Group 9

Yardwrk

Ian Adams: Database Enthusiast
Jake Epperson: Design Lead
Quinn Ormond: Project Manager
Dave Storey: Backend Lead

Project Overview

Yardwrk creates an easy to use interface where cache valley residents can post outside jobs/chores that need to be performed in exchange for payment. Individuals (especially teenagers and young people) looking to earn money can sign up and complete these jobs.

The goal of this project was to create a website that allows customers to hire individuals to do various yard working jobs. Customers would be able to submit jobs that they want done, and workers would be able to accept these jobs in exchange for money. Importantly, the hard working website owners will receive 10% of all transactions.

Major Design decisions.

When designing the website we wanted a very easy and convenient structure that would allow anyone to use the website. It seems apparent that many customers will likely be older. As such we deemed it prudent to take a very straight to the point kind of design with the hopes that even those not well versed in technology and websites will easily be able to find a worker. Very simple design.

Following this idea we created dashboards that allow you to quickly view any information that you are readily seeking.

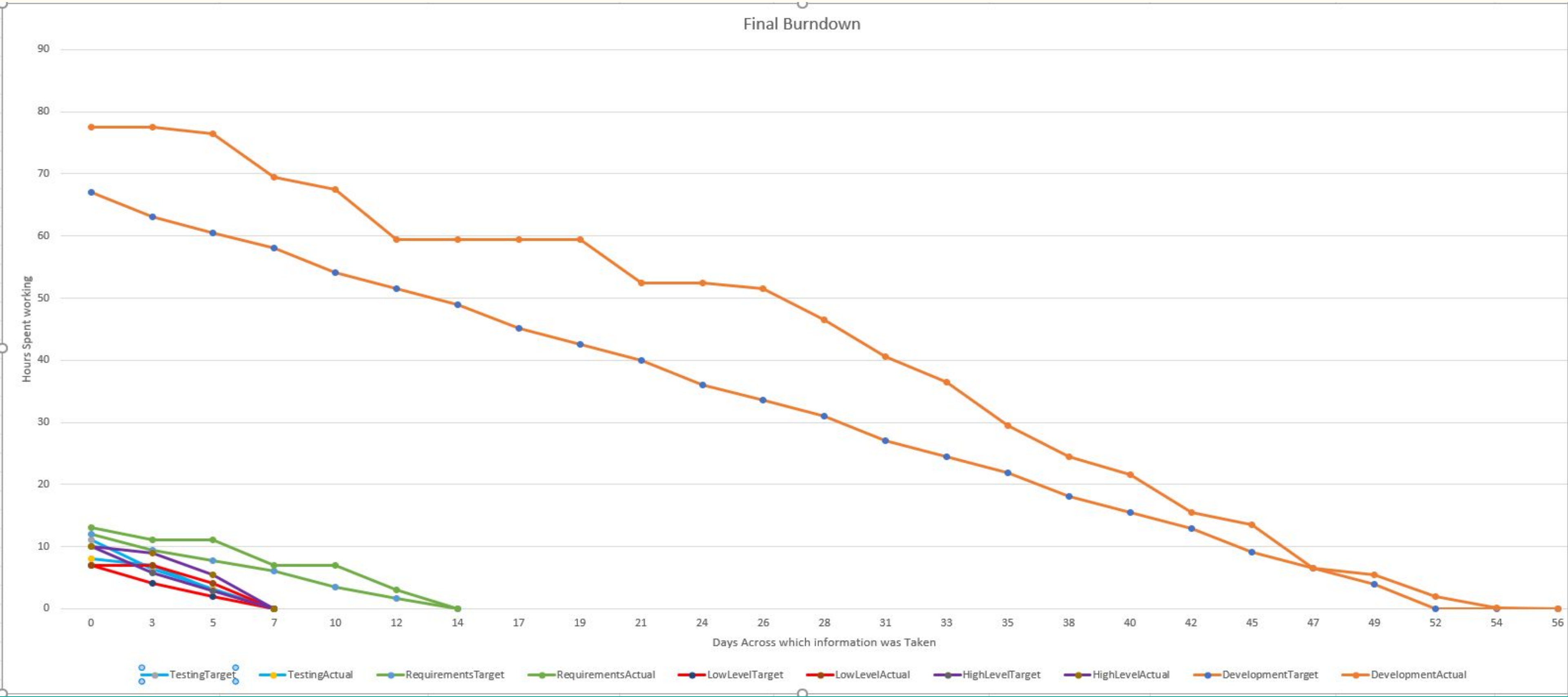
This reflected our abilities well as we are not used to this large of group projects. So instead of trying to connect a large amount of webpages, we created a few and spent our time on them, similar to how we are used to assignments in classes.

Adapting agile practices to improve productivity.

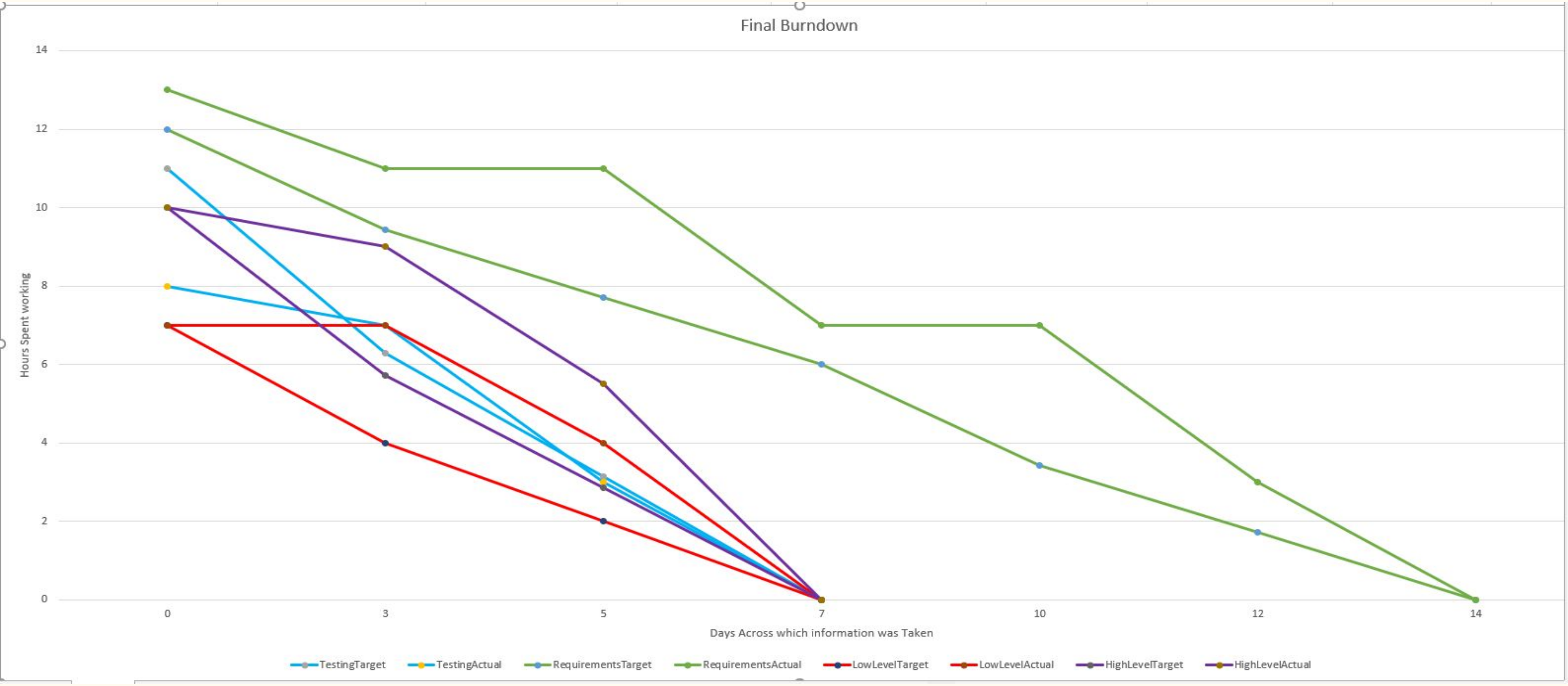
When we first began using the scrum board and issues, we honestly found it to be kind of a hassle. By the end of the project, it was the first thing we sought out to do. Even though sprint 4 never required proof of a sprint meeting, we did so.

I believe that when looking at our sprint practices you should view them as a learning curve that we found successful. As we practiced sprint productivity we believe that we got much better at showing what we're doing (backlog info.) By doing so we also got further off in in sprint velocity. We found it to be more helpful and at the same time our estimates didn't immediately get much better. More practice will be necessary.

Burn Down


Final Burndown
Hours Spent working
90
80
70
60
50
40
30
20
10
0
0
3
5
7
10
12
14
17
19
21
24
26
28
31
33
35
38
40
42
45
47
49
52
54
56
Days Across which information was Taken
TestingTarget
TestingActual
RequirementsTarget
RequirementsActual
LowLevelTarget
LowLevelActual
HighLevelTarget
HighLevelActual
DevelopmentTarget
DevelopmentActual

Burn Down (A better look shorter information)



Velocity of each sprint

Sprint 1 velocity - 1.083333333

Sprint 2 velocity - 1.045454545

Sprint 3 velocity - 1.140350877

Sprint 4 velocity - 1.322580645

Overall velocity - 1.14792985

Ready for deployment

It’s time to get that money.

