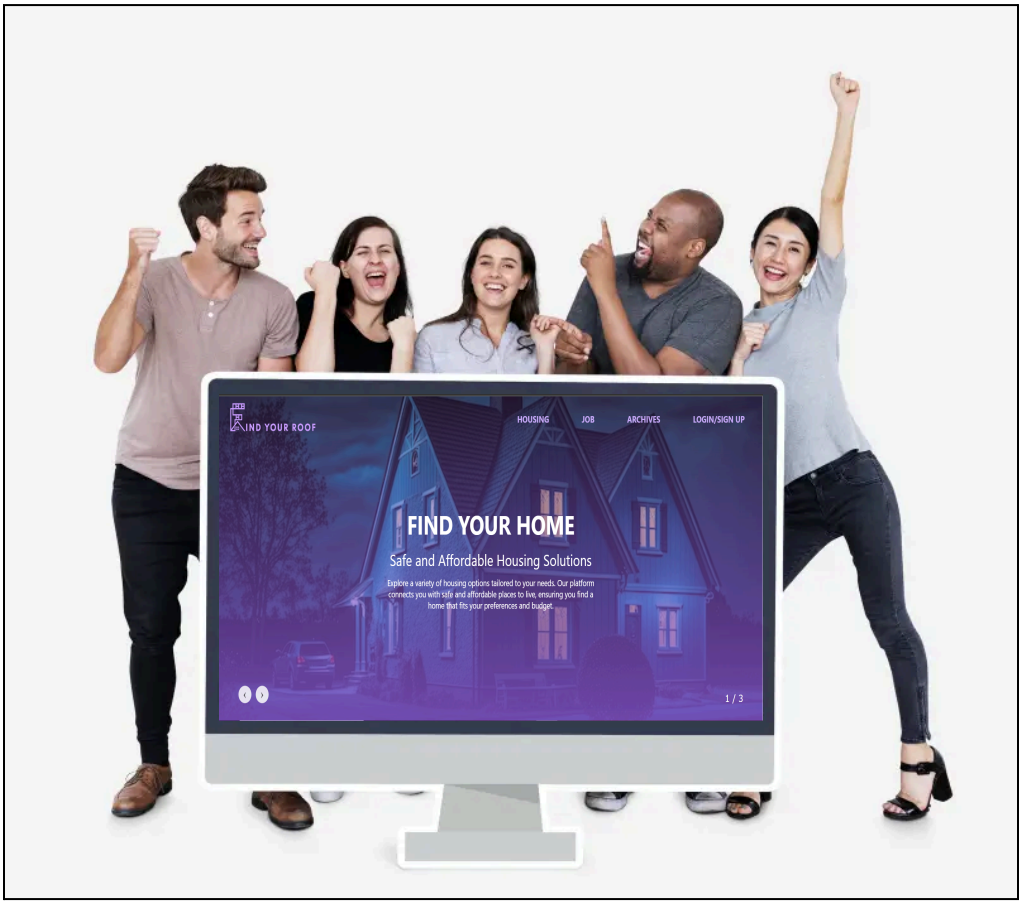
FindYourRoof Final Report



An application designed to help homeless individuals rebuild and thrive

Prepared by
Frank Mensah, David Serrano, Michael Jedziniak, Vibha Navale
***for use in* CS 442**
at the
University of Illinois Chicago

Spring 2024

Table of Contents

List of Figures

List of Tables

No Tables for this Report

I Project Description

1 Project Overview

FindYourRoof is an all-in-one application aimed at addressing the complex issue of homelessness in the Chicagoland Area [1]. It's more than just a temporary solution; it offers a structured pathway for individuals to reintegrate back into society. The platform efficiently matches individuals with suitable housing options and promising job opportunities, while also offering a secure environment for document storage and facilitating impactful mentorship connections. By weaving together these features, FindYourRoof fosters a strong rehabilitation for homeless individuals by providing various tools to help them in their endeavors. This collective approach is geared towards driving significant and lasting change in society.

2 Project Domain

Regarding the domain of this project, the application itself is built off of NextJS which utilizes React Components to have a well-designed system of reuse of code throughout the front-end. Moreover, Supabase was utilized to help manage the database system to help store data from the various APIs used and data stored from the user. Regarding APIs, for holding the Housing data, US Real Estate was used; for holding the Job data, JSearch was used.

3 Relationship to Other Documents

For this document, the Final Developer Project Report of the "FindYourRoof" [1] was used to construct various sections of this report. Moreover the glossary and various terms from "FindYourRoof" [1] are utilized throughout this document to best align with the clear project vision that S. Burney, R. Dhotre, N. Patel and R. Rizvi provided.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

FindYourRoof: The all-in-one application designed to assist homeless individuals in finding affordable housing, employment opportunities, and communication with mentors and support networks.

AI Engine: The artificial intelligence system used for job matching, which matches individuals based on their skill sets, past experiences, and training needs.

