Requirements

Introduction and Context

Finding time for lawn care can be a hassle and looking for lawn care workers can be even more of a struggle. To help fix this problem, this project aims to provide an application for homeowners to make finding lawn care workers stress-free and easy.

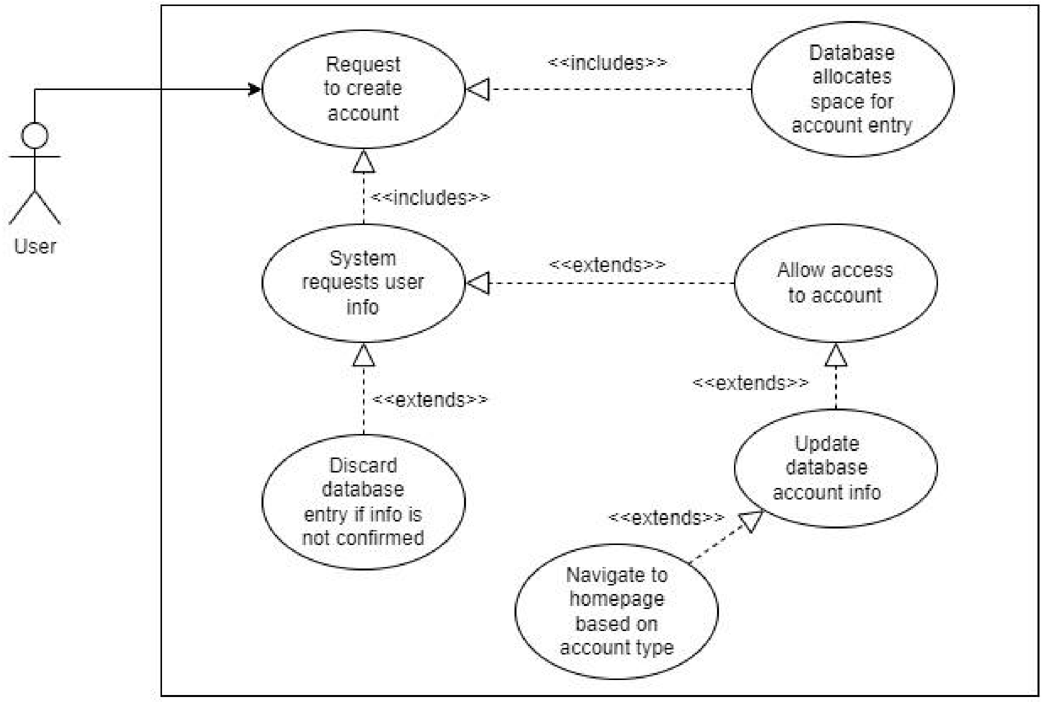
In this app, users will be able to post a job to the app where local workers will be able to then view the job. The posted job will contain the job description, pictures of the lawn, location of the home, the payout, estimated time of labor, when the job needs to be done, and any other additional notes the worker will need to know. The job post will then be available for the all workers to see until someone accepts the job.

The worker will then go do the job based on the information from the job post. Once completed, the payment will be processed, and the homeowner will be able to leave a rating for the worker. If the job was not completed, the homeowner will be able to submit a complaint stating how the job was not completed, and if approved, the homeowner will be refunded.

By giving workers and homeowners a platform like this, finding lawn care jobs should never be a problem again.

Users and Their Goals

User Creates an Account


User
Request to create account
<<includes>>
Database allocates space for account entry
<<includes>>
System requests user info
<<extends>>
Allow access to account
<<extends>>
<<extends>>
Discard database entry if info is not confirmed
Update database account info
<<extends>>
Navigate to homepage based on account type

Participating Actor: User

Entry Conditions:

