Project Title: Fixer Uppers

Team Name: Databoss

Database Design
May 09, 2025

Group Captain: Lesli Perez Garcia
Members: Lesli Perez Garcia, Edwin Serna, Gabriel Herrera, Edwin Lozano, Miguel Torres

1. Introduction & Database Design

This project represents the final stage of our system development process. In Part A, we identified the core problem and developed an initial database design using an Entity-Relationship (E-R) model. Part B focused on designing the system architecture, including a State Transition Diagram and detailed page descriptions. While the foundational designs from Parts A and B have remained largely consistent, this final phase is centered on the full implementation of the Fixer-Uppers website. It brings together the earlier planning and design work into a functioning, user-ready application.

1.1 Problem Statement

Finding a reliable professional to enter your home and perform necessary repairs, such as cleaning or plumbing, can be a major challenge. Many people living in the same city offer these services, but they are often found through unreliable sources like scattered online listings or newspaper ads, making it difficult to verify their credibility. This lack of a trustworthy platform creates frustration for homeowners in need of quick repairs and for skilled professionals looking for consistent work.

This project aims to bridge that gap by creating FixerUppers.com, a platform that connects homeowners with trusted professionals. Users can easily post job requests when they need repairs or services, while qualified contractors can browse and accept jobs that match their skills. Once a job is accepted, either party can initiate direct messaging to coordinate details. Contractors can update the job status as they progress, ensuring transparency. To build trust and accountability, both users and contractors can rate each other based on their experience. This rating system creates confidence in the platform, making it easier for homeowners to find dependable help and for professionals to secure reliable clients.

FixerUppers.com hopes to provide a solution that eliminates the uncertainty of finding dependable and safe home service providers. With a user-friendly system, the platform will ensure that homeowners can hire with confidence and skilled professionals get work opportunities.

1.2 E-R Design

The entity relation diagram shows the relations of each table and how they interact with each other. Users can send direct messages to contractors, post job requests, and create reviews for contractors. Users also get rated by the contractors they hire, so contractors can also trust the clients.

Contractors are, by default, users and have to opt-into becoming a contractor. Like the users, contractors have the ability to send direct messages, create reviews, and receive reviews from their clients. They will have the ability to take on jobs from job requests and can find jobs based on their job title.

Users can create job requests that upon creation have an “open” status. Contractors are then able to view these job requests if the request type fits what the contractors offer. Contractors are able to see the offering price up front and when a job is accepted, the status automatically changes to “in progress”. Once a job is complete, the contractor is able to change the status to “closed”. A contractor will also be able to view all of the jobs they have

accepted in one place. The arrows in Figure 1 denote the cardinality constraints, such as many-to-one, while the diamond shapes represent relationship sets.

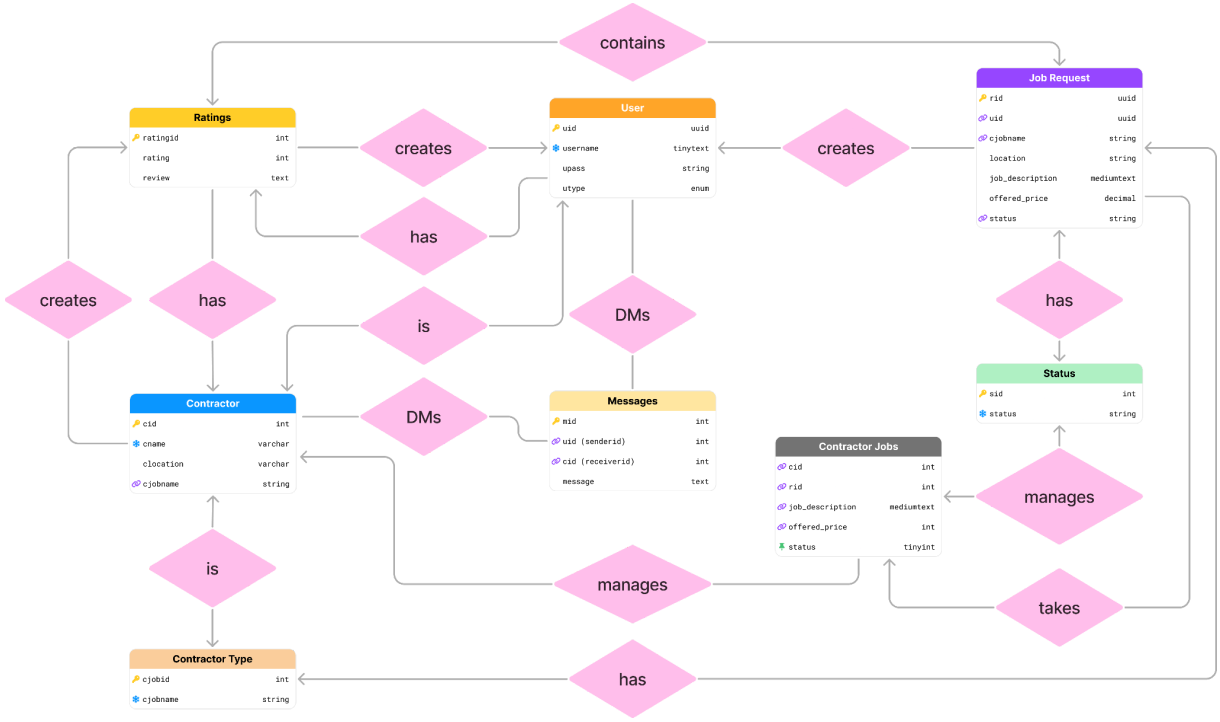

Figure 1: E-R Design illustrating table relationships for FixerUppers.com. For full details on table attributes, refer to Figure 2 and Sections 2.5 - 2.7.

1.3 Table & Example Naming Conventions

For the user of this document, the database table naming will be based on the following:

Primary Key: The primary key is an attribute that uniquely identifies each record in the table. Primary would be denoted as the first letter of the table and the logical name of the field. Exception is for the "Ratings" table as they share the same first letter, changing the first condition from the first letter of the name to the full name. For example, **uid** is a primary key in the "User" table as "u" is given by "user" and "id" is the logical name.

Unique Attribute: A unique attribute is a condition that a certain attribute must have unique values to prevent the duplication of data and identity. For example, *cname* is a unique value in the "Contractor" table as we prevent two or more different contractors from sharing a common name.

Foreign Key: A foreign key is an attribute in a table that is used in a relationship with another table attribute which carries unique identification as primary key or unique attribute. For example, uid in the "Job Request" table has a unique identification as primary key in the "User" table as the primary key. Another example is the cjobname in the "Contractor" table holds a foreign key from the "Contractor Type" table *cjobname* which is a unique attribute.

For the tables shown within this document would carry this format:

🔑Key Logo - Primary Key
❄️Snowflake - Unique Attribute
🔗Chain Link - Foreign Key

On the other hand, any writing and examples defining these attributes would carry this format to implicitly define the naming of the attributes:

Bold - Primary Key
Italicized - Unique Attribute
<u>Underlined</u> - Foreign Key

1.4 Relational Database Design

The FixerUppers database consists of 8 tables, the attributes and relations of these tables are shown in Figure 2.