Document Repository: An encrypted cloud-based system where individuals can securely store critical personal documents.

Geospatial Integration: A system that uses location data to find nearby housing and jobs for homeless people, especially close to bus or train stops.

Cloud Service Providers: Companies that give online storage and service where data and software can be accessed from anywhere with a network.

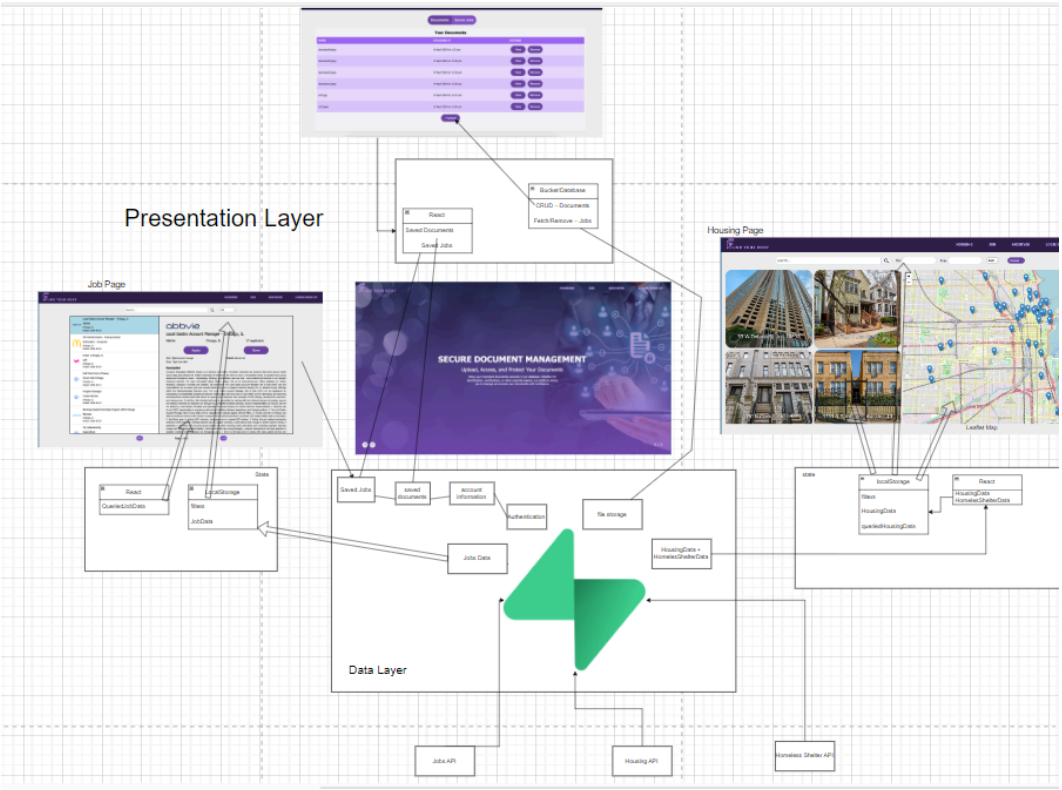
Volunteer: Individuals who can donate to Non-Profit Organizations and participate in workshop events.

Mentor: Individuals who offer one-on-one guidance and support to homeless individuals through the application.

4b UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [2]. Any exceptions are noted where used. Shown below is our architecture diagram that highlights the structure of our application, the communication with supabase, and the organization of it all.

Figure 1 - Architecture Diagram



4c Data Dictionary for Any Included Models

The information that is recorded in the application of various users includes:

1. Homeless Individual Profile: Name, Contact Information, Saved jobs, Document Repository (critical documents)

The overall data storage and flows involve:

1. Document Repository Data: Encrypted cloud-based storage for critical personal documents of homeless individuals.
2. Geospatial Data: Data related to housing and employment options within the city.

Technical Specifications for Interfaces involve:

1. User Registration and Login Interface: Capturing user data for registration and authentication.
2. Job and Housing Application Interfaces: Allows users to apply for jobs and housing.

Document Upload and Access Interface:

1. Allows users to upload and access critical documents securely.

II Project Deliverables

Due to the scope of this project, we constructed the main functionality of the Housing and Job search which are two of the major features of this application [1]. Moreover, we have constructed and finished the sign-up/log-in feature, the document storage, as well as the Local Non-Profit Websites feature. The only feature that was not implemented due to time constraints was the Mentorship Platform feature where the user can find a mentor for various professional and academic reasons.

1 First Release

From January 28th to February 18th, the home page and housing feature were designed and implemented. The home page holds all the main information of FindYourRoof and all the features the website presents to the user. The housing page contains a searching feature where the user can find a wide range of housing with various filters to narrow the results. Moreover, a major part of this release was the initial construction of the application where the reusable React Components, Supabase, and the NextJS page directories were all set up to help with the future development of features.

