BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Purrfect Match: A Web-Based Platform for Promoting Feline Adoption

Supervisor
Lect. dr. Surdu Sabina

Author
Hognogi Ana-Maria Cristina

2024

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ÎN LIMBA ENGLEZĂ

LUCRARE DE LICENŢĂ

Potrivire Perrfectă: O Platformă Web pentru Promovarea Adopției Feline

Conducător științific
Lect. dr. Surdu Sabina

Absolvent
Hognogi Ana-Maria Cristina

2024

ABSTRACT

This thesis addresses the increasing number of homeless cats worldwide and the challenges of cross-border cat adoptions. It presents the development of PurrfectMatch, a web-based platform designed to streamline the adoption process and connect individuals across different countries, thereby promoting feline welfare. The platform aims to fill the gap created by the lack of efficient, user-friendly global adoption platforms, addressing the significant issue of feline homelessness by offering a seamless, accessible solution.

The original contribution of this thesis is the creation of PurrfectMatch, which integrates Human-Computer Interaction (HCI) principles with a commitment to animal welfare. Unlike existing pet adoption websites, PurrfectMatch features global reach and a communication module that enables direct interactions among users and pet guardians. The platform provides detailed cat profiles, ensuring adopters are well-informed, and facilitates meaningful connections leading to responsible adoptions rather than operating on an e-commerce model.

The thesis is organized into several chapters. **Chapter 1** introduces the research, highlighting the need for innovative feline adoption solutions. **Chapter 2** provides a theoretical overview, reviewing literature in HCI and web development. **Chapter 3** details the system architecture, including backend development, data management, and frontend design. **Chapter 4** covers the design and implementation phases, showcasing PurrfectMatch's unique features. **Chapter 5** concludes the thesis by reviewing the findings and proposing future development directions.

Hognogi Ana-Maria Cristina

Contents

Chapter 1

Introduction

1.1 Motivation and Objectives

The motivation behind this thesis arises from the increasing number of homeless cats worldwide and the challenges associated with cat adoptions, especially in a cross-border context. The thesis aims to address these issues by developing PurrfectMatch, a web-based platform designed to streamline the adoption process, connecting individuals across different countries and promoting cross-border adoptions about feline welfare. Feline homelessness is a global concern, with millions of cats living without a permanent home or the care they require. This situation is exacerbated by the lack of efficient, user-friendly platforms that can facilitate the adoption process on a global scale. PurrfectMatch aims to fill this gap by providing a seamless, efficient, and accessible platform that connects homeless cats with potential adopters worldwide.

1.2 Original Contribution

The original contribution of this thesis lies in the creation of PurrfectMatch, a web-based platform that integrates Human-Computer Interaction (HCI) principles with a deep commitment to animal welfare. Unlike existing websites or social media pages that facilitate pet adoptions, PurrfectMatch stands out with its global reach and a communication module that involves direct and personal interaction among users and pet guardians. Moreover, the platform is designed to offer a detailed insight into each cat's profile, including health status and additional information, ensuring that adopters are fully informed for the responsibility of pet ownership.

Distinguishing itself further from typical pet adoption websites, PurrfectMatch does not operate on an e-commerce model. It is not about 'ordering' a pet; instead, it's about facilitating meaningful connections that lead to responsible adoptions.

Users can send adoption requests, and guardians have the ability to choose a suitable home for their cat based on comprehensive user profiles. Additionally, the platform provides an extensive guide on cat care, customized for various breeds, ensuring adopters have all the information needed to care for their new pets. Furthermore, PurrfectMatch is designed to be fully accessible, allowing users to navigate the platform with ease. This ensures a smooth and intuitive user experience, making the platform inclusive and user-friendly.

1.3 Structure

The thesis is meticulously organized into various chapters, each designed to carefully analyze into the creation and execution of Purrfect Match. The initial chapter, the Introduction, lays the foundation for the study, elucidating the underlying motivation and the definitive objectives behind the research. This chapter sets the stage, offering a comprehensive background that outlines the pressing need for innovative solutions in the realm of feline adoption and how PurrfectMatch endeavors to meet this demand through a unique blend of technology and animal welfare principles.

The second chapter is the Theoretical Overview, which explores the conceptual foundations of PurrfectMatch. It reviews relevant literature in HCI and web development, highlighting key principles and methodologies that inform the design and implementation of user-friendly web applications. The chapter also delves into the importance of real-time communication technologies, such as WebSockets, in enhancing user interactions and overall experience on the platform.

The third chapter System Architecture offers an in-depth exploration of the technical architecture of PurrfectMatch. It elucidates the application's backend development using Node.js and Express, data management with SQLite and Sequelize for ORM, and frontend design using React with a RESTful API. This chapter aims to detail how these technologies converge to create a seamless, user-friendly platform that addresses the needs of cats, adopters, and shelters globally.

The fourth chapter is the Application, which showcases the unique features and functionalities that distinguish PurrfectMatch from other platforms. It highlights the application's user-centric design, its global approach to cat adoption, and the innovative communication module that facilitates interactions among all users. This chapter also illustrates how the application streamlines the adoption process, ensuring users are well-informed and engaged throughout their journey to responsible pet ownership.

In the fifth chapter, the thesis will conclude by summarizing the key contributions of PurrfectMatch. The chapter will highlight the platform's impact on animal welfare and user engagement. It will also identify areas for further improvements.

Chapter 2

Theoretical Overview

2.1 Human-Computer Interaction

2.1.1 Introduction to HCI

STANFORD provides the following definition of Human-Computer Interaction: [Uni16]

"Human-Computer Interaction (HCI) is an interdisciplinary field focusing on the design of computer technology and the interaction between humans (the users) and computers. While initially concerned with computers, HCI has since expanded to cover almost all forms of information technology design."

Human-Computer Interaction (HCI) is described as a multidisciplinary field that synthesizes knowledge from computer science, psychology, and design to understand and enhance how people interact with technology. (Figure 2.1) This field aims to improve user experiences by designing systems that are intuitive and effective for human use. HCI covers a broad spectrum, from theoretical foundations to practical applications across various technologies. The primary goal of HCI is to create systems that are not only functional but also user-friendly, ensuring that the interaction between humans and computers is as seamless as possible.

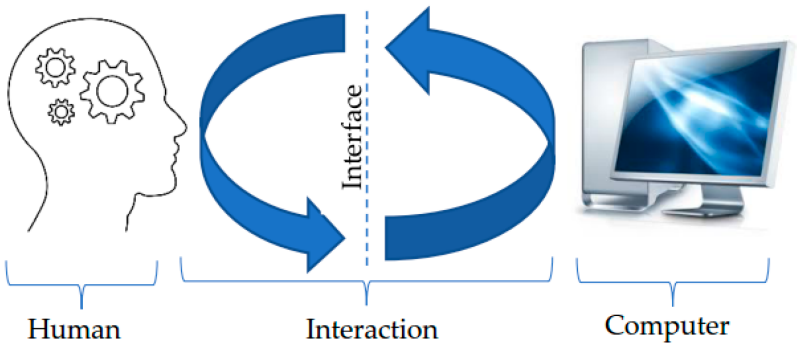

Figure 2.1: High-level diagram of a typical human–computer interaction [Kat21]

2.1.2 Historical Context and Evolution of HCI

The evolution of Human-Computer Interaction (HCI) dates back to the early days of computing, characterized by command-line based interactions. Early HCI research focused on making these interactions more efficient and error-free. [DFAB03] As technology progressed, the need for more user-friendly interfaces became apparent, leading to the development of graphical user interfaces (GUIs) in the 1980s. This era marked a significant shift in HCI, emphasizing the importance of usability and the user's cognitive processes in system design. [DFAB03]

In the 1990s, the rise of the internet brought new challenges and opportunities for HCI. Web usability became a critical focus area, and researchers began to explore how people navigate and use websites. This period also saw the emergence of mobile computing, which introduced new interaction paradigms and further expanded the scope of HCI research. The 21st century has continued this trend with the advent of ubiquitous computing, virtual reality, and artificial intelligence, all of which present new challenges and possibilities for HCI. [DFAB03]

2.1.3 Core Principles and Theories of HCI

2.1.3.1 Usability

One of the core principles of HCI is usability, which refers to how easily users can learn to operate the system, prepare inputs, and interpret its outputs. (Figure 2.2) Usability is often evaluated based on effectiveness, efficiency, and satisfaction. Effective usability ensures that users can complete their tasks accurately. Efficiency refers to the speed and resource consumption with which tasks are completed. Satisfaction measures the user's comfort and positive feelings about using the system. [DFAB03]

Figure 2.2: Components of Human-Computer Interaction [Ram23]

A key aspect of usability is the principle that technology should be designed with the user in mind, ensuring that it meets their needs and preferences. Key aspects of usability include learnability, efficiency, memorability, error prevention, and satisfaction.

- **Learnability:** Learnability in usability refers to how easily new users can start interacting with a system and successfully accomplish their goals. Jakob Nielsen explains that learnability involves the user's capability to reach a specified level of proficiency with the system after a certain amount of training. [Nie93]. This element is crucial because it affects how quickly users can become effective when using new technology.

- **Efficiency:** Efficiency evaluates the speed at which users can complete tasks once they are accustomed to the user interface. Nielsen defines efficiency as the degree to which users can perform tasks using a system after they have become proficient with it. [Nie93]. Efficiency is vital as it directly impacts productivity and user satisfaction, particularly in repetitive or task-oriented environments.

- **Memorability:** Memorability deals with the user's ability to return to the system after a period of not using it and still perform tasks without needing to relearn everything. Nielsen explains that a user interface is considered memorable when users can recall enough information to use it effectively during future interactions. [Nie93]. Good memorability minimizes the user's learning curve and supports casual or intermittent use without frustration.

