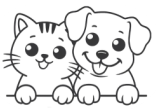


FUR-EVER FOUND

Project Report

1. Introduction

Our project, Fur-Ever Found, aimed to create a React-based web application designed to assist in reuniting lost pets with their owners. The project showcases the skills we acquired during our CFG course, including front-end development with React, state management, and API integration.

Aim

The primary aim of our project was to develop a user-friendly platform that facilitates the process of reporting and finding lost pets. We focused on creating a minimum viable product (MVP) to test the need for our concept and build a following first, thereby increasing the likelihood of successful reunions between pets and owners.

Objectives

1. Designing an Intuitive User Interface and Experience - Ensuring ease of use for all demographics who would own a pet.
2. Implementing Robust Functionality - Allowing users to post lost pet reports through submitting a simple form, including the ability to upload images and provide detailed descriptions.
3. Integrating a Filter Feature - Enabling users to look for reported lost pets based on various criteria such as location and type of animal through a filter.
4. Providing Support and Reducing Stress - Being a friendly and approachable team that helps alleviate the burden during the traumatic time of losing a pet. We aimed to make the process easy and act as the contact point between someone finding a pet and the owner to eliminate security/privacy concerns and time-wasting.

2. Background

Initially, we evaluated competitors such as Pets Reunited, finding them challenging to navigate and often requiring high payments from vulnerable users. Our goal for the MVP was a user-friendly experience, considering the trauma of losing a pet. We developed personas to grasp the problem and craft an appropriate solution. An example of the persona is detailed below:

Persona: Edith

Name: Edith Johnson | Age: 50 | Occupation: Librarian | Location: Bristol, UK
Pets: Multiple cats, each with unique personalities | Tech-Savvy Level: Basic

Background: Edith is a librarian living in a cosy home in Bristol. She has a deep love for cats and currently cares for several, each bringing their own charm and mischief to her life. Edith prefers a quieter, more private life and avoids social media due to concerns about sharing personal information online. Her experience with technology is limited, primarily using it for email and simple web browsing.

Goals:

- To find a lost cat quickly and safely.
- To use a platform that respects her privacy and does not require creating a public profile.
- To have a straightforward way to post about her lost pet without dealing with the complexities of social media.
- To be guided through the process.

Frustrations:

- Existing platforms often require too much personal information, which Edith is uncomfortable sharing.
- She finds social media frustrating and dislikes the exposure that comes with it.
- She struggles with complex interfaces and needs a simple, easy-to-use platform.

Combining competitors and persona insights, we defined our MVP focus to advertise lost pets, including a homepage, lost pets page, pet details, submission form, and about us page.

3. Specification and Design

When discussing our ideas about what the platform was aiming to deliver, we agreed that we wanted to make a user-friendly, accessible design with an easy and intuitive flow. The platform has two main functionalities: to allow users to post lost pets and to also allow them to browse the database of lost pets, in case they have found a pet and are looking to contact the owner. Highlighting these functionalities, we agreed on the following requirements:

Technical Requirements

- To choose a version control system that allows strong teamwork and collaboration throughout the project, leading to efficient problem-solving and a cohesive final product.
- Emphasis on improved user experience (UX) through iterative design changes.
- Using React, JavaScript, HTML, and CSS.
- The use of reusable components to allow for less coding and ensure consistent page design.
- Using a system to effectively manage states by implementing Redux.
- A simple system for data management to allow the team focus on front-end functionality.
- Technical choices (e.g. button design) that make the platform more accessible
- Utilising APIs available to enhance the UX.
- Unit testing our components.

The “Implementation and Execution” section describes how we successfully met and implemented our technical requirements.

Non-Technical Requirements

Usability

- User-friendly interface with intuitive navigation with consistency across all pages. We wanted to avoid a design where the user needs to take more than one step to access one of these functions. This is why the user can access both the posting form and the page with lost pets in two different ways: from the navigation bar, and from the step by step diagram on the homepage.
- Clear instructions and prompts for users to guide them through the processes by adding a step-by-step visual guide on how to use the website.
- Consistency with every element of the main page, from the header image and logo to the mission statement and overall design, reinforces the platform's purpose. This cohesive presentation effectively communicates what Fur-Ever Found does, providing users with instant clarity and assurance.

User Engagement

- Clear call-to-action buttons (e.g., Post Lost Pet, Browse Lost Pets).
- Informative and engaging content about the team.

Legal and Compliance

- We made space for our privacy policy and terms of service whilst they are being drawn up. We made sure they are always available in the footer and included them on the submission form.

Support and Communication

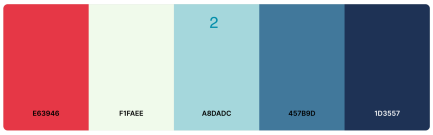
- Regular updates and communication channels are always visible (e.g., newsletter and social media).
- All emails and contact us functionality made seamless by automatically creating an email, with the subject line pre-populated where appropriate.