The website is beyond presentable at this point. It has the functionality to connect customers and workers quite quickly. Money transfer is working great. All functionality that was included in original design is over 96%.

With all four of our members actively digging through the site there is no bugs that will inhibit the users experience. Meaning that without the user actively trying to inhibit themselves they should have no capability to do so.

We are ready!

For Burndown Information as well as a better look at the charts. Follow this link.

https://github.com/davevstorey/9-usu-cs3450-sp2022/tree/master/docs/FinalBurnDown

Requirement 1: User Authentication and Access

A safe and secure user authentication system is essential to almost every web app. Requirement one specifies that any and all user information is only available to those who have been authenticated successfully. Additionally, users will have the ability to act as a customer, worker, or owner dependant upon their accounts privileges.

Requirement 1: User Authentication and Access

FURPS Description

Functionality - Create, sign into, and view an account securely

Usability - A login screen will give users appropriate instructions to either create or login to an account. Account information will be available to logged in users through a webpage

Reliability - The reliability of this feature largely depends on user authentication tools provided by the django framework

Performance - The django framework allows for quick and efficient user creation and authentication

Supportability - This system is fairly simple to maintain as the majority of its functionality is handled by the django framework leaving just the front-end capabilities left to the developer to support

MoSCoW Description

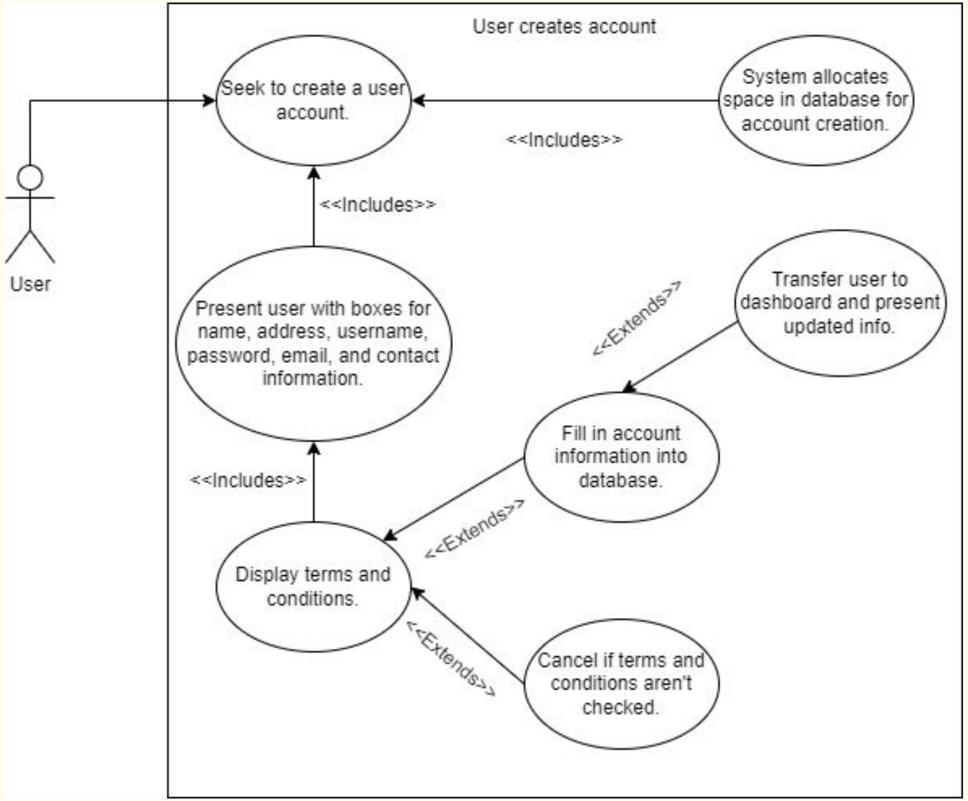
Must Have - The ability to create, sign into, and view an account. Account privileges.

Should Have - Safe and secure authentication. Protection against duplicate usernames and emails.

Could Have - The ability to change your password

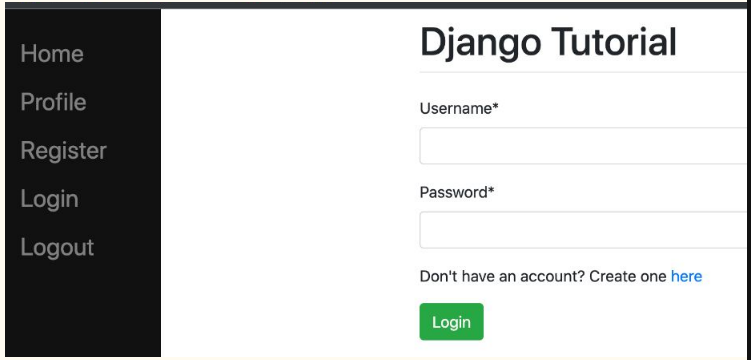
Won't Have - Email verification process

Requirement 1: Use Case Diagram


User creates account
Seek to create a user account.
System allocates space in database for account creation.
<<Includes>>
<<Includes>>
User
Present user with boxes for name, address, username, password, email, and contact information.
Transfer user to dashboard and present updated info.
<<Extends>>
Fill in account information into database.
<<Includes>>
<<Extends>>
Display terms and conditions.
<<Extends>>
Cancel if terms and conditions aren't checked.

Requirement 1: High Fidelity Prototype

Django project with custom user model and authentication system. Can be found in ‘docs/prototypes/highFidelity/djangoProject’.

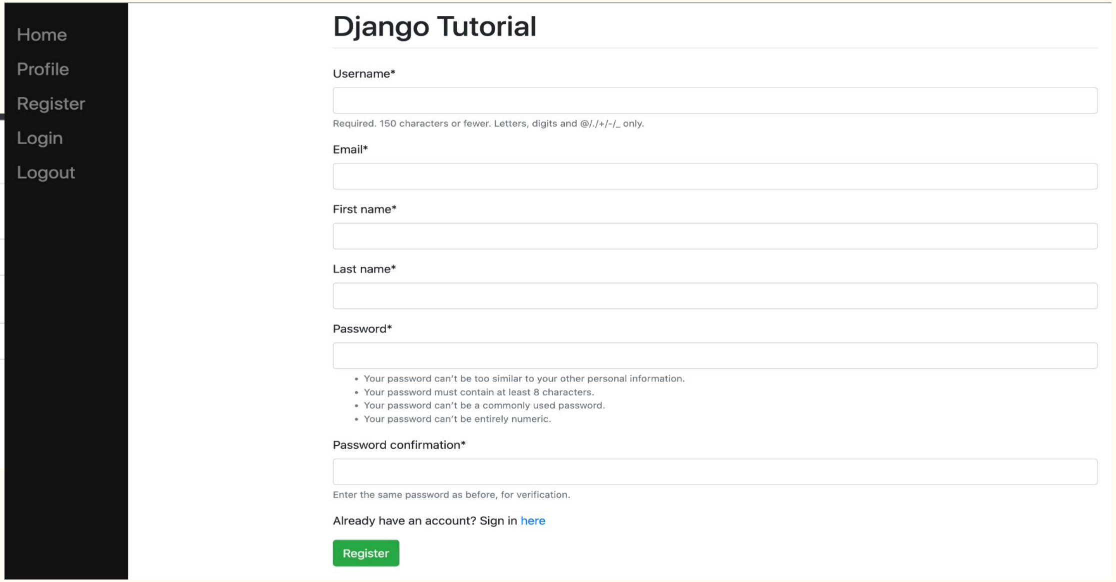


Django Tutorial

Username*

Password*

Don't have an account? Create one here



Django Tutorial

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

First name*

Last name*

Password*

- Your password can’t be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can’t be a commonly used password.
- Your password can’t be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Already have an account? Sign in here

Requirement 1: Scrum Tasks and Testing

Scrum Tasks

- User Model
- Sign In View
- Account Creation View
- Account view
- Account creation backend logic
- Login/Logout backend logic