- **Error Prevention:** Error prevention in HCI focuses on reducing the chance that users will make errors during their interaction with a system and on helping them recover from errors that do occur. Norman suggests that effective design can preventively address issues, preventing them from arising initially. [Nor13]. Emphasizing error prevention enhances the usability of a system by making it more reliable and user-friendly.

- **Satisfaction:** Satisfaction refers to how pleasant an interface is to use. Norman explains that satisfaction encompasses the positive emotions of pleasure and achievement users feel when their needs and desires are met. [Nor13]. Satisfaction is a subjective measure but is crucial for ensuring user loyalty and continued use of a system.

By prioritizing these aspects, designers can create technologies that not only meet functional requirements but also provide an intuitive and enjoyable user experience. This approach leads to higher user satisfaction, improved productivity, and loyalty towards the technology.

2.1.3.2 User Experience (UX)

User Experience (UX) extends beyond usability to encompass all aspects of the user's interaction with the system, including emotional responses, perceptions, and overall satisfaction. (Figure 2.3) [DFAB03] UX design aims to create systems that are not only functional but also enjoyable to use. Key factors influencing UX include the system's aesthetics, the user's previous experiences, and the context in which the interaction occurs.

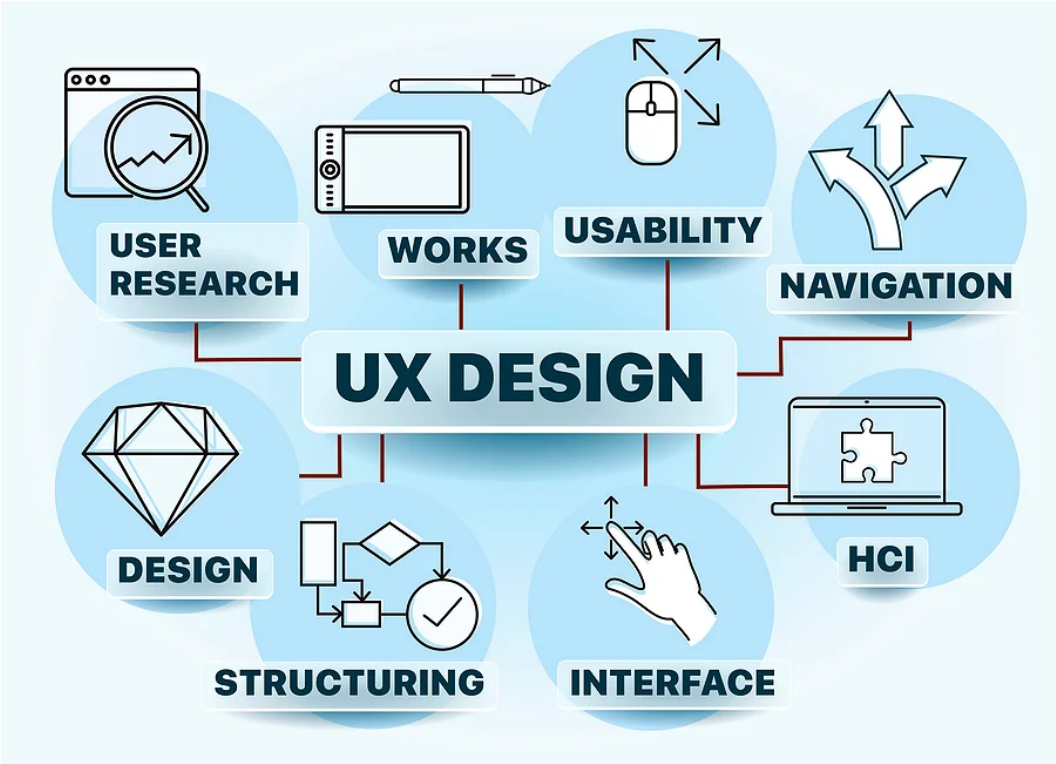

Figure 2.3: User Experience Design [Kee22]

Additionally, UX is deeply connected to the concept of user engagement, which measures the level of a user's involvement and interest in the system. Engaged users are more likely to have positive experiences and exhibit higher levels of satisfaction. The effectiveness of UX design can be evaluated through various methods, including user testing, surveys, and analytics, which provide insights into user behavior and preferences. [Bec20]

Furthermore, emotional design plays a crucial role in UX by creating interfaces that evoke positive emotions and foster a connection between the user and the system. This approach often involves using visual design elements, such as color schemes and typography, as well as interactive elements that are both functional and delightful. [DFAB03] Incorporating accessibility considerations into UX design ensures that the system is usable by people with diverse abilities, enhancing the overall inclusivity and reach of the technology. [Bie22]

2.1.3.3 Cognitive Load and Human Information Processing

Understanding how humans process information is crucial in HCI. Cognitive load refers to the amount of mental effort required to use a system. (Figure 2.4) Systems that require high cognitive load can lead to user frustration and errors. [DFAB03] HCI aims to minimize cognitive load by designing interfaces that align with the user's natural thought processes and by providing clear, concise information.

Human information processing models, such as the Model Human Processor, provide frameworks for predicting how users will interact with systems. These models help designers anticipate potential issues and optimize interfaces to support efficient and error-free interactions. [DFAB03]

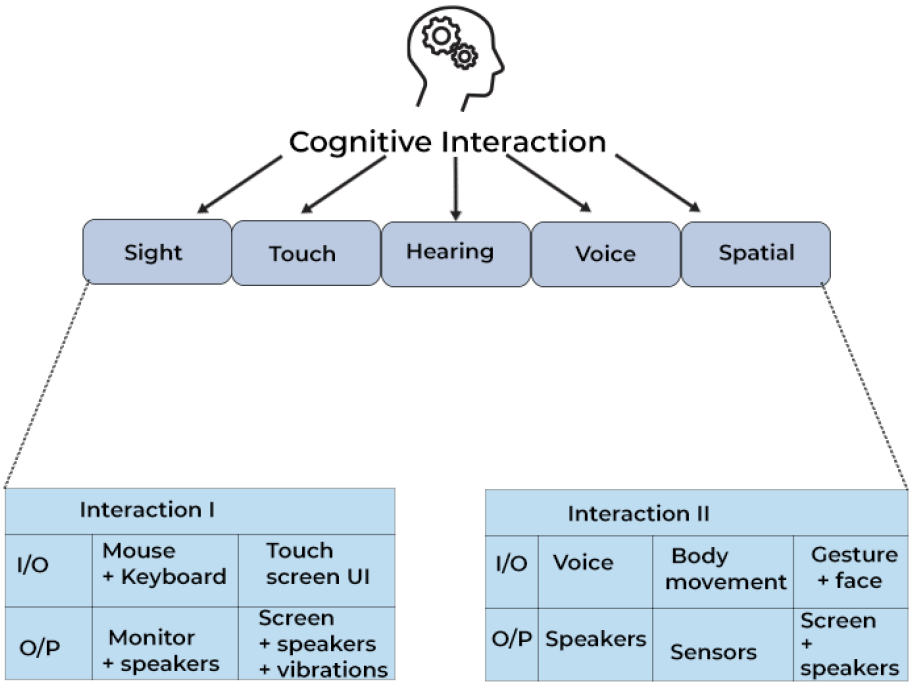

Figure 2.4: Diagram of Human-Computer Interaction [Kan22]

2.1.3.4 Affordances and Signifiers

Affordances refer to the properties of an object that indicate how it can be used. For example, a button affords pressing, and a handle affords pulling. In HCI, designers use affordances to create intuitive interfaces. Signifiers are visual indicators that highlight or clarify affordances, guiding users toward the intended interaction. [Bie22] Effective use of affordances and signifiers can significantly enhance usability and reduce the learning curve for new users.

2.1.4 Methodologies and Techniques in HCI Research

2.1.4.1 User-Centered Design (UCD)

User-Centered Design (UCD) is a design philosophy that emphasizes placing the user at the center of the development process. (Figure 2.5) UCD involves iterative cycles of user research, prototyping, and testing. By continuously involving users and incorporating their feedback, designers can create systems that truly meet their needs and expectations.[Bec20]

UCD typically follows a four-step process: understanding user needs, designing solutions, evaluating prototypes, and implementing the final design. Each step involves close collaboration with users, ensuring that their perspectives are integrated throughout the design process. [Bec20]

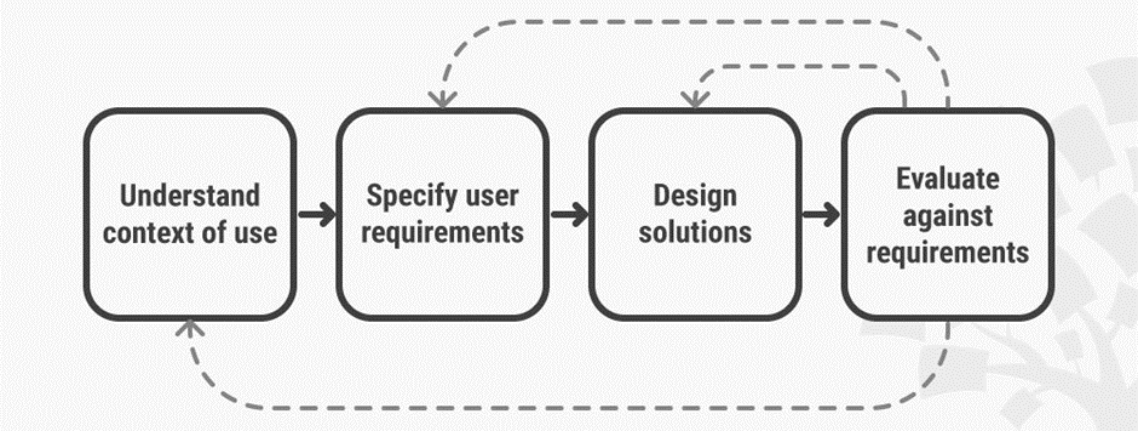

Figure 2.5: User-centered design - stages of design and development [Fou16]

2.1.5 HCI in Practice: Case Study of a Web Application

2.1.5.1 Introduction to the Case Study

This case study focuses on a web application designed for the adoption of cats, where users can place cats for adoption and adopt cats directly from other users without the involvement of shelters. The application aims to provide an intuitive and seamless user experience, ensuring that both those looking to adopt and those placing cats for adoption can easily navigate the platform and achieve their goals efficiently.