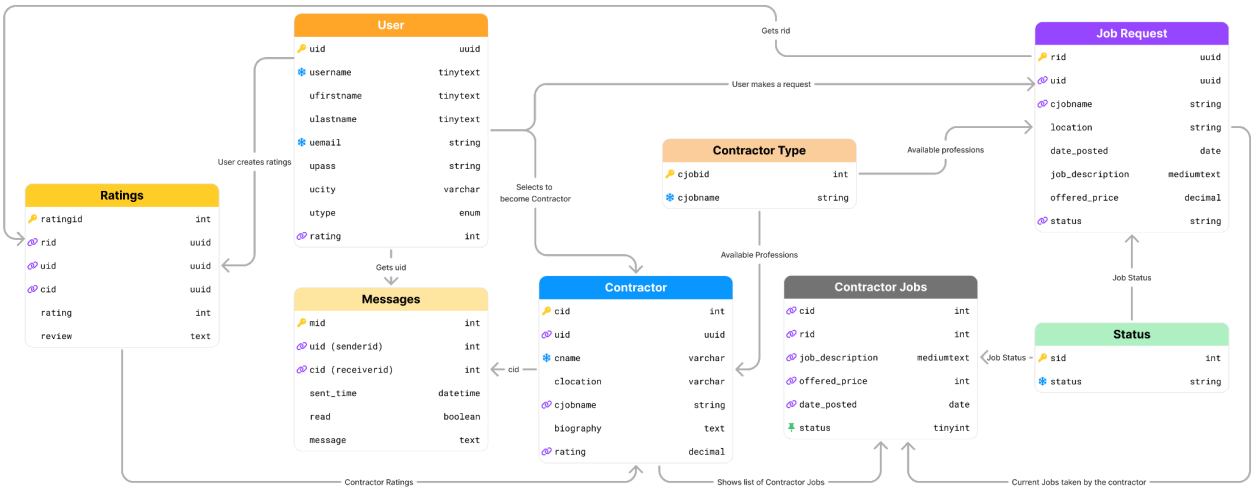

Figure 2: Complete Relational Database Design for FixerUppers.com, with relationships represented by arrows indicating foreign keys.

1.5 User Tables

Description: This "User" table is where user data and defining characteristics will be stored. This table will be the parent table for the contractors table based on the utype.

Users Example								
uid	*username*	ufirstname	ulastname	*uemail*	upass	ucity	utype	<u>rating</u>
U115	mikelRocks	Michel	Gonzalez	M@gmail.com	pass123	Pharr	Client	4.62
U615	jenyiscool	Jennifer	Ross	Jen@outlook.com	Jenny23	Mission	Contractor	3.91

Table 2: Example of a populated Users Table.

uid: Holds the unique global ID for each user and will be the primary key. Type: Integer

username: Users will have a unique ID that they will not be able to view. Type: Text (16)

ufirstname: User's first name. Type: Text (30)

ulastname: User's last name. Type: Text (30)

uemail: User's email must be unique. Type: Text (30)

upass: User's password. Type: Text (64)

ucity: The city they are from to show nearby contractors. Type: Text (30)

utype: The user's role as either just a client or both a client and contractor. Type: Text (64)

rating: The user's rating, which is a foreign key from the "Ratings" table. Type: Real

1.6 Jobs Tables

Description: The "Job Request" table stores all information for jobs posted. It involves a many-to-one relationship with the "User" table. This is because many jobs can be held by one user. Another relation that the "Job Request" holds is with "Status" as a many-to-one. The same reasoning appears as many job requests can have the same one status. Lastly, "Job Request" holds a many-to-one relationship with "Contractor Type", with the same purpose to hold a category that many job requests can have one contractor type.

Job Request Example							
rid	uid	cjobname	location	date_posted	job_description	offered_price	status
0	U115	plumber	Pharr	2025/02/03	Needs to replace sink	$700.00	Open
1	U615	cleaner	Edinburg	2025/02/10	Needs help cleaning	$50.00	In Progress

Table 3: Example of a populated Job Request Table.

rid: This is the primary key for the "Job Request" table. Type: Integer.

uid*:* This is a foreign key linking the "Job Request" table to "User" table. Type: Integer.

cjobname*:* This is a foreign key linking "Job Request" table to "Contractor Type" table. Type: Text(30).

location: This variable would specify the location that the job is going to take. Type: Text(30).

date_posted: This is a date variable that would contain the data that the user posted the job. Type: Date

job_description: This variable would hold a description given by the users. Type: Text(1000).

offered_price: This would be a decimal value to save the price for the job request created. Type: Decimal (7,2).

status: This is a foreign key linking the "Job Request" table to "Status" table. Type: Text(30).

1.7 Contractor Tables

Description: This table displays the unique contractor business name, user ID, contractors's location, type of job performed, small biography of how their business started and how well they approach their job, and rating for the specified job.

Contractor Example						
cid	uid	*cname*	clocation	cjobname	biography	rating
0	U115	Mario Bros	McAllen	plumber	Working since 1970...	4.71
1	U615	Mari Cleans	Edingburg	Cleaning	Quality cleaning...	2.45

Table 4: Example of a populated Contractor Table.

cid: Holds the contractor ID for each contractor. It serves as the local primary key for each contractor. Type: Integer

uid: Holds the unique global ID for each user. It is a foreign key retrieved from the User table.

cname: The name of the contractor's business. This name must be unique and cannot be repeated. Type: Text (30)

clocation: Specifies the contractor's location and the city where they provide their services. Type: Text (30)

cjobname: Specifies the type of service provided. This is selected from a dropdown menu populated with data from the Contractor Type table. Each profession is unique, and users can select only one.

biography: A short description of the business, including its foundation, principles, service quality, and approach toward clients. Type: Text (30)

rating: A foreign key from the “Ratings” table. It aggregates all ratings received by the contractor and calculates an average, allowing users to assess their service quality.
Type: Real.

2. System Architecture Design

In this part, we transitioned from conceptual planning to detailed system architecture design. This phase focused on outlining how the Fixer-Uppers website would function structurally. Key deliverables included the State Transition Diagram, which mapped out user interactions and system behavior, as well as detailed page descriptions that defined the layout and functionality of each component. These architectural plans served as a blueprint for the final implementation developed in Part C.

