Uber for Vendors (PRDv2)

by

Github Reapers

Authors

- Frank Lee (Lead) franklee@ucsb.edu
- Raul Pulido (Scribe) raulido@outlook.com
- Edward Yuen (Developer) edwardyuen@umail.ucsb.edu
- Wei-Yee (Developer) weiyee@umail.ucsb.edu
- Eric Shen (Developer) eric10@umail.ucsb.edu

Intro

Motivation: What problem are we trying to solve?

Vendors are people/companies that provide property-related services such as plumbers and electricians. The current process of hiring vendors and scheduling appointments is messy, inefficient, and can even involve third parties; the vendor, the landlord, and the tenant must agree on when to meet. Additionally, vendors and clients have to handle multiple different jobs concurrently while looking for new jobs. This makes it difficult to properly schedule appointment times, and often prohibits an open communication pipeline between the vendors and the tenants themselves. For example, a popular method of finding vendors is to go on craigslist and post a listing in the general labor category and wait for vendors to call. This process is very inefficient since it relies on the fact that clients are available throughout the day to receive calls at random, and because it doesn't give clients direct access to vendor's qualifications. To make matters worse, client's are usually not fully aware of their landlord availability which may lead to complications in the case where landlords want to oversee vendor work.

Misc: Why is this problem important?

Currently, one of the most popular real estate practices is to buy properties, fix them, and sell them for a higher price for a profit. This practice is beneficial both for the landowner and for vendors hired to fix the property issues. By expediting the interaction between vendors, landowners, and tenants we are making the process much more efficient. Clients are able to find vendors much quicker and vendors are able to complete a lot more jobs. This is important because we are essentially increasing the

potential income gain that both clients and vendors can make. But most importantly, by making this process more efficient we are also indirectly helping in the beautification of communities.

Addressing the lack of efficient interaction between clients and vendors also helps sponsor independent vendors who might struggle competing with current major contractors. The process of finding jobs is so fragmented that at often times clients may miss out on potential vendors due to the lack of promotion. For example, vendors trying to find work on craigslist might miss opportunities that clients post on a facebook group. By centralizing the process on to one web application we are helping independent vendors find more opportunities.

Apartment landowners might lack the proper funding to have an onsite repair/maintenance crew. By increasing the interaction between clients and vendors we are also helping individual landlords address tenant issues much quicker and potentially cheaper.

Background: How is this problem addressed today?

Often, the tenant informs the landlord of a problem. Eventually, the landlord will look through multiple sources to find and reach out to an external vendor. Then the landlord must manually schedule an appointment time with the vendor. Often, this scheduling is inconvenient for the tenant as it may take a long time and may even conflict with the tenant's timetable.

The result of the current way this problem is addressed is that the landlord may have no way to see if they are overpaying a vendor or if the vendor is qualified enough for the job. The vendor might also not be able to properly schedule an appointment time due to lack of communication and miss potential jobs due to poor scheduling practice. Tenants may also be left frustrated from waiting unnecessary lengths of time for an issue to be completed despite not knowing the landlord's struggle.

Goals

Our attempt at addressing the problems described above is Uber for Vendors, a web application dedicated to making the three-way interaction between vendors, tenants, and landlords much better. Why go through the hassle of putting so much effort in setting up multiple different jobs when a web application can do all of that for you. Forget having to navigate multiple different websites to find vendors, when you can find them all on one site. Uber for Vendors will address scheduling problems by automatically listing compatible vendors to tenant-created jobs. Uber for Vendors will also assign vendors to listings optimally, so that no scheduling conflicts are possible and so that Vendors can fill their schedules to their preference.

By centralizing the entire process on Uber for Vendors we are able to automate repetitive sub-interactions between vendors, tenants, and landlords. Removing a large amount of the current overhead cost for all parties when submitting a new job.

Innovation

Uber for Vendors is a web application developed with ruby on rails and react. Users(Vendors,Tenants,Landlords) will be able to login using their google accounts and will be greeted by a landing page relevant with their account's jobs and calendar. All three users will be able to supply Uber for Vendors with their google calendar, which Uber for Vendors will use to automate job scheduling for all three users. Additionally, Uber for Vendors will guide all three users through the interaction process by breaking down the interaction into 3 separate web application interactions. First, tenants will be able to submit a job to Uber for Vendors via the job submission page. Second, Landowners will be notified by Uber for Vendors about their tenant's job submission and have the option to choose between various actions such as canceling the job, indicating that they want to be present during the job, choose between a list of available vendors, and add on to the description of the job. Third, Vendors will be able to indicate their availability and job qualifications to Uber for Vendors so they can be automatically selected for work and accept jobs. To summarize, Uber for Vendors will take care of the matchmaking and repetitive steps in the property-related service market, so that users have minimal overhead in what would seem like a complicated process.

Misc: Assumptions

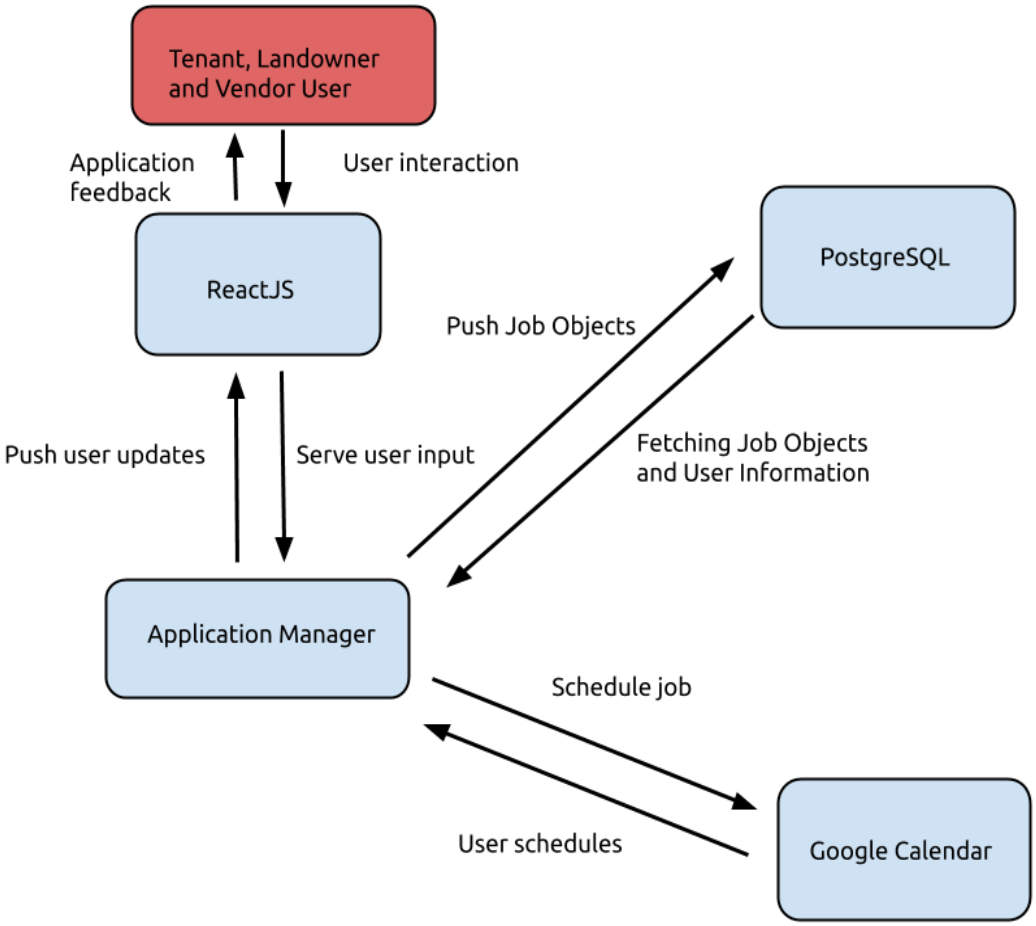
One of the assumptions this app makes is that all three parties will be involved in the process. In the case where the interaction is only between tenant and vendor the advantage of using this web application is less beneficial, but will definitely still be useful. The major focus of the app is to make the three-way interaction easier, but the automated matchmaking would definitely serve to help both.