2.1.5.2 User-Centered Design Approach

The development of the web application followed a User-Centered Design (UCD) approach, ensuring that users' needs and preferences were prioritized throughout

the design process. The first step involved conducting user research to understand the motivations, behaviors, and pain points of potential users. This research included observational studies with individuals who had experience with cat adoption or had used similar online platforms. [Bec20]

Based on the insights gained from user research, personas were created to represent different types of users, such as cat owners looking to rehome their pets and individuals seeking to adopt a cat. These personas guided the design decisions, ensuring that the application addressed the specific needs and preferences of its target audience.

2.1.5.3 Key Features and Usability Enhancements

The key features that were implemented based on HCI principles to enhance the usability of the web application:

- **Intuitive Navigation**: A clean and straightforward navigation menu allows users to easily access different sections of the application, such as *"Feline Friends Catalog"*, *"Give a Cat for Adoption"*, *"My Profile"* and so on. Clear labels and a consistent layout ensure that users can find what they need without confusion. [DFAB03]

- **Advanced Search and Filters**: The search functionality includes filters for location, age, breed, and other characteristics, helping users quickly find cats that match their preferences. The search results are presented in a visually appealing grid with photos and brief descriptions, making it easy to browse available cats. [DFAB03]

- **User Profiles and Messaging**: Each user has a profile page where they can manage their listings, view their favorites, and communicate with other users. The messaging system is designed to facilitate smooth and secure communication between users, with notifications and conversation history easily accessible. [Bec20]

- **Security and Privacy Features**: Robust security measures are implemented to protect user data and ensure safe interactions. Features such as secure messaging, profile verification, and privacy settings help maintain a trustworthy environment. [Bec20]

- **Responsive Design**: The application is designed to be fully responsive, ensuring a seamless experience across different devices, including desktops, tablets, and phones. This responsiveness is crucial for users who may want to access the platform from anywhere. [DFAB03]

2.2 Web Development

2.2.1 Introduction to Web Development

Web development is a broad field that includes the creation and maintenance of websites and web applications. It involves a combination of client-side *(front-end)* and server-side *(back-end)* programming (Figure 2.6), as well as the design and implementation of user interfaces and experiences. Modern web development strives to create responsive, efficient, and user-friendly web applications that cater to the needs of both users and businesses.

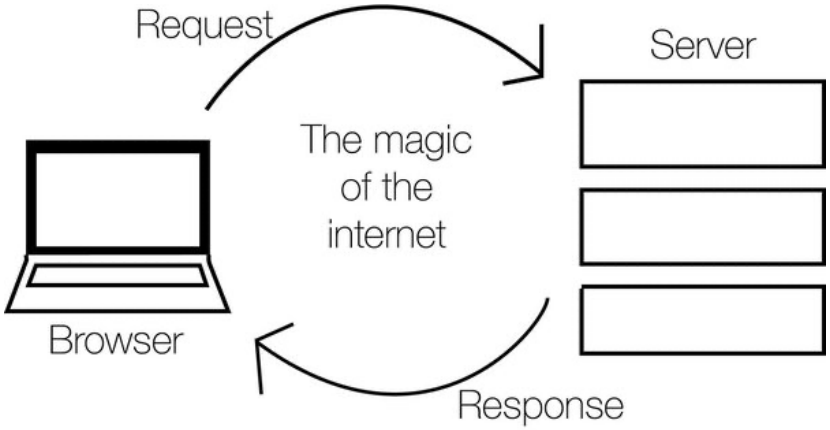

Figure 2.6: Communication between front-end and back-end

Web development integrates various technologies and practices, such as HTML, CSS, JavaScript, and server-side languages like Node.js, to build interactive and dynamic web applications. Understanding the theoretical underpinnings of these technologies is essential for creating robust and maintainable web solutions.

2.2.2 Core Technologies in Web Development

2.2.2.1 HTML and CSS

HTML (Hypertext Markup Language) is the fundamental language for constructing web pages. It outlines the structure of a web page by designating elements like headings, paragraphs, links, and images. HTML is essential for developing both the content and layout of a web page. [Rob07]

CSS (Cascading Style Sheets) is used to style and format HTML elements. CSS allows developers to apply styles such as colors, fonts, and spacing, as well as layout techniques like flexbox and grid, to create visually appealing and responsive designs. [McF09] By separating content (HTML) from presentation (CSS), developers can create cleaner and more maintainable codebases.

2.2.2.2 JavaScript and Front-End Frameworks

JavaScript is a flexible programming language that runs in the browser, allowing for the creation of dynamic and interactive web pages. It is used to manipulate the Document Object Model (DOM), handle events, and communicate with servers through AJAX requests. [Hav24] JavaScript is fundamental to modern web development, allowing developers to create rich user experiences.

Front-end frameworks and libraries like React, Angular, and Vue.js extend JavaScript to offer structured approaches for developing complex web applications. [Sto16] For instance, React focuses on component-based architecture and unidirectional data flow, simplifying state management and enabling the creation of reusable UI components.

2.2.2.3 Server-Side Development

Server-side development involves creating the backend logic and databases that power web applications. Technologies like Node.js, Express, and databases such as MongoDB, PostgreSQL, and SQLite are commonly used in this field. Node.js is a runtime environment that allows JavaScript to be utilized for server-side scripting, providing a unified language for both client and server code. [Bro19]

Express is a minimalist web framework for Node.js that simplifies the creation of web servers and APIs. It offers robust routing, middleware, and templating capabilities, facilitating the development of scalable and maintainable server-side applications. [Bro19]

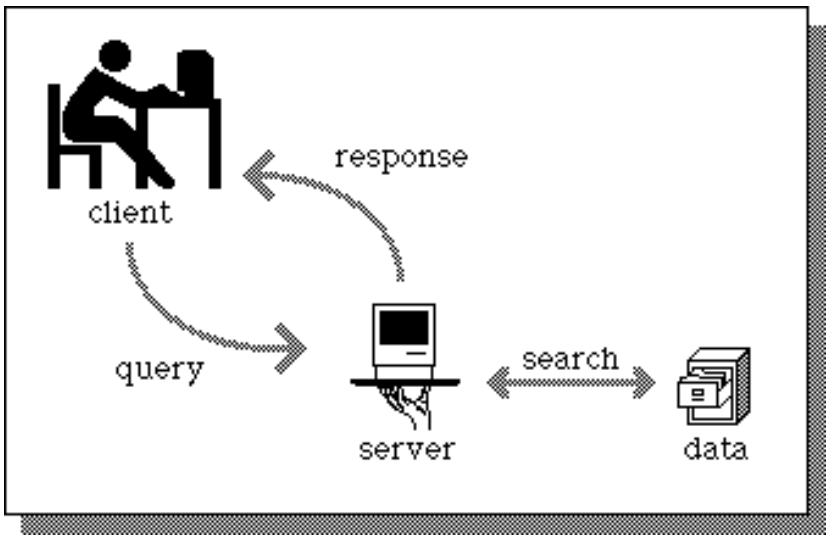

Figure 2.7: Front-end, Back-end and Database-side [Cza18]

2.2.2.4 WebSockets for Real-Time Communication

Amar Gupta defines WebSockets as *"a protocol that enables full-duplex communication channels over a single TCP connection. Unlike traditional HTTP requests, which are stateless and require a new connection for each request"* [Gup23] or response cycle, WebSockets allow for continuous, two-way communication between the client and server. [Sim15] (Figure 2.8)

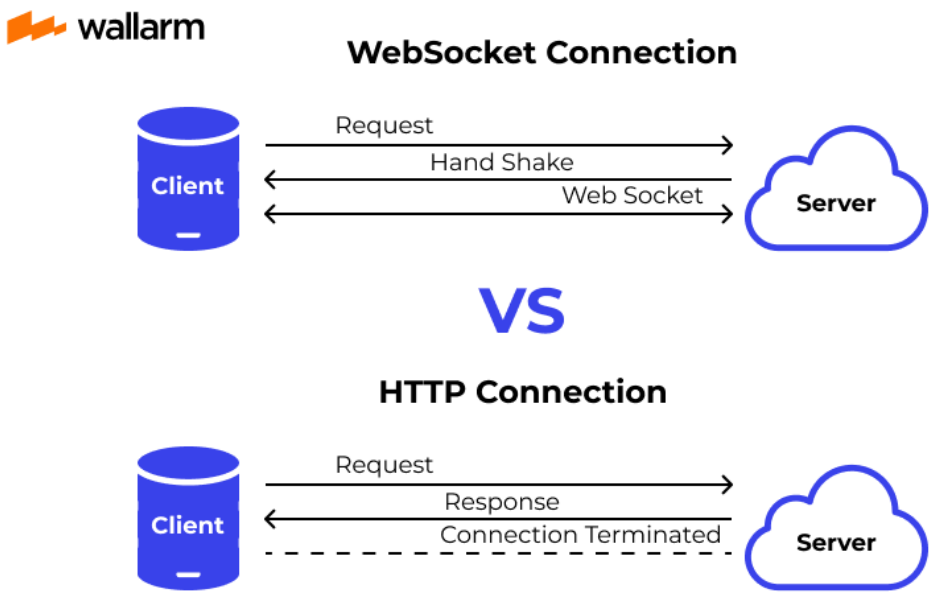

Figure 2.8: How do WebSockets work? [Bes24]

This technology is particularly useful for real-time applications such as chat systems, live notifications, and online gaming. WebSockets reduce latency and improve the responsiveness of web applications by maintaining an open connection and enabling instant data transfer.