2.1 State Transition Diagram

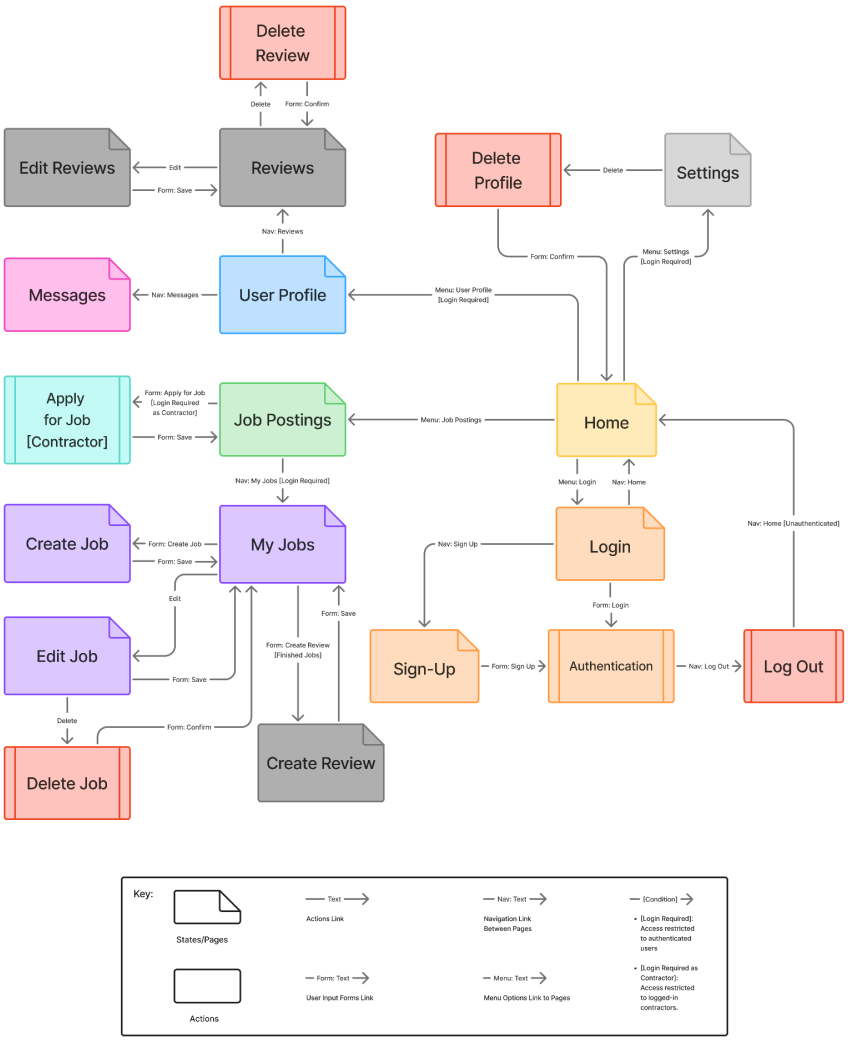

Figure 3: State Transition Diagram

2. 2 Architecture Page Descriptions

1. **Home**

   Description: The home page has several key sections: The About Us explains what the site offers and a set of instructions on how to post their own job to connect with contractors. The Anonymous Review section shows feedback from users who interacted with our site to complete their tasks. Users can log into their accounts from the Login tab and start making requests on Job Postings. The Contact Information is located at the bottom of the page for questions, comments, and/or concerns.

   Input: None

   Output: Views on ‘About’ section, pictures, reviews, ‘Login’ tab, ‘Job Postings’ tab, and contact info.

   Pseudocode:
   Begin
   Display all components and navigation menu
   End

2. **Login**

   Description: A basic login page with the site logo displayed above the username and/or email input. Includes a button to sign-up if the user does not already have an account registered.

   Input:
   - username or uemail
   - upass

   Output: Automatically redirects to ‘Home’ after logging in, signup component.

   Pseudocode:
   Begin
   Display logo, two input boxes, enter button, and sign-up button
   Authenticate username/email and password with database
   Return to ‘Home’ page or route to sign-up
   End

3. **Sign-up**

Description: Users will have to sign up/log in if they want to create or accept job postings. This information will be stored in the database and can be retrieved efficiently with user id.

Input:

All Users:

- username
- ufirstname
- ulastname
- uemail
- upass
- city

If Contractor checkbox is checked:

- cname
- clocation
- cjobname

Output: Redirect to 'Home' after creation

Pseudocode:

Begin

Display all input boxes and checkbox
If contractor checkbox is clicked then allow user to input data for contractors
Else leave contractor inputs uneditable
Generate a unique uid for the user
If user checks the contractor checkbox, a cjobid is generated
Return to 'Home' after completion and.

End

4. **Job Postings**

Description: Contains a table with unique job postings and the ability to filter through them. Each posting will have the first name and last name initial of the person who posted the job and their username. It will also include a general location of the job so a contractor can filter from their preferences, but not give an exact location for safety. Additionally, a detailed job description and the offered price will be included. Each posting will have a status, by default only open jobs that have not been accepted by a contractor yet are displayed.

Input: None

Output: Jobs table, ‘My Jobs’ tab, ‘Login’ tab, ‘Home’ tab, ‘User profile’ tab, 'Settings’ icon

Pseudocode:
Begin
    Display table of job postings and navigation menu
    Filter tab that can display postings based off user preferences
    Search bar at the top of the jobs table
    If user is logged in as a contractor, they will have the ability to apply for jobs
    Else the user will be redirected to ‘Login’
End

5. **My Jobs**

Description: Contains a table with the jobs that a contractor or user created. Shows job status and details. Contractors are able to see the jobs that they have accepted and are able to change the status of those jobs. Users are able to see a list of job applicants and choose one to take up the job. Upon completion of a job, both clients and contractors are prompted to leave a review. The page consists of a top menu that includes ‘Jobs’, ‘About’, and ‘Contact’ along with the user menu. The ‘Jobs’ link would have all of the job postings while ‘About’ and ‘Contact’ would link to the ‘Home’ page. To facilitate the search for a specific job, there would be a search bar on the top, and a filter menu on the left side.

Input: None

Output: Redirected to ‘Creation/Edit Jobs’ or ‘Create Review’ pages

Pseudocode:
Begin
    Display table of user’s job postings and navigation menu
    Filter tab that can display postings based off user preferences
    Ability to edit posted jobs
    Display link to post reviews on completed jobs
End

6. **Edit Jobs**

Description: Contains input boxes for the following: job name, job location, job description and offered price. It allows modification of the status of a current job. The pre-loaded content of the page includes user information such as username and rating. Then there would be the job name on top followed by the posted period and the job type, and the offered price on the right top corner. Contractors can only update a client’s job status and post reviews after a job is completed.

Input:

- cjobname
- location
- job_description
- offered_price
- status

Output: Redirection to ‘My Jobs’ page

Pseudocode:
Begin
    Display previously inserted inputs with the ability to manipulate the text
    Posted job will have the status as open upon creation
    Return to ‘My Jobs’ when edits are saved
End

7. Create Jobs

Description: Contains input boxes for a user to fill such as: job name, job location, job description and offered price. The pre-loaded content of the page will include the user information such as username and rating. After the form is filled, the user is redirected to the ‘My Jobs’ page and will be able to view/edit their job posting. The time stamp will be included on the post.

Input:

- cjobname
- location
- job_description
- offered_price

Output: Redirection to ‘My Jobs’ page

Pseudocode:
Begin
    Display inputs that the user must fill out to post job
    Posted job will have the status as open by default
    A rid will be generated once the user clicks submit
    Return to ‘My Jobs’ when posted
End

8. User Profile

Description: Users will see their profile or other user profile details. Profile details include: username, first name, last name initial, rating, open jobs posted by the user, and reviews.

Input: uid

Output: Display User Profile details

Pseudocode:
Begin
    Display all data for the user, reviews, average ratings, and navigation menu
End

9. Messages

Description: This section includes a list of all users with whom messages have been exchanged. It allows users to select a contact to message, view previous conversations, and send new messages. The message layout features a chat interface where users can continue conversations with the selected contact, along with a search bar at the top, and a list of chats on the left side.

Input: message

Output: sent and received messages

Pseudocode:
Begin
    Display recent recipient chat and input message box
    Navigation tab between all other recipients sorted with the most recent messages at the top
End

10. Create Review

Description: This page includes a simple feature where either the user or contractor can leave a review for each other. The review will include an overall rating and a comment. The layout will display the information of the user or contractor being rated at the top left, with the job name and type in the center at the top. Below that, users can provide a rating, write a review, and have the option to attach images.

Input:
- rating
- review
- images (optional)

Output: Redirection to ‘My Jobs’ after submitted

Pseudocode:

Begin

Display two inputs:

1-5 score for user

Input box for review comment

Display attachment icon

A ratingid will be generated after submitting review

Return to ‘My Jobs’ after completion

End

11. Edit Review

Description: Similar to the ‘Create Review’ page, the user will have the ability to edit their posted reviews. The previous data will automatically propagate to the insert text boxes but will allow the user to manipulate the data as they wish.

Input:

- rating
- review
- images (optional)

Output: Redirection to ‘My Jobs’ after submitted

Pseudocode:

Begin

Display two inputs with previous data:

1-5 score for user

Input box for review comment

Display attachment icon

Returns to ‘My Jobs’ after completion

End

12. Settings

Description: Contains information about the user, holds personal information, notification settings, and account details. Personal information includes: location, first name, last name, email, and account type. Notification settings let a user change the type of notifications they want to receive. Lastly, account settings give the user the option to delete their account or change their password. The settings page will feature a menu on the left, offering options for personal information, notifications, and account settings. The details for the selected option will be displayed on the right side of the page.