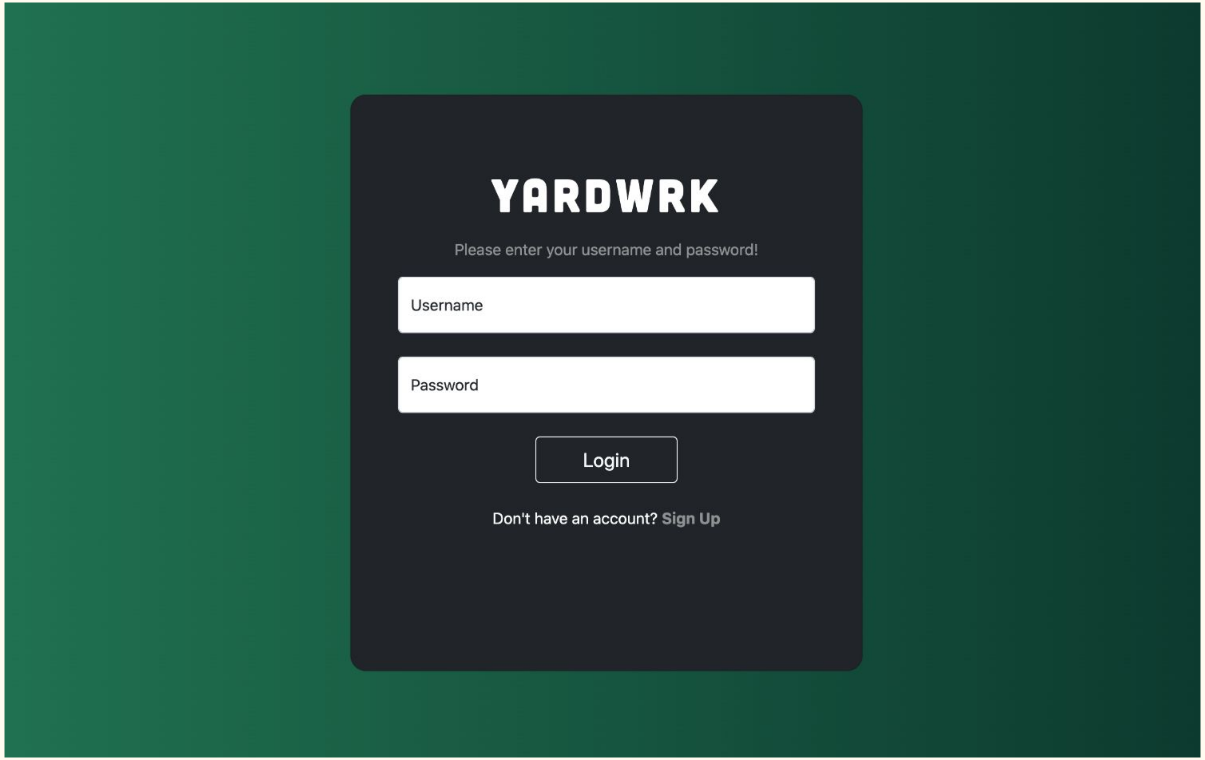
Testing

Testing was mainly done through regression testing. After new changes had been implemented we would verify that the requirements previously fulfilled would still pass.

You can find a test for this in docs/SystemTesting.pdf Index 1: User Authentication and Access

Requirement 1: Implementation

Sign in view with link to create an account if you don't have one already

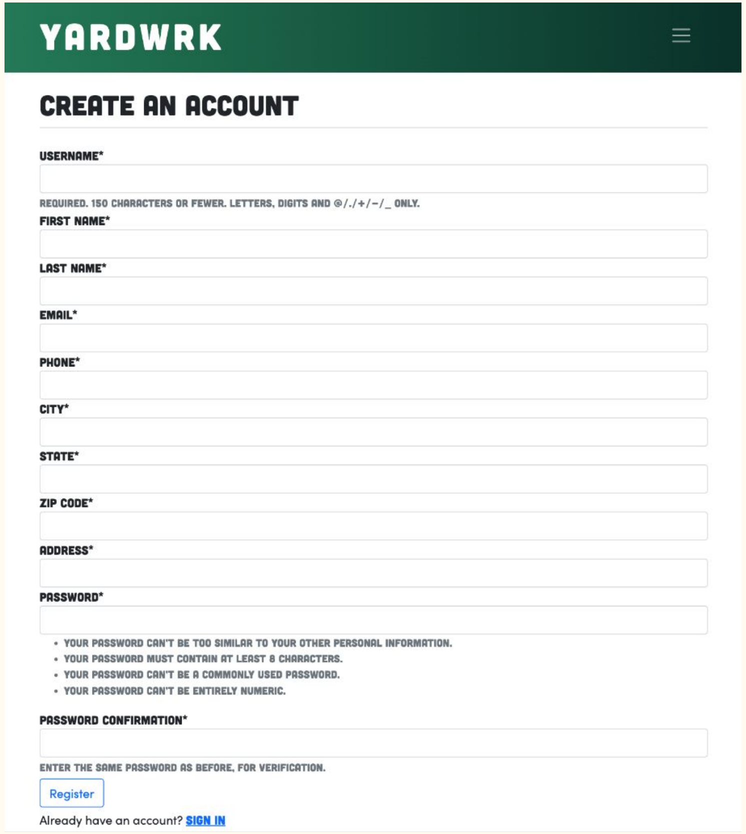

YARDWRK

Please enter your username and password!

Don't have an account? Sign Up

Requirement 1: Implementation

View to create an account.



YARDWRK

CREATE AN ACCOUNT

USERNAME*

REQUIRED. 150 CHARACTERS OR FEWER. LETTERS, DIGITS AND @/./+/-/_ ONLY.

FIRST NAME*

LAST NAME*

EMAIL*

PHONE*

CITY*

STATE*

ZIP CODE*

ADDRESS*

PASSWORD*

- YOUR PASSWORD CAN'T BE TOO SIMILAR TO YOUR OTHER PERSONAL INFORMATION.
- YOUR PASSWORD MUST CONTAIN AT LEAST 8 CHARACTERS.
- YOUR PASSWORD CAN'T BE A COMMONLY USED PASSWORD.
- YOUR PASSWORD CAN'T BE ENTIRELY NUMERIC.

PASSWORD CONFIRMATION*

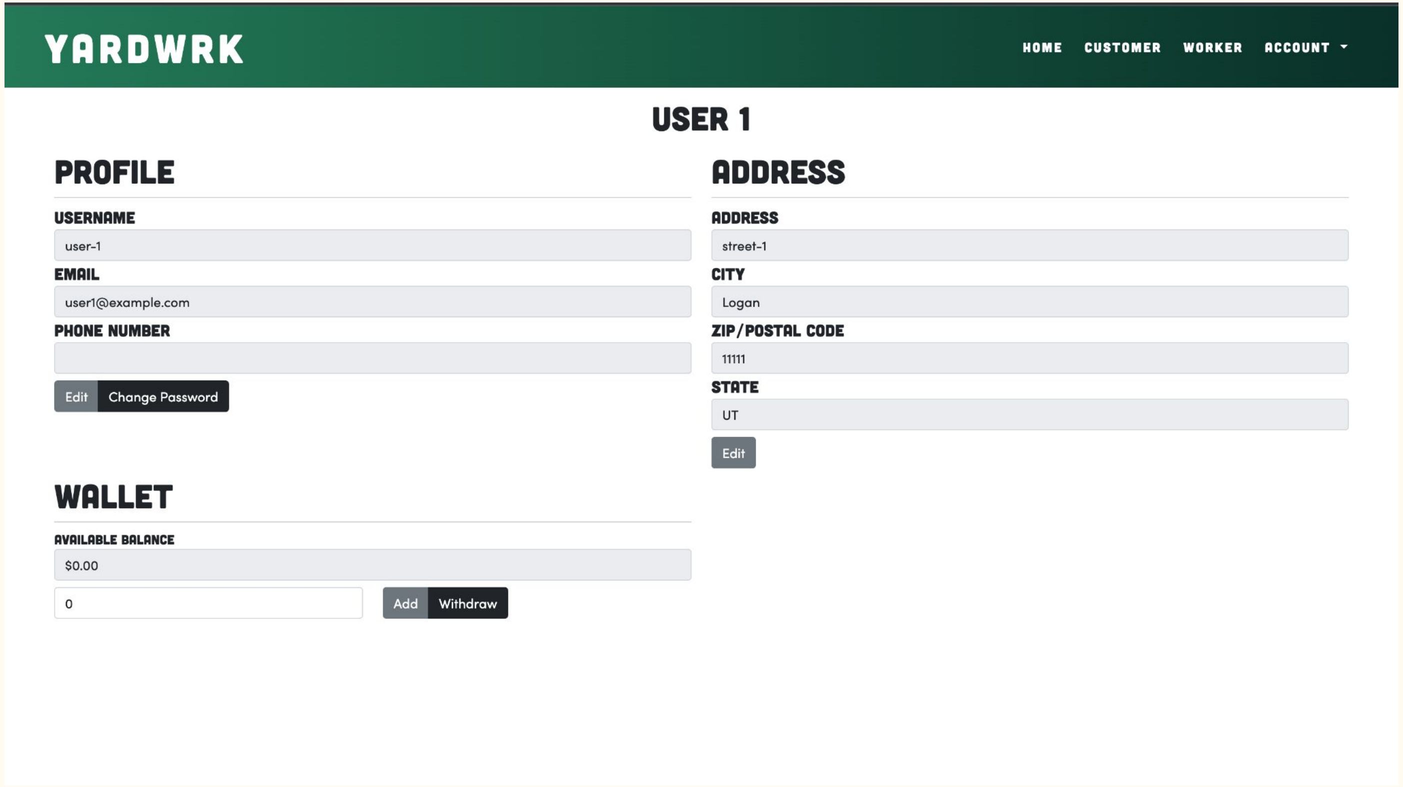
ENTER THE SAME PASSWORD AS BEFORE, FOR VERIFICATION.