2.2.3 Principles of Clean Code and Web Design

2.2.3.1 Writing Clean and Maintainable Code

Clean code is crucial for creating maintainable and scalable web applications. It involves writing code that is easy to read, understand, and modify. Key principles of clean code include meaningful variable names, modular functions, and adherence to coding standards. [Mar08]

One of the most important aspects of clean code is the Single Responsibility Principle (SRP), which states that each module or function should have only one reason to change. [Mar08] This principle helps in keeping the codebase organized and reducing the complexity of debugging and extending the application.

2.2.3.2 User-Centered Design and Usability

Effective web design goes beyond aesthetics to emphasize usability. User-centered design (UCD) prioritizes creating websites that meet users' needs and preferences. This approach involves understanding user behavior, conducting usability testing, and iteratively refining the design based on user feedback. [Kru13]

Key principles of UCD include simplicity, consistency, and intuitive navigation. Simplicity ensures that users can easily understand and interact with the website. [Kru13] Consistency helps users develop familiarity with the interface, reducing the learning curve. Intuitive navigation allows users to find information quickly and efficiently.

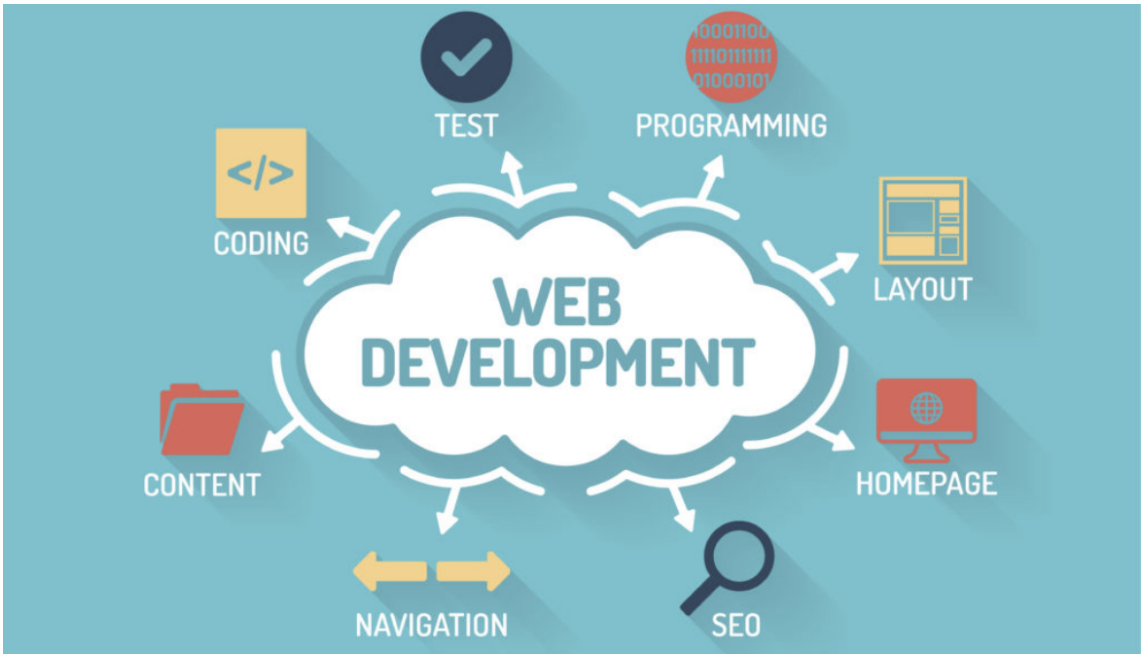

Figure 2.9: Building Dynamic Websites [aca23]

2.2.4 Advanced Techniques in Web Development

2.2.4.1 Single Page Applications (SPAs)

Single Page Applications (SPAs) load a single HTML page and dynamically update content, offering a fast, seamless user experience similar to native apps, using frameworks like React or Angular. Multi-Page Applications (MPAs), on the other hand, consist of multiple HTML pages that reload entirely upon navigation, providing better SEO and accessibility but often resulting in slower performance. SPAs excel in speed and user experience, while MPAs are simpler to secure and manage at scale. [MJ13]

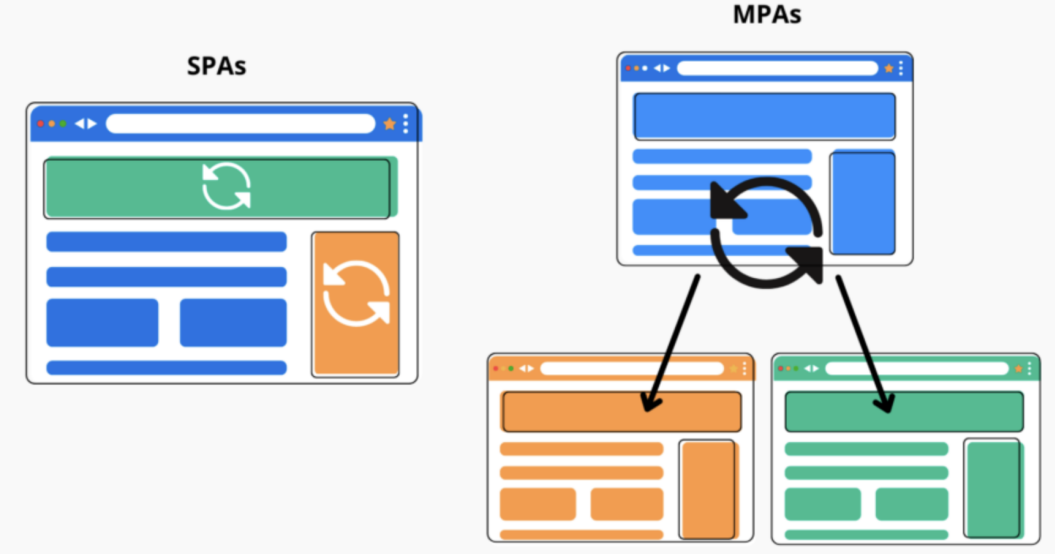

Figure 2.10: Single-page Applications vs Multiple-page Applications [Car23]

2.2.4.2 Responsive Web Design (RWD)

Responsive Web Design (RWD) is a design strategy that ensures web applications look and perform well on a variety of devices and screen sizes. RWD employs flexible grids, fluid layouts, and media queries to adapt the layout and content based on the screen size. [McF09]

By using RWD, developers can create a single web application that offers an optimal user experience on desktops, tablets, and mobile devices. This approach simplifies development and maintenance by ensuring consistency across all these devices.

2.2.4.3 WebSockets in Action: Real-Time Communication

WebSockets are a game-changer for real-time web applications. Traditional HTTP requests are inefficient for real-time communication due to the overhead of establishing a new connection for each request. WebSockets address this limitation by establishing a persistent connection between the client and server, allowing for instant data exchange. [Sim15]

For example, in a web-based chat application, WebSockets enable real-time messaging by maintaining an open connection. When a user sends a message, it is instantly transmitted to the server and broadcast to all connected clients without the need for repeated HTTP requests. This approach minimizes latency and provides a more responsive user experience.

2.2.5 Case Study: Web Application for Cat Adoption

2.2.5.1 Overview of the Application

This case study examines a web application designed for user-to-user cat adoption. The application allows users to place cats for adoption and connect with potential adopters without involving shelters. Key features include user registration, cat listings, search functionality, and messaging between users.

2.2.5.2 Front-End Implementation

The front-end of the application is developed using HTML, CSS, and JavaScript, in conjunction with the React framework. React's component-based architecture enables the creation of modular and reusable UI components, which streamlines the development process and simplifies code maintenance. [Sto16]

CSS is used to style the application. Media queries and flexible grids are employed to create a consistent user experience across devices. [McF09] Additionally, Framer Motion is used for animations and transitions, providing a smooth and visually appealing experience for users as they navigate through the application.

2.2.5.3 Back-End Implementation

The back-end is powered by Node.js and Express, providing a robust and scalable server-side infrastructure. This setup ensures high performance and flexibility, capable of handling a large number of concurrent connections efficiently. The application uses a RESTful API to manage CRUD (Create, Read, Update, Delete) operations for user profiles and cat listings, enabling seamless data management and integration with the front-end. [Bro19]

SQLite is used as the database to store user and cat data, offering flexibility and scalability for the application. SQLite's lightweight nature makes it an ideal choice for a wide range of devices and deployment scenarios, ensuring that the application can scale as needed without compromising on performance. [Bro19]

To enhance security, the application employs JWT (JSON Web Tokens) for user authentication, ensuring that user sessions are secure and that data privacy is maintained. This approach provides a stateless, scalable solution for managing user sessions across the platform.

2.2.5.4 Real-Time Messaging with WebSockets

A critical feature of the application is the real-time messaging system, which enables users to communicate instantly. This system is implemented using WebSockets, allowing for continuous two-way communication between clients and the server.

[Sim15] When a user sends a message, it is immediately relayed to the recipient without the need for repeated HTTP requests, providing a seamless and responsive chat experience. This ensures that conversations are as fluid and natural as possible, closely mimicking face-to-face interactions.

In addition to facilitating personal conversations, WebSockets are integral to the adoption process within the platform. They provide real-time updates and interactions between users, such as notifications about new adoption requests, instant status updates on the adoption process, and live interactions between potential adopters. This real-time capability ensures a smooth and efficient adoption experience for both parties involved, significantly enhancing user satisfaction and engagement. [Sim15]

WebSockets are also utilized for a chat box, which is another key feature of the platform. This chat box enables users to connect with others within the community to share advice, ask questions, and support each other. The real-time nature of WebSockets ensures that messages are delivered instantly, creating a dynamic and interactive environment.