Figure 2 - Home Page

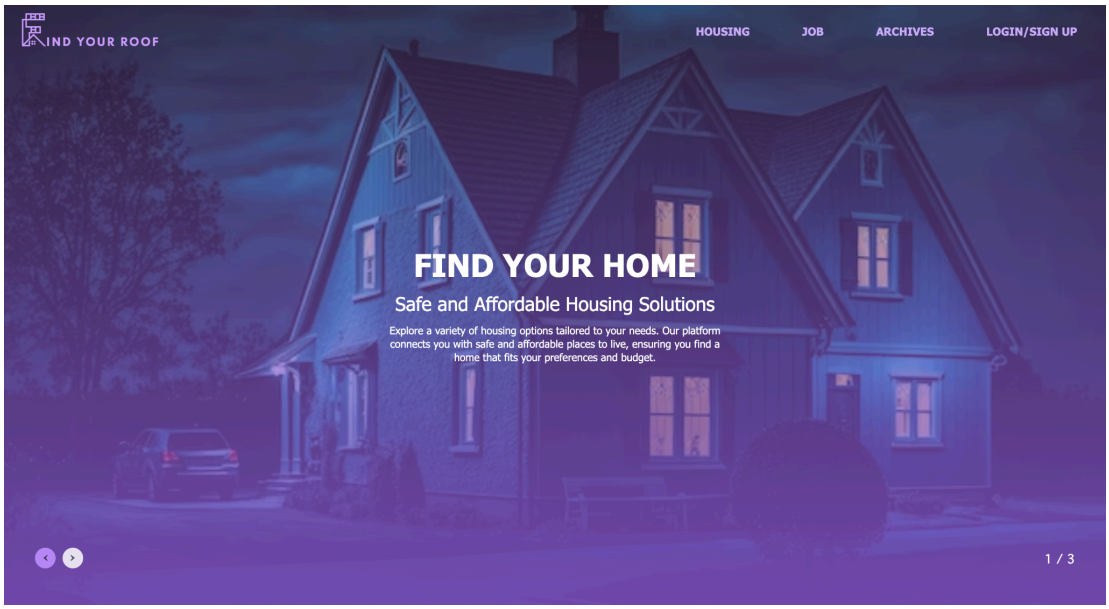
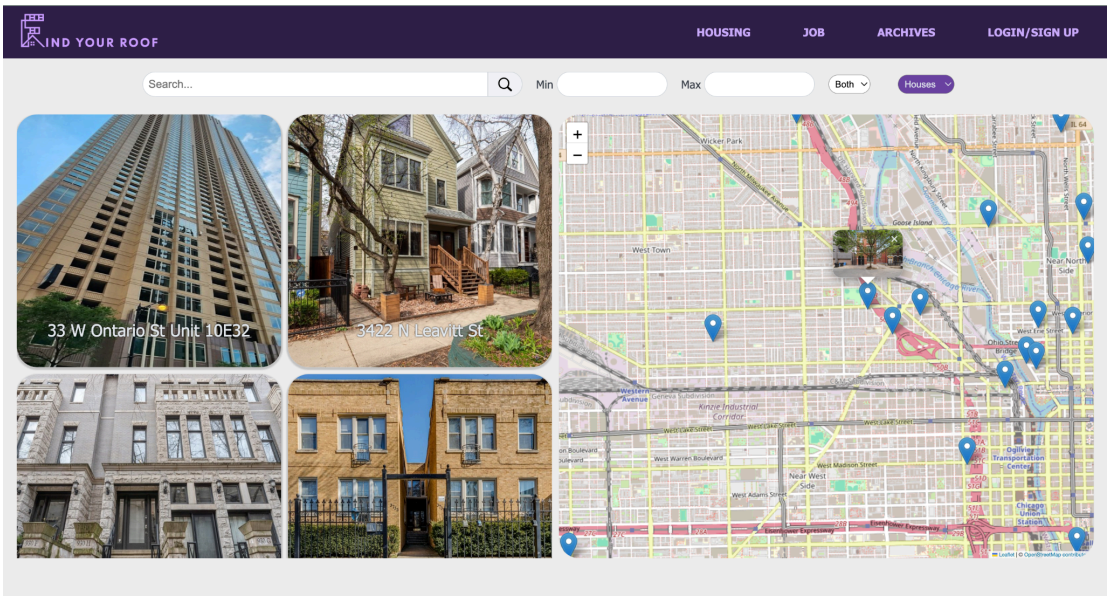


Figure 3 - Housing Page



2 Second Release

From February 25th to March 24th, the job search page and user authentication feature were implemented in this release, along with addressing some bugs from the Housing Page. On the job page, the user is able to search from a wide range of jobs in the Chicagoland Area and filter those results to better fit their needs. From those results, and on click, the user is directed to the webpage to where they can apply for the job. Moreover, the development of the user authentication feature was complete

and the user can save information such as jobs and storing of documents when they visit the site. Also, this release handles a mainstream of the color scheme and design system of the application to keep all of the CSS and designs aligned throughout the various pages and to prepare for a Mobile View that will be implemented in the third release.