Register

Already have an account? SIGN IN

Requirement 1: Implementation

Pofile/account view



USER 1

PROFILE

USERNAME
user-1

EMAIL
user1@example.com

PHONE NUMBER

ADDRESS

ADDRESS
street-1

CITY
Logan

ZIP/POSTAL CODE
11111

STATE
UT

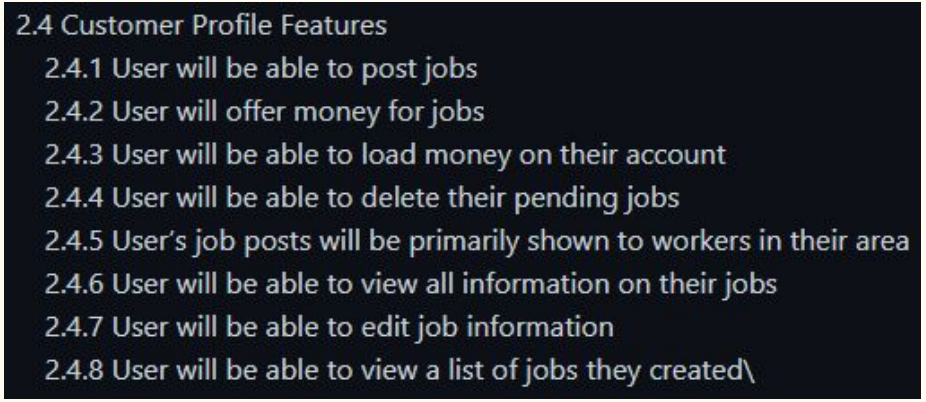
WALLET

AVAILABLE BALANCE
$0.00

0

Requirement 2: Posting a Job

This requirement is essential. It specifies that a user is allowed to post a job that will later be accepted by a worker. This is half of the fundamental usage of our web application. No job, no money. The requirement is found under 2.4 Customer Profile Features, listed as 2.4.1. All requirements in this section are dependent upon this requirements success.



2.4 Customer Profile Features

- 2.4.1 User will be able to post jobs
- 2.4.2 User will offer money for jobs
- 2.4.3 User will be able to load money on their account
- 2.4.4 User will be able to delete their pending jobs
- 2.4.5 User's job posts will be primarily shown to workers in their area
- 2.4.6 User will be able to view all information on their jobs
- 2.4.7 User will be able to edit job information
- 2.4.8 User will be able to view a list of jobs they created\

Requirement 2: Posting a Job

FURPS Description

Functionality - to post a job to the web application that a worker can accept.

Usability - A button on the customer dashboard will bring you to a page where you can enter job details. Click the create button and that job is entered into the database and has been created.

Reliability - This features reliability is tied to the overall functionality of our application. If this doesn't work, the application is pointless.

Performance - Through Django's database and view models, this feature is fast and efficient.

Supportability - The logic for this is supported through Django views and html forms. Easy to update if needed.

MoSCoW Description

Must Have - Functionality to post a job to the Django database.

Should Have - Various fields that describe the job. Description, amount paid, and job type. Clean and concise UI.

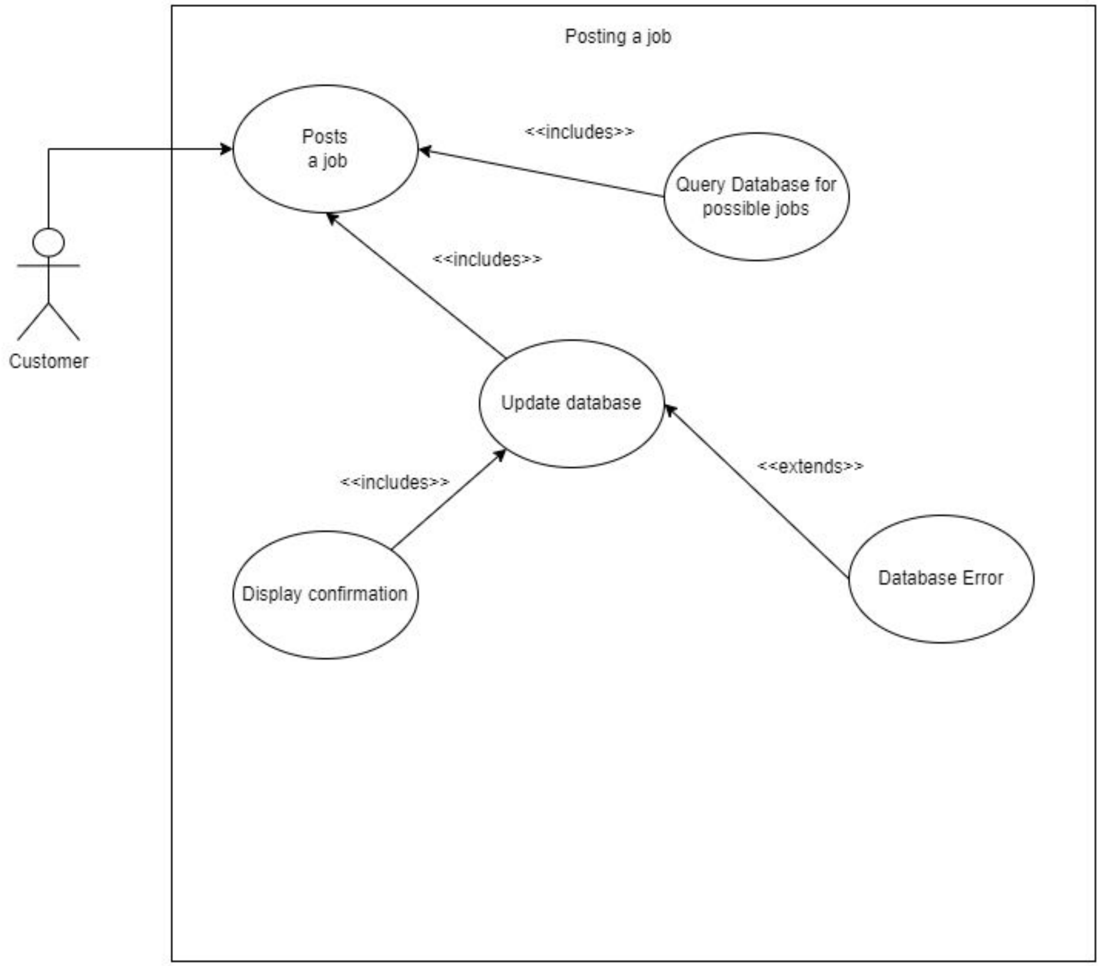
Could Have - A feature that doesn't allow the user to give a monetary reward greater than what is already in their balance.

Won't Have - Won't check for duplicates. You can post the same type of job as many times as you like.

Requirement 2: Use Case Diagram

This Use Case Diagram can be found in docs/useCaseDiagrams.md as Diagram 1

We see that the customer has access to posting a job. Everything else that happens is under the hood, as the database grabs possible job types to display to the user, updates the database on completion, and displays confirmation by showing the posted job in the customer’s dashboard.

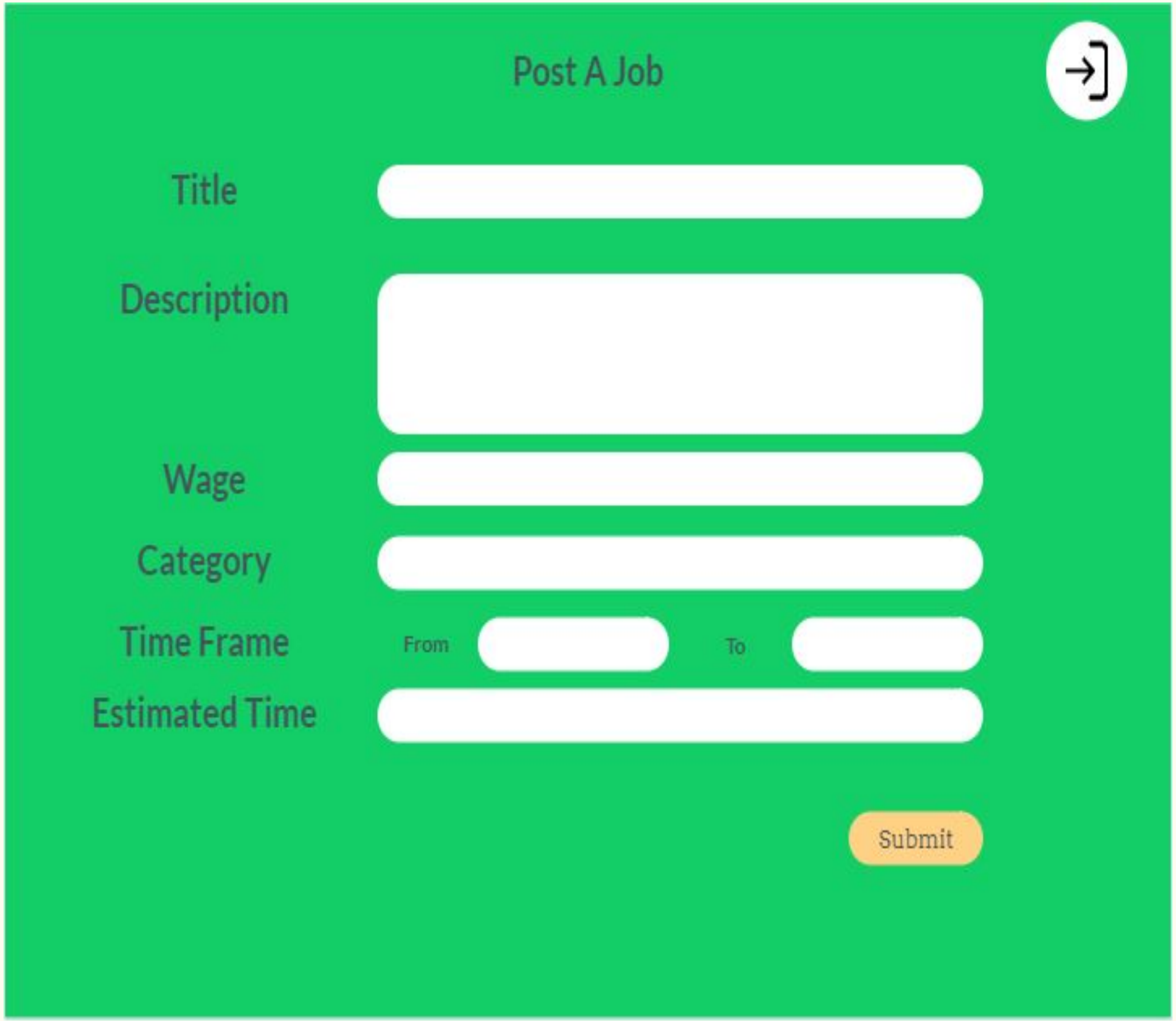


Requirement 2:
Low Fidelity Prototype

This prototype can be found docs/prototypes/lowFidelity/low_fi_post_job.png

This was mainly used to show what the user would see upon clicking the job creation link.