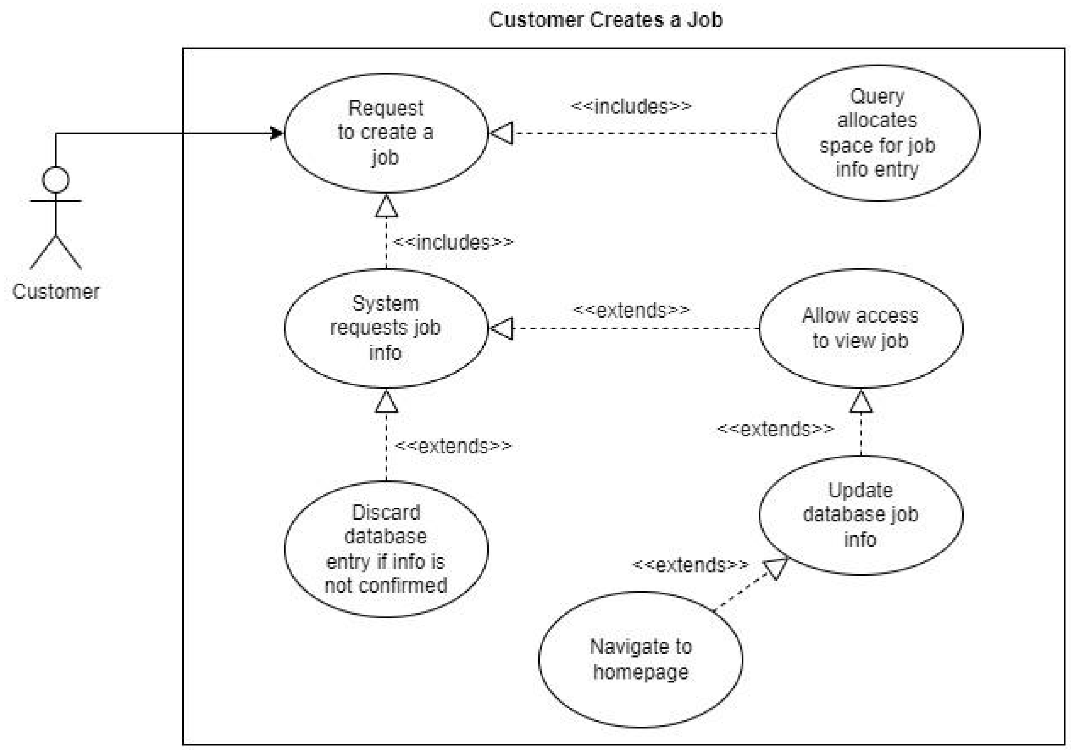
- User wants to create an account

Exit Conditions:

- Account is created
- User doesn't sign up and account is not created

Event Flow:

- User requests to create an account
- System requests the user's information
  - If user signs up, account is created and they're navigated to homepage
  - If user cancels, account is not created
- System updates users database
- User is navigated to the homepage of the account they created



Participating Actor: Customer

Entry Conditions:

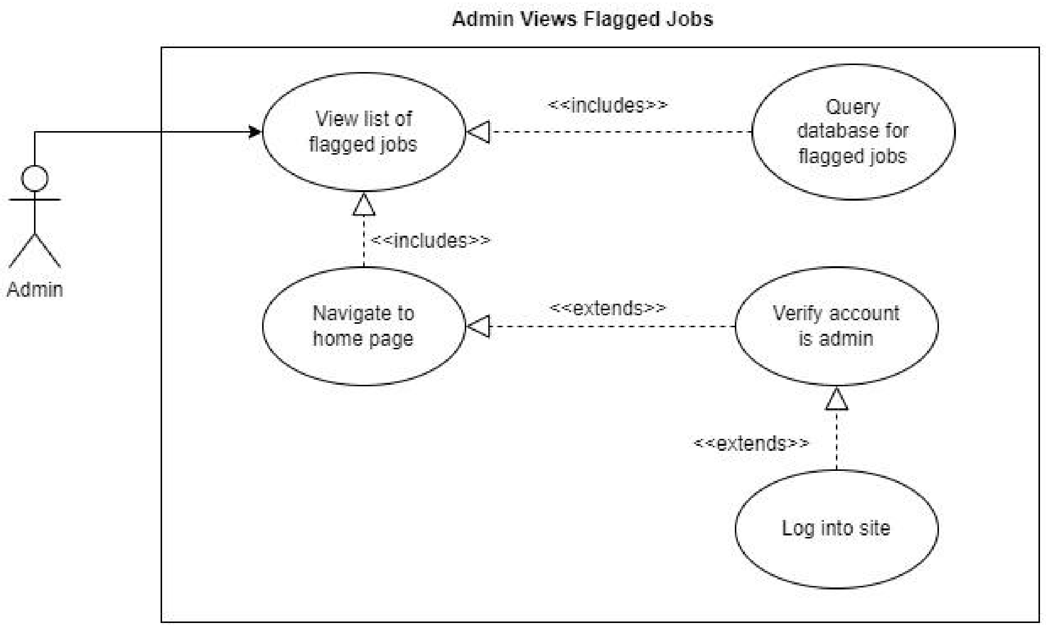
- Customer wants to create a job
- Account type is Customer

