1.pdf Resume1.pdf</div> <div data-bbox="269 529 420 550">Upload New Resume</div>	<div data-bbox="914 275 1040 315"><h3>Profile</h3></div> <div data-bbox="1308 264 1422 289">Back To jobs</div> <div data-bbox="558 333 659 361">Location:</div> <div data-bbox="1107 333 1185 361"># Jobs:</div> <div data-bbox="558 451 753 478">Change Password</div> <div data-bbox="1107 451 1201 478"># Offers:</div>
<div data-bbox="269 583 420 613"><h3>Saved Links</h3></div>	<div data-bbox="914 573 1040 602"><h3>Resources</h3></div> <div data-bbox="573 648 799 835">CCO Other Things Depending On The Individual</div>

Sign Up Page

Sign Up

[Create Profile](#)

Mobile UI
Login Page



The image shows a mobile app login screen. At the top, the word "Officium" is written in a blue, cursive script font, with a blue pen icon at the end of the word. Below the logo, there are two white rectangular input fields with black borders. The first field is labeled "Username" and the second is labeled "Password". Below these fields is a blue rectangular button with the text "Sign in" in white. At the bottom of the screen, there are three links: "Sign in with Twitter", "Sign in with Google", and "Sign in with Github", all in a small, black, sans-serif font. Below these links is a "Sign Up" link, also in a small, black, sans-serif font, which is underlined.

Officium

Username

Password

Sign in

[Sign in with Twitter](#) [Sign in with Google](#) [Sign in with Github](#)

[Sign Up](#)

Sign Up Page

Sign Up

Jobs Page

<u>New Job</u>	
Company	
Job Title:	
Status:	Date Applied:
Location:	Next Step Due:
Company	
Job Title:	
Status:	Date Applied:
Location:	Next Step Due:
Company	
Job Title:	
Status:	Date Applied:
Location:	Next Step Due:
Company	
Job Title:	
Status:	Date Applied:
Location:	Next Step Due:
Company	
Job Title:	
Status:	Date Applied:
Location:	Next Step Due:
<u>Filters</u>	<u>Profile</u>

Profile Page[Back](#)

Profile

Location

Jobs:

[Change Password](#)

Offers:

Resumes

Resume1.pdf

Resume1.pdf

Resume1.pdf

Resume1.pdf

Saved Links

Resources

CCO

Other Things

Depending

On The Individual

Add Job Wizard

[Cancel](#)**New Job**

Company

Job Title

Salary

Date of Last Contact

[Back](#)[Next](#)