Another assumption made is that there are enough vendors on the application to fulfill the job demands submitted by tenants. If a massive amount of jobs are submitted to Uber for Vendors it is completely possible for a Vendor's schedule to be completely filled. So we are assuming that the amount of vendors available is enough to keep waiting periods to be within a reasonable amount of time.

The most noticeable assumption is that each user has a google account. A lot of Uber for Vendors functionality relies on the fact that users will give Uber for Vendors access to their google account both for authentication and for job scheduling. Without access we have no way of knowing a user's availability or a user's identification. Allowing non-authenticated accounts to use Uber for Vendors could lead to malicious use of the web application.

System Architecture Overview

High Level Diagram



User Interaction and Design

All Users

The first instance of interaction between any user and our application is Signing up for our app. The users will be able to input their email, password, name, the type of User they are (Tenant, Landlord, Vendor), and other important account credentials upon creating an account. Upon doing so, that User will be able to login through our signin web interface and access their profile and user type specific features tied to their account.

Tenant Users

Tenant Users will be able to sync their calendar with our application in order to provide inputs of what dates and times they are available for a vendor to come. When a Tenant user needs a service done, they can fill out a request form and submit it through our application. The request form will include the service type, a short description of the issue, and other relevant details to help move forward with the problem. The Tenant User will then have to wait a period of time for our algorithm to fully schedule the appropriate appointment before being notified of a successful assignment. This assignment will automatically be added into the Tenant's calendar and a notification will be sent to confirm the appointment. The Tenant will have the option to cancel or re-schedule an appointment if they become busy later.

Landlord Users

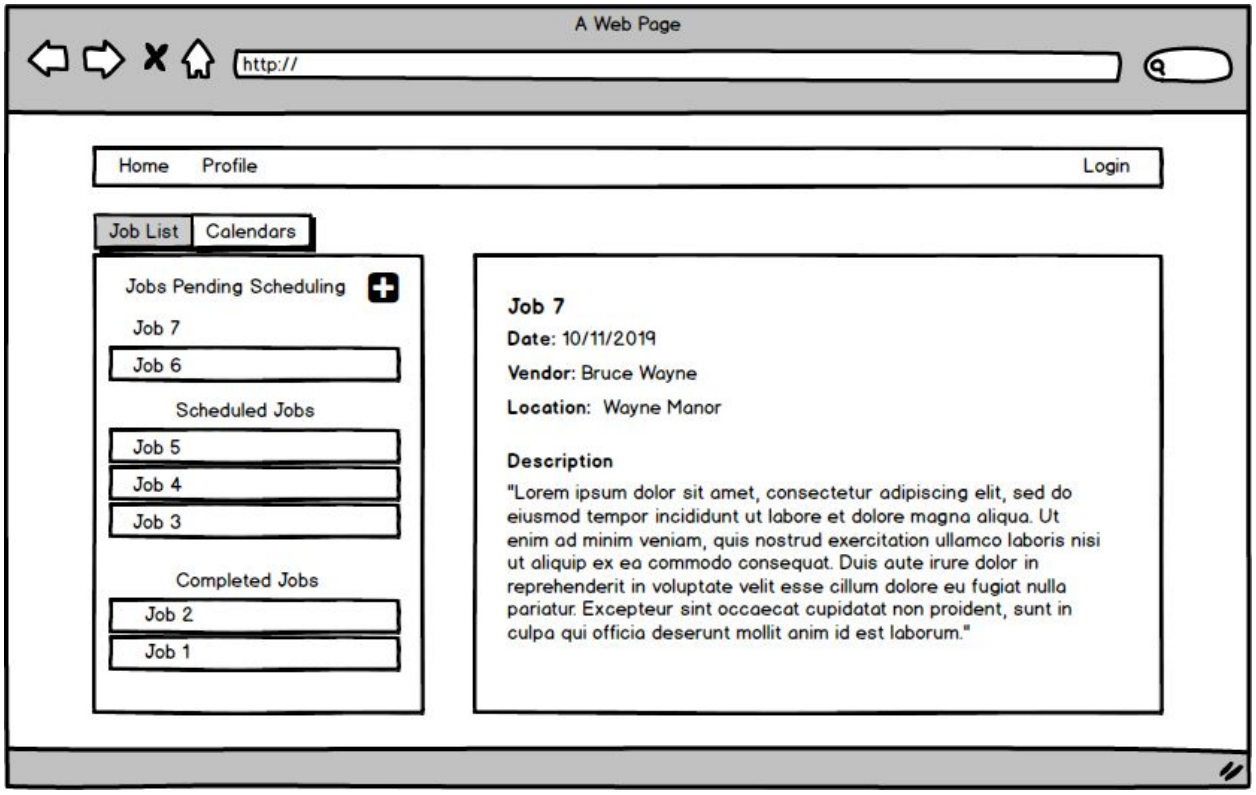
Landlord Users will also have access to a calendar page that lists the different service appointments for their tenants. Landlords will also be able to merge their personal calendar to the tenant service one so that they can see when these services are done in respect to their own schedule. Whenever a Tenant requests for a service to be done, the respective Landlord will be notified as well. As the authorizing party, the Landlord must give approval for the service before the service is officialized. In addition, the Landlord can choose to either blacklist or whitelist vendors in order to form a set of acceptable vendors the Landlord is satisfied with. The Landlord will have access to reviews of different Vendors and can decide to either manually pick out the ones he wants or use pre-built filtering to find a desired set (best reviews, fastest work, local workers, etc.) The Landlord can also choose for the appointment to be aligned with their own schedule so they can come monitor the service being done (which will also notify the Tenant of the Landlord's decision). In addition, if the Landlord decides the service is no longer necessary, they can also choose to cancel the service.