By integrating WebSockets throughout its features, the platform provides a robust, real-time communication experience that enhances user interaction, streamlines processes, and fosters a strong, connected community. This technology not only improves the effectiveness of the chat box but also transforms the overall user experience, making it more engaging and efficient.

Chapter 3

System Architecture

3.1 Overview of the System Architecture

PurrfectMatch is designed as a full-stack web application, seamlessly integrating frontend and backend technologies to provide an optimal user experience. The platform utilizes a decoupled architecture, which ensures a clear separation of concerns and improves maintainability. This chapter will explore the specifics of the system architecture, including the backend and frontend technologies employed, the database management system, and other related technologies.

3.2 Integration and Workflow

The integration and workflow of a web application involve the seamless coordination of different technologies and layers within the software architecture to ensure optimal performance and user experience. (Figure 3.1) A well-designed decoupled architecture strategy is crucial for maintaining a clean and efficient codebase, which facilitates easier maintenance. The following sections outline the primary components and tools involved in a typical web application's integration and workflow.

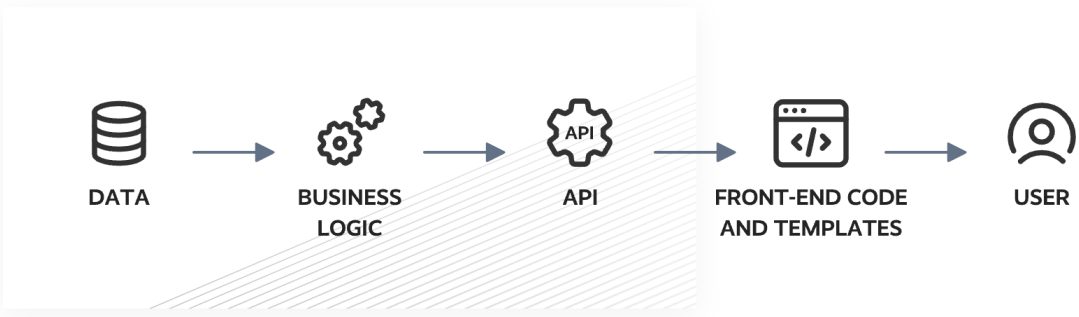

Figure 3.1: Decoupled web architecture [And]

3.2.1 Backend

The backend of a web application handles the server-side logic, database interactions, and business rules. One of the most common patterns for backend integration is the use of RESTful APIs, which allow different parts of the application to communicate over HTTP. [Kle17] RESTful APIs are a fundamental aspect of backend integration, providing a standard way to expose services and data to the frontend and other systems.

Backend integration often involves the use of middleware frameworks such as Express.js in a Node.js environment. Express.js simplifies the creation of server-side logic and routing, enabling developers to build scalable and maintainable web applications. [CM20a]

3.2.2 Frontend

The frontend of a web application is responsible for the client-side rendering and user interface interactions. Modern frontend development typically involves frameworks like React, Angular, or Vue.js. These frameworks enable the creation of dynamic and responsive user interfaces by utilizing component-based architectures. [BP20] React's component model allows for reusable and maintainable UI elements, enhancing the overall development workflow.

3.3 Backend Technologies

PurrfectMatch's backend is developed with Node.js and Express, offering robust solutions for managing HTTP requests. Node.js, with its event-driven, non-blocking I/O model, is particularly suited for creating scalable network applications. Meanwhile, Express serves as a lightweight and adaptable web application framework, streamlining the creation of powerful APIs and web services. [Mea18]

3.3.1 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, enabling developers to write server-side code using JavaScript and facilitating full-stack development with a single programming language. One of the key advantages of Node.js is its asynchronous nature, which supports high concurrency and efficient handling of multiple requests simultaneously. This makes Node.js particularly well-suited for real-time applications and APIs. [Mea18]

Node.js utilizes an event-driven architecture, where the server runs on a single-threaded event loop. This design allows it to handle concurrent operations effi-

ciently without blocking the execution of other tasks. This is achieved through the use of asynchronous callbacks and promises, allowing Node.js to manage thousands of connections with minimal overhead. This architecture is ideal for I/O-heavy operations, such as reading and writing to a database or interacting with external APIs.

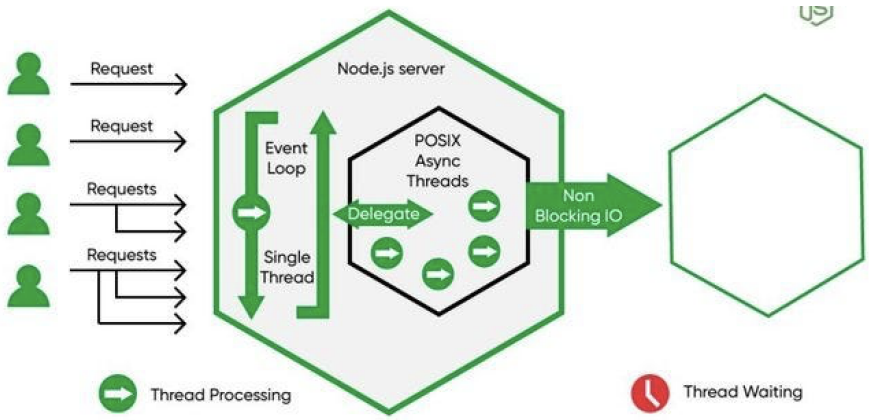

Figure 3.2: Diagram Flow of Node.js [Ade23]

3.3.2 Express

Express is a lightweight web framework for Node.js, providing an extensive array of features for developing both web and mobile applications. It streamlines the handling of HTTP requests, routing, and middleware integration, making it easier for developers to manage these processes. [Mea18] Express supports a variety of middleware functions to manipulate request and response objects, handle different HTTP methods, and set up a variety of routing configurations. This flexibility allows developers to quickly and efficiently create RESTful APIs, resulting in a powerful and adaptable backend infrastructure.

Express allows for the organization of application logic into middleware functions, which can be used to handle requests, perform authentication, validate data, and manage sessions. Middleware functions are executed sequentially, allowing for modular and reusable code. This makes it easier to manage the complexity of large applications and maintain a clean and organized codebase.

3.3.2.1 Key Features of Express

Routing: Express provides a powerful routing mechanism, allowing developers to define URL patterns and map them to specific request handlers. This makes it easy to create RESTful endpoints for CRUD operations. Routing in Express is simple yet powerful, enabling the definition of routes using HTTP methods to handle different types of requests. Route parameters and query strings can be easily parsed

and accessed, providing flexibility in handling dynamic URLs and complex request patterns.

Middleware: Middleware functions can be used to process requests before they reach the final route handler. Common middleware includes authentication checks, input validation, and logging. In Express, middleware functions form a sequence that interacts with the request object (*req*), the response object (*res*), and the next middleware function in the application's request-response cycle. [Mea18]

These functions enable processing of requests and responses, executing code, modifying request and response objects, and passing control to subsequent middleware functions in the chain. This allows for preprocessing tasks such as parsing request bodies, setting response headers, or performing security checks before the actual route handler executes.

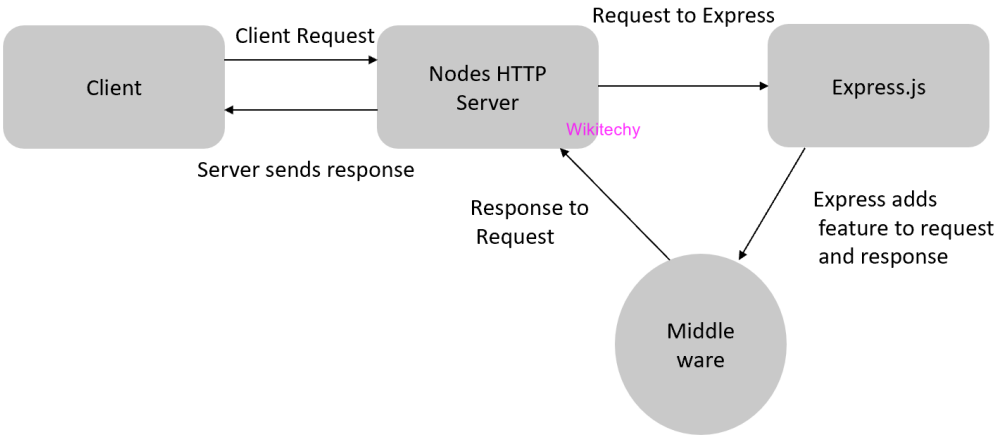

Figure 3.3: What is ExpressJs [Wik]

Template Engines: Express supports various template engines, such as Pug, for rendering dynamic HTML content on the server side. Template engines in Express allow developers to create dynamic web pages by embedding server-side code within HTML. These engines can process templates to produce HTML output, integrating data from the backend seamlessly into the user interface.

Static Files: Express can serve static files such as images, CSS, and JavaScript directly to the client, improving performance and simplifying the deployment process. Serving static files is a common requirement for web applications, and Express handles this efficiently through built-in middleware like express.static.

3.4 Frontend Technologies

The frontend of PurrfectMatch is developed using React, a popular JavaScript library for building user interfaces, and Vite, a fast and modern build tool. These

technologies work together to provide a dynamic and responsive user experience.

3.4.1 React

React is a declarative, component-based library that enables developers to build complex user interfaces from small, reusable pieces of code known as components. It employs a virtual DOM to efficiently update and render the UI in response to state changes, ensuring high performance and a smooth user experience. [BP20] React's component-based architecture promotes reusability and modularity, simplifying the management of the codebase.

React components can be stateful or stateless, with stateful components managing their internal state and responding to user interactions. Components can be composed to create complex UIs, with data passed between components through props. This approach encourages a unidirectional data flow, making it easier to understand and debug the application's behavior.