Figure 4 - Job Page

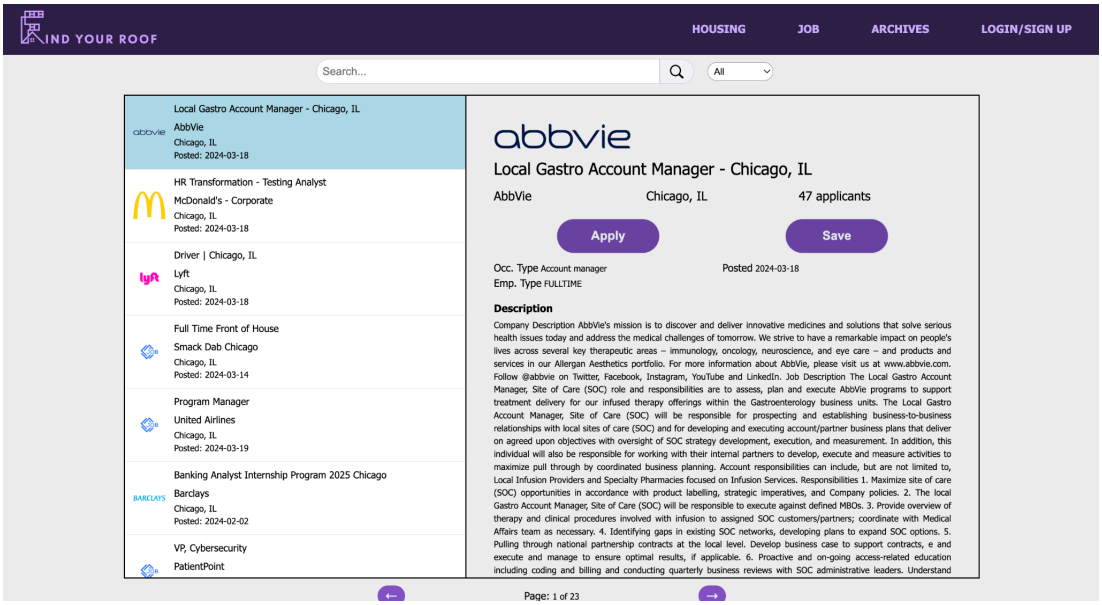
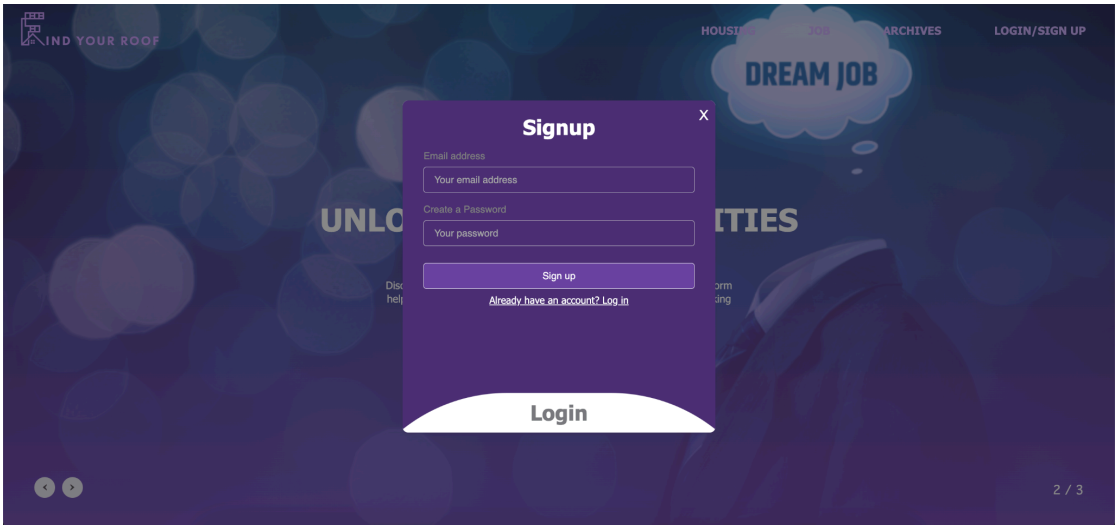