Input:

- ufirstname
- ulastname
- ucity
- uemail
- upass
- utype

Output: Personal Information Settings, Notification Settings, Account Settings

Pseudocode:

Begin

    Display all components and navigation menu

End

13. Actions

Description: Certain pages will call these actions which will manipulate/affect database records or the user's signed-in status.

1. Delete Review

Description: A button that would allow users/contractors to delete reviews they have posted. Reviews a user/contractor receives cannot be deleted.

Input:

- ratingid

Output: Deletion of the review.

Pseudocode:

Begin

    Get ratingid
    Search ratingid in Rating Table
    Delete ratingid entry

End

2. Delete Profile

Description: A button that allows the user to delete their account if they no longer want to continue enrollment in the website.

Input:

- uid

Output: Deletion of the user account

Pseudocode:
Begin
    Get uid
    Search uid in User Table
    Delete uid entry
End

3. **Delete Job**

Description: A button that allows users to delete their created jobs from the ‘My Jobs’ page.

Input:
- rid

Output: Deletion of the job request

Pseudocode:
Begin
    Get rid
    Search rid in Job Request Table
    Delete rid entry
End

4. **Apply for Job**

Description: A button will be available that allows contractors to apply for a job posted by a user. Only contractors will have access to the "Apply for Job" button.

Input:
- Contractor uid
- rid

Output: Links Contractor to the job accepted

Pseudocode:
Begin
    Retrieve rid
    Search rid in Job Request Table
    Get the contractor uid linked to the entry for the rid
End

5. **Authentication**

Description: A button that allows the user to confirm their data is authorized and correct.

Input:
- username or uemail
- upass

Output: User is now logged in with a user account

Pseudocode:
Begin
    Get username or email input
    Search username or email in User Table
    If not found:
        No authentication
    Else:
    Search upass
    If upass matches uid:
        Authentication is true
        Begin Session
    Else:
        No authentication
End

6. **Log Out**

Description: A button that allows the user to log out of their session.

Input:
- uid

Output: Users who are not logged in will be redirected to the 'Home' page.

Pseudocode:
Begin
    Log out user of if session is active
    Redirect to 'Home' Page
End

3. Implementation

The final implementation phase of the Fixer-Uppers project focused on transforming earlier design and architectural plans into a fully functional web platform that connects homeowners with local service professionals. The website was developed using Next.js for the front-end and routing, AWS for deployment and backend infrastructure, and JavaScript

and CSS for the core logic and styling. This stack was chosen to ensure scalability, maintainability, and a modern user experience. When a user opens the website they first see the Home page.

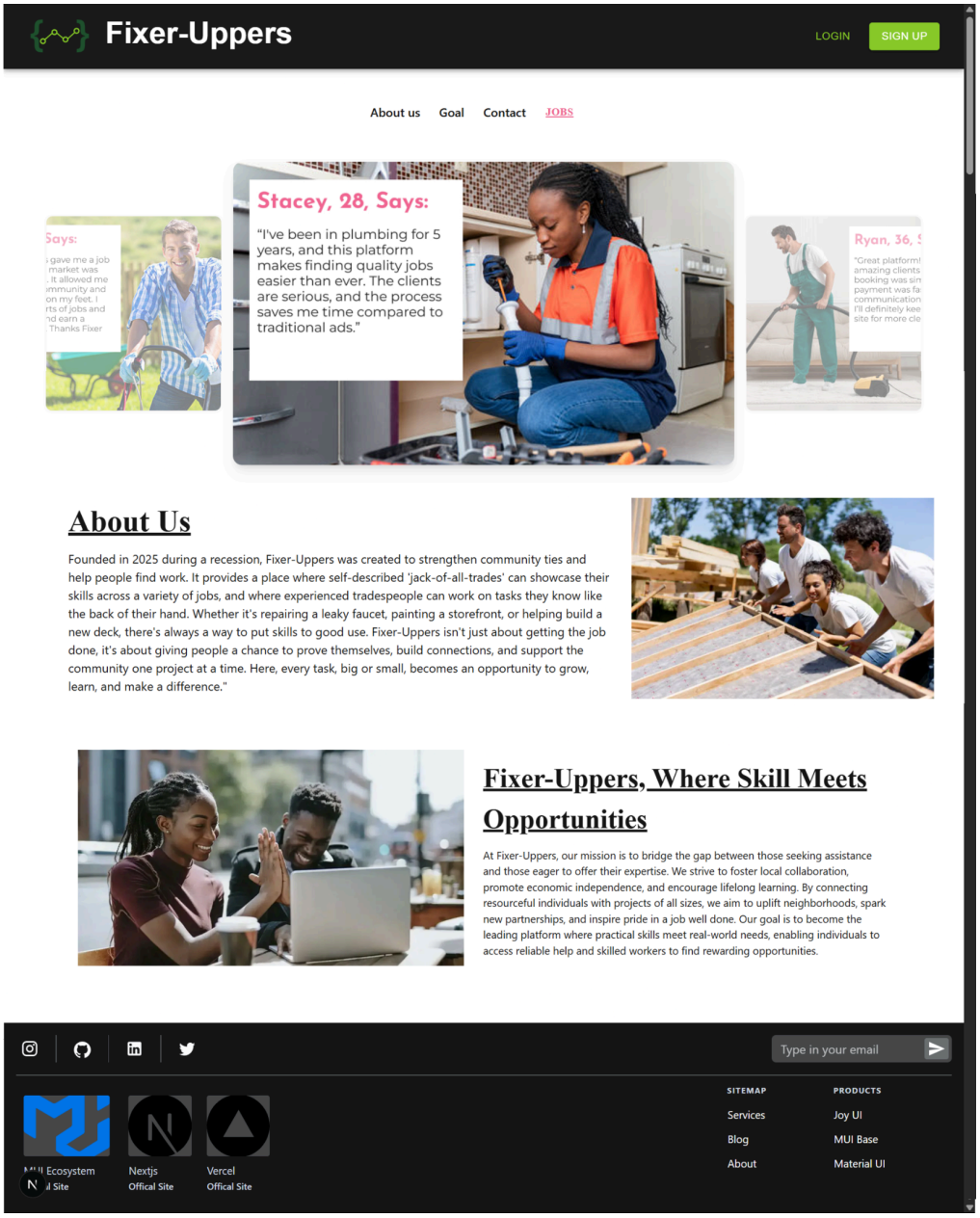

Figure 4: Fixer-Uppers Home Page.

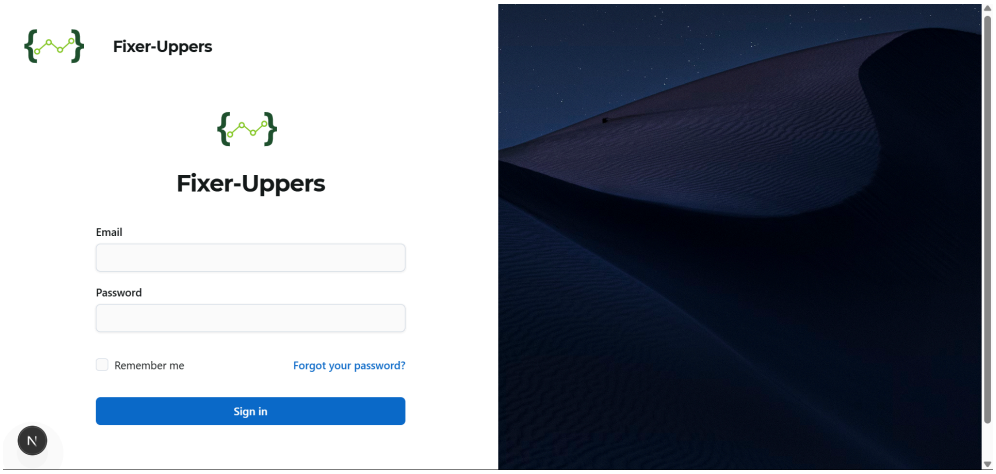

Figure 5: Fixer-Uppers Login Page.

