Requirements Definition
Group 1

Job Find

Introduction and Context

The purpose of this project is to build a system for facilitating community yardwork job offers. It will serve as a profitable means to match young workers with customers offering jobs. The site owner will not employ anyone, but rather act as a job finding provider that receives a surcharge for making a connection between the customer and worker.

To populate the system with listings, customers will have the ability to sign up and post the type of job they have available, and when they are hoping to have it completed. They will also add a balance to conveniently handle payment to the worker that completes their job listings. Once they have listed a job, the system will attempt to match it with an available worker.

The system will allow workers to create accounts and set their availability. Jobs will be rewarded to workers based on this availability. They may specify the type of jobs they're willing to accept. After a job is assigned to a worker, they will have access to contact information for the customer and be expected to complete the job.

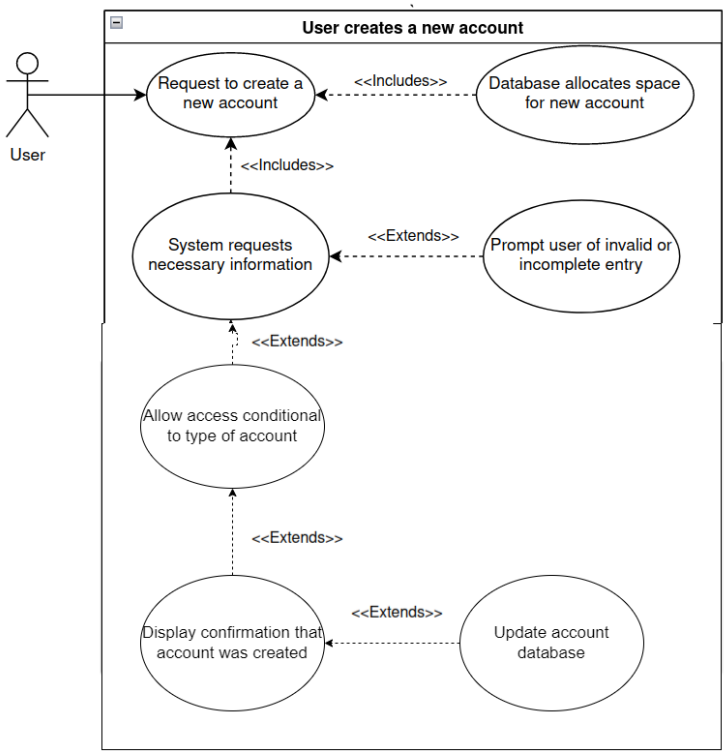
Upon completion, the worker will notify the system that the job is complete, and money will be transferred accordingly. 90% of a listed job wage will be rewarded to the worker, and 10% to the site owner. Both parties will have the opportunity to make a dispute if needed.

- This system will provide a convenient means for young people wanting to make money to connect with customers needing work done around their yard without having to create an individual advertising scheme.

Users and Goals

The purpose of this section is to describe the system's actors and how they aim to interact with the system using UML use case diagrams.

Figure 1 – User creates a new account



Participating actors: Users

Entry conditions:

- The user wants to create an account

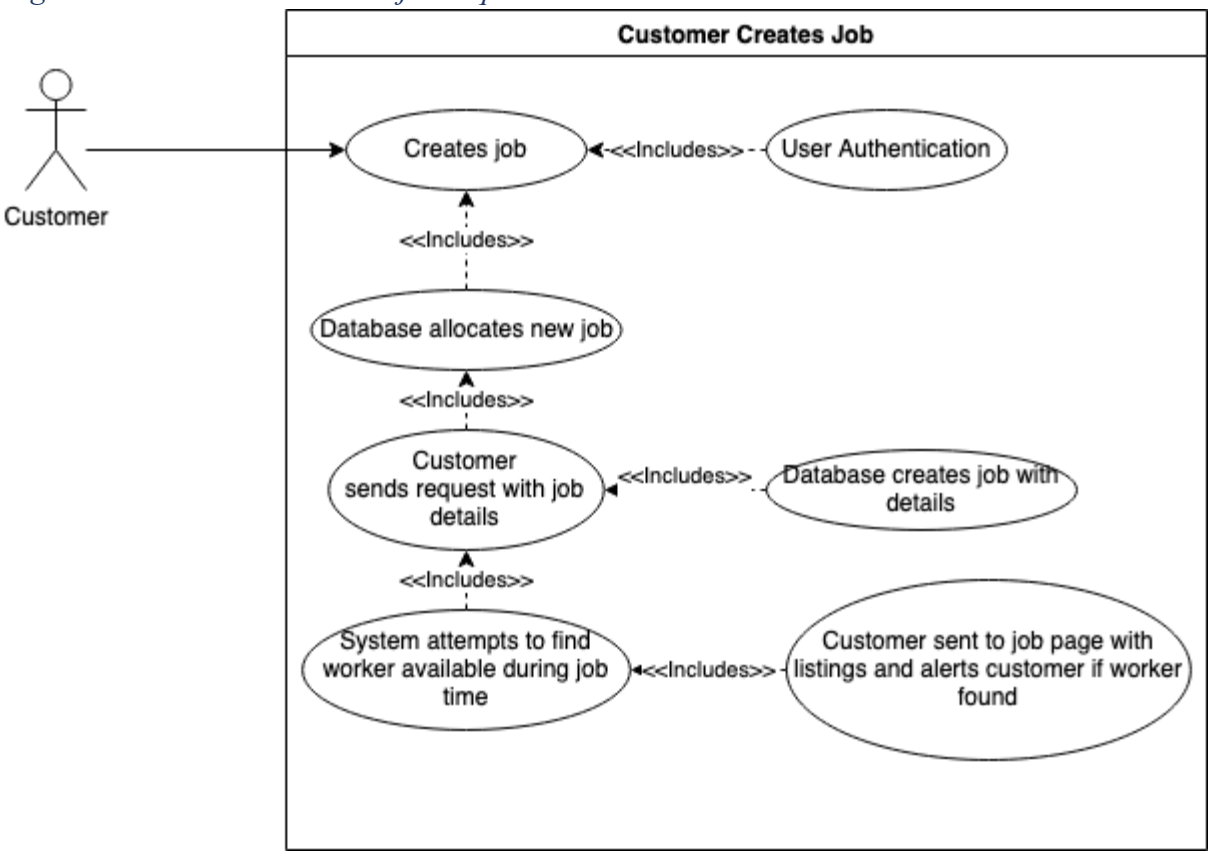
Exit conditions:

- The user creates a new account
- The user cancels account creation

Event flow:

1. The customer clicks on the link to create an account
2. The customer fills in the necessary information
    1. The system validates the information with instruction as needed
3. The system attempts to create an account with appropriate access
4. The user is notified if the account was created or not

Figure 2 – Customer creates a job request



Participating actors: Customers

Entry conditions:

- The customer is logged in and clicks ‘create new listing’