Vendor User

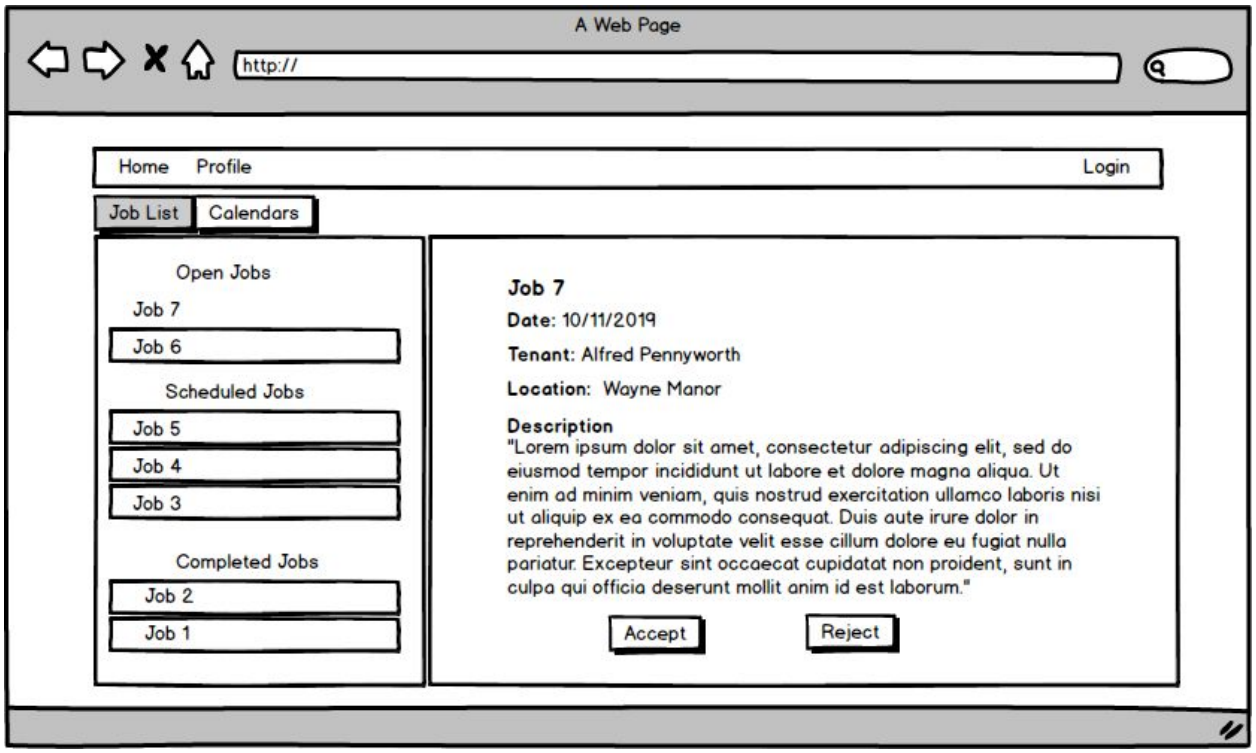
The Vendor User will have to sync their calendar into our application so that we have the available service times. The task time shown on the Tenant side will be estimated, but each Vendor will have the option to put the time it takes them to personally complete the task. These appointments will automatically be scheduled so a Vendor will be able to let these appointments accrue without having to put too much effort into discovering jobs. These jobs will be assigned with an appropriate buffer time for the Vendor to respond to in order to avoid last minute cancelations. The Vendor will be able to see the jobs, the general description, and other details in the calendar.

UI MockUps

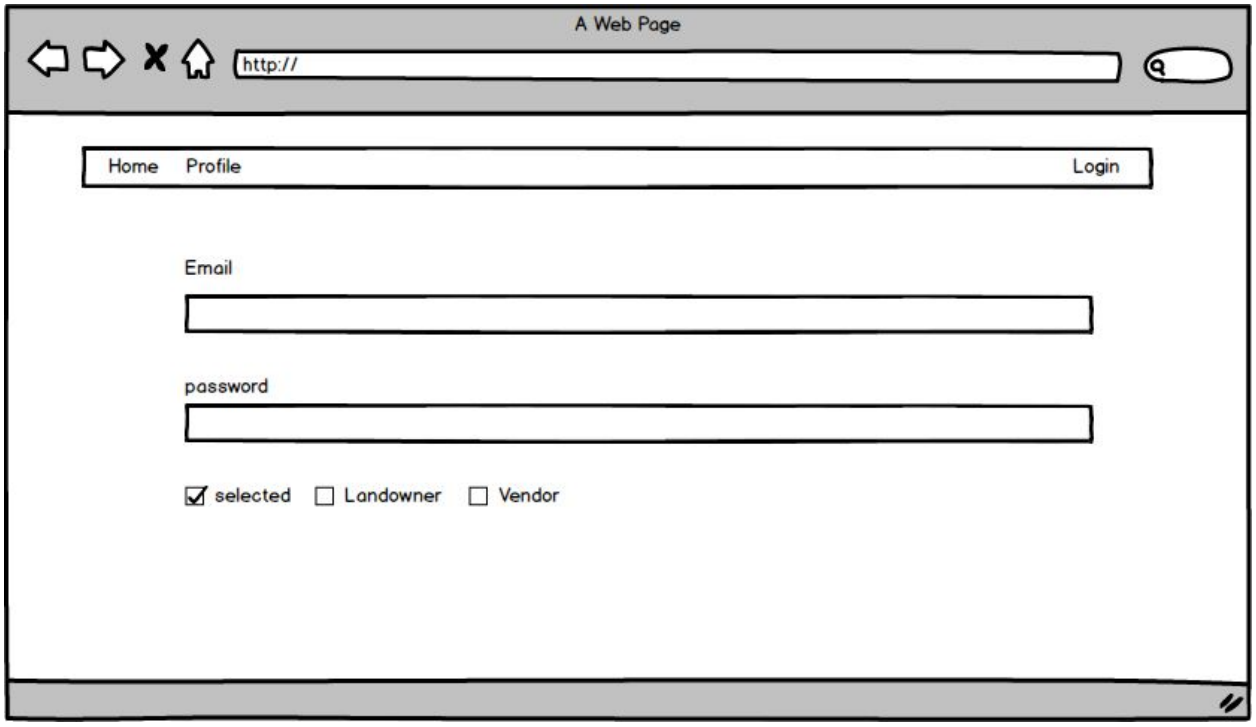
Landing page with Job Listing selected for the Tenant and Landowner User.



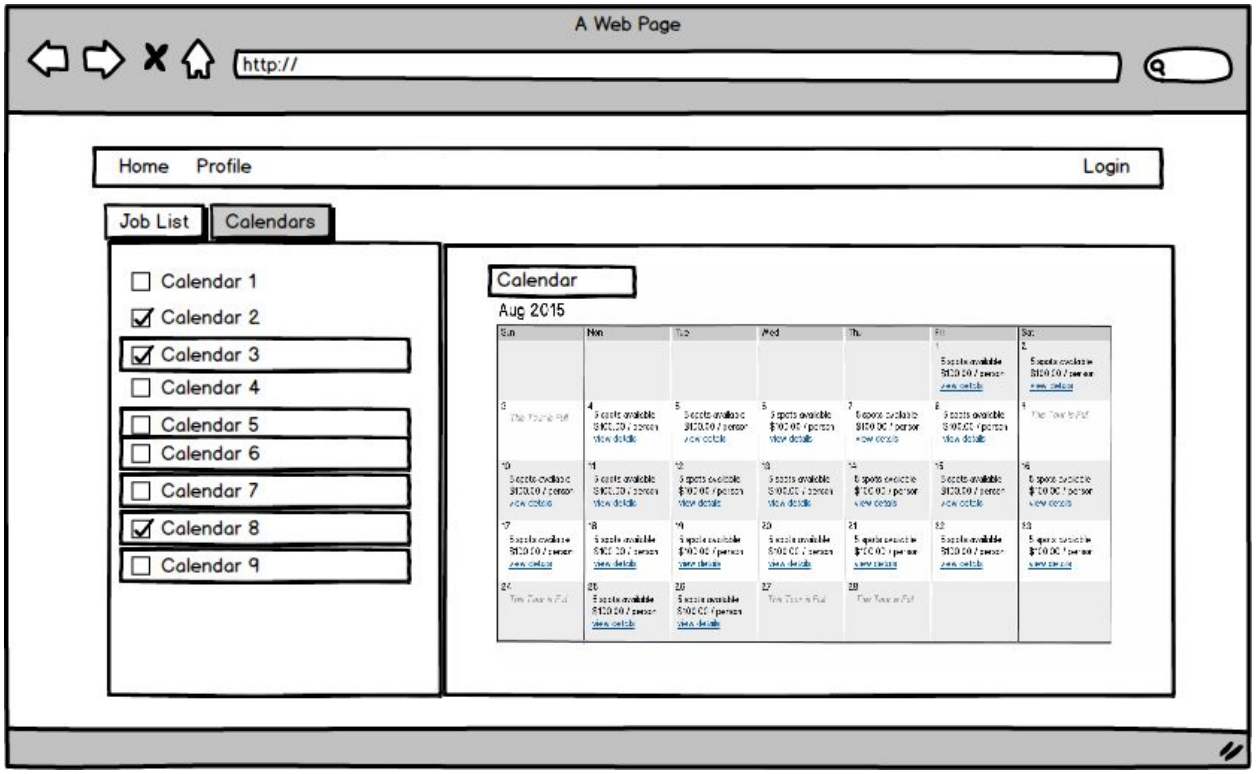
Landing page with Job Listing selected for the Vendor User.