Figure 5 - Sign-up/Log-in Page

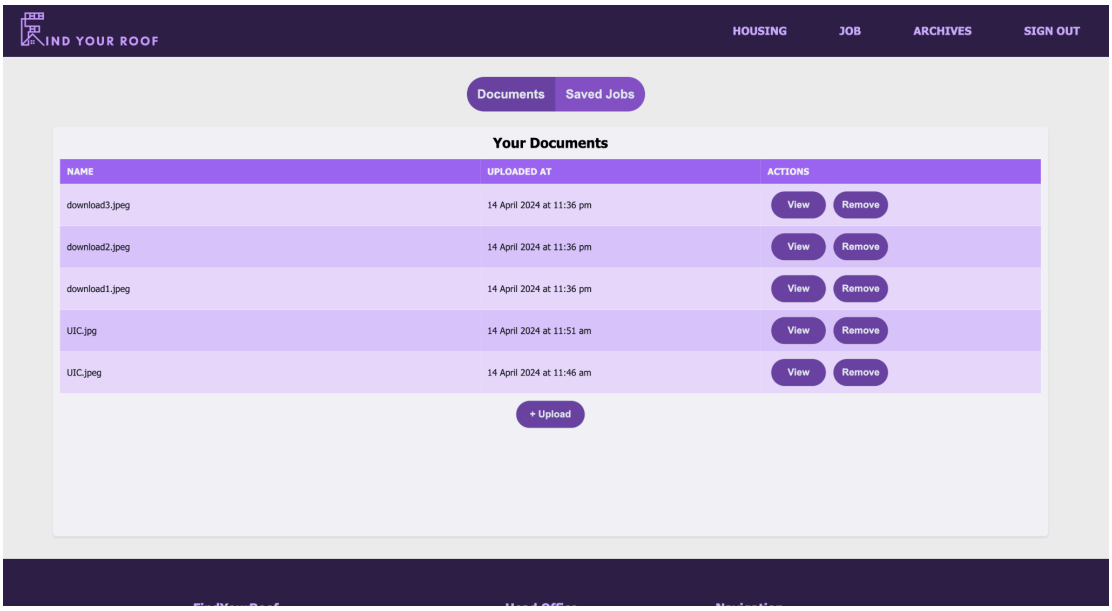


3 Third Release

From March 31th to April 14th, this release handled the implementation of the mobile view of the application, document storage i.e. Archives, and local non-profit viewer. Using the mainstream design from the second release, the mobile view was able to be

constructed and finished in a quick manner. On the Archives page, the user is able to view their saved jobs and also store any important documents on the site for future use. Moreover, the local non-profit viewer was added to the home page so the user can see the various non-profits and be directed to the respective site. This release also focused on minor improvements in the website in terms of bug fixes, addition of appropriate toast or alert messages. The web application was also thoroughly tested and deployed using Netlify.

Figure 6 - Document Page



4 Comparison with Original Project Design Document

Compared to the proposed project from Fall 2023 [1], we were able to complete the majority of the features that the original report outlined. These were housing and job search, local non-profit organizations, and document storage. We were also able to construct a mobile view of the application which was not originally planned from the original report but we knew it would be a good addition to the application. The only feature that was not implemented was mentorship. Overall, we were able to closely implement the original project outline to great prosperity but still have areas of improvement in the future.

III Testing

1 Items to be Tested

1. User Authentication
    a. Sign up
        i. Email confirmation
    b. Log in

2. Housing Page
    a. Map Integration
    b. Search
    c. Cards
    d. Drawer
3. Job Page
    a. Job list
        i. Page navigation
    b. Job Description Box
    c. Search
4. Archives Page
    a. Document Storage
    b. Saved Jobs

All of our code was reviewed and manually tested by the reviewer(s) i.e., we tested each other's code before merging each Pull Request to the main branch on Github.

2 Test Specifications

ID#1 - User Authentication: Sign Up

Description: The user should be able to successfully sign up.

Items covered by this test: User sign up

Requirements addressed by this test: Access requirements.

Environmental needs: Access to Supabase user authentication table.

Intercase Dependencies: Test case ID#2 depends on the success of this test.

Test Procedures: User creates a new account using an email address and password.

Input Specification: The user must provide a valid email address in the correct format along with a password. Upon clicking the "Sign Up" button, the user will receive a confirmation email. Logging in requires confirmation of this email.

Output Specifications: On successful sign up, the user is redirected to the homepage of FindYourRoof.

Pass/Fail Criteria: If sign up is unsuccessful, the user should be presented with a prompt or message indicating the failure.

ID#2 - User Authentication: Log In

Description: The user should be able to successfully log into the web application.

Items covered by this test: User log in

Requirements addressed by this test: Access requirements.

Environmental needs: Access to Supabase user authentication table.

Intercase Dependencies: Depends on success of test ID#1.

Test Procedures: User logs into the web app using an already registered email address and password.

Input Specification: The user must provide a valid email address in the correct format and an existing user profile, along with the correct password. Upon clicking the "Log In" button, the user will be able to access all pages of the application.

Output Specifications: On successful login, the user should be redirected to the homepage of FindYourRoof or the page they attempted to log into the app, for instance the “Archives” page.

Pass/Fail Criteria: If login fails, the user should be presented with a prompt or message indicating the failure.

ID#3 - Housing: Map Integration

Description: The Map shows the correct locations of the housing that are for sale.