Exit Conditions:

- Job is created
- Job is canceled

Event Flow:

- Customer requests to create a job
- System requests job information
- Customer inputs information
    - Customer confirms
    - Customer doesn't confirm and job is canceled
- System updates jobs database
- Customer navigates back to homepage



Participating Actor: Admin

Entry Conditions:

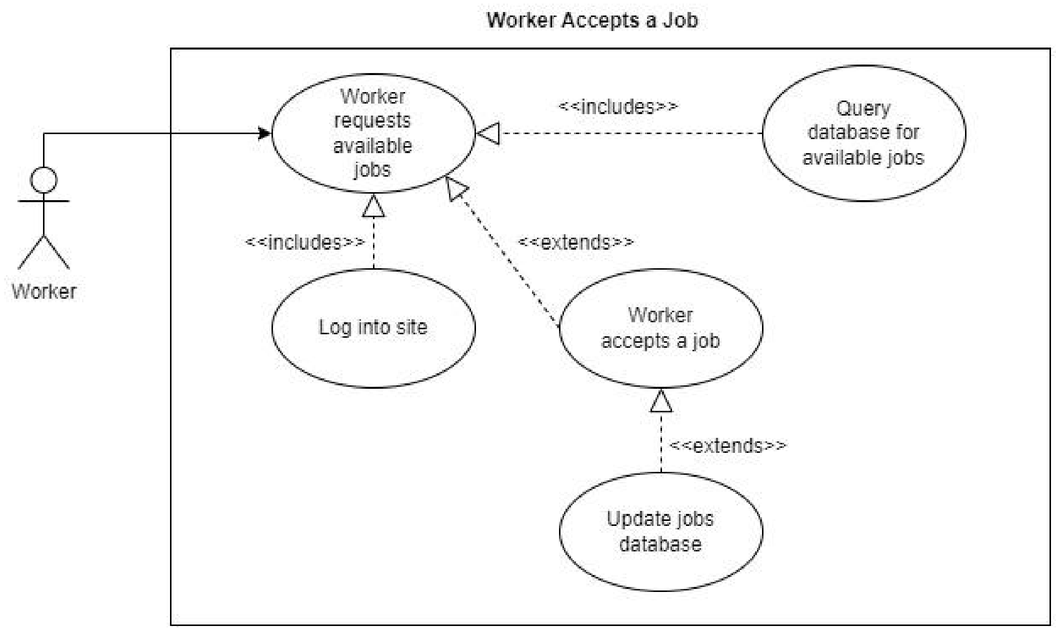
- Verify account is Admin

Exit Conditions:

- Admin views flagged jobs

Event Flow:

- Admin logs into account
- Admin requests to view flagged jobs
- System displays flagged jobs



Participating Actor: Worker

Entry Conditions:

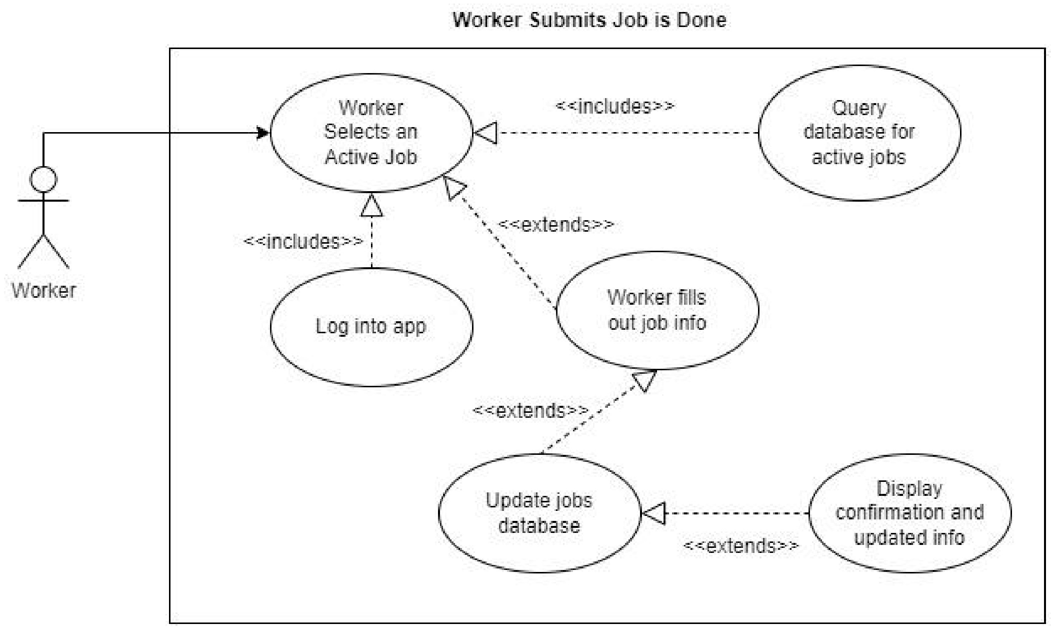
- Verify account is Worker
- Worker visits available jobs page

Exit Conditions:

- Worker accepts a job
- Worker doesn't accept job

Event Flow:

- Worker is logged in
- Worker visits available jobs
- Worker accepts a job
- System updates jobs database


Worker Submits Job is Done
Worker
Worker Selects an Active Job
<<includes>>
Query database for active jobs
<<extends>>
<<includes>>
Log into app
Worker fills out job info
<<extends>>
Update jobs database
Display confirmation and updated info
<<extends>>

Participating Actor: Worker

Entry Conditions:

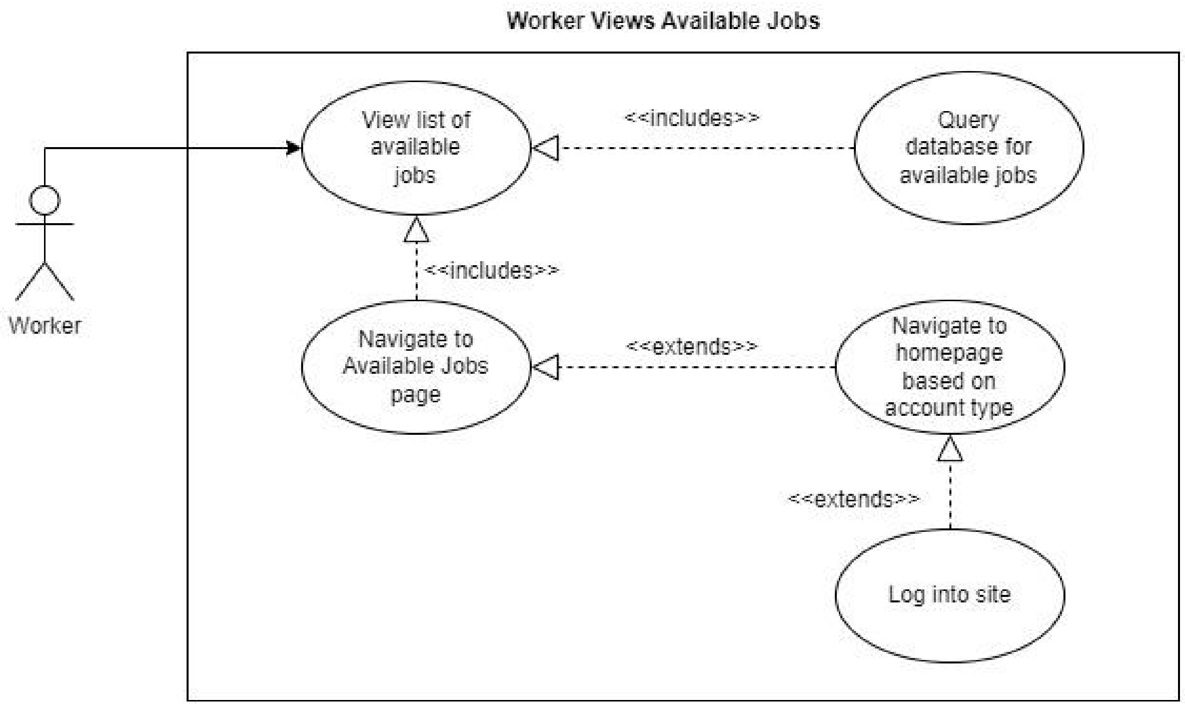
- Worker has active job

Exit Conditions:

- Worker submits job

Event Flow:

- Worker logs into account
- Worker selects active job and submits job is done
- System transfers money into Worker's account


Worker Views Available Jobs
Worker
View list of available jobs
<<includes>>
Query database for available jobs
<<includes>>
Navigate to Available Jobs page
<<extends>>
Navigate to homepage based on account type
<<extends>>
Log into site

Participating Actor: Worker

Entry Condition:

- Verify account is Worker

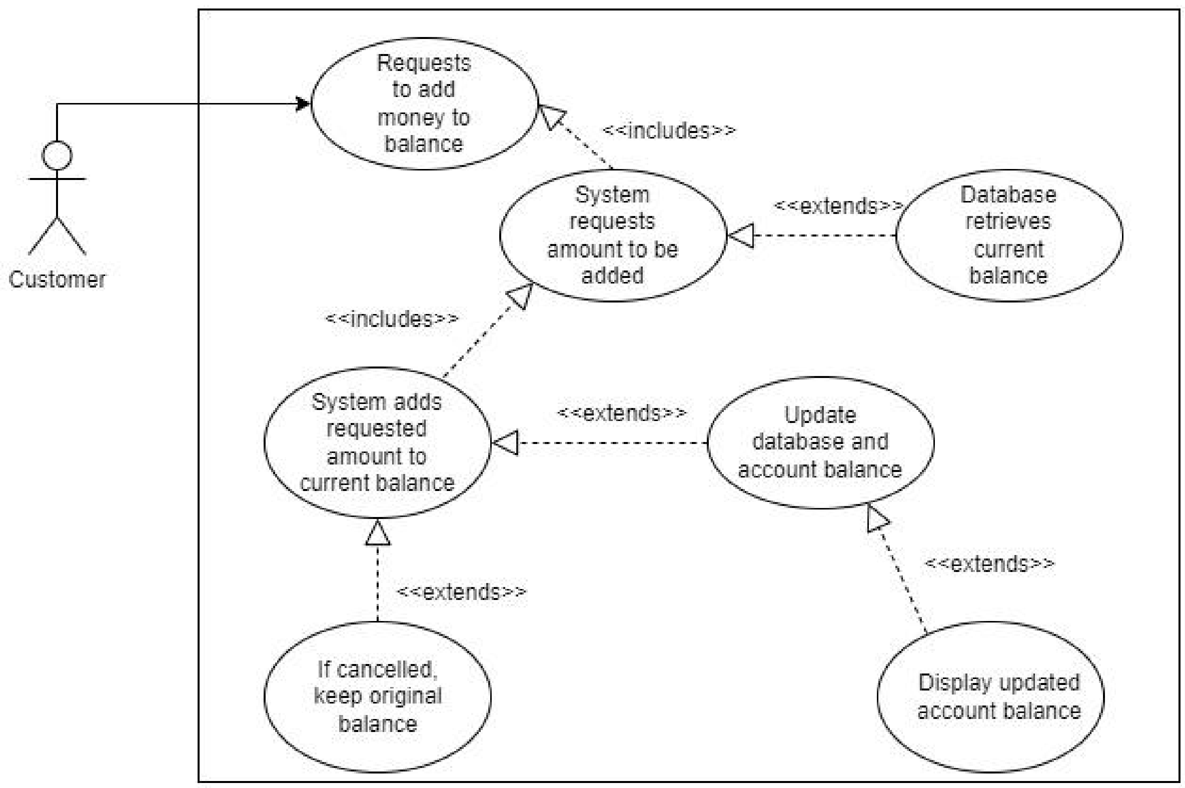
Exit Condition:

- Worker views available jobs

Event Flow:

- Worker logs into account
- Requests to see all available jobs
- System displays available jobs

Customer Adds Money to Account Balance


Customer
Requests to add money to balance
System requests amount to be added
<<includes>>
Database retrieves current balance
<<extends>>
<<includes>>
System adds requested amount to current balance
<<extends>>
Update database and account balance
<<extends>>
<<extends>>
If cancelled, keep original balance
Display updated account balance