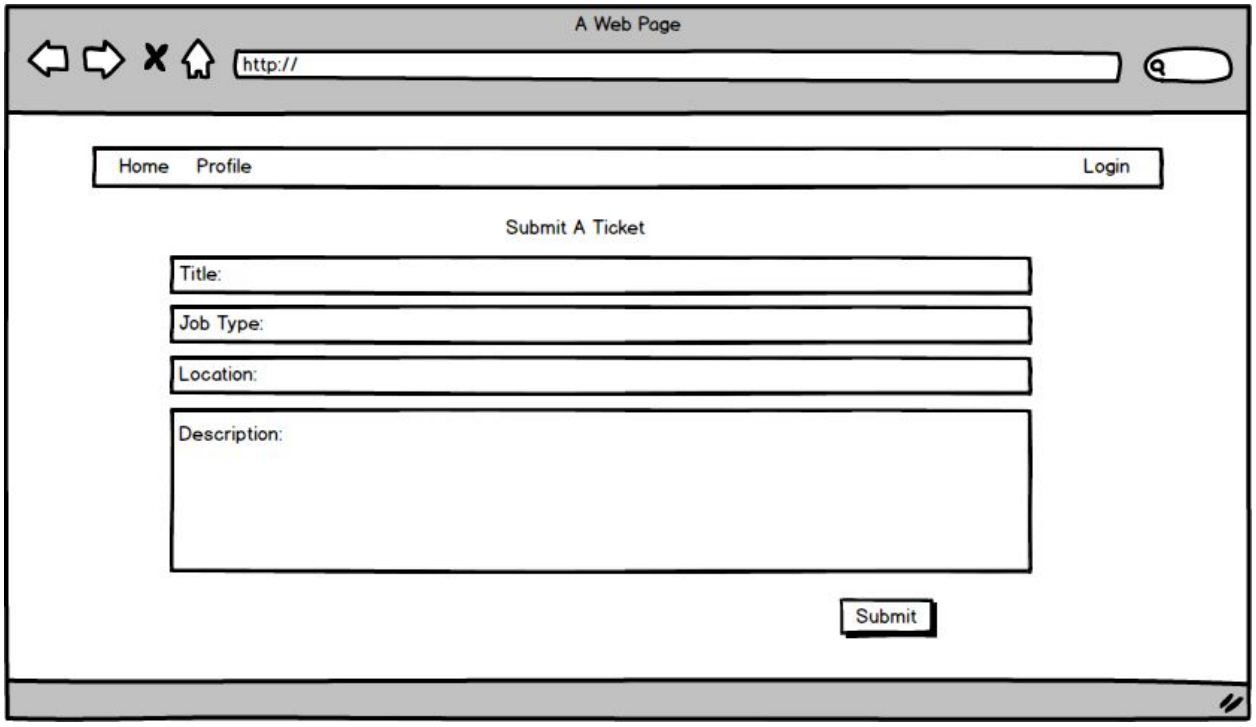
Login Page for the Tenant, Landowner, and Vendor



Landing page with Calendar listing selected for the Tenant, Landowner, and Vendor



Ticket Submission page for the Tenant User



Prototyping Code, Tests and Metrics

GitHub: https://github.com/franklee26/appfolio-uber-for-vendors

Sample commits:

- Integrating Google Calendar API with a new rails Calendar controller: https://github.com/franklee26/appfolio-uber-for-vendors/commit/077dde85639c37b1ed4b6b362e12256120377899
- Removed development.log from remote branch: https://github.com/franklee26/appfolio-uber-for-vendors/commit/013740925b8c967cfc41b71c15bfa35b27a19ed6
- Created Landowner page and model: https://github.com/franklee26/appfolio-uber-for-vendors/commit/b606d930e61474d7d61c02aa02f1167092b14f19
- Adding Tenant resource to show Tenants https://github.com/franklee26/appfolio-uber-for-vendors/commit/7ca63ea7474cc32764e4bb7a75dcd52e7ef538df
- Added Tenant tests, Tenant Controller and a new login page https://github.com/franklee26/appfolio-uber-for-vendors/commit/418ceb12092a1f91ca93a1ebc555fbe341c3dcf5
- Basic list of vendors and their occupation

https://github.com/franklee26/appfolio-uber-for-vendors/commit/3a2d61019e955ee35195c7dad1b64af87700eae5

- Vendor search page https://github.com/franklee26/appfolio-uber-for-vendors/commit/ec68027fa647f875c20e5d39adc2cf365e2e7973
- Integrated Google::POST call to add calendar event https://github.com/franklee26/appfolio-uber-for-vendors/pull/21/commits/2aad62dde470452854f5d85843a6d323cba630eb
- Created Sessions controller for Google::API login routing https://github.com/franklee26/appfolio-uber-for-vendors/commit/1a91050fa0dcf5102482e606ac4daeda8b0591a4
- Added Job controller, model and views w/ RESTful API hooks https://github.com/franklee26/appfolio-uber-for-vendors/pull/17/commits/f44d5bdf8bd381750762c1fcad340a45c4a624fe
- Test Cases for Job Model related functions https://github.com/franklee26/appfolio-uber-for-vendors/commit/de2dbcf108836dcbc340e67ab20da6b5057bcb86

Requirements

User Stories

User Story #1: Signup Page
Actors: Tenant, Vendor, and Landowners
Pre-conditions: User must have access to the web application and locate the signup button.
Use-case: As a user, I can choose among three signup options signifying whether I am a tenant, a landlord, or a vendor. I need a valid email and a password to sign up.
Acceptance Test:

- There exists a single sign up page that gives users three sign-up options (tenant, landlord, vendor)
- If a user clicks on one of the signup buttons/links, he will then be prompted to fill in a valid email and a valid password. (email : [string]@[string].[string] and the password has 10+ characters)
- If user submission is valid, then the email and password attributes are pushed into the database and are greeted with a success page.

- If the user submission is invalid, then attributes aren't pushed into DB and are prompted with error page.