Design and Architecture

Decision-making process

To ensure everyone's agreement on key decisions, we used Slack polls for finalising our idea, colour schemes, brand name, and logo design. Everyone had the chance to process the information, suggest, and vote.

A blue base was chosen for our colour palette to create a calm feeling, while red was used for buttons and alerts to contrast and highlight their function, enhancing the website's accessibility.



Wireframes

Given we outlined the requirements and the concept, we created wireframes using Figma. The wireframes helped us to visualise and made the coding easier as we were aiming towards the design.

Our Website

The final result of our website and a breakdown of its pages are below:

Screen 1: Main Page

The Fur-Ever Found home screen features a navigation bar, site name, purpose, and an image. Users can access 'Lost Pets', 'About Us', and 'Post Now' pages from the navigation bar. Scrolling down reveals an explanation of the site's function, buttons to 'Post Lost Pet Screen' (Screen 4) and 'Browse Pet Screen' (Screen 2) , and a footer with the logo, useful links, and a newsletter subscription form.

Screen 2: Lost Pets

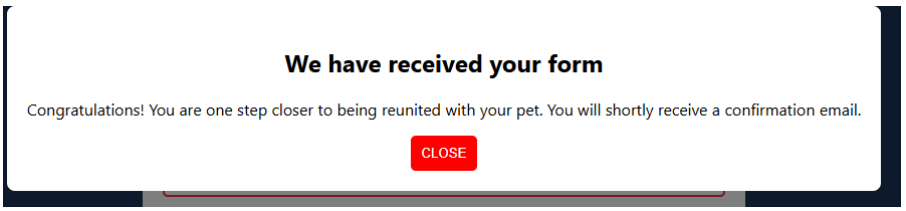
This is the pet information screen, accessible from the main screen's navigation bar. It displays the lost pet's name, city, and features an interactive card for each pet. Clicking on a card redirects to the pet's detailed page.

Screen 3: Detailed Pet Screen

This section provides details about each lost pet. Users access this page by clicking a pet image and can email the Fur-Ever Found Team if they've seen the pet. This email prepopulates the subject line. They can return to the main pet screen via the "Return to all pets" button or use the navigation bar for other screens.