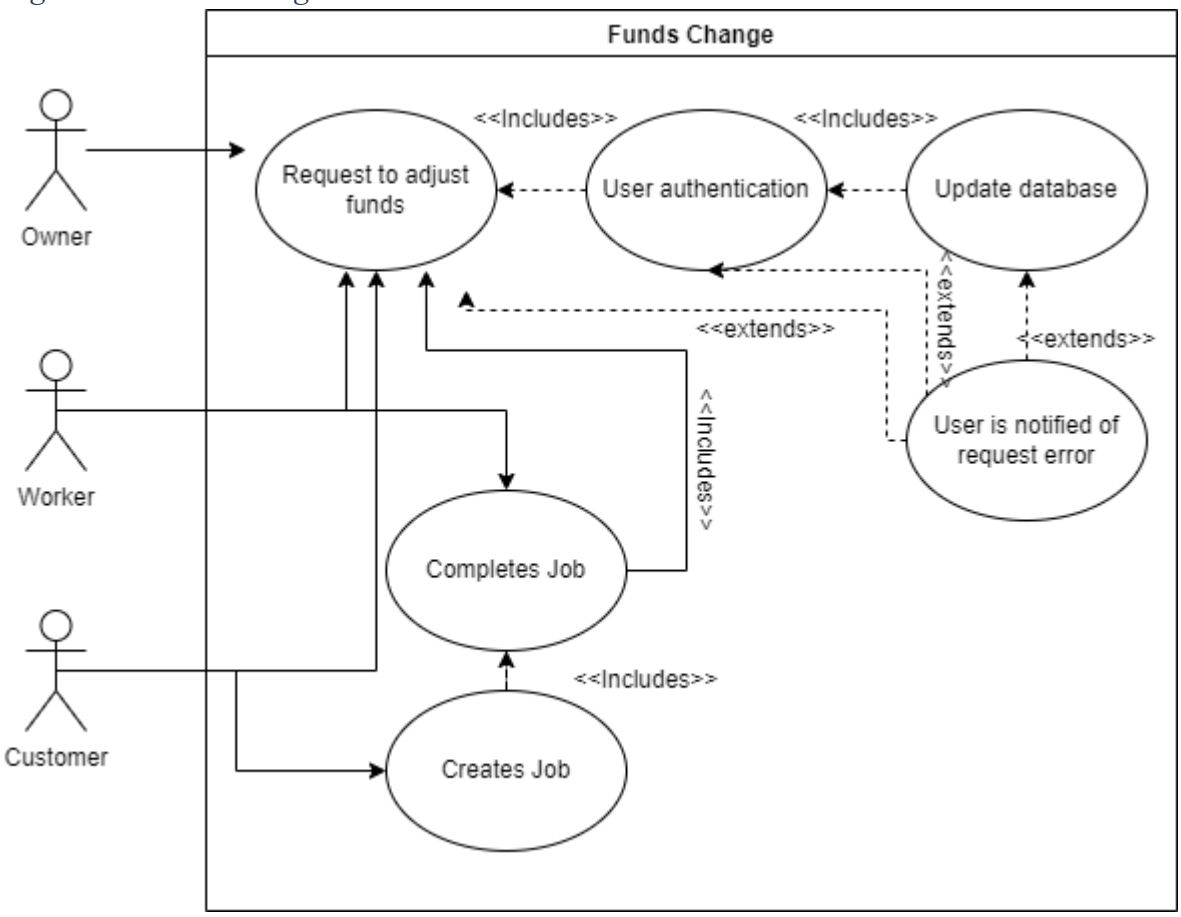
Exit conditions:

- The customer creates a new job request
- The customer cancels the creation of the job request

Event flow:

1. The customer logs in
2. The customer requests for job creation from the ‘jobs’ page
3. The customer inputs information pertinent to the job
    1. The customer can cancel their request at any point in this process
    2. If the customer follows through with the request, they are notified that the job was created correctly

Figure 3 – Funds change



Participating actors: Customers, Owner, Workers

Entry conditions:

- The user is logged in

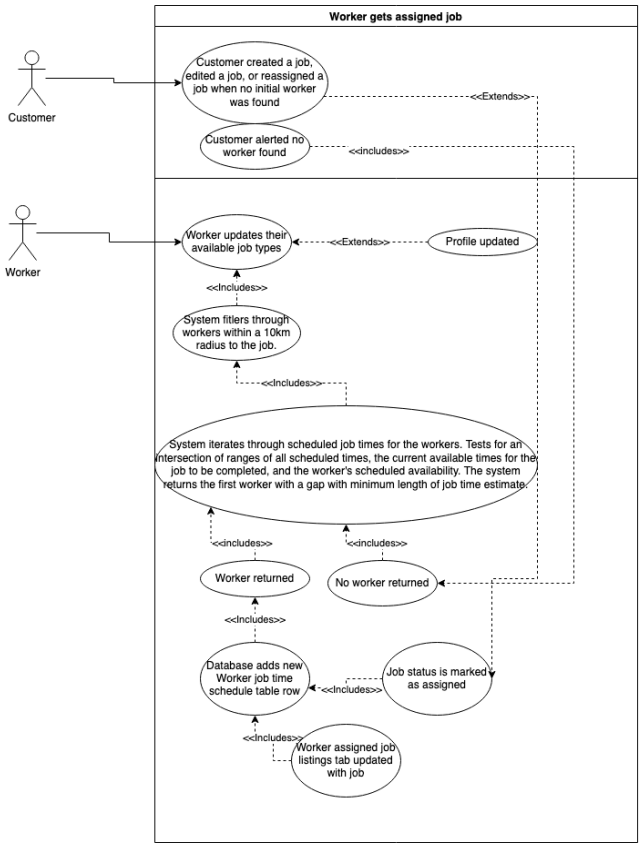
Exit conditions:

- Funds are moved around and user balances change

Event flow:

1. A user signs in and goes to the edit account page OR a customer creates a job
2. A user requests that funds be added to the count, either by editing the account or job completion
3. Funds are changed in the database and reflected in the UI

Figure 4 – Worker is assigned a job



Participating actors: Workers, Customers

Entry conditions:

- The customer creates a job listing
- The customer edits a job listing
- The customer reassigns a job listing that was initially unassignable

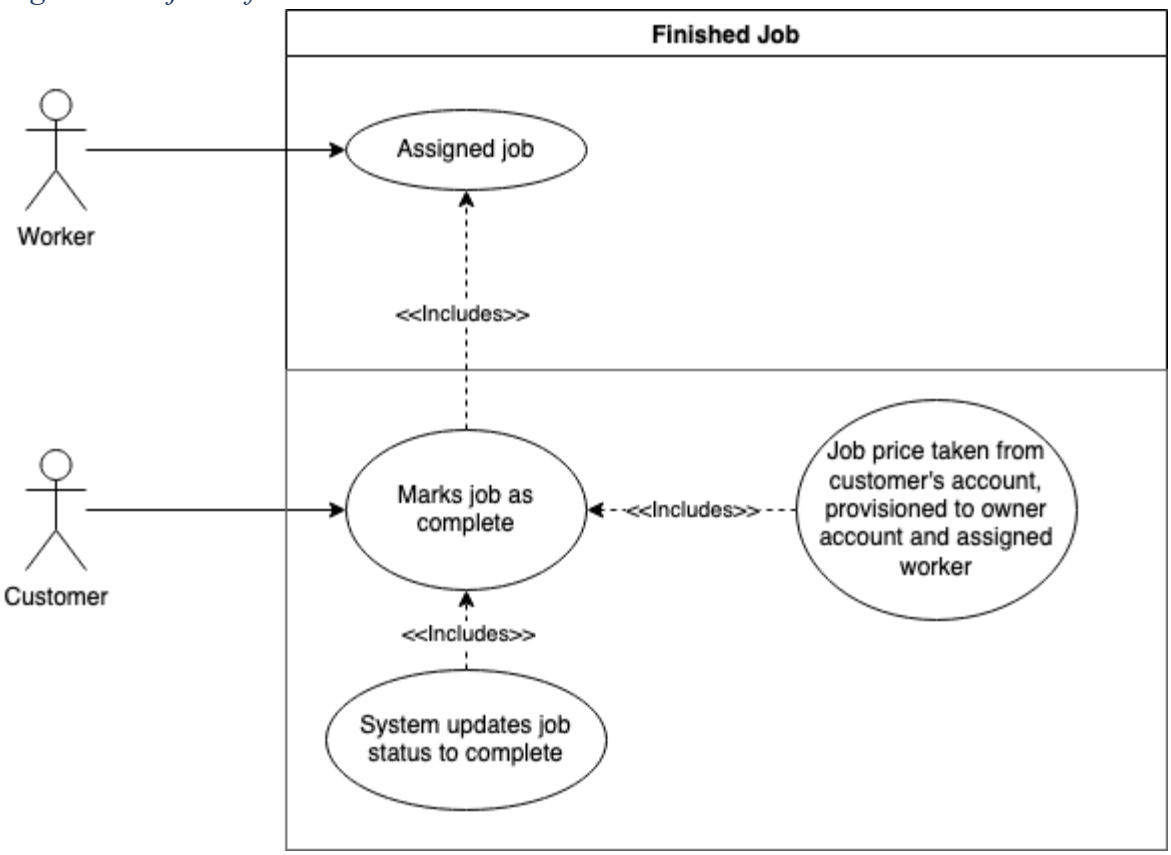
Exit conditions:

- A worker is assigned the job
- There are no available workers so the customer is alerted and will have to try again later

Event flow:

1. The customer creates a job
2. The system filters workers by applicable criteria
    1. The job is assigned to the worker with the first applicable availability that has a time block greater than or equal to the time estimate, and does not conflict with a previous job assignment.
    2. The job is not assigned at all and customer is alerted

Figure 5 – A job is finished



Participating actors: Worker, Customer, Owner

Entry conditions:

- The customer indicates that the job is finished

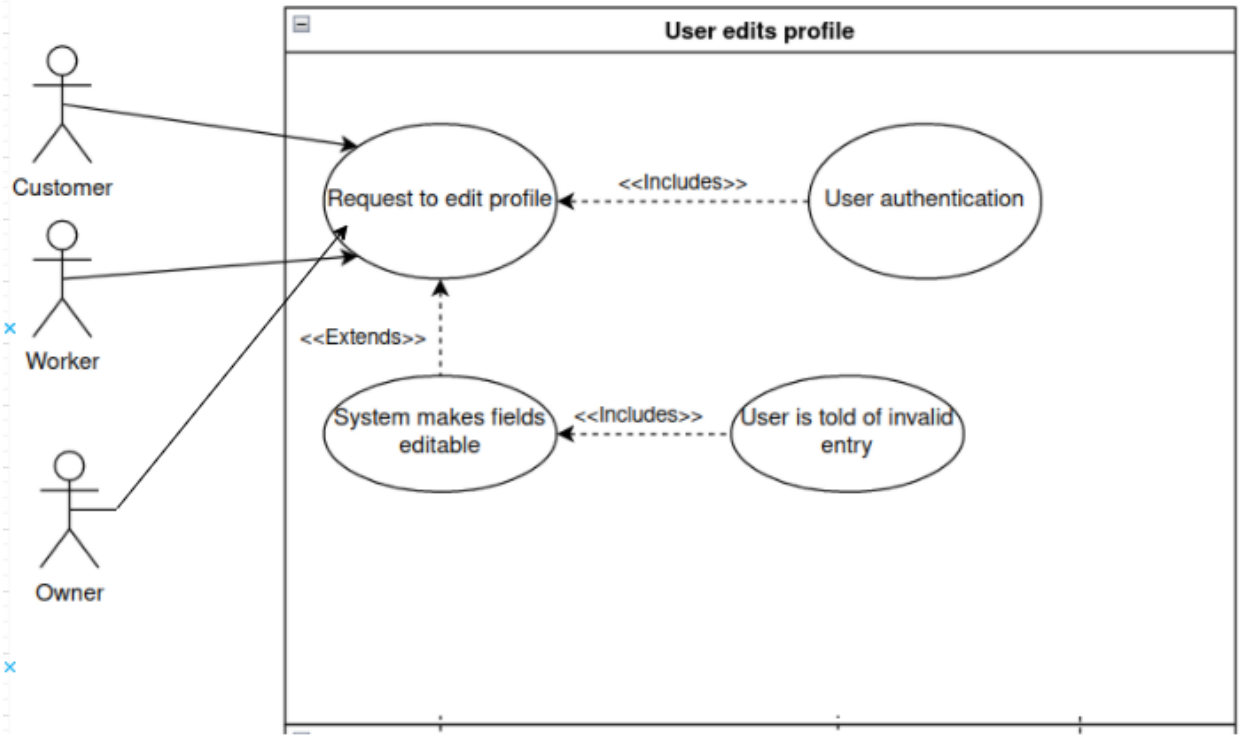
Exit conditions:

- The job is marked complete, and funds move
- The job is not marked complete, and funds do not move

Event flow:

1. The customer indicates that the job is done
2. The worker is paid for their work, the owner receives a commission
3. The system marks the job as completed
4. The customer reviews the work done and enters a review into the app
    1. The customer or worker can file a dispute if needed

Figure 6 – User edits profile



Participating Actors: Workers, Customers, Owner

Entry conditions:

- The user is logged in and clicks on their profile

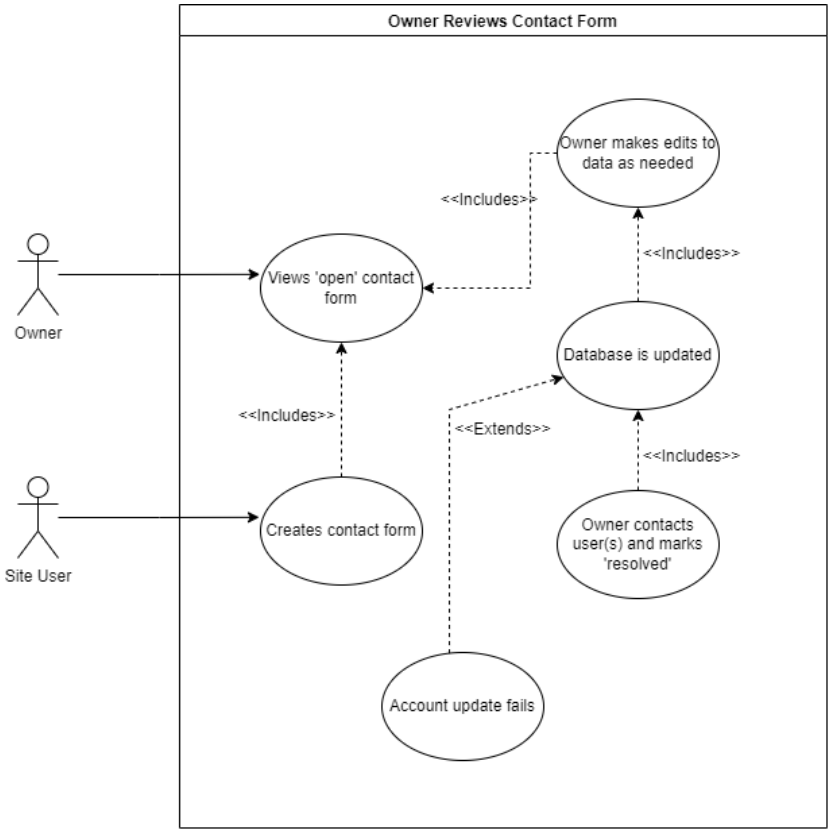
Exit condition:

- The user is notified whether their requested changes were accepted
- The user cancels the process, and no changes are made

Event flow:

1. A user logs in to their account
2. They request a change to their account information
    1. Owners may request a change to any users account information
3. The input is validated
4. The changes are made
    1. The user is notified if updates are unsuccessful

Figure 7 – Owner reviews a contact form



Participating actors: Owner

Entry conditions: The owner enters the contact form tool from the owner portal

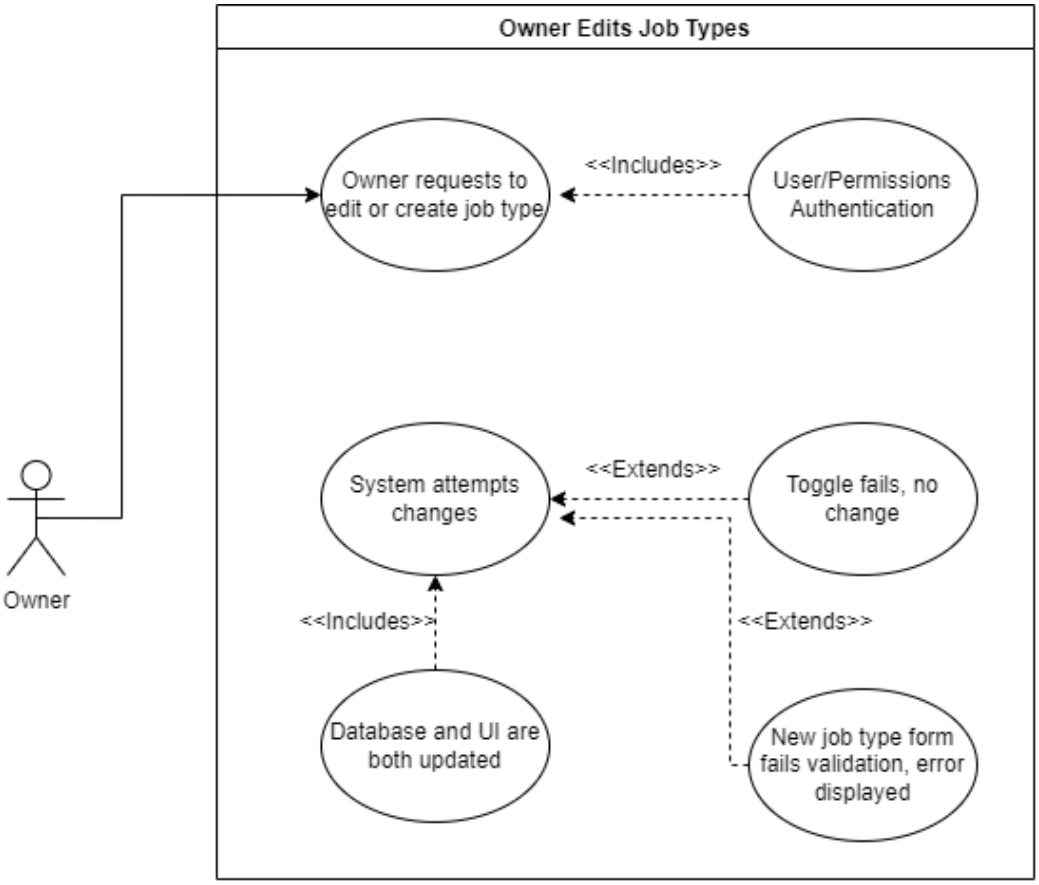
Exit conditions:

- The owner chooses to ignore the form
- The owner updates the status of a dispute from ‘open’ to ‘resolved’

Event flow:

1. The owner logs into their account and goes to the owner portal
2. The owner enters the contact form tool and views a form
3. The owner may choose to contact the user and edit jobs/profiles as needed
4. The owner marks the dispute ‘resolved’

Figure 8 – Owner edits job types



Participating actors: Owner

Entry conditions: The owner is logged in and accesses the job types tool from the owner portal

Exit conditions:

- The job status is created OR toggled between ‘active’ or ‘archived’
- The edits fail, an error is displayed, and nothing is changed
- The owner closes the screen before editing or adding a job type

Event flow:

1. The owner logs in to their account and requests to view job types from the owner portal
2. The owner requests to toggle a job type, or add a new one
    1. If adding a new one, the information is validated when saved and an error may be displayed
3. The owner can cancel changes before they are saved or save the changes to the database
4. Otherwise, the job type is either created or edited successfully, or an error is displayed

Functional Requirements

This section focuses on requirements related to how the system is expected to behave.

1. User Authentication and Account Management
    1.1. The initial page will ask the user to sign in or sign up.
    1.2. Registration
        1.2.1. If the user clicks on the signup button, they will be taken to a page with a form.
            1.2.1.1. Users must sign up with a name, password, email, phone number, avatar, and role. An initial balance and address (or coordinates) are optional.
            1.2.1.2. Users will have the option to add a profile picture or graphic.
            1.2.1.3. There will be a button to confirm registration.
                1.2.1.3.1. If there are missing fields or other errors, a message will be displayed back to the user.
            1.2.1.4. Upon a successfully created account, workers will be taken to an extended registration page.
                1.2.1.4.1. Worker Registration
                    1.2.1.4.1.1. A worker must enter their availability.
                    1.2.1.4.1.2. A worker must enter the types of jobs they're willing to accept.
            1.2.1.5. Upon completing the registration process the user is taken to the home page.
    1.3. Logging in
        1.3.1. Users with active accounts will sign in with an email and password each session.
            1.3.1.1. There will be fields for the email and password on the initial page, along with a button to sign them in.
        1.3.2. Incorrect attempts will result in an error being displayed back to the user.
            1.3.2.1. The error will let the user know that the email or password was invalid, or that another error was encountered, prompting them to try again.
    1.4. Account Management (All Users)
        1.4.1. A user must be logged in to access account management features.

1.4.2. Any type of user will be allowed to change any account details aside from the account type. This includes email, password, phone number, address/coordinates, and profile picture.

1.4.3. A user must be able to view and edit account balance.

1.4.4. Workers must be able to edit availability.

2. Customer Account Features

2.1. The key functionality of this user is the ability to post job offers using a built-in template accessible from a home page.