Participating Actor: Customer

Entry Conditions:

- Verify account is Customer

Exit Conditions:

- Money is added to balance
- Action is cancelled

Event Flow:

- Customer requests to add money to balance
- Customer inputs the money they wish to add
- System saves new account balance
    - If cancelled, no changes are made to the balance

Functional Requirements

1. User Authentication and Access
    - 1.1 The user must login to the app using an email and password
        - 1.1.1 If it’s the user’s first login, they must create an account inputting their email and password in the sign up page
            - 1.1.1.1 The app will make sure the account information is secure
        - 1.1.2 If it’s not the user’s first login, they will be given access to the app. If if anything is incorrect in the login process, the user can try again
    - 1.2 Users can have 3 different accounts: Admin, Worker, Customer
        - 1.2.1 The Admin is the owner of the application. It will not be accessible to the public
        - 1.2.2 A Customer is the homeowner and the one needing the lawn care
        - 1.2.3 A Worker is the person who will accept the lawn care job for the Customer

2. User Profile
    - 2.1 The user profile will display needed information about the user where they can view the information
    - 2.2 Login credentials will be viewable to the user
    - 2.3 Their location will be viewable here to the public
    - 2.4 Their balance will be shown on the user profile page where they can deposit into their account

3. Owner/Admin Features
    - 3.1 The owner account that contains admin features will not be accessible to the public
    - 3.2 Owners will have the power to interfere with employer/worker interactions if it Is needed
        - 3.2.1 They will be able to review disputes sent to the app
        - 3.2.2 If appropriate, they can reverse transactions that have taken place

4. Customer Features
    - 4.1 Creation of a Customer account
        - 4.1.1 Customers will give their name, email, and account password upon creation of an account
    - 4.2 Creation of a job
        - 4.2.1 When creating a job the user will enter a location that the job will take place at
        - 4.2.2 The user will give a description of the job that is needed
        - 4.2.3 A picture of the job is not required, but will be recommended
        - 4.2.4 A payout will be selected here for when the job is finished
        - 4.2.5 The job will be available for all Worker to see in the 'Available Jobs' page
    - 4.3 After the job
        - 4.3.1 After the job, the transaction will process automatically
        - 4.3.2 If the user is not satisfied witht he job, they will be about to dispute the transaction
            - 4.3.2.1 They will give a description of the issues that occurred
            - 4.3.2.2 An owner will review the dispute and decide (See 3)

5. Worker Features
    - 5.1 Creation of a Worker account
        - 5.1.1. Workers will give their name, email, and account password upon creation of an account
        - 5.1.2 They will be able to see their own balance
    - 5.2 Viewing a job
        - 5.2.1 When a job is created by a Customer, a Worker will be able to see it in the 'Available Jobs' page
        - 5.2.2 Before accepting, the worker will be able to view the job information
        - 5.2.3 The worker will have the option to decline the job
        - 5.2.4 If accepted, the worker will be assigned that job, and the job will be moved into the 'Active Jobs' page

5.3 After the job

5.3.1 Once the job is done, the transaction will go through

5.3.2 The worker will be able to leave final comments

5.3.3 If the Customer is unhappy, they can dispute the transaction (See 4.3)

5.3.4 If everything goes through, then the job is terminated

Non-Functional Requirements

1. The system must use a database

1.1 The system's database must store the user account information, including the following fields: Name, Email, Password, Balance

1.2 They system must store information about the location of the customer/worker represented in the system

1.3 The system must keep track of jobs that are currently open

2. The team will use the Git version control system, with GitHub as a remote repository

3. The system must be deployable

3.1 Can be downloaded onto an Android device

Future Features

This section contains a list of features that are beyond the scope of the project but could be implemented in future versions.

1. The systems interface could be converted to a map where the locations of the workers nearby are shown
2. The app could have a chat box that appears for the Worker who accepts the job to be able to communicate with the Customer throughout the app
3. The app could show a progress bar to the Customer so they can see how close they are to having their job completed

Glossary

This section contains a list of important terms and their definitions

User - refers to any of the three types of users in the system (Customers, Workers, Owner

Customer - a user that uses the system to receive yard work (pays)

Worker - a user that uses the system to find yard work (receives money)

Owner - a user that receives and manages the transferring of money

System - refers to the application that the project aims to build