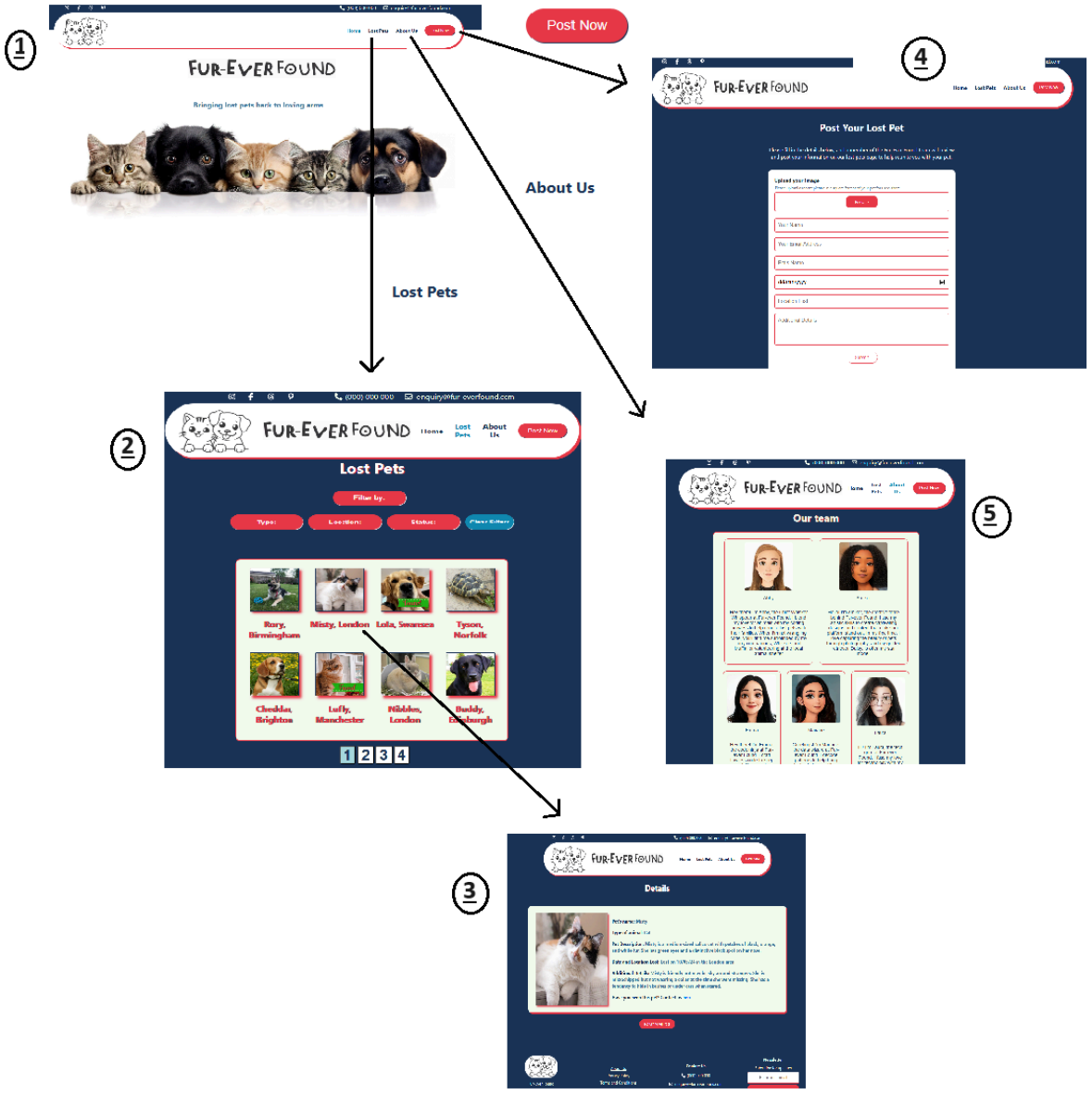
Screen 4: Post Lost Pet

From the navigation bar, users can access a form to report lost pets. They can enter pet details, add a photo, and use Google Maps to specify the last known location. After the form is submitted, a confirmation pop-up appears.



Screen 5: About Us

This page showcases the Fur-Ever Found team, aiming to establish rapport with owners in search of their lost pets. It reassures them by introducing the dedicated team behind the pet recovery efforts.



Comparison Between the Wireframes and the Final Product

The final product aligns with our initial wireframes with some adjustments. We removed the login and registration pages due to time constraints. We added diagrams to the homepage for accessibility and our contact details in the footer. Owners' personal details were removed for safety and all correspondence is handled by the Fur-Ever Found Team. A pop-up window was added to confirm form submissions.

4. Implementation and Execution

Development approach and team member roles

Our development approach followed Agile methodologies, with Amiker serving as the Scrum Master.

In this project, all team members acted as designers, developers, and testers. This holistic approach ensured that each team member had a comprehensive understanding of the entire project lifecycle and promoted a shared responsibility for the project's success. Some of the agile elements we used in our project are:

- **Iterative Approach:** The project was developed in iterative cycles, with regular reviews and refinements.
- **Refactoring:** The code was continuously refactored to improve performance and maintainability.
- **Code Reviews:** Pull requests on GitHub were reviewed by at least two team members before being merged into the weekly branch.
- **Daily Stand-ups:** Daily meetings helped identify and resolve blockers, ensuring smooth progress and collaboration.

For our project design and ideas to be executed, we made use of specific tools and libraries including:

Libraries and Frameworks: React, Google Places API, Redux, Redux Thunk, Jest, FontAwesome.

Collaboration and Communication Tools: Jira for project management, Zoom for meetings, Slack for instant communication, Google Drive for document sharing, GitHub for version control and code management.

Implementation Process

Based on the technical requirements explained in the previous section, we implemented the project using:

React Web Application

- We chose React for its component-based architecture, allowing for reusable and maintainable code. React's virtual DOM improves performance, which is crucial for a responsive user experience.
- HTML & CSS were used for structuring and styling the application. CSS modules were implemented for scoped and maintainable styles.

State Management

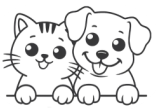
- We used redux for managing the global state of the application, ensuring a predictable state container. This was particularly useful for managing the state of the pet data, user inputs, and application status.
- Simplified the setup and logic for Redux by using Redux Toolkit, providing us with a better developer experience.

API Integration

- We integrated Google Maps API for location-based services, such as autocomplete for the location input when submitting a lost pet. A custom service was built to handle the loading and initialisation of the Google Maps API to ensure best practices.

Data Management

- Initially, pet objects are stored in a local array for simplicity during the development phase. This allows us to focus on the frontend functionality without the complexity of setting up a backend for the first phase.



FUR-EVER FOUND

User Interface

- Clear and consistent navigation bar across all pages.
- Easy to use form for posting lost pets with image upload and location mapping.
- Filter functionality for browsing lost pets.
- Consistent branding throughout – including both the navigation bar and footer.

Accessibility

- Alt text for all images and diagrams.
- Contrasting colour scheme.

Scalability

- Using components and having modular architecture to help future enhancements.