Items covered by this test: Map Container

Requirements addressed by this test: Map viewer requirement

Environmental needs: Access to Supabase DB

Intercase Dependencies: NA

Test Procedures: Navigate to the Housing Page, then view Map results.

Input Specification: No specific input needed from the tester. Only navigate to the Housing Page.

Output Specifications: Map container has all locations of available housing that is found in the Supabase DB.

Pass/Fail Criteria: All data points found in Supabase are displayed on the Map Container.

ID#4 - Housing: Search

Description: User should be able to search for any available housing in a given address input.

Items covered by this test: Housing List Container

Requirements addressed by this test: Housing Search requirement

Environmental needs: Access to website

Intercase Dependencies: NA

Test Procedures: Navigate to the Housing Page, then view Map results.

Input Specification: For sake of clarity, search for 'Clark St'

Output Specifications: '930 N Clark St Unit H'

Pass/Fail Criteria: From the input, the tester is able to find the given house in the output section above.

ID#5 - Housing: Cards

Description: Cards contained in the Housing List should show the correct image and address, and display a drawer when clicked.

Items covered by this test: Card Container

Requirements addressed by this test: Housing Card Requirements

Environmental needs: Access to Supabase DB

Intercase Dependencies: NA

Test Procedures: Navigate to the Housing Page.

Input Specification: View the '33 W Ontario St Unit 10E32' card container

Output Specifications: Refer to Supabase DB to see the needed image. When clicked a drawer container is outputted.

Pass/Fail Criteria: The correct image for the input is the same as in Supabase and a drawer is displayed when clicked.

ID#6 - Housing: Drawer

Description: Drawer should display the right details of the card that was clicked or the pin from the map.

Items covered by this test: Drawer Container.

Requirements addressed by this test: Housing Drawer Requirement

Environmental needs: Access to Supabase DB

Intercase Dependencies: NA

Test Procedures: Navigate to Housing Page.

Input Specification: Select the ‘33 W Ontario St Unit 10E32’ card container

Output Specifications: Output of ‘33 W Ontario St Unit 10E32 21000$ sale Since 2024-03-25‘

Pass/Fail Criteria: The outputted data from the given input is the same as viewed in Supabase.

ID#7 - Job: Job list

Description: The Job lists show the correct job listings that are open to be applied.

Items covered by this test: Job List Container

Requirements addressed by this test: Job search requirement

Environmental needs: Access to Supabase DB

Intercase Dependencies: NA

Test Procedures: Navigate to the Job Page, then view Job List results

Input Specification: No specific input needed from the tester. Only navigate to the Job Page.

Output Specifications: Job List container displays all job posts when job loads

Pass/Fail Criteria: All job posts found in Supabase are displayed on the Job List Container

ID#8 - Job: Job Description Box

Description: When selecting a job from the Job List Container, the Job Description should be updated to show the selected job information.

Items covered by this test: Job Description Container

Requirements addressed by this test: Job Description Requirement

Environmental needs: Access to website.

Intercase Dependencies: NA

Test Procedures: Navigate to the Job Page

Input Specification: Select the “local Gastro Account Manager - Chicago, IL”

Output Specifications: Output of the "local Gastro Account Manager - Chicago, IL" job description for AbbVie.

Pass/Fail Criteria: From the tester input, the correct output as stated above is retrieved.

ID#9 - Job: Search

Description: User should be able to search for any available job with any given job title input

Items covered by this test: Search and Job List Container

Requirements addressed by this test: Job Search requirement

Environmental needs: Access to website

Intercase Dependencies: NA

Test Procedures: Navigate to the Housing Page, then view Map results.

Input Specification: For sake of clarity, search for 'HR Transformation'

Output Specifications: 'HR Transformation - Testing Analyst' from McDonalds.

Pass/Fail Criteria: From the tester input, the correct output as stated above is retrieved.

ID#10 - Archives: Document Storage

Description: User should be able to access, remove and view the list of documents previously uploaded, and upload new files.

Items covered by this test: Archives page.

Requirements addressed by this test: Access and Document Storage requirements.

Environmental needs: Access to Supabase storage bucket and user authentication.

Intercase Dependencies: Depends on test ID#2 (Successful user login).

Test Procedures: Users can view the list of files previously uploaded and perform actions on them.

Input Specification: The user is required to be logged into the app to access the Archives page and its contents.

Output Specifications: If the user is logged in, they should be able to access the list of files, if any, by default and can upload new files, remove existing files and view individual files in the "Documents" view.

Pass/Fail Criteria: If user is not logged in, they should be prompted with a message indicating that they are required to be logged in to access the contents of the Archives page.

ID#11 - Archives: Saved Jobs

Description: User should be able to access, remove and view the list of jobs previously saved.

Items covered by this test: Archives page.

Requirements addressed by this test: Access and Saved jobs requirements.