The fields are self explanatory. If a invalid value was entered, and the submit button pressed, error messages would display notifying the user.



Requirement 2: Scrum and Testing

Scrum Tasks:

- Job module creation : Ian
- User module creation: Jake
- Customer module creation: Dave
- Job creation backend: Dave
- Job creation frontend: Jake
- Job creation form: Jake
- Job creation success page: Ian
- Job creation account balance prerequisite: Dave
- Customer dashboard backend: Dave
- Customer dashboard frontend: Jake

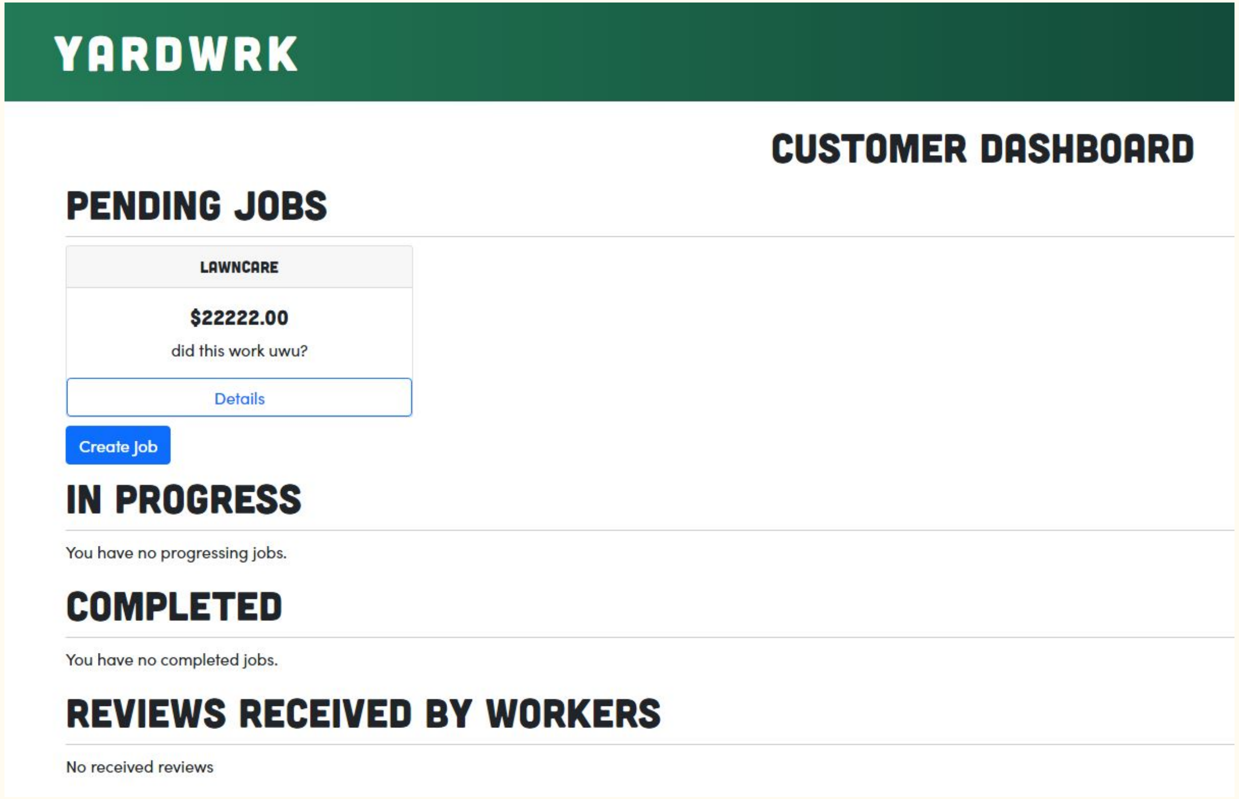
Testing:

Testing was mainly done through regression testing. After new changes had been implemented we would verify that the requirements previously fulfilled would still pass.

You can find a test for this in docs/SystemTesting.pdf Index 3: Customer Actions

Requirement 2: Implementation

This is the customer dashboard, posting a job starts after a customer hits the 'Create Job' button.



YARDWRK

CUSTOMER DASHBOARD

PENDING JOBS

LAWNCARE

$22222.00

did this work uwu?

IN PROGRESS

You have no progressing jobs.

COMPLETED

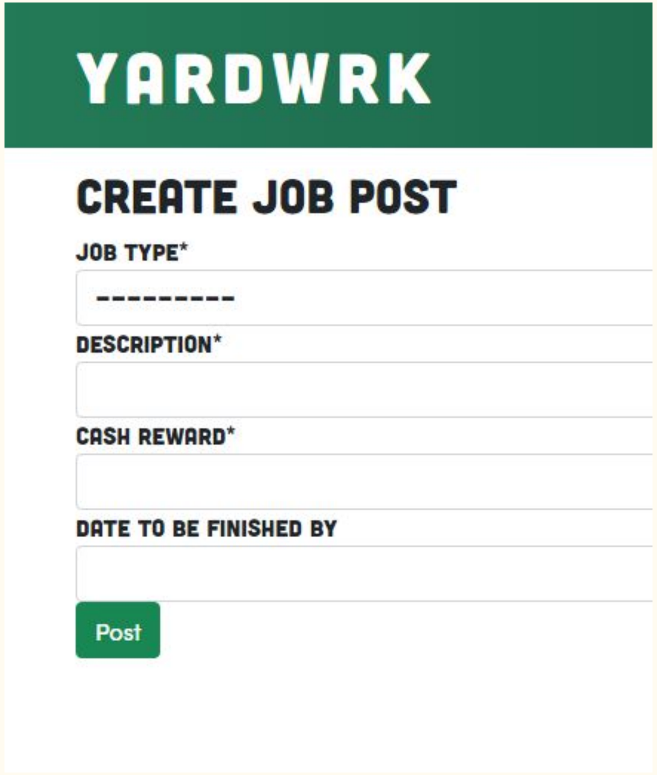
You have no completed jobs.

REVIEWS RECEIVED BY WORKERS

No received reviews

Requirement 2: Implementation

The customer is then taken to this page, where they can fill out the job information.


YARDWRK

CREATE JOB POST

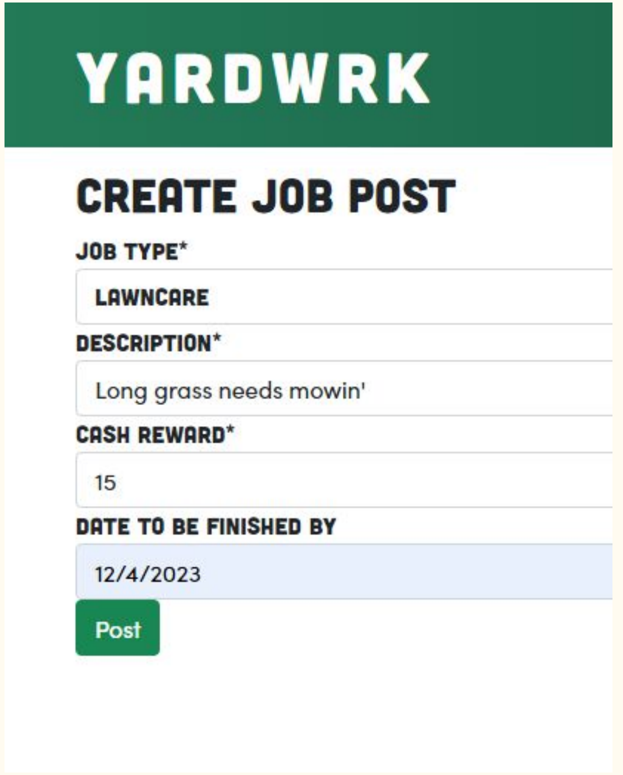
JOB TYPE*

DESCRIPTION*

CASH REWARD*

DATE TO BE FINISHED BY

Post


YARDWRK

CREATE JOB POST

JOB TYPE*

LAWNCARE

DESCRIPTION*

Long grass needs mowin'