User Story #2: Login Page
Actors: Vendor, Landlord, and Tenant
Pre-conditions: Users must have signed up for an account before attempting to log in and must have access to the web-app.
Use-case: As a user, I can fill the login form with my account credentials, submit my account credentials, and have access to my Uber for Vendors account.
Acceptance Criteria:

- The account information is retrieved from the application's database, and is displayed on my application indicating that login was successful
- The account credentials submitted are invalid and I receive an error message from the application indicating that login was unsuccessful

User Story #3: Vendor Submit Availability
Actor: Vendor
Pre-conditions: Vendors are signed in and have internet access.
Use-case: As a vendor, I can put in my schedule availability and submit it to the back end.
Acceptance Criteria:

- Vendors are greeted with a list of times for every hour from 9am-6pm
- Vendors are able to select and deselect available times but must select at least one before submitting
- If at least one time is checked then the time is pushed into the Vendor database and Vendor is greeted with a success page
- If no times are checked then nothing is pushed into the database and the Vendor is prompted with an error page

User Story #4: Tenant Submit Availability
Actor: Tenant
Pre-condition: Users will already have signed up with an account and must be logged into their account.
Use-Case: As a Tenant user, I can put in my schedule availability and submit it to the back end so the vendor service can be done at a time convenient for me
Acceptance Criteria:

- Tenant are greeted with a list of times for every hour from 9am-6pm
- Tenant are able to select and deselect available times but must select at least one before submitting

- If at least one time is checked then the time is pushed into the Tenant database and Tenant is greeted with a success page
- If no times are checked then nothing is pushed into the database and the Tenant is prompted with an error page

User Story #5: Landlord Submit Availability
Actor: Landlord
Pre-Condition: Users will already have signed up an account and have logged into their account through the login page.
Use-case: As a Landlord user, I can put in my schedule availability and submit it to the back end so I can oversee the work done if I want.
Acceptance Criteria:
- Landlord are greeted with a list of times for every hour from 9am-6pm
- Landlord are able to select and deselect available times but must select at least one before submitting
- If at least one time is checked then the time is pushed into the Landlord database and Landlord is greeted with a success page
- If no times are checked then nothing is pushed into the database and the Landlord is prompted with an error page
-

User Story #6: Tenant requests Job
Actor: Vendor
Pre-condition: Tenant has an account.
Use-case: As a tenant, I can request a job so I can get a service done
Acceptance Criteria:
- On this page, the Tenant will have to pick the type of job (which will influence the time the job will take, mvp will not include this). This will route the job to the landlord and will provide the vendor for this job a list of times that the vendor can pick to take the job.

User Story #7: Landlord search for Vendor
Actor: Landlord
Pre-condition: Landlord has logged into his account and has accessed find landlord page
Use-case: As a landlord, I can search through a list of vendors to select a vendor I want to work with.
Acceptance Criteria:
- Will list pages of vendor profiles

- Allows landlords to filter vendors based on what they specialize in ex: search for all electricians
- Allows landlord to filter the list by location of the vendor
- Will provide a search bar for the landlords to search for a specific vendor

User Story #8: Vendor Calendar
Roles: Vendor
Preconditions: Vendors must have signed up for an account before attempting to log in and must have access to the web application.
Use-case: As a vendor, I can see a list of jobs assigned to me so I know what jobs I have to do that week.
Acceptance Criteria:

- A page has a calendar that contains all of the vendor's scheduled jobs
- A job contains these basic info for now including the following:
    - Job type
    - Scheduled time (the full range)
    - (Maybe name and address of user)

User Story #9: Jobs/Vendors displayed on a Map
Actor: Landlord, Vendor, Tenant
Pre-Condition: Users are logged in.
Use-case: As a Landlord user, I can choose to display the location of my job so that vendors can better judge the feasibility of completing the job. Vendors can then better decide if they want to switch jobs. As a Vendor user, I can choose to display my work location so clients can better see if vendors are able to complete their jobs.
Acceptance Criteria:

- Job/Vendor is displayed on a map interface with a basic title.
- The vendor can click on jobs displayed on the map to see more information about the job.
- The landlord can click on vendors on the map to see the vendor's profile.

User Story #10: Vendor home page
Actor: Vendor
Pre-condition: Vendor has already created an account
Use-case: As a vendor, I can see my home page where it displays my full name, email, phone number and the job(s) I can accept.
Acceptance Criteria:

- If a vendor is logged in, then he/she can go to their homepage where only the full name, email, phone number and jobs are displayed.

User Story #11: Google OAuth Login Integration
Actor: Vendor, Tenant, Landlord
Pre-condition: None
Use-case: As a user, I can have my account created or logged in through Google OAuth
Acceptance Criteria:

- Logging in through Google OAuth will allow users to be created or logged in (determined by a search for the user through credentials)
- The Google Calendar controller will store user information in the sessions controller, so the sessions controller can serve the proper pages

User Story #12: Google Calendar Integration
Actor: Vendor, Tenant, Landlord
Pre-condition: User is logged in and has an account saved into the database
Use-case: As a user, I can have access to view my current calendars on the web-application and the events associated with each calendar
Acceptance Criteria:

- Users can view all of their calendars on their landing page and the 10 upcoming events associated with each of their calendars

User Story #13: Google Calendar Integration #2 with FreeBusy Calls
Actor: Vendor, Tenant, Landlord
Pre-condition: User is logged in and has an account saved into the database
Use-case: As a user, I can add an event to my personal calendar through the web-application during times that are free
Acceptance Criteria:

- Users can view 1-hour free time-slots in their individual calendars (done with FreeBusy)
- After users select to add an event to their calendar, it will reflect on their actual calendars when they log in through google

User Story #14: Sessions Controller
Actor: Vendor, Tenant, Landlord
Pre-condition: User is logged in and has an account saved into the database
Use-case: As a user, I can view personalized pages based on my credentials and information. I can also choose to logout and destroy the current session, and log back in later.
Acceptance Criteria:

- The Sessions controller will need to keep track of the user that is logged in (through credentials)
- The Sessions controller serves either a landing page or profile page personalized for the user who is logged in
- If a user logs out, the sessions controller will destroy the temporary information about the user logged in, redirect to the homepage, and allow new users to login

User Story #15: Landlord views vendors
Actor: Landlord
Pre-condition: A landlord user is logged on
Use-case: As a landlord, I can do a search on vendors associated with my property and view their profiles
Acceptance Criteria:

- Landlords can run a search for all of their associated vendors

User Story #16: Vendor Profile Page
Actor: Vendor
Pre-condition: Vendor has already created an account
Use-case: As a vendor, I can view my current profile details and edit/delete/add more information associated with my account. For example, I can add a job occupation, delete a job occupation, or edit my profile bio.
Acceptance Criteria:

- If a vendor is logged in, then he/she can go to their profile page where only information related to that vendor's account is displayed and is customizable.

User Story #17: Logout
Actor: Vendor, Landlord, Tenant
Pre-condition: User is currently signed into the web application.
Use-case: As a user to Uber for Vendors that is currently signed in, I have the ability to logout to indicate the end of my interaction with the web application by clicking on a logout button on the navbar.
Acceptance Criteria:

- Upon clicking on the logout button any functionality associated with my account that is not supposed to be active during the end of my interaction with the web application is halted. For example, as a tenant upon being logged out any job submissions with my account should not be possible. However, as a vendor I should still be able to be assigned to jobs while being logged out.

User Story #18: Rate a vendor
Actor: Tenant, Vendor
Pre-condition: A job submitted by a tenant has been completed by a Vendor.
Use-case: As a Tenant I can rate a vendor after he has completed a job I submitted. I can pick between 1 to 5 stars.
Acceptance Criteria:

- Upon job completion, which is indicated once the time interval associated with that job has passed, I am prompted with a message that gives me the option to rate a Vendor. I can then click on the amount of stars I want to give the vendor and press submit. A thank you message will appear indicating the end of the interaction.