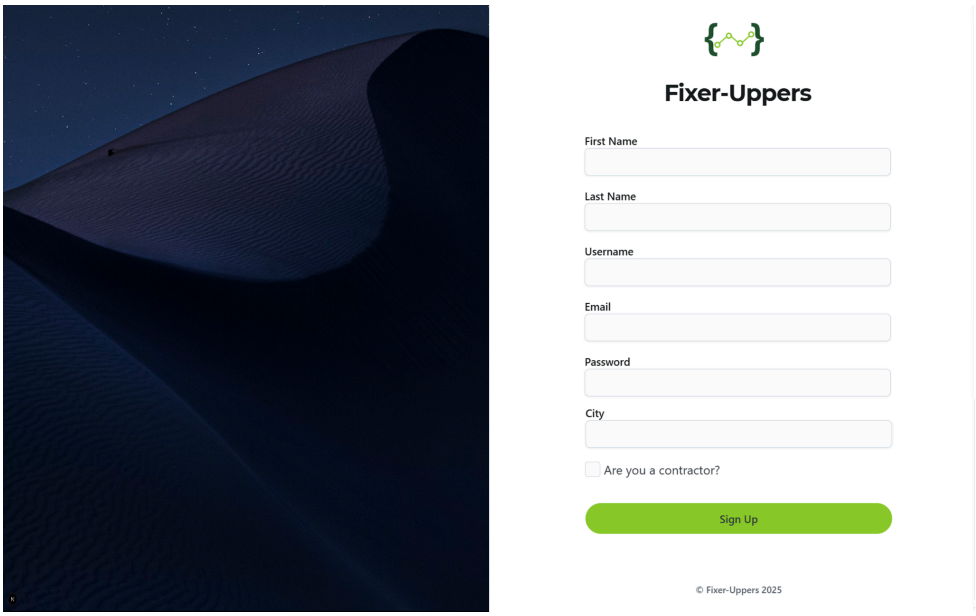

Figure 5: Fixer-Uppers Sign Up Page.

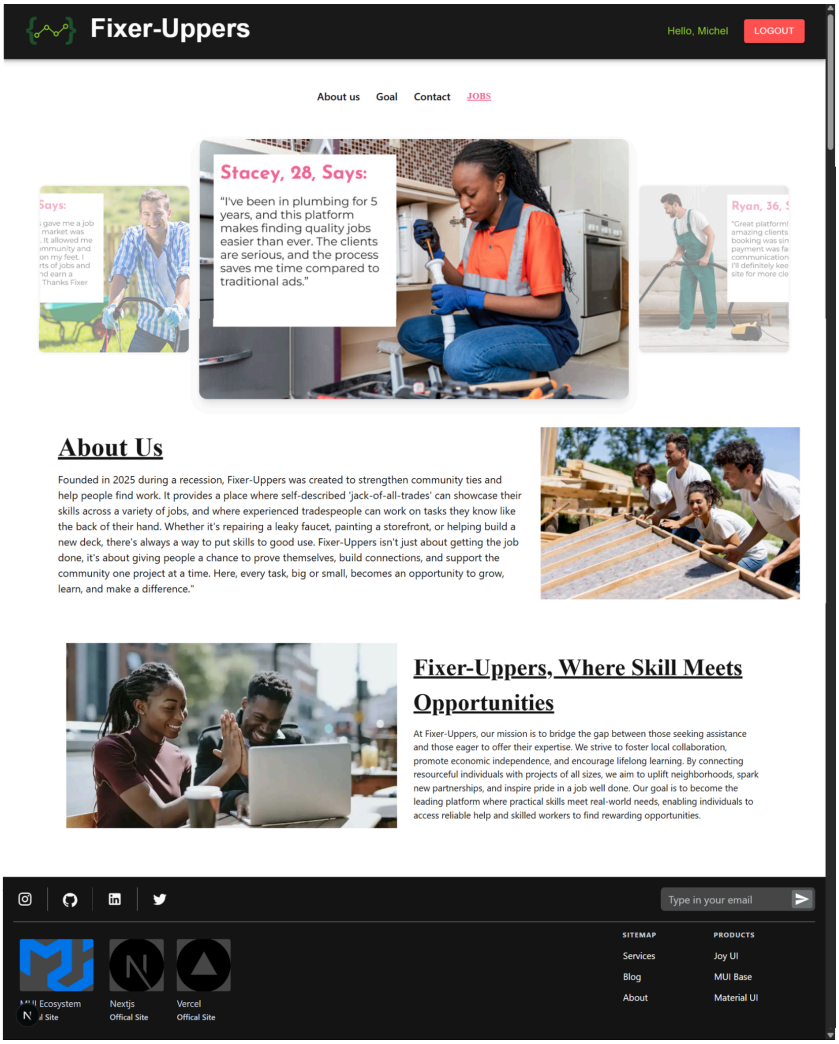

Figure 6: Fixer-Uppers Sign Up Page Logged In.

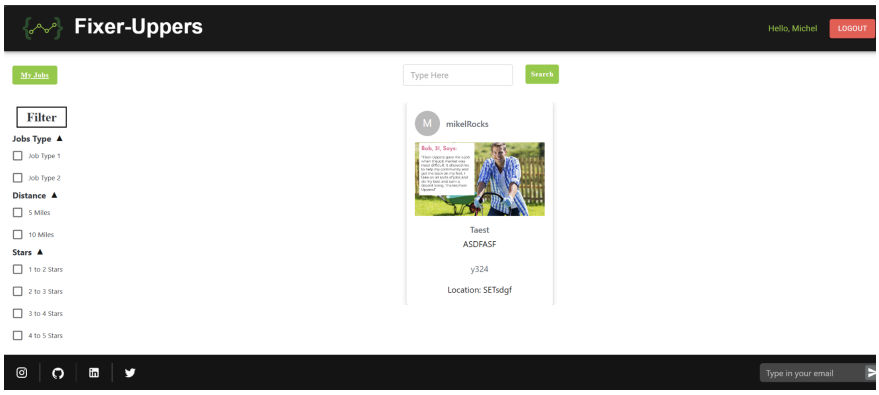

Figure 7: Fixer-Uppers Jobs Page.

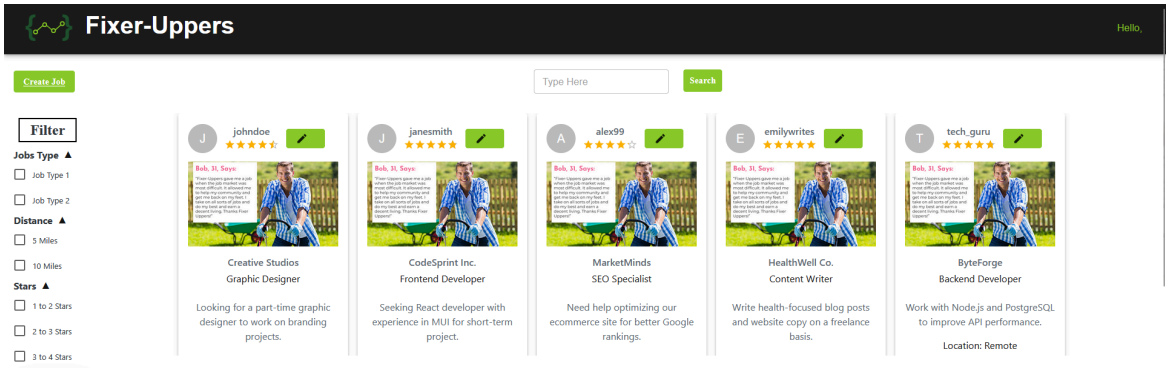

Figure 8: Fixer-Uppers My Jobs Page.