CASH REWARD*

15

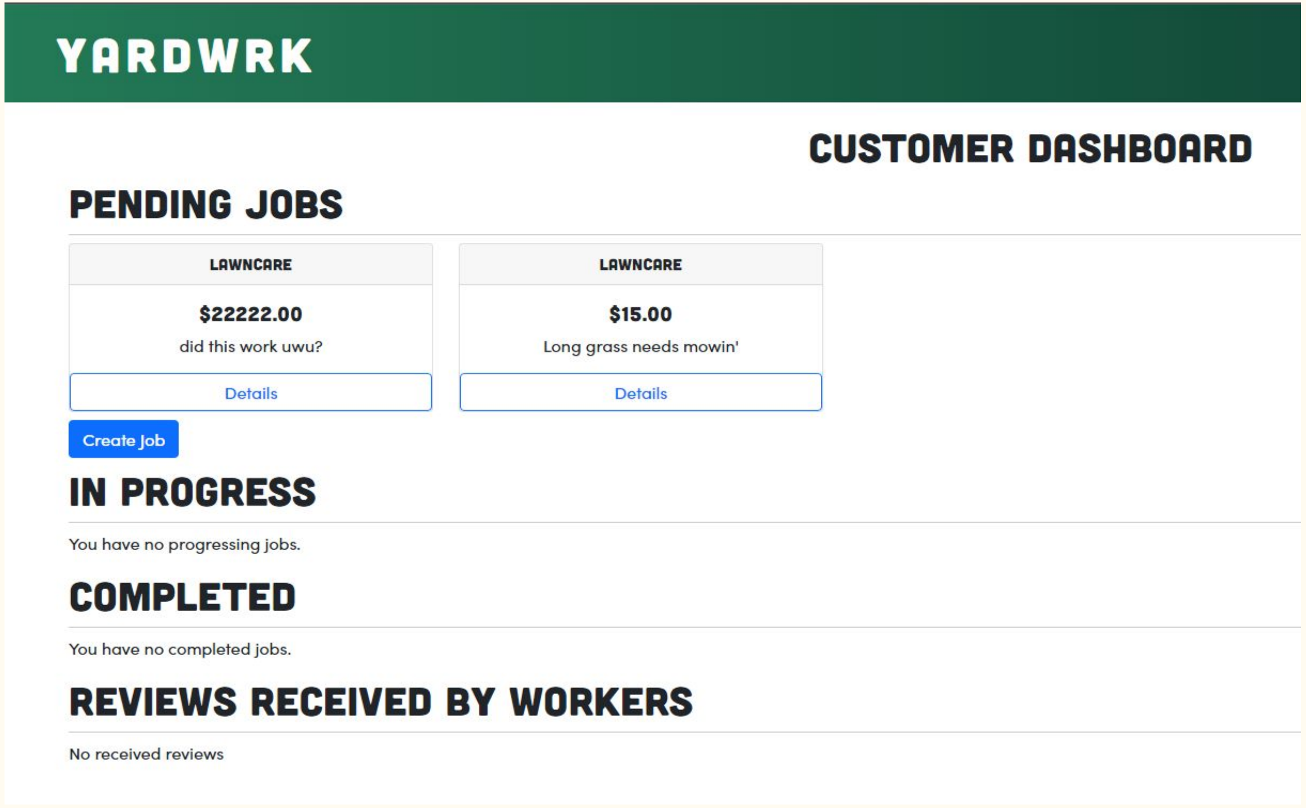
DATE TO BE FINISHED BY

12/4/2023

Post

Requirement 2: Implementation

After a successful job posting, the customer is returned to their dashboard and the new job can be seen here. This verifies that the job has been posted.


YARDWRK

CUSTOMER DASHBOARD

PENDING JOBS

LAWNCARE
$22222.00
did this work uwu?
Details

LAWNCARE
$15.00
Long grass needs mowin'
Details

Create Job

IN PROGRESS

You have no progressing jobs.

COMPLETED

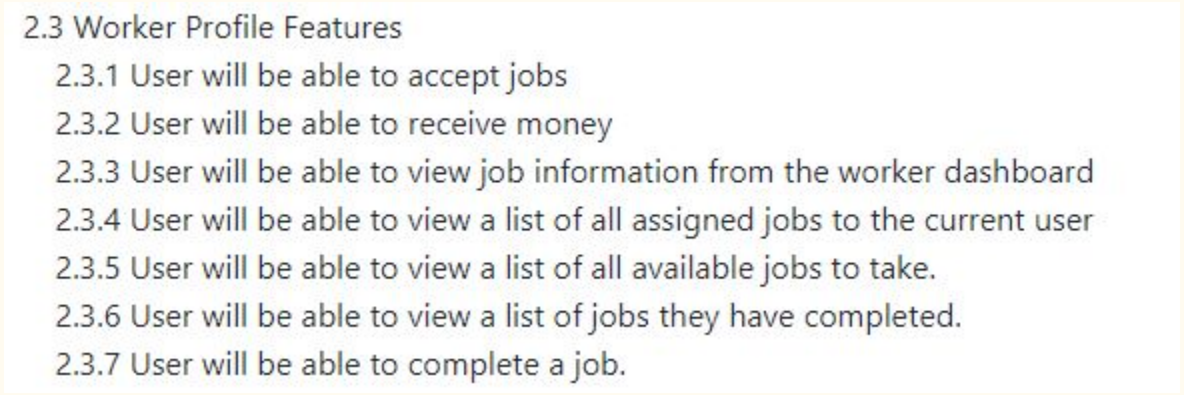
You have no completed jobs.

REVIEWS RECEIVED BY WORKERS

No received reviews

Requirement 3: Accepting/Taking a Job

This requirement involves allowing a worker to accept a job on the website. It was an important requirement, because being able to accept jobs to make money is foundational to a job-finding application. It is found under requirement 2.3, worker profile features, in our requirements definition, specifically requirements 2.3.1, 2.3.3, 2.3.4, and 2.3.5 as shown in the snippet below.



2.3 Worker Profile Features

- 2.3.1 User will be able to accept jobs
- 2.3.2 User will be able to receive money
- 2.3.3 User will be able to view job information from the worker dashboard
- 2.3.4 User will be able to view a list of all assigned jobs to the current user
- 2.3.5 User will be able to view a list of all available jobs to take.
- 2.3.6 User will be able to view a list of jobs they have completed.
- 2.3.7 User will be able to complete a job.

Requirement 3: Accepting/Taking a Job

FURPS Description

Functionality - to assign a job in the database a reference to a worker that accepts it.

Usability - A simple button on a job's details page allows a worker to accept it. On click, the user is redirected to a page that confirms acceptance and allows them to review the job details.

Reliability - This features reliability is tied to the integrity of the database.

Performance - Through the use of the Django framework and SQLite, this feature runs very efficiently.

Supportability - This feature's logic is organized and encapsulated within Django views, so it should be quite trivial to update.

MoSCoW Description

Must Have - Functionality to assign a job to a specific worker

Should Have - A nice, easy to use UI, and a well organized design (in code and UI) that can be reused

Could Have - A feature that notifies the user when requested and allows them to reject the worker

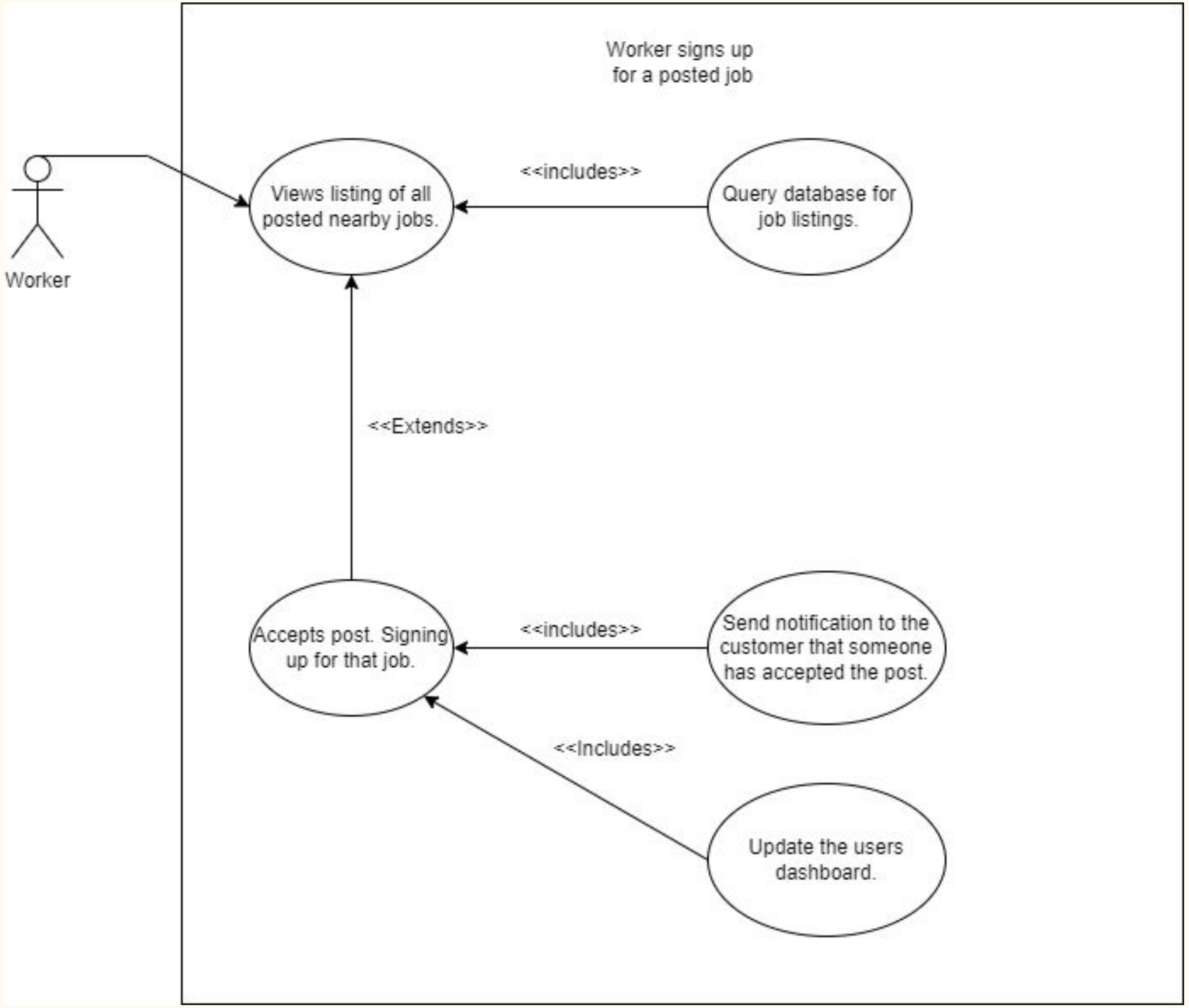
Won't Have - N/A (Feature was simple enough that we only need to use the above three descriptors)