User Story #19: Rate a tenant
Actor: Tenant, Vendor
Pre-condition: A job submitted by a tenant has been completed by the vendor.
Use-case: As a vendor I can rate a tenant after completing their job.
Acceptance Criteria:

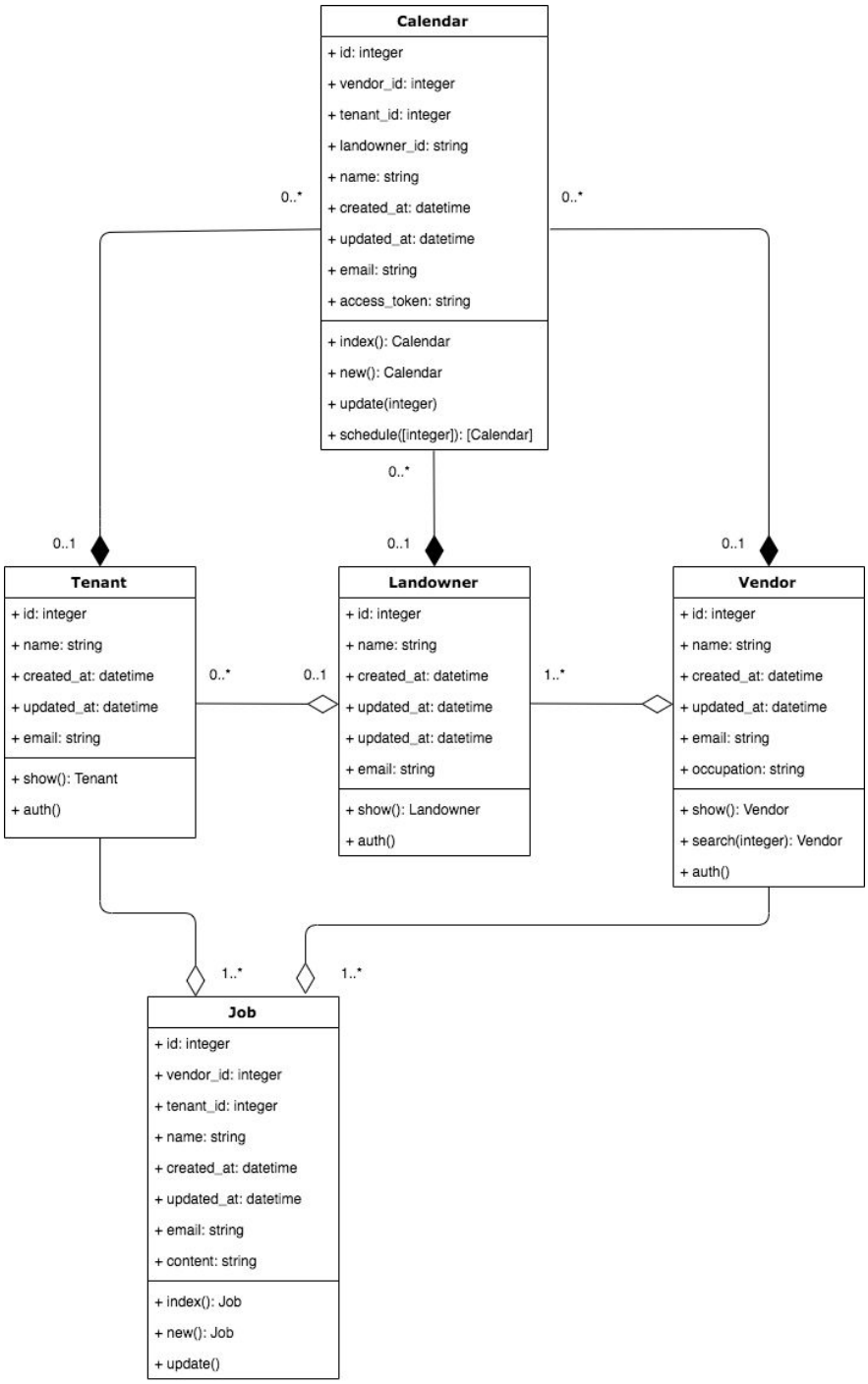
- Similar to vendor rating, vendors are prompted with a message that gives them the option to rate a tenant. I can then click on the amount of stars I want to give the tenant and press submit. A thank you message will appear indicating the end of the interaction.

User Story #20: View Job Description
Actor: Vendor
Pre-condition: Have recieved jobs from Uber for Vendors that are pending acceptance or have been accepted.
Use-case: As a vendor I can click on a job html container to view more details about a job, such as description, associated tenant, and the time it will take place.
Acceptance Criteria:

- Vendors can click through multiple jobs which will display a separate container on the side with the description of said job.