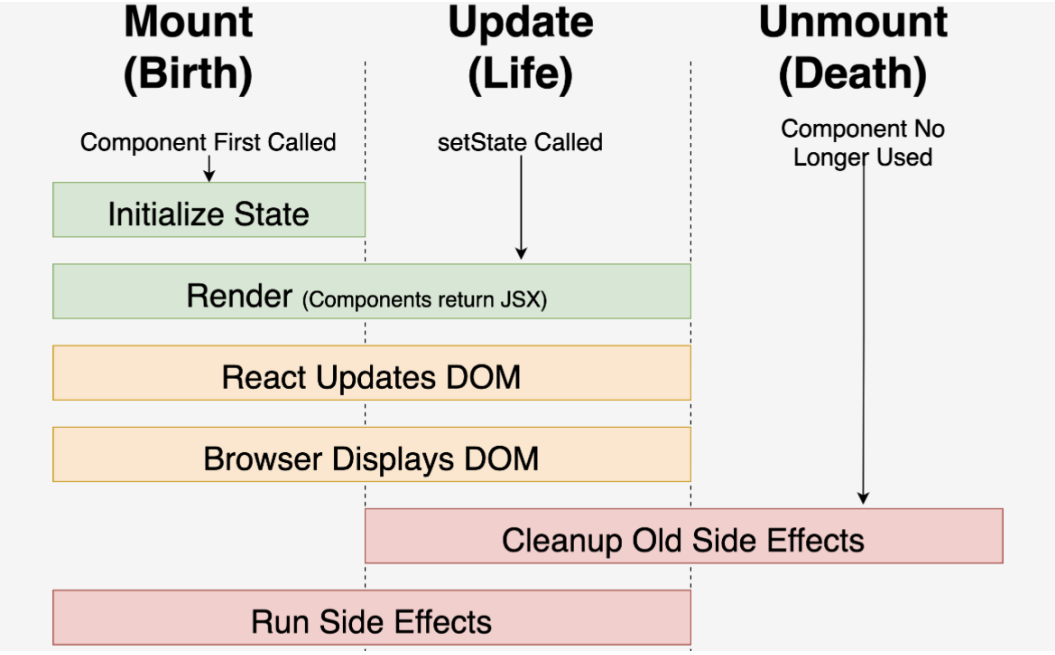

Figure 3.4: Diagram Flow of React [Bra19]

3.4.1.1 Hooks

Introduced in React 16.8, Hooks provide a way to use state and other React features in functional components. The *useState* hook allows developers to add state to functional components, while *useEffect* lets them perform side effects such as data fetching or subscribing to events. [BP20] Other hooks, like *useContext* and *useReducer*, offer additional functionality for managing global state and more complex

state logic, respectively. Hooks simplify code by reducing the need for class components and lifecycle methods, making it easier to share logic across components.

3.4.1.2 State Management in React

State management in React is a crucial aspect of building dynamic and interactive applications. React components can maintain internal state, which allows them to manage and respond to user interactions and other events. Developers can use the *useState* hook to initialize and update state variables, ensuring that the component re-renders when state changes occur. [BP20]

For example, *useState* allows a component to track user input in a form, providing real-time feedback and validation. Additionally, more complex state logic can be handled using the *useReducer* hook, which is ideal for managing state transitions in larger applications. Effective state management helps developers build more efficient and responsive applications, ensuring a seamless user experience.

3.4.1.3 Context API

The Context API in React is used for managing global state across a React application. It allows developers to create context objects that can be accessed by any component within the context provider. This is particularly useful for passing down data that needs to be available to multiple components, such as user authentication status or theme settings. [Sto16] By avoiding *"prop drilling"* (the process of passing data through many levels of components), the Context API enhances code readability and maintainability.

3.4.1.4 Virtual DOM

React's virtual DOM serves as an efficient, lightweight copy of the actual DOM. When a component's state changes, React updates the virtual DOM and performs a comparison with its previous version. This process, called reconciliation, detects the differences and updates only the specific parts of the real DOM that have changed. [Sto16] This efficient diffing mechanism minimizes direct manipulations of the real DOM, leading to quicker rendering times and a more responsive user interface.

3.4.1.5 JSX

JSX (JavaScript XML) is a syntax extension for JavaScript, enabling developers to write HTML-like code directly within JavaScript. By visually representing the UI's structure, JSX enhances code readability. It is transpiled to JavaScript before execution, enabling developers to combine the logic and presentation layers seamlessly.

[Sto16] JSX also supports embedding JavaScript expressions within the markup, providing a powerful way to dynamically generate content.

3.4.1.6 Single Page Applications (SPA)

React is often used to build Single Page Applications (SPAs), where the entire application runs on a single web page. SPAs dynamically update the content without requiring a full page reload, providing a faster and more seamless user experience. This is achieved through client-side routing, where React Router, a popular library for React applications, plays a crucial role.

React Router allows developers to define routes within their applications and manage navigation between different views. Each route corresponds to a specific component, which React dynamically renders based on the URL. This enables the application to handle complex navigation structures while maintaining a single page structure. [Sto16]

3.4.2 Vite

Vite is a modern build tool that offers a significant improvement over traditional tools like Webpack and Babel. It provides fast development and build times by leveraging native ES modules in the browser and a highly optimized build process. Vite's hot module replacement (HMR) feature ensures that changes are reflected instantly during development, enhancing the developer experience and productivity.

Vite is designed to be highly configurable, with support for various plugins and integrations. It can handle both JavaScript and TypeScript projects, providing a smooth development experience regardless of the chosen language. Vite's build process is optimized for performance, producing highly optimized bundles for production deployment. [BP20]

3.5 Database Management System

PurrfectMatch utilizes SQLite as its database management system, with Sequelize as the Object-Relational Mapping (ORM) tool. SQLite is a lightweight, disk-based database that is easy to set up and requires minimal configuration. It is well-suited for web applications that require a simple and reliable database solution. [Owe06]

3.5.1 SQLite

SQLite is a self-contained, serverless, and zero-configuration database engine. It is highly reliable and performs well across a variety of applications. One of its key

advantages is its simplicity, as it stores the entire database in a single file on disk. This makes it easy to manage and deploy, particularly for smaller applications and development environments.[Owe06]

3.5.2 Sequelize

Sequelize is a promise-based ORM for Node.js that supports various SQL dialects, including SQLite. It provides a simple and powerful API for defining models, querying the database, and handling relationships between entities. By using Sequelize, developers can write database queries in JavaScript, abstracting the complexities of SQL and making the codebase more maintainable and readable. [Owe06]

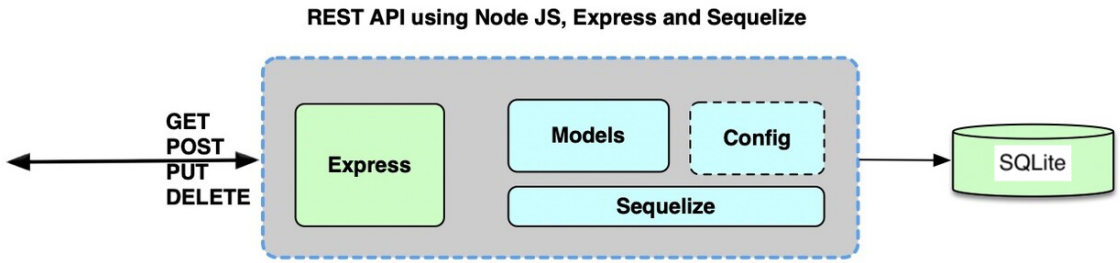

Figure 3.5: Simple REST API using Node.js, Express and Sequelize [L.21]

3.5.2.1 Key Features of Sequelize

Model Definition: Sequelize allows developers to define models that represent database tables, with attributes that map to table columns. This ensures a consistent and structured schema. Models can also include custom methods and validations, making it easier to enforce business rules and data integrity directly within the model definition. [CM20b]

Querying: Sequelize provides a powerful querying API, allowing developers to perform complex queries using a simple and intuitive syntax. This includes support for filtering, sorting, and aggregating data. Additionally, Sequelize offers raw queries for more advanced use cases where direct SQL statements are necessary, providing the flexibility to handle unique requirements without sacrificing the convenience of the ORM. [CM20b]

Associations: Sequelize supports defining relationships between models, such as one-to-one, one-to-many, and many-to-many associations. This allows for efficient querying and manipulation of related data. Sequelize's eager and lazy loading

capabilities enable fetching associated data as needed, optimizing performance and reducing unnecessary database calls. [CM20b]

Migrations: Sequelize includes built-in support for database migrations, enabling developers to manage schema changes over time in a systematic and version-controlled manner. Migration files can be versioned and rolled back if necessary, ensuring a safe and controlled evolution of the database schema. [CM20b]

Validation and Constraints: Sequelize offers extensive validation options and constraints at the model level, ensuring data accuracy and consistency. Developers can define custom validation rules and constraints such as unique, not null, and foreign key constraints, enhancing data integrity and preventing invalid data from being persisted in the database. [CM20b]

Transactions: Sequelize offers comprehensive transaction support, allowing developers to combine multiple database operations into a single, atomic transaction. This ensures that all operations are either fully completed or entirely rolled back, preserving data consistency and integrity. By handling transaction management, Sequelize simplifies the execution of complex workflows that involve multiple interdependent database operations. [CM20b]

3.5.2.2 Migrations with Sequelize

Database migrations are essential for managing changes to the database schema over time. Sequelize provides a robust migration tool that allows developers to define and execute migrations, ensuring that the database schema evolves in a controlled and consistent manner.