Requirement 3:
Use Case Diagram

This Use Case Diagram can be found in our useCaseDiagrams.md as Diagram 2.

We see that a worker has the ability to access a listing of posted jobs, queried from the database, and can filter out ones that are not nearby.

The worker can accept a post and sign up for the job, which will update the dashboard and notify the customer that someone has accepted their post.


Worker signs up for a posted job
Worker
Views listing of all posted nearby jobs.
<<includes>>
Query database for job listings.
<<Extends>>
Accepts post. Signing up for that job.
<<includes>>
Send notification to the customer that someone has accepted the post.
<<Includes>>
Update the users dashboard.

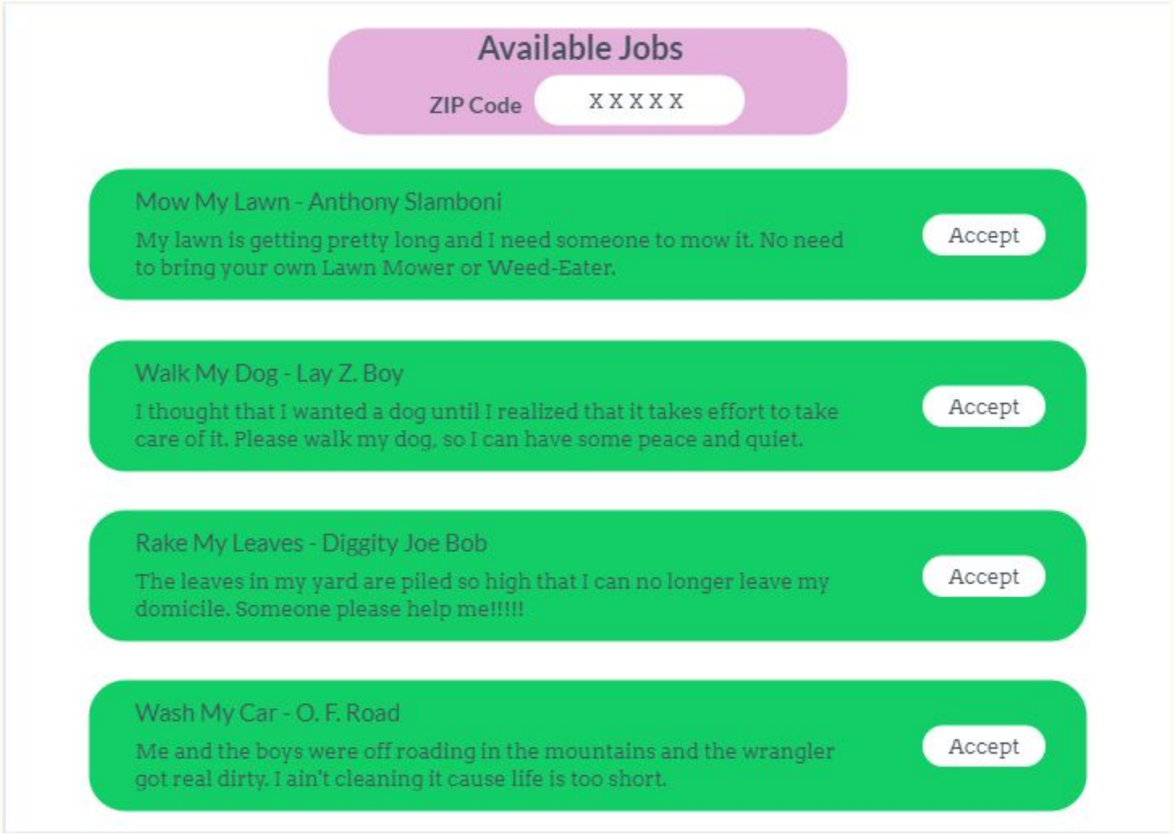
Requirement 3:
Low Fidelity Prototype

This prototype can be found in our prototypes directory as low_fi_take_jobs.png

The idea here was to model a job filtering system and a way that jobs could be listed for the worker to accept.

The entry box at the top would filter jobs based on a ZIP code chosen by the user.

The jobs listed below would be a dynamic list of available jobs for the worker to choose from that displayed data about the job and had a button to accept the job.



Available Jobs

ZIP Code XXXXX

Mow My Lawn - Anthony Slamboni

My lawn is getting pretty long and I need someone to mow it. No need to bring your own Lawn Mower or Weed-Eater.

Walk My Dog - Lay Z. Boy

I thought that I wanted a dog until I realized that it takes effort to take care of it. Please walk my dog, so I can have some peace and quiet.

Rake My Leaves - Diggity Joe Bob

The leaves in my yard are piled so high that I can no longer leave my domicile. Someone please help me!!!!!

Wash My Car - O. F. Road

Me and the boys were off roading in the mountains and the wrangler got real dirty. I ain't cleaning it cause life is too short.

Requirement 3: Scrum Tasks and Testing

Scrum Tasks:

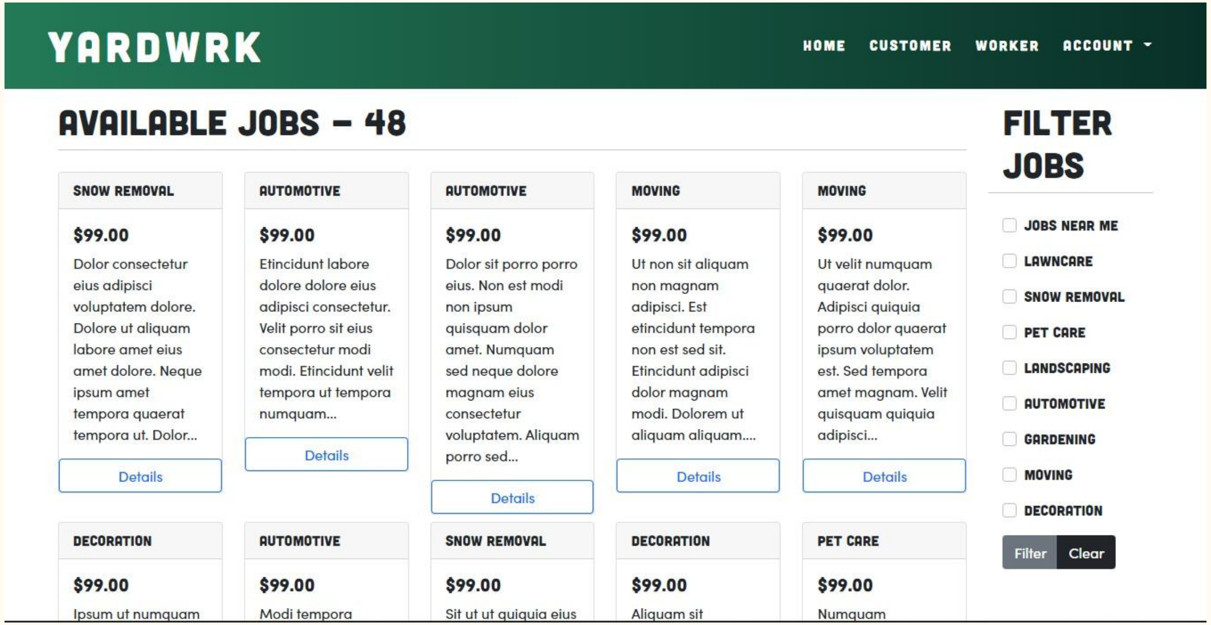
- Job Model: Ian
- User Model: Jake
- Worker Model: Dave
- Worker Dashboard: Ian
- Home Page: Jake
- Job Details Page: Jake, Ian, and Dave
- Job Assignment Back-end: Ian
- Job Acceptance Page: Jake and Ian

Testing

Testing for this feature was done mostly through regression testing. The procedure for testing this requirement can be found in SystemTesting.pdf under Index 4: Worker Actions

Requirement 3: Implementation

This is the home page. People who want to do jobs can go here to browse through them. Each job card has a details button that will lead to a page with more information.