Security

- Users do not need to create profiles to post lost pets, ensuring their personal details remain confidential. The Fur-Ever Found team approves all posts to ensure they are safe and accurate before being made public.
- Adding validation to email for the newsletter and the submission form (no empty fields and image shape).

Maintaining and deployment

- Github strategy to ensure feature branches were used on weekly branches before merging into main to ensure adequate version control.

Testing

- Unit testing for key components

Phase two will add more for performance to be as fast as possible, additional security and responsive design.

During the implementation process, we decided to add extra features to the platform that we had not envisioned before such as:

- Design adjustments were made to improve the overall user experience based on UI/UX lesson
- Filtering on the lost pets page was implemented to enhance usability.
- A dedicated section on the home page provides a visual guide on how the website works. This section includes step-by-step instructions and visual aids, ensuring users can easily understand and navigate the website's functionalities.

During the implementation of the project we experienced the challenges below:

- Addressing merge conflicts, which were resolved during daily stand-up meetings.
- Adapting to design changes to enhance UX and accessibility
- The size and shape of images uploaded by users were causing layout issues on the lost pet page. To address this, restrictions were implemented, requiring uploaded images to be square-shaped to maintain the integrity of the card design.

5. Testing and Evaluation

Testing Strategy

We implemented 52 unit tests across 14 test suites. We took an analytical approach, creating tests based on the components' requirements. We focused on tests which reflected how the components would be interacted with by users in the application, e.g. that functions were fired as expected. We considered snapshot testing for our pages but felt this would not be as useful in the initial development phase where significant daily changes were being made to the UI. These could be useful to implement in the future to protect against unexpected changes.

Functional and User Testing

We used a component-driven approach with unit testing for each component, using React Testing Library and Jest. We used redux-mock-store for redux-based components due to its compatibility with React and ease of use. This allowed us to test the submission form in isolation, ensuring the redux logic did not impact our tests.

Tests required mocking to handle errors, notably where useNavigate was called inside a component. Jest's requireActual method was used for this. Mocking was also used to ensure function props, like onClick, fired as expected without testing specific function results.

We conducted thorough regression testing after every change to prevent integration issues. While working on complex pages like the Lost Pets page, we executed tests such as resetting filters and navigating pages before raising a merge request. Our review process included local testing of the feature branch's code. This rigorous process, though time-consuming, ensured high-quality code and instilled a sense of shared responsibility and pride.

System Limitations

Mocking of features: Due to time and practical constraints, some parts of our application are mocked. While a real-world application would use a backend database, we prioritised creating our React application and used a local array to store the pets. This means the 'Post a pet' form data is not sent to a database. We store the information in local storage to demonstrate what data would be sent (and validated server-side) in future. For our project, this decision meant we could amend the local array and change the pet object structure easily. As a result, we now have a good idea of the schema that our database would require.

Dependencies on external services: Using the external Google Maps API means that a problem with the API beyond our control would result in it not working inside our application. It also adds a slight layer of complication when setting up the project as users are required to add their own API key and additional security measures will need to be used when going live and using a Fur-Ever Found API key. However, we felt the benefits were worth the limitations. It improved user experience (users can select an address from a dropdown) and added an extra layer of validation by helping prevent mistakes with filling in the form. Going forward we would need to assess the use of the Google Maps API vs. the user experience by gathering data from the error logs to understand the impact of integrating this. This data would allow us to assess whether we could accept a user input when the API is unable to load, especially as we have an approvals process in place.

Exception Handling

We've implemented exception handling and field validations in our application to enhance user experience and reduce errors. For instance, all fields on the pet submission form are mandatory, ensuring we collect the necessary information for pet reunification. The image upload also undergoes validation to meet shape requirements, minimising post-approval delays. Finally, within the Google Maps API, when Autocomplete is initialised, actions are dispatched to handle errors.

6. Conclusion

Through this project, we were able to create an MVP for Fur-Ever Found which is a functional, user-friendly web app that allows users to report lost pets, search for them using select filters, and increase the chances of reuniting them with their owners.

Throughout the project, we encountered several challenges, including what features we wanted to include in the MVP, time and learning a new skill. These challenges provided valuable learning experiences and opportunities to apply problem-solving skills in a practical context.

Looking forward, there are several enhancements that could be made to the application. These include adding a fully functional database, adding a place to post pets that have been found and they are looking for the owner, adding login pages for both the Fur-Ever Found team and the users to add ease of approving submissions and integrating real-time notifications, improving search algorithms with AI by comparing the images of the lost to the found, and integrating the use of social media to increase the likeability of reuniting pets with owners. Additionally, further user testing and feedback could help refine the application to better meet the needs of its users.

In conclusion, this project was a significant step in our learning journey, allowing us to apply theoretical knowledge in a real-world scenario. It demonstrated the importance of user-centred design, the integration of front-end technologies, and the need for continuous improvement in software development.