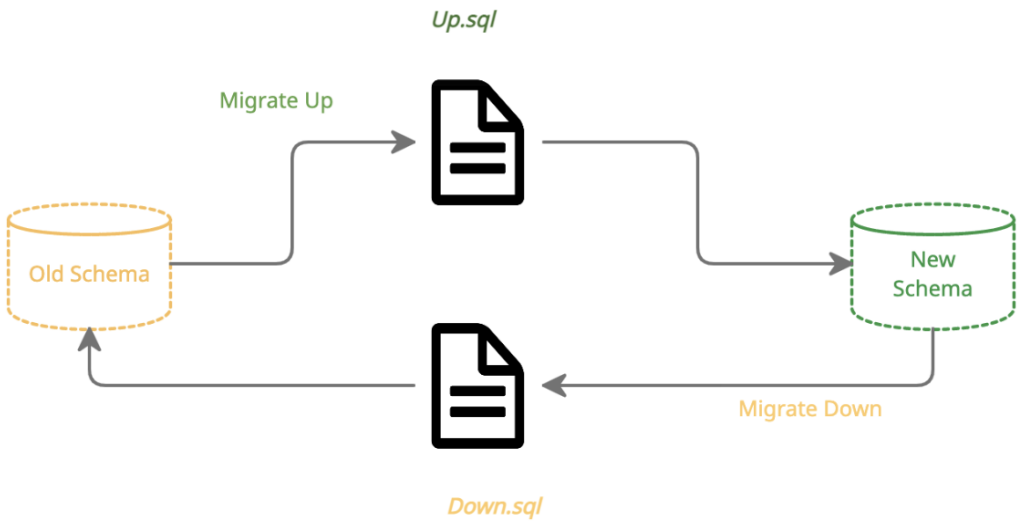

Figure 3.6: Database Migration UP/Down [Gor21]

Key Concepts of Migrations:

- **Migration Files**: Migrations are defined in JavaScript files that specify the changes to be made to the database schema. Each migration file includes *up* and *down* functions, which describe how to apply and revert the changes, respectively.
- **Migration Commands**: Sequelize's CLI (Command Line Interface) provides commands to create, run, and revert migrations. These commands ensure that migrations are applied in the correct order and that the database schema is always in a consistent state.
- **Version Control**: By storing migration files in a version control system, such as Git, developers can track changes to the database schema and collaborate effectively with other team members. This ensures that the entire development team is aware of schema changes and can apply them to their local development environments.

3.6 Security

Ensuring the security of the application is crucial for providing a safe experience. To protect user data and maintain the integrity of the application, PurrfectMatch implements several security measures, including input validation to ensure that all user inputs are validated and sanitized, preventing injection attacks; secure authentication and authorization mechanisms to verify user identities and control access to resources; and data encryption to safeguard sensitive data both in transit and at rest, protecting it from unauthorized access. [Hof20]

Additionally, in a decoupled architecture, employing HTTPS for secure communication between the frontend and backend is essential. HTTPS encrypts data transmitted over the network, protecting it from interception and ensuring secure communication. This security measure is crucial for maintaining the integrity and confidentiality of sensitive information

3.7 Conclusion

In conclusion, the system architecture of PurrfectMatch seamlessly integrates modern frontend and backend technologies to create a robust, scalable, and user-friendly web application. By leveraging Node.js and Express on the backend, the platform handles high concurrency and complex business logic efficiently. React and Vite

on the frontend ensure a dynamic, responsive user interface that enhances user engagement and satisfaction. SQLite and Sequelize manage the data layer effectively, offering a reliable solution for database management and schema migrations.

Adhering to the decoupled architecture allows for a clear separation of concerns, promoting an organized and maintainable codebase. This structure not only simplifies development but also supports future enhancements. Additionally, the platform implements comprehensive security measures, including input validation, secure authentication, and data encryption, ensuring a safe and secure user experience.

By combining these technologies and architectural principles, PurrfectMatch delivers a high-quality web application that effectively meets user needs while providing a foundation for ongoing improvement. This thoughtful integration ensures that the platform remains flexible, efficient, and user-centric, resulting in a reliable and engaging user experience.

Chapter 4

Application

4.1 Introduction

This chapter provides a comprehensive overview of the development and implementation of the PurrfectMatch web application. PurrfectMatch is designed to facilitate cat adoptions by providing a platform where users can list cats for adoption and find cats to adopt directly from other users. The application includes several key features: an authentication system, cat profiles management, a favorites list, an adoption request system, a care guide, and user profile management. This chapter covers the problem statement and specifications, design and architecture, implementation details, and a user guide.

4.2 Development Overview

4.2.1 Problem Statement and Specifications

The primary goal of PurrfectMatch is to create a user-friendly web application that streamlines the process of cat adoption. The platform aims to connect cat owners looking to rehome their pets with individuals seeking to adopt cats.

The primary issue that PurrfectMatch addresses is the inefficiency and lack of personalized interaction in current pet adoption processes. Many existing platforms do not provide sufficient information about pets, nor do they facilitate direct communication between potential adopters and current pet guardians. This often leads to uninformed decisions, mismatches between pets and owners, and ultimately, a higher rate of pets being returned to shelters.

4.2.2 Design and Architecture

The design and architecture of PurrfectMatch are meticulously crafted to ensure a seamless user experience and efficient handling of data. The application adheres to the decoupled architecture pattern, which separates the frontend and backend into distinct layers that communicate through well-defined APIs. In Figure 4.1, we have a use case diagram that illustrates the interactions between users and various components of the system, providing a clear visualization of the application's workflow.

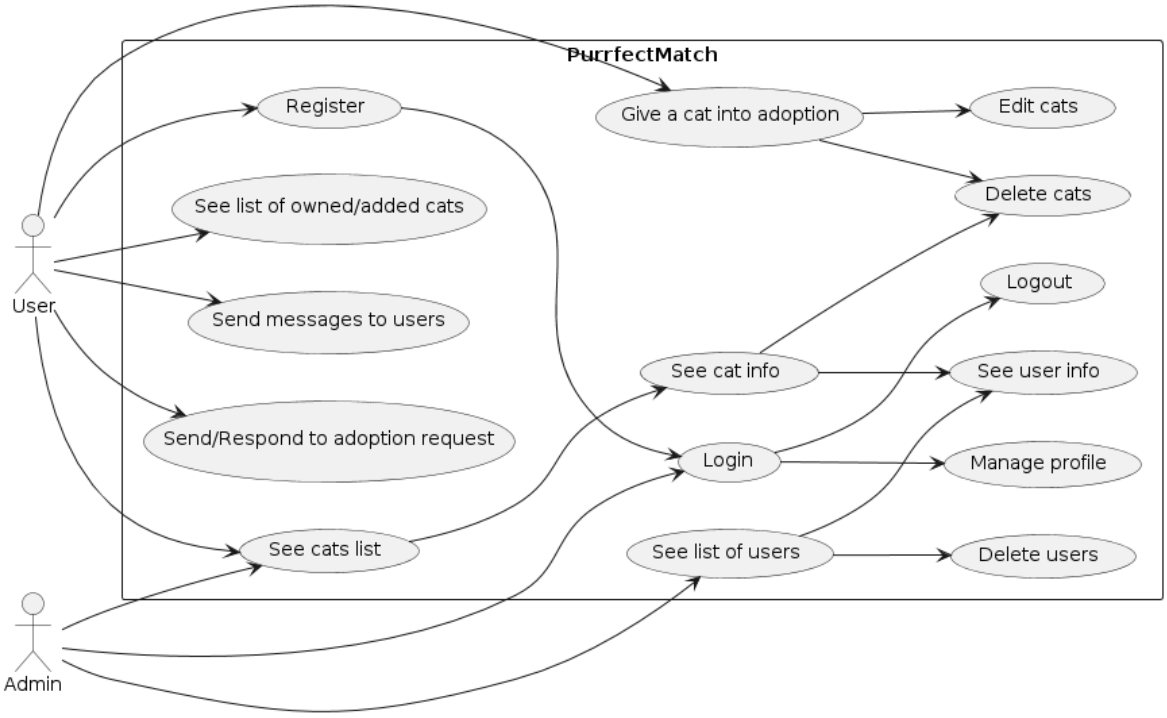

Figure 4.1: Use case diagram of the PurrfectMatch application

Frontend: The frontend of PurrfectMatch is developed using React, a popular JavaScript library for building user interfaces. React is chosen for its component-based architecture, which allows for reusable UI components and efficient rendering. The application uses Vite as a build tool, which offers fast and optimized development by providing instant feedback and hot module replacement. React handles the user interface, including the display and interaction of cat profiles, user authentication forms, favorites list, and adoption request forms. The frontend communicates with the backend through RESTful API endpoints to fetch and submit data, ensuring a dynamic and responsive user experience.

Backend: The backend is built using Node.js with the Express framework. Node.js is a powerful JavaScript runtime that enables server-side scripting, while Express is a minimal and flexible web application framework that provides a robust set of fea-

tures for web and mobile applications. The backend handles HTTP requests, processes business logic, manages user authentication, and interacts with the database. It is responsible for operations such as creating, reading, updating, and deleting cat profiles, managing user sessions, and handling adoption requests. The backend exposes a series of API endpoints that the frontend interacts with. For instance, there are endpoints for user authentication *(/register, /login)*, managing cat profiles *(/cats)*, handling favorites *(/favorites)*, and processing adoption requests *(/adopt)*. In Figure 4.2, we have the sequence diagram of the cat adoption process, detailing the step-by-step interactions between the user, frontend, backend, and databaset.