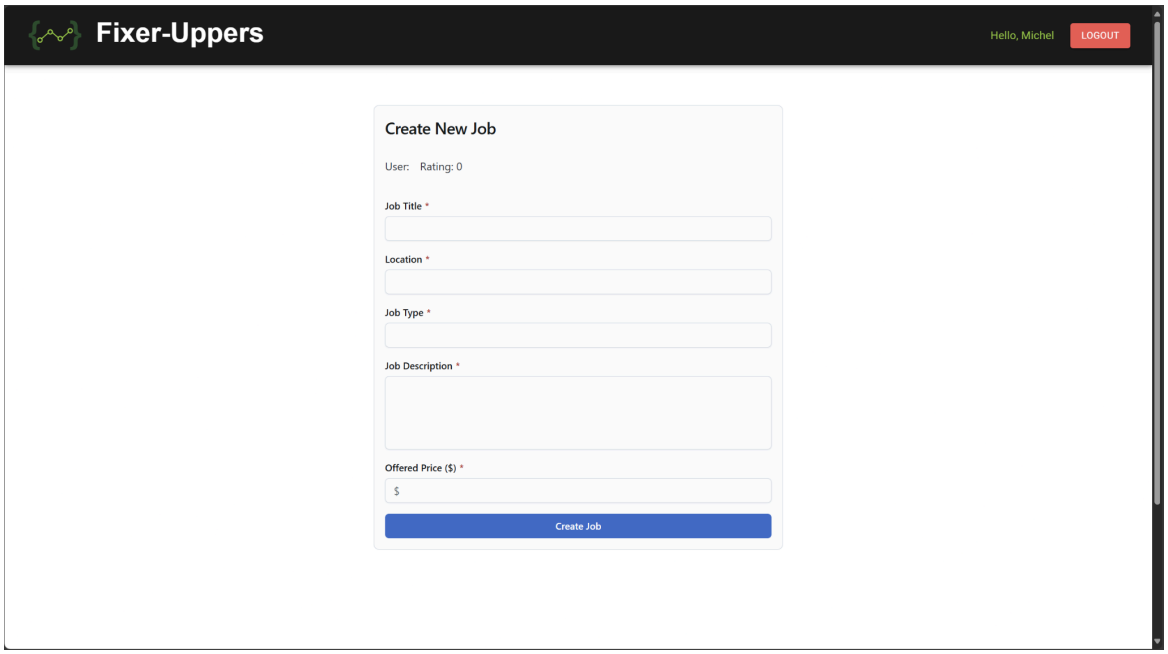

Figure 9: Fixer-Uppers Create a Job Page.

4. Usability

Fixer-Uppers was designed to provide a smooth and intuitive experience for both homeowners and service professionals. The user interface was designed on Figma and emphasizes clarity and ease of navigation, with structured pages that guide users through the process of posting jobs, browsing job listings, and managing their accounts. A cohesive and modern color scheme was chosen to enhance visual appeal and aesthetics and keep users engaged. The layout and design elements maintain consistency across pages, ensuring users can easily understand and use the website. Extensive attention was given to visual design using Figma, where all core pages, like login, sign-up, home, and job listings, were planned and laid out with usability in mind. These designs prioritize accessibility, legibility, and user flow, helping to create a welcoming and professional interface that supports the platform's functionality.

4. 1 Website Color Palette

The color palette for the website was inspired by our team logo, which features black, lime green, and forest green. Typically, color palettes consist of two dark colors and three lighter shades. For this project, we chose black and forest green as the two dark colors, while white, lime green, and bright pink served as the lighter tones. This combination of colors provides the website with a vibrant pop and enhances its overall aesthetic appeal, as shown in Figure 4.

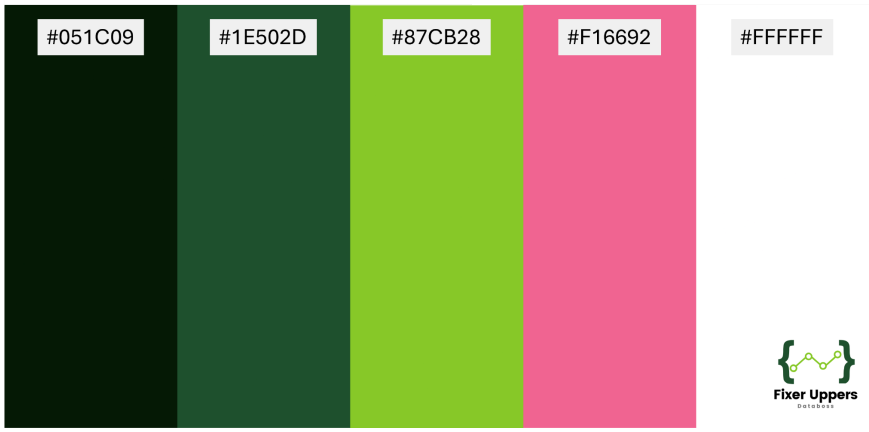

Figure 4: Fixer-Uppers Color Palette.

4. 2 Figma Pages

The following figures showcase the key pages of the FixerUppers.com platform, designed in Figma to reflect a clean, accessible, and professional user experience. The layout, colors, and navigation elements were carefully chosen to align with the project goals.

1. Home Page:

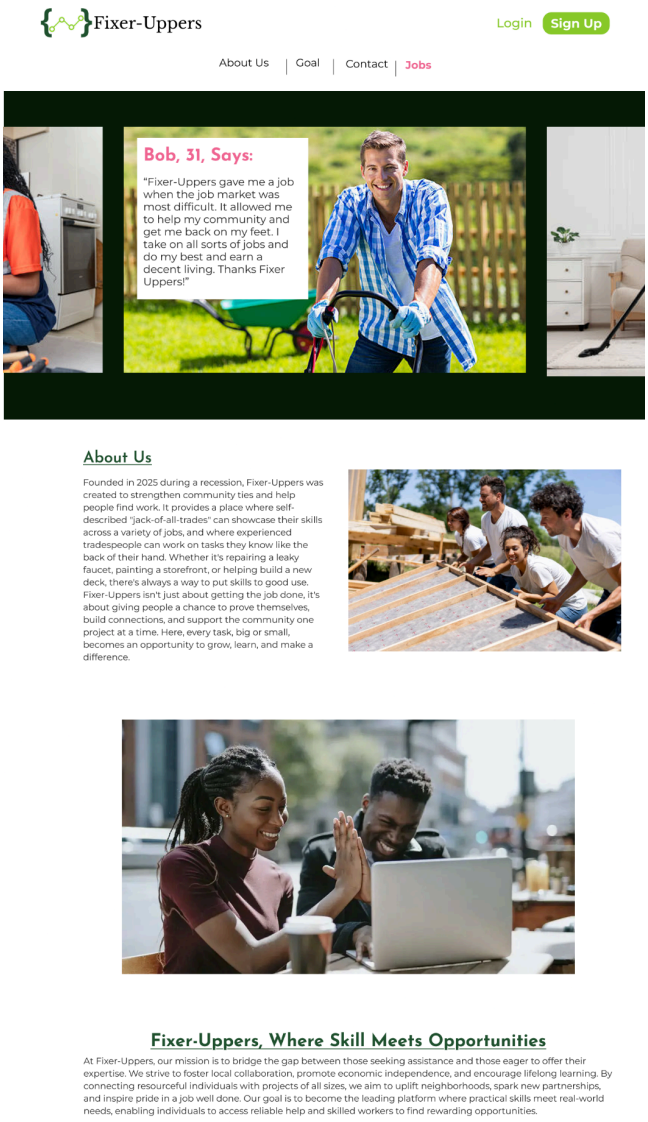

Figure 5: Fixer-Uppers Home Page.