Environmental needs: Access to Supabase database and user authentication.

Intercase Dependencies: Depends on test ID#2 (Successful user login).

Test Procedures: Users can view the list of jobs previously saved from the "Jobs" page and perform actions on them.

Input Specification: The user is required to be logged into the app to access the Archives page and its contents.

Output Specifications: If the user is logged in, they should be able to access the list of jobs saved, if any. They should also be able to remove them from the list and the view should immediately reflect the changes.

Pass/Fail Criteria: If user is not logged in, they should be prompted with a message indicating that they are required to be logged in to access the contents of the Archives page.

3 Test Results

ID#1 - User Authentication: Sign Up

Date(s) of Execution: 04/11/2024

Staff conducting tests: Frank Mensah

Expected Results: Redirected the Home Page with change to the navbar for 'Login/Sign-up' to be 'Logout' showing the user has been created.

Actual Results: Same actions as from the Expected Results

Test Status: Pass

ID#2 - User Authentication: Log In

Date(s) of Execution: 04/11/2024

Staff conducting tests: Frank Mensah

Expected Results: Redirected the Home Page with change to the navbar for 'Login/Sign-up' to be 'Logout' showing the user has been logged in.

Actual Results: Same actions as from the Expected Results

Test Status: Pass

ID#3 - Housing: Map Integration

Date(s) of Execution: 04/07/2024

Staff conducting tests: Michael Jedziniak

Expected Results: Refer to Supabase DB

Actual Results: Displays the map which is interactive

Test Status: Pass

ID#4 - Housing: Search

Date(s) of Execution: 04/07/2024

Staff conducting tests: David Serrano

Expected Results: Refer to Supabase DB for 'Clark St'

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#5 - Housing: Cards

Date(s) of Execution: 04/10/2024

Staff conducting tests: Vibha Navale

Expected Results: Refer to Supabase DB for '33 W Ontario St Unit 10E32'

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#6 - Housing: Drawer

Date(s) of Execution: 04/10/2024

Staff conducting tests: Michael Jedziniak

Expected Results: Refer to Supabase DB for '33 W Ontario St Unit 10E32'

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#7 - Job: Job list

Date(s) of Execution: 04/15/2024

Staff conducting tests: David Serrano

Expected Results: Refer to Supabase DB

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#8 - Job: Job Description Box

Date(s) of Execution: 04/15/2024

Staff conducting tests: Vibha Navale

Expected Results: Refer to Supabase DB for "Local Gastro Account Manager - Chicago, IL"

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#9 - Job: Search

Date(s) of Execution: 04/15/2024

Staff conducting tests: Michael Jedziniak

Expected Results: Refer to Supabase DB for "HR Transformation" from McDonalds

Actual Results: Same output as from Supabase DB

Test Status: Pass

ID#10 - Archives: Document Storage

Date(s) of Execution: 04/09/2024

Staff conducting tests: Vibha Navale

Expected Results: Access the files the user has submitted into the system.

Actual Results: Same files were there as expected.

Test Status: Pass

ID#11 - Archives: Saved Jobs

Date(s) of Execution: 04/09/2024

Staff conducting tests: David Serrano

Expected Results: Access the saved jobs the user has stored into the system.

Actual Results: Same saved jobs were there as expected.

Test Status: Pass

4 Regression Testing

Not applicable to this Final Report.

IV Inspection

1 Items to be Inspected

Every Pull Request (PR) was thoroughly reviewed and tested by a groupmate (other than the one who created the PR) before merging to the main branch, as seen on our Github repository.

2 Inspection Procedures

In a document on ClickUp (our CASE management tool), we detailed the commit guidelines. Every PR adhered to these guidelines.

An overview of the guidelines we wrote ourselves based on past experience:

- **Commiting to the main branch:**
  - Create a sub-branch
  - Get it reviewed before merging it to the main/master branch
  - Merge the branch into the main branch only after it is reviewed, tested and approved.
- **Commit messages:**

    - Keep it concise – Only specify major changes made in the code wrt to the task & MR (Eg.: Base repo setup, Supabase setup, Bug fix: <Describe bug>)
    - Do not mention any future changes that you plan to work on.
- **Keep the Pull Requests (PR) short and concise:**
    - Split the task into different sub-tasks if there are too many file changes in the PR and create PR for each sub-task.
    - This allows the reviewer(s) not to spend too many hours of their day reviewing and testing the code.
    - The developer can focus on other tasks while their MR is being reviewed and will not have to make too many review changes if the MR is short.
    - Longer MRs may also lead to more merge conflicts.
- **Descriptions on PRs:**
    - Add proper descriptions to your PRs - Describe what the changes in your MR are about.
- **Squash commits and merge:**
    - Squash all the commits before merging your PR to the master branch. This combines multiple commits into a single commit.
    - Select the option for "Squash and Merge" instead of the "Merge pull request".
- **Avoiding merge conflicts:**
    - Take the latest pull from the master/main branch before pushing commits to your PRs.
    - Run the prettier lint fix before pushing commits such that everyone has the same coding format.
- **Do not merge PRs before they are approved:**
    - The reviewer should only review and approve the PR.
    - The creator of the PR should work on the review comments and if none, merge the PR.
    - The reviewer should NOT merge the PR.
    - Since there is no button to approve the MR, the reviewer can leave a comment on the MR (eg.: Approved.) and the MR can be merged by the creator.
    - It is also recommended to message the creator after you are done with the code review (Can be done on the group chat by tagging them). This way, everyone will be aware of the MRs being reviewed and merged.