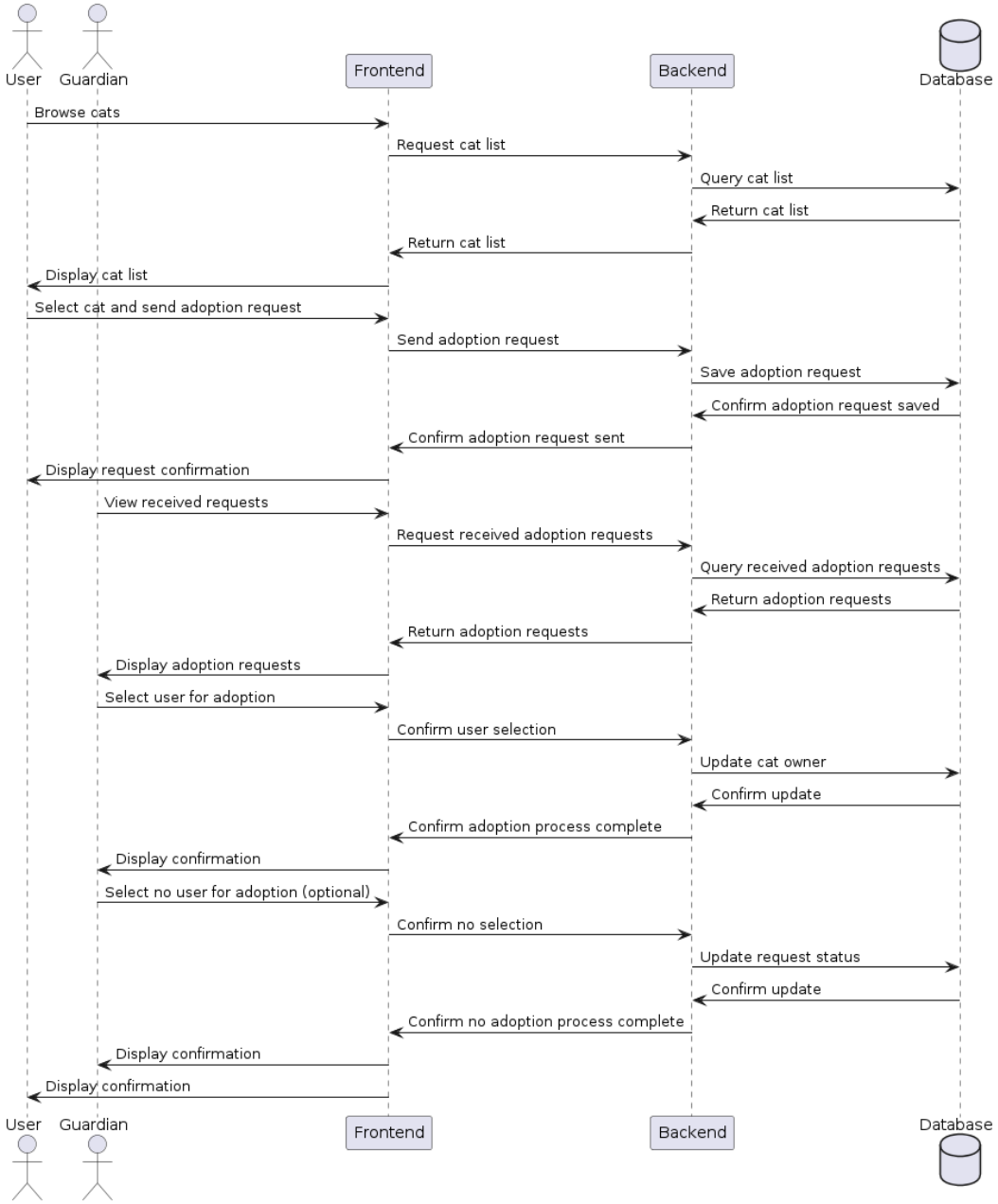

Figure 4.2: Sequence diagram of the adoption process

Database: The database management system used is SQLite, chosen for its lightweight

nature and ease of integration with Node.js. Sequelize, an ORM (Object-Relational Mapping) tool, is used to handle database interactions, making it easier to work with SQL databases by using JavaScript objects. Sequelize also facilitates database migrations, which are used to update the database schema over time without losing data. This setup allows the application to manage user data, cat profiles, and adoption requests efficiently. The database schema includes tables such as users, cats, chat sessions and adoption requests. In Figure 4.3, we have the entire database schema, displaying all tables and their relationships, including the primary tables, as well as the auxiliary tables connected to these primary entities.

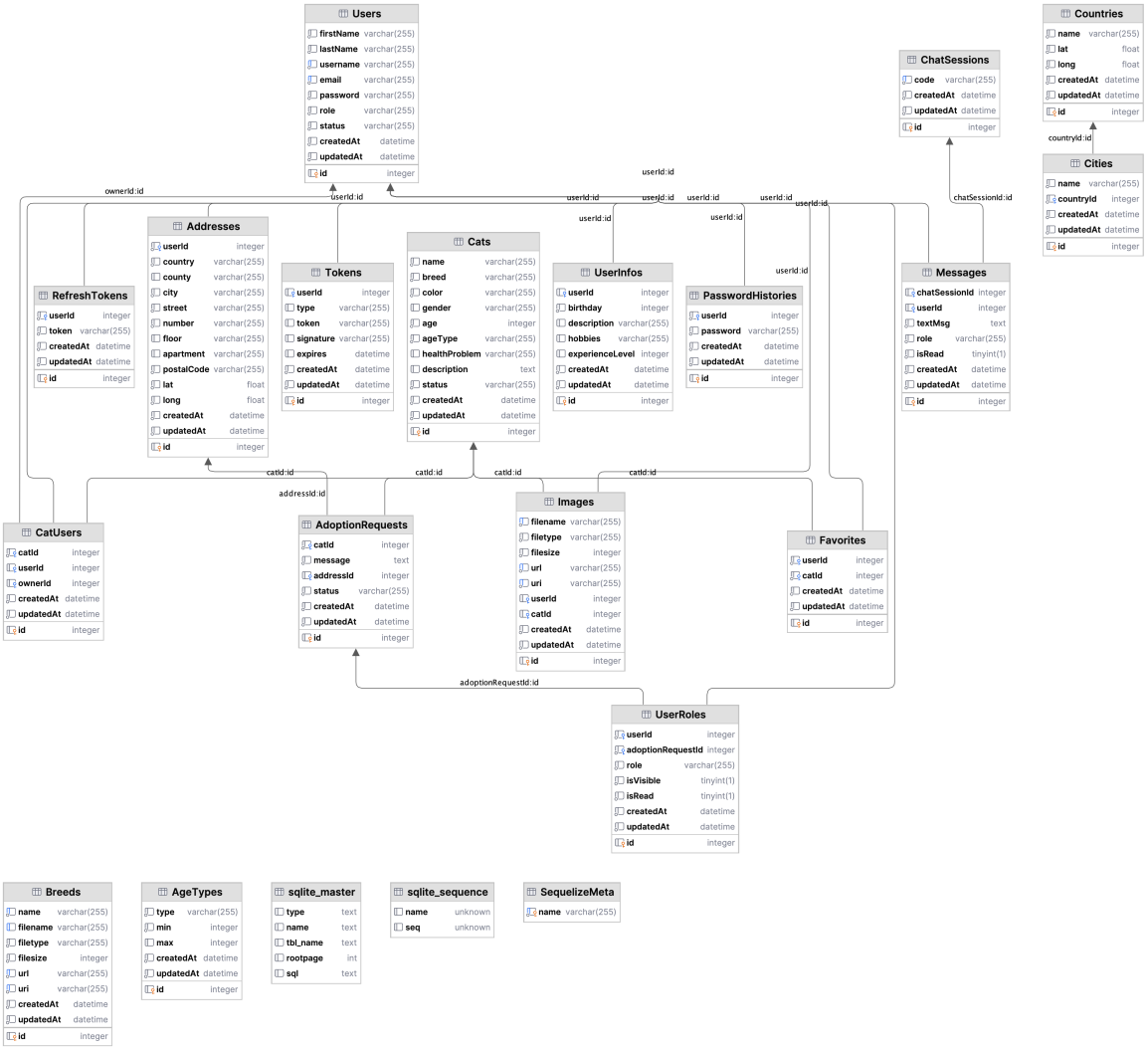

Figure 4.3: Database diagram for the PurrfectMatch application

Technologies Involved: The technologies involved in PurrfectMatch include React for the frontend, providing a dynamic and responsive user interface. Vite is used as the build tool, enhancing the development experience with fast feedback and optimization. On the backend, Node.js with Express handles HTTP requests and server-side logic, providing a robust and scalable server environment. SQLite,

managed through Sequelize, is used as the database management system, ensuring efficient data storage and retrieval. The application follows the RESTful architecture, with clearly defined API endpoints for various functionalities such as user authentic.

4.3 Implementation

The implementation of PurrfectMatch involves a well-structured approach to developing both the frontend and backend, ensuring a modular design that promotes maintainability. This section outlines the setup of the development environment, the development process for the frontend and backend, and the integration of key features such as authentication, cat profiles management, favorites list, adoption requests, care guide, and user profiles.

4.3.1 Setting Up the Development Environment

4.3.1.1 Server Setup

The backend environment is set up using Node.js and npm (Node Package Manager). Node.js provides a JavaScript runtime environment for server-side development. Express is used as the framework for handling HTTP requests and defining API endpoints. To set up the server, initialize a new Node.js project (command *npm init -y*) and install the necessary packages. Additionally, a Python script was used to generate images for mock data, aiding in testing the application with realistic data.

1. Navigate to the server directory:

   ```
   cd Project/server
   ```

2. Download images for configured data:

   2.1. Navigate to the configurations' directory:

   ```
   cd configurations/downloads
   ```

   2.2. Run the following command to download the images:

   Note: This step will download the images for cats and breeds (this will be used on the client side).

   ```
   python3 main.py
   ```

3. Install the dependencies:

   Note: This step will install all the necessary dependencies for the server, set up the database tables, and initialize the configured data.

```
npm install
```

4. Start the server:

```
nodemon
```

4.3.1.2 Database Setup

Sequelize is used for database interactions