2. Sign Up (Client):

Figure: Fixer-Uppers Client Sign Up Page.

3. Sign Up (Contractor):

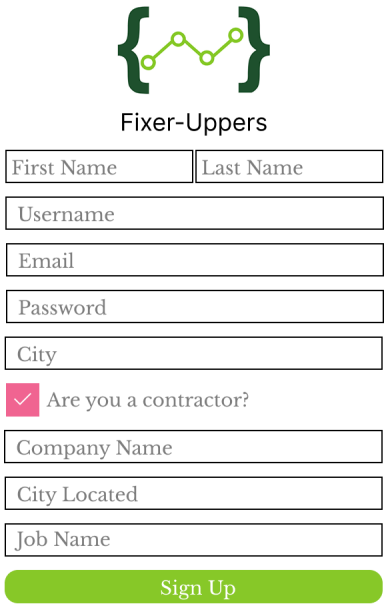

Figure: Fixer-Uppers Contractor Sign Up Page.

4. Sign In:

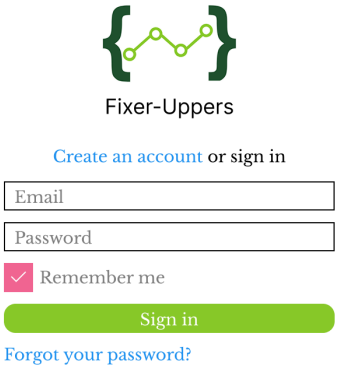

Figure: Fixer-Uppers Login Page.

5. Job Listings:

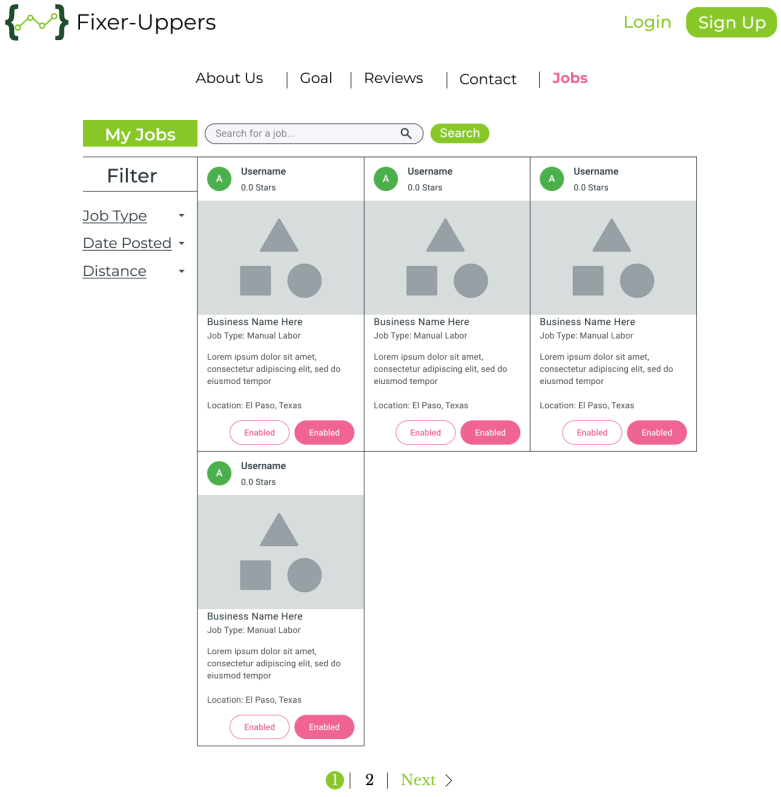

Figure: Fixer-Uppers Job Listings Page.

6. My Jobs (Edit):

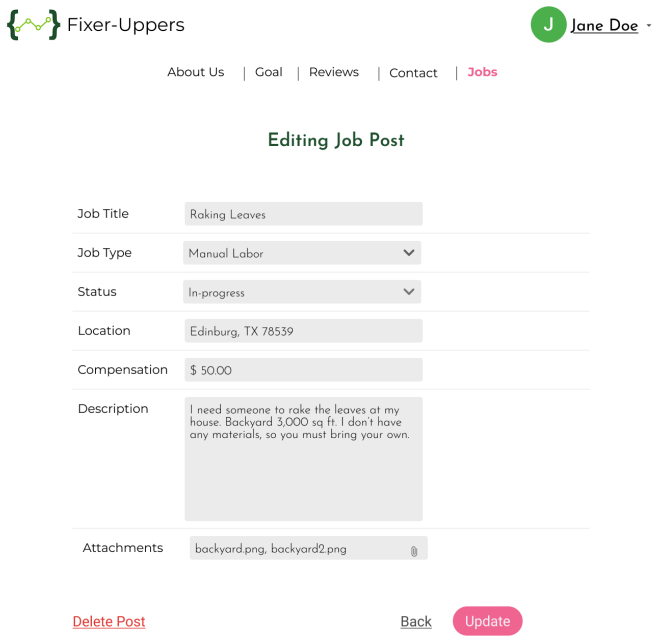

Figure: Fixer-Uppers Edit a Job Page.

7. My Jobs (Create):

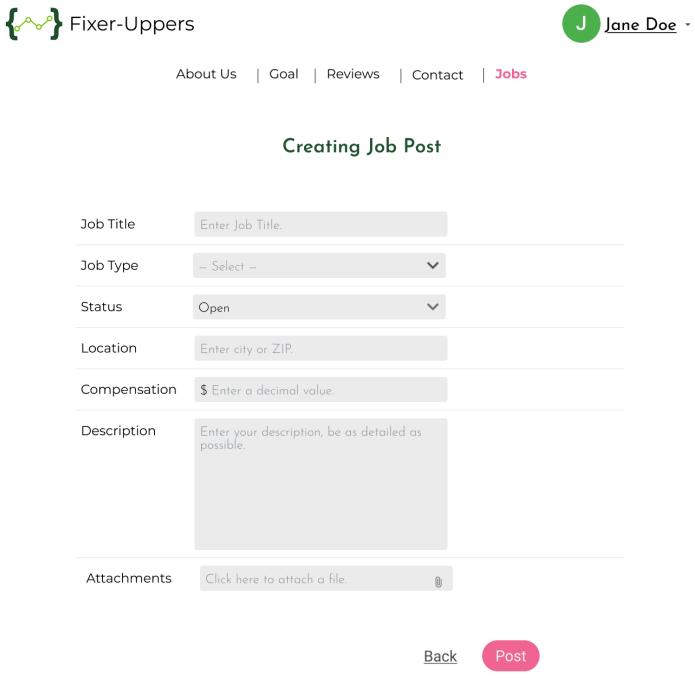

Figure: Fixer-Uppers Create a Job Page.