3 Inspection Results

Every single PR. Here is the list of PRs.

V Recommendations and Conclusions

- Adherence to strict commit guidelines allowed us to work together seamlessly.
- Before having a detailed commit guidelines document and not enough communication in the first few weeks, we faced merge conflicts from time to time and did not follow a common coding pattern. After multiple discussions during our weekly meetings, we agreed upon using reusable components, following a modular approach when implementing our code and setup linting which allowed us to have more concise and cleaner code with little to no conflicts or bugs.
- Additionally, we documented our findings, discussed with the group and after implementation, tested out each feature before merging the branches into the main branch. The analysis documentations we wrote and shared with the group improved code quality and greatly reduced the time we otherwise would have spent on refactoring code.

VI Project Issues

1 Open Issues

Looking back to the original outline of the project [1], the Mentorship Platforms feature still will need to be constructed and implemented for future releases of the application. This feature will need to be made to allow the user to find a mentor to help with professional and academic needs. The way of implementing this can be very different from developer to developer. As well as, finding information regarding accredited mentors to help the user will need to be researched and found to be used in the application.

2 Waiting Room

Regarding potential waiting room features, there are a lot of new React Components that can be used to help add more modern CSS designs to various pages. For instance, a use of an accordion component would be beneficial for the job details. This is due to the fact the current way of displaying it is using the job description provided from the person who posted the job which can be very long when displaying on the page. Adding an accordion to these text segments would help with user experience and provide a better and familiar design for many users.

3 Ideas for Solutions

Not applicable to this report

4 Project Retrospective

Overall, the majority of the technologies and software engineering methodologies we used worked well for all members of the group. Regarding the technologies used, Supabase and utilization of React Components worked well to eliminate a lot of potential issues when trying to store data and reuse of code throughout different pages

of the application. For methodology methods, our weekly meetings worked well with each other and the use of continuous integration was also beneficial.

Regarding tools that did not work as well as we have planned, ClickUp was a big issue with us. Due to its odd UI setup and lack of features such as a setup workflow from backlog to open issues, we had to spend a lot of time to figure out how to best use this tool with the limited features it had. As a result, further into the development of the project, we utilized ClickUp less and less with each release.

VII Glossary

Class: A blueprint for creating objects, defining their properties (attributes), and behaviors (methods).

UML (Unified Modeling Language): A standardized visual modeling language used for specifying, visualizing, constructing, and documenting software systems.

Deployment Diagram: A diagram that shows the physical arrangement of software components on hardware nodes.

Client-Server Architecture: A network architecture where client computers request services or resources from server computers.

Microservices Architecture: A software architecture where applications are built as a collection of loosely coupled services.

Singleton Design Pattern: A design pattern ensuring a class has only one instance and provides a global point of access to it.

Agile Methodology: An iterative approach to software development emphasizing flexibility, customer collaboration, and incremental delivery.

API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other.

MVC (Model-View-Controller): An architectural pattern separating an application into three interconnected components: Model (data), View (UI), and Controller (logic).

Data Persistence: The ability of data to outlast the process that created it, often achieved through storage in databases or files.

Encryption: The process of converting data into a code to prevent unauthorized access.

Risk Management: The process of identifying, assessing, and mitigating potential risks that could affect the project.

Continuous Integration (CI): A development practice where developers integrate code into a shared repository frequently, often several times a day.

API Gateway: An intermediary between clients and backend services, managing requests, and handling security, load balancing, and more.

GUI (Graphical User Interface): The visual part of a computer program that allows users to interact using graphical elements.

Homelessness: The state of lacking permanent housing and being unable to secure or maintain housing due to financial or other constraints.

Affordable Housing: Housing deemed affordable based on an individual or family's income, often costing no more than 30% of monthly income.

Case Manager: A social services professional who coordinates services and resources to assist homeless individuals.

Transitional Housing: Temporary accommodations provided with support services to facilitate the movement of homeless individuals to permanent housing.

Supported Employment: Employment assistance programs that provide additional support like training and counseling to individuals facing challenges getting hired.

VIII References / Bibliography

[1] S. Burney, R. Dhotre, N. Patel and R. Rizvi, "FindYourRoof Project Final Report", 2023.

[2] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

IX Index

Project Issues - 22