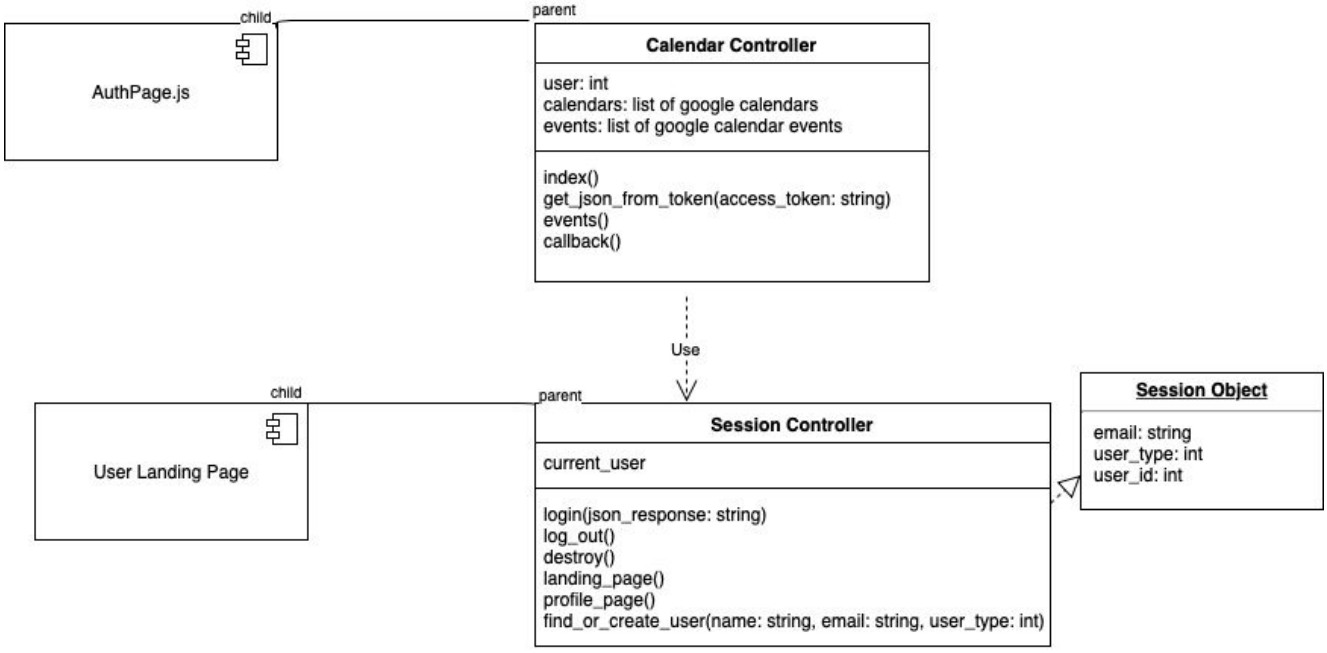
System Models

UML Design

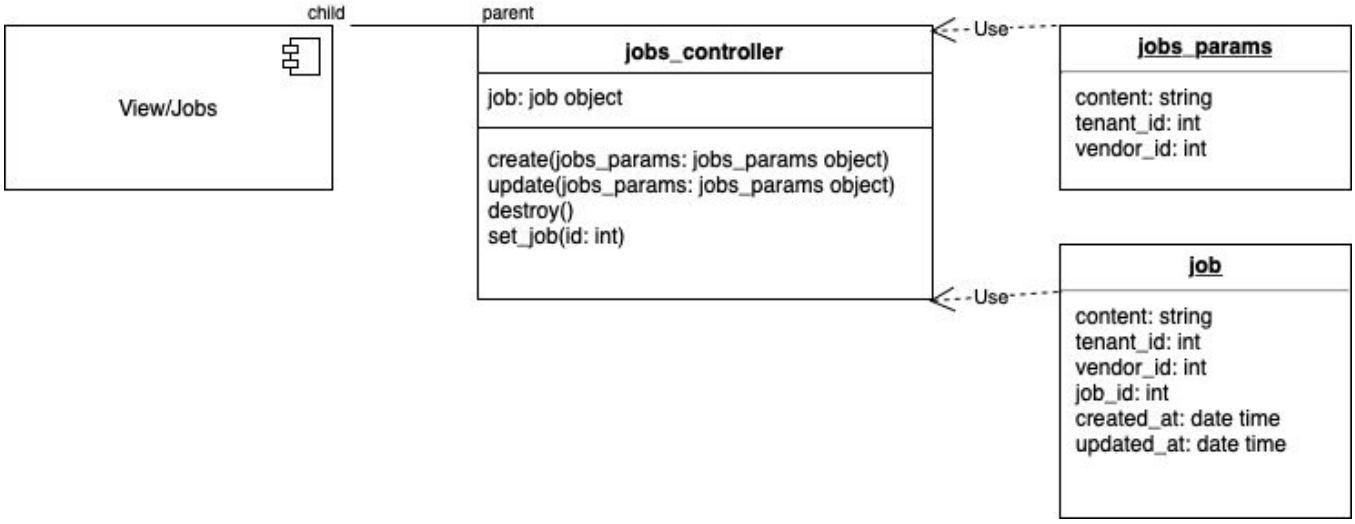


Class/Module Sequence Diagrams

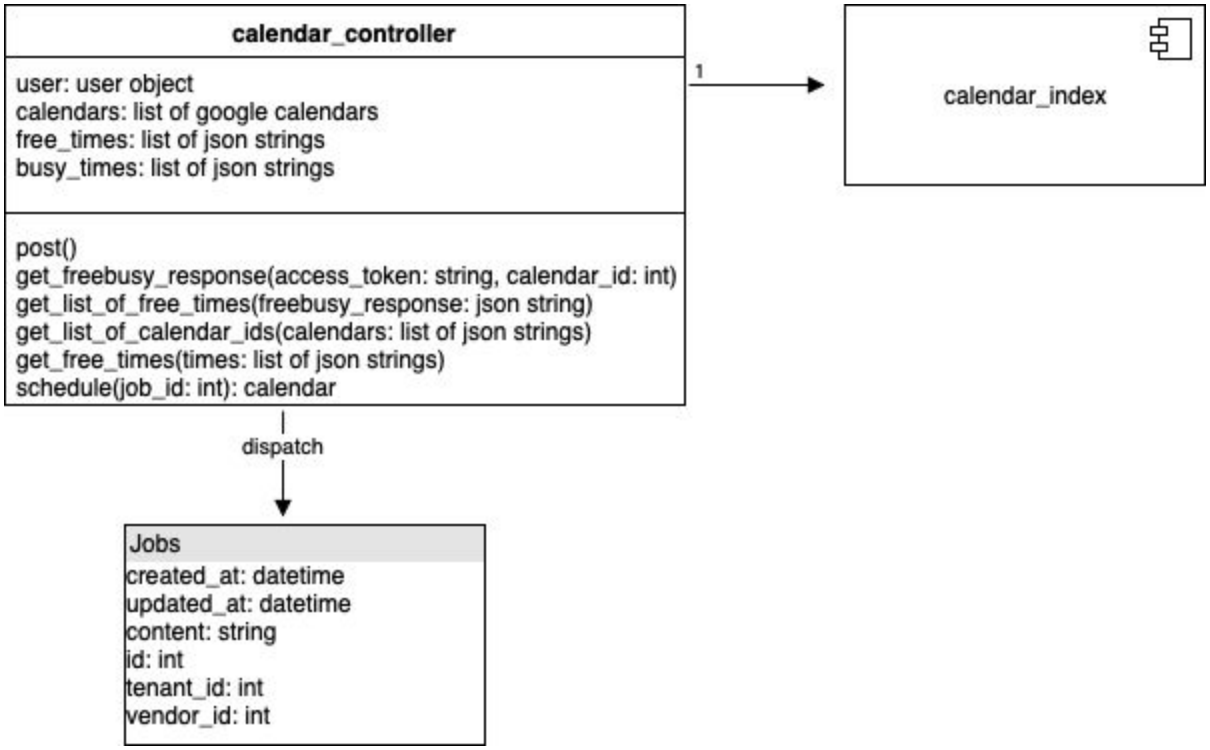
Class Diagram Sequence for User Login



Class Diagram for Job Submission/Edit

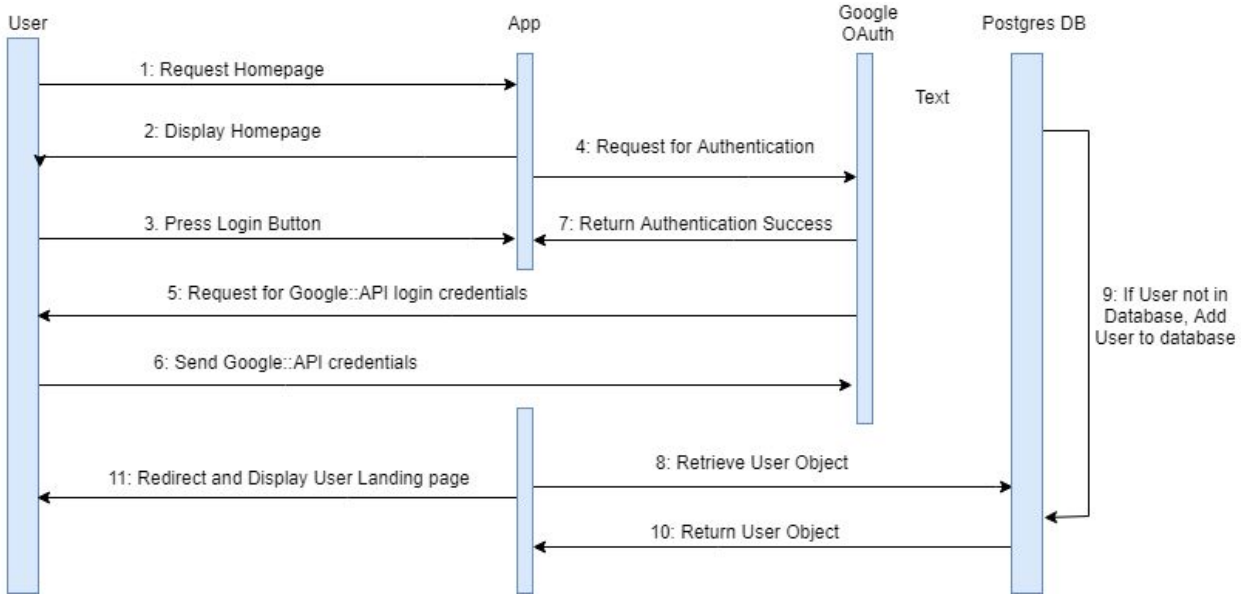


Class Sequence Diagram for Scheduling

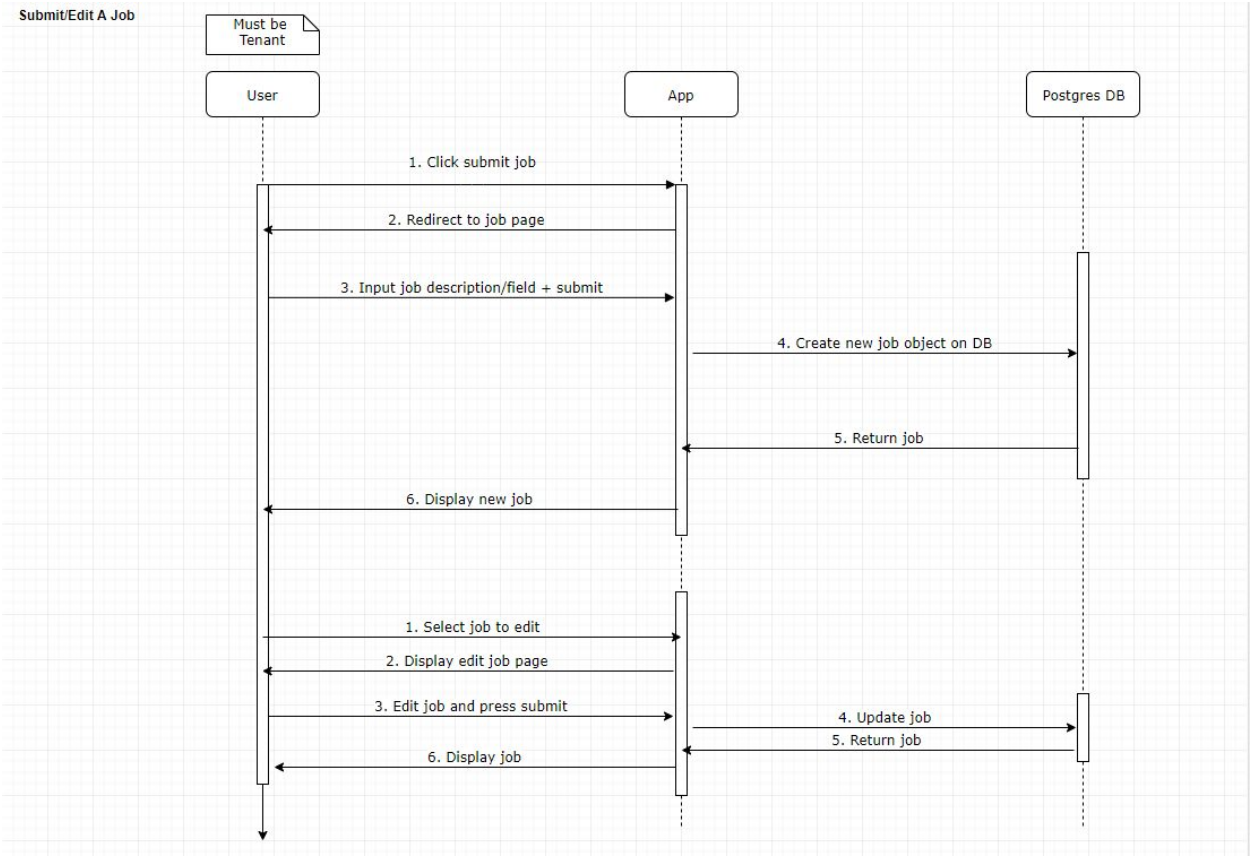


UI Sequence Diagrams

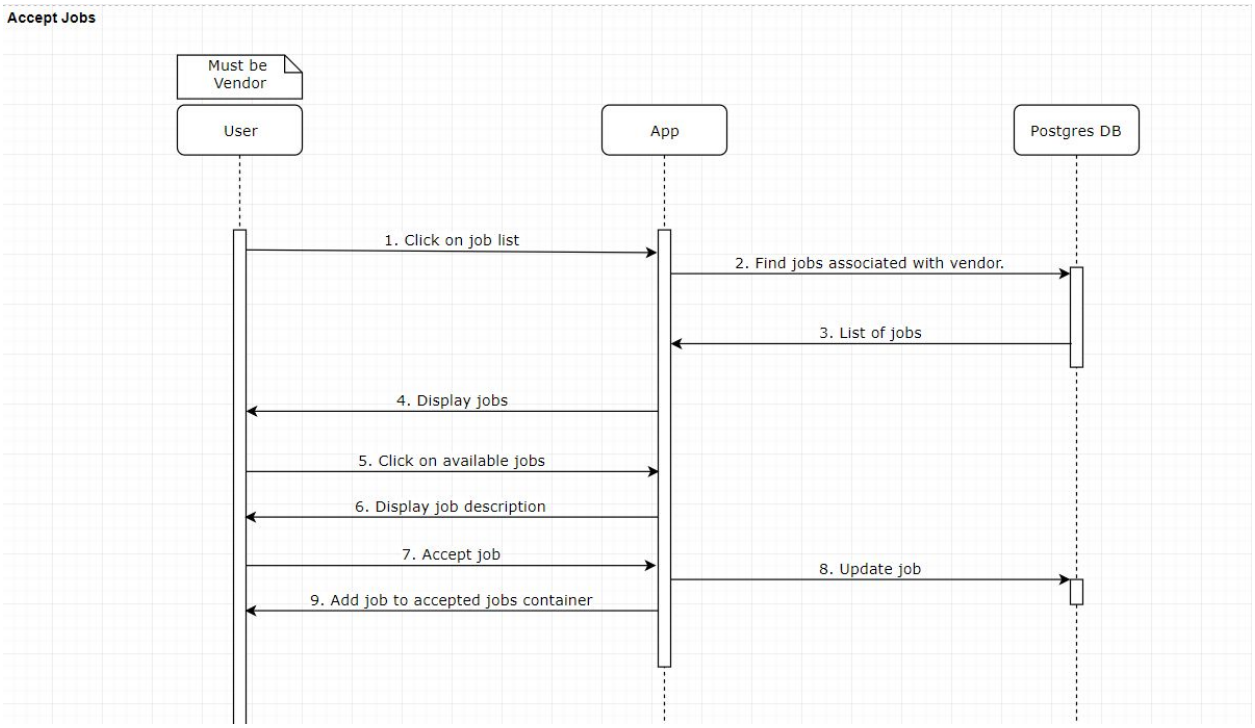
Sequence Diagram for User Login



Sequence Diagram for Tenant Job Submission/Edit



Sequence Diagram for Vendor Job Acceptance



Appendix

Technologies Employed

Ruby on Rails: A web application development framework that runs Ruby on the backend and provides an MVC interface

Ruby: Multi-paradigm language

Google Calendar API: Standard API for RESTful interactions as well further support for scheduling analysis.

ReactsJS: A Javascript framework used for making user interfaces on web applications.

Bootstrap: A CSS framework used to implement effective user interfaces as well as creating mobile-friendly responsive web applications.

PostgreSQL: Object-relational database management system.

Javascript: Multi-paradigm language

Git: Versioning control to allow a feature branch workflow

Google OAuth: The ability to use Google Authentication API on different web applications to supplement their own web application's authentication.