8. View User Profile:

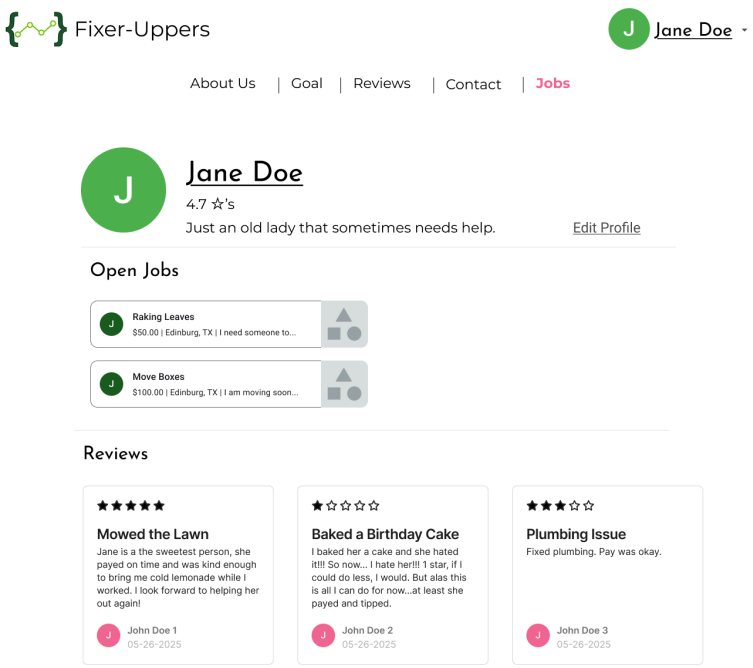

Figure: Fixer-Uppers View Profile Page.

5. Testing

Quality Assurance encompasses all activities to ensure software meets quality standards. In the Fixer-Uppers project, this included not only design and implementation but also a hands-on approach to testing the website's functionality and user experience. We conducted thorough manual testing to validate the system which included:

- Exploratory Testing: Team members and volunteers interacted with the site as real users would, navigating through pages, posting jobs, signing up as clients or contractors, and logging out. This approach allowed us to quickly identify usability issues, unexpected bugs, and areas where the user flow could be improved. Volunteers helped us see where the UX/UI was lacking as they had never used the website before.
- Ad Hoc Testing: We performed unscripted, targeted testing sessions-attempting to "break" the site by entering unusual data, rapidly switching between pages, and simulating edge cases.

- Usability Checks: By manually exploring the interface, we assessed the clarity of navigation, the intuitiveness of forms, and the overall user experience. Manual usability testing is crucial for ensuring the application is accessible and user-friendly, as automated tools cannot fully evaluate these aspects.

Manual testing was selected for this because the application was still evolving, and requirements were subject to change, making it more practical to adapt test scenarios on the fly rather than maintain automated scripts.

6. Member Contributions

6. 1 Lesli Perez Garcia

For Part A of the project, I was involved in the core database design work. I contributed to the creation of the E-R design image, wrote and refined the problem statement, and developed the E-R design description to clearly communicate the relationships and logic behind our data model. As team leader, I revised and submitted the final report for this section, ensuring that our documentation was thorough and well presented and I created the logo and cover page for our team.

In Part B, my main focus was on the system architecture design. I was responsible for creating the state transition diagram, mapping out the various user interactions and system states for the Fixer-Uppers platform. Additionally, I edited the team's initial drafts for the architecture page descriptions, to improve grammar and clarity. I also handled the revision and submission process for this section, making sure our architectural documentation was complete and cohesive.

For Part C, I contributed to both the design and implementation of the project. I hand-drew all the initial page layouts as low-fidelity wireframes, which served as the blueprint for our later digital designs in Figma. I chose the project's color palette, designed and implemented key pages such as the home page, My Jobs (edit and create), and the view user profile page, and made general tweaks to enhance usability and appearance on Figma. On the technical side, I improved the navigation bar to display a personalized greeting and logout button after login, which required changes across the database, route files, and navigation component. I also conducted manual exploratory and ad hoc testing for quality assurance, and I was responsible for writing the report for Part C, ensuring the final documentation accurately reflected our work and met project standards.

6. 2 Edwin Serna

I contributed to the development of our project by helping with the design of the E-R diagram in Part A by identifying issues in the relational schema, particularly in how tables interact with one another. I also assisted in creating an AWS account to host the website and played a key role in defining primary keys, selecting unique attributes, and determining appropriate foreign keys for the database tables.

In Part B, I supported the design of the overall database architecture and provided detailed descriptions for each component and page. Also, I helped refine our relational diagram by figuring out what pages/components we should try to accomplish by the end of Part C.

In Part C, I was responsible for initializing the project setup, which included creating the GitHub repository, selecting the Next.js framework, and configuring the project template. Making wireframes using Figma and designing how the pages will look and function depending on user input. Additionally, I developed the front-end components for the homepage, sign-up, and sign-in pages, and integrated them with the backend using https methods. Finally, I helped with the manual exploration and ad hoc testing for QA.

6. 3 Gabriel Herrera

First, I worked on the design of the database in the first part of the project in the Figma created for the design. This included the management of the attributes, keys, and database interaction. Then, in part B, I helped with the design of the architecture by writing the document and reviewing the grammar and format. Lastly, on part C, I work on the front-end part of the Jobs and My Jobs page. These files include the showing of the jobs for the public user to see examples of jobs and a selected user seeing the jobs they have completed or are in the process of completing. These pages also include a visual filter section for the user to use and navigation to edit jobs and create jobs pages.

6. 4 Edwin Lozano

I played a key role in developing the front-end interface for job creation and editing functionalities. This involved designing intuitive templates to effectively capture and display job-related information while ensuring a seamless user experience. I contributed to back-end integration, enabling robust database operations to accurately store and update user-submitted data for both new job postings and existing job modifications.

Additionally, I implemented profile editing capabilities, allowing users to update personal information while ensuring real-time system reflection of these changes. My work enhanced data flow coordination between front-end and back-end systems, significantly improving overall reliability and user satisfaction.

I also contributed to the design and implementation of critical database tables, including those supporting job creation workflows and user management systems. For Part B of the project, I helped architect the structural logic for 'create' and 'edit' table operations, ensuring consistent data handling across the application.

6. 5 Miguel Torres

For Part A, my contribution was coming up with ideas for some of the column names for the tables we wanted to use for our database design and E-R design. We collectively designed and came up with a plan of how the tables will relate to each other on a figmajam board. In our database design report, I had made some of the descriptions for the columns, their data types and use case.

For Part B, my contribution to this part included writing some of the pseudo-like code for the System Architecture Design report. This included writing inputs and outputs for each system and subsystem. The descriptions we made for these systems would later come into Part C of the project.

Finally for Part C, my contribution to this part is that I had worked on programming much of the backend/web-server and our database implementation using the better-sqlite3

javascript library. This database includes various tables used in our project under the "db/database.js" file. I had based the column names and keys for all these tables with our database design we agreed to in Part A and Part B project submissions. Another thing I had worked on is implementing the API routes to be able to interact with this database. The folders of "/api" and "/auth" include most of the code I wrote to build out these API's. These endpoints are fetched by our frontend to retrieve data or create new data from our database.

7. Conclusion

Building the Fixer-Uppers website showed us how important a solid database design is when creating a reliable and user-friendly system for solving real-world problems. We started by understanding user needs, then created an E-R model that helped guide the development of a working website connecting homeowners with trusted service professionals. The database was designed with clear tables, logical relationships, and rules that help keep data accurate and secure. We followed best practices like reducing repetition and planning for future growth, which helps the platform run smoothly and stay reliable as it expands.

Fixer-Uppers makes it easy for users to post and accept jobs, and it also builds trust through its review system. The user interface is designed to be clean and easy to navigate, which improves the experience for both homeowners and contractors. In the end, this project showed how strong database design and how thoughtful planning can lead to better user interactions and a more effective system overall.

8. References

1. Tor, Y., & Chen, Z. (2003, June 16). *ANTES system requirements specification developer's guide*. ANTES Technical Report Series. Computing and Information Technology Center (CITeC), and Department of Computer Science, University of Texas–Pan American.