2.1.1. They will be able to specify job type (e.g., lawn mowing)

2.1.2. They will be able to specify a time frame for when the job is to be completed.

2.1.3. They will have the ability to specify the estimated time for completion.

2.1.4. They will have the ability to specify payment.

2.1.4.1. They must have enough funds in their account for the job to be created.

2.2. They will have the ability to add money to their account.

2.2.1. Note that for the scope of this project, we will not be using real money. Instead, they will enter an arbitrary number for the system to work with.

2.3. They must have the ability to open a dispute on specific jobs for the owner to review.

3. Worker Account Features

3.1. The key functionality to this type of user is the ability to be assigned to jobs posted by customers.

3.2. They must be able to set and edit their availability to take jobs, meaning they will only be assigned jobs that fit within their set availability.

3.3. They must be able to set and edit which types of jobs they are willing to be assigned.

3.4. They must be able to view jobs assigned to them by the system.

3.5. They will receive 90% compensation for each completed job.

3.5.1. Each job will display the amount of compensation they will receive.

3.6. They can mark when a job has been completed.

3.6.1. This will result in funds being transferred to their account.

3.7. They must be able to open a dispute on specific jobs for the owner to review.

4. Owner Account Features

4.1. A special account will be created for the system owner, and an owner account cannot be created through the typical registration process.

4.2. They will receive 10% of compensation for each completed job.

4.3. They will have a portal only accessible to them for system management purposes.

4.3.1. They can add and archive types of jobs.

4.3.1.1. This would involve adding and archiving job types from the drop-down list customers see when posting a job offer.

4.3.1.2. Adding a job would allow customers to create job offers with that type and workers to declare they are willing to do the job type.

4.3.1.3. Archiving a job will not affect current job offers but will remove it from the form when creating new jobs.

4.3.2. They have the privilege to edit user accounts, including the ability to move money between accounts for reimbursement purposes.

4.3.3. They can view contact forms and mark them as open or resolved.

4.3.4. They can view all jobs in the system along with the job details.

4.4. In addition to these special features, they will also have the same access to processes that customer and worker accounts have.

5. The application should be somewhat usable on mobile.

Non-Functional Requirements

This section focuses on requirements related to how the system will work.

1. The system must utilize a database.
    1.1. The database will store user information.
        1.1.1. This includes permissions, names, password hashes, emails, addresses (and coordinates), phone numbers, account balances, and profile pictures.
            1.1.1.1. Worker accounts will store availabilities and job types they're willing to accept as well.
    1.2. The database will be used to store information on job postings.
        1.2.1. This includes job type, location, estimated completion time, compensation, desired completion window, customer, customer contact information, status, extra notes, and worker (if assigned).
            1.2.1.1. The customer contact information must only be accessible by the customer who created the offer, the worker assigned to it, and the owner.
2. The system design must be data-driven and dynamic.
    2.1. Updates to the database should affect other areas of the system.
3. The team will use Git as a version control system, with GitHub.
    3.1. The team will use GitHub Project to manage backlogs.
4. The system must be deployable either locally or by a cloud service.

Future Features

This section focuses on possible additions for future versions that aren't necessary to the MVP.

1. The system could implement a filter by location feature to account for the distance between workers and customers.

    1.1. Workers would have the option to set maximum distances.

2. The system could implement a preference system where customers could add 'favorite' workers to give priority to for their job offers.
3. The system could implement a means for user ratings.
4. The system could implement the ability for a customer to set up a recurring job offer.
5. An alert system via email/SMS/Push could be implemented to let users know when jobs are available and accepted.
6. The database could track the history of jobs and a report could be displayed back to users.
7. The system could implement a tip system to adjust wages.

Glossary

This section defines a list of terms relevant to the project.

Owner - a user with elevated privileges that profits from connections and can manage jobs, job types, user accounts, and user conflicts

Worker - a user that is looking to complete job offers for compensation

Customer - a user that is looking to provide compensation for a job offer

Job - a specific task related to yard work managed by the owner and specified by the customer for a worker to complete

System/Program - the software application this project aims to create; the product being built

User - an owner, customer, or worker with specific access according to the user type

Database - storage for system data to be stored (e.g., user and job details)