AVAILABLE JOBS – 48

SNOW REMOVAL

$99.00

Dolor consectetur eius adipisci voluptatem dolore. Dolore ut aliquam labore amet eius amet dolore. Neque ipsum amet tempora quaerat tempora ut. Dolor...

Details

AUTOMOTIVE

$99.00

Etincidunt labore dolore dolore eius adipisci consectetur. Velit porro sit eius consectetur modi modi. Etincidunt velit tempora ut tempora numquam...

Details

AUTOMOTIVE

$99.00

Dolor sit porro porro eius. Non est modi non ipsum quisquam dolor amet. Numquam sed neque dolore magnam eius consectetur voluptatem. Aliquam porro sed...

Details

MOVING

$99.00

Ut non sit aliquam non magnam adipisci. Est etincidunt tempora non est sed sit. Etincidunt adipisci dolor magnam modi. Dolorem ut aliquam aliquam....

Details

MOVING

$99.00

Ut velit numquam quaerat dolor. Adipisci quiquia porro dolor quaerat ipsum voluptatem est. Sed tempora amet magnam. Velit quisquam quiquia adipisci...

Details

DECORATION

$99.00

Ipsum ut numquam

AUTOMOTIVE

$99.00

Modi tempora

SNOW REMOVAL

$99.00

Sit ut ut quiquia eius

DECORATION

$99.00

Aliquam sit

PET CARE

$99.00

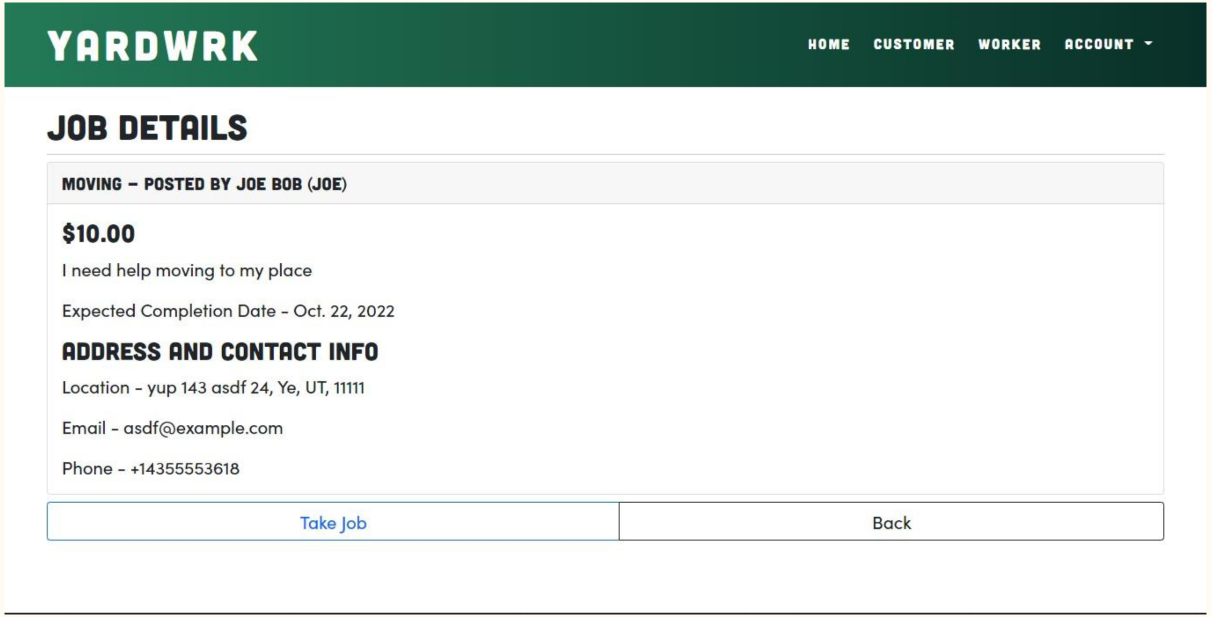
Numquam

FILTER JOBS

- ☐ JOBS NEAR ME
- ☐ LAWNCARE
- ☐ SNOW REMOVAL
- ☐ PET CARE
- ☐ LANDSCAPING
- ☐ AUTOMOTIVE
- ☐ GARDENING
- ☐ MOVING
- ☐ DECORATION

Requirement 3: Implementation

This is the job details page. It shows info about the job and gives the user the ability to assign it to themselves.



JOB DETAILS

MOVING – POSTED BY JOE BOB (JOE)

$10.00

I need help moving to my place

Expected Completion Date - Oct. 22, 2022

ADDRESS AND CONTACT INFO

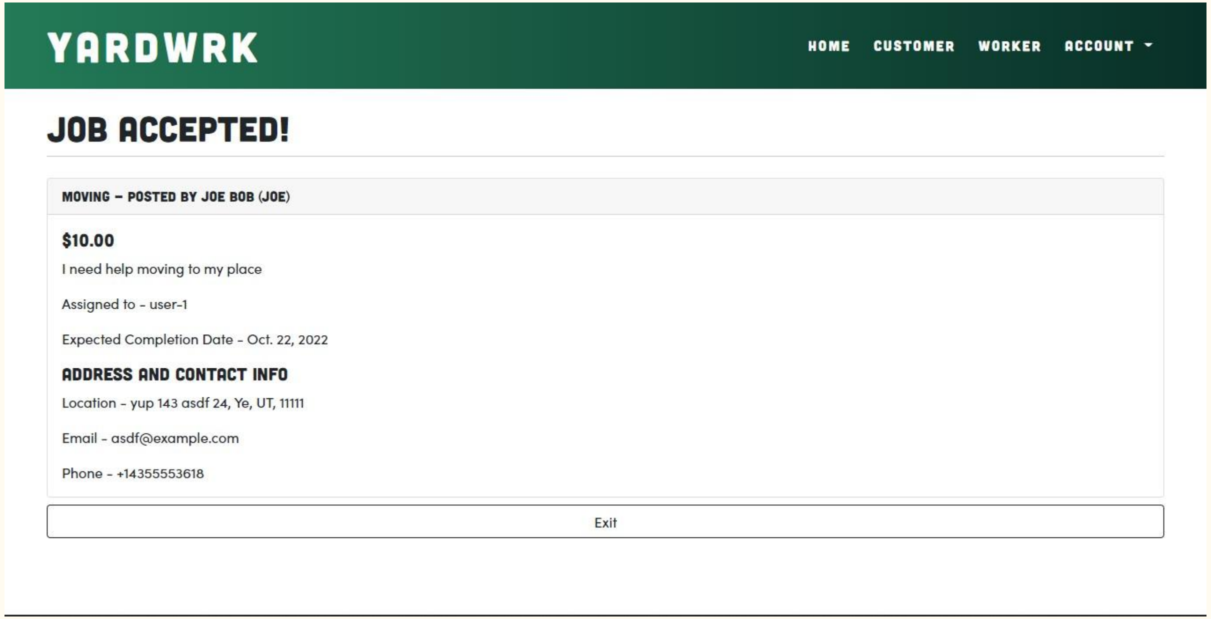
Location - yup 143 asdf 24, Ye, UT, 11111

Email - asdf@example.com

Phone - +14355553618

Requirement 3: Implementation

This is the acceptance page that displays after you take a job.



JOB ACCEPTED!

MOVING – POSTED BY JOE BOB (JOE)

$10.00

I need help moving to my place

Assigned to - user-1

Expected Completion Date - Oct. 22, 2022

ADDRESS AND CONTACT INFO

Location - yup 143 asdf 24, Ye, UT, 11111

Email - asdf@example.com

Phone - +14355553618

Hyperlinks to four screen capture videos describing a completed use case.

Quinn - User modifies account information
https://github.com/davevstorey/9-usu-cs3450-sp2022/blob/master/docs/VideoWalkThroughs/modifyAccountInformation.mp4

Ian - User Changes their account Balance
https://github.com/davevstorey/9-usu-cs3450-sp2022/blob/master/docs/VideoWalkThroughs/IanAdamsUseCase3Demo.mp4

Jake -
https://github.com/davevstorey/9-usu-cs3450-sp2022/blob/master/docs/VideoWalkThroughs/WorkerAcceptsJob480p.mp4

Dave - Customer Posts a Job
https://github.com/davevstorey/9-usu-cs3450-sp2022/blob/master/docs/VideoWalkThroughs/PostAJob.mp4

Major Resources Utilized

Bootstrap. (n.d.). *Introduction*. Introduction · Bootstrap v5.1. Retrieved April 14, 2022, from https://getbootstrap.com/docs/5.1/getting-started/introduction/

Django Software Foundation. (n.d.). *Many-to-one relationships.* Djangoproject. Retrieved April 14, 2022, from https://docs.djangoproject.com/en/4.0/topics/db/examples/many_to_one/

Django Software Foundation. (n.d.). *Model field reference.* Djangoproject. Retrieved April 14, 2022, from https://docs.djangoproject.com/en/4.0/ref/models/fields

Documentation. Django. (n.d.). Retrieved April 14, 2022, from https://docs.djangoproject.com/en/4.0/intro/tutorial02/

Where the world builds software. GitHub. (n.d.). Retrieved April 6, 2022, from https://github.com/

Questions?

- Now’s the time. Feel free to ask us any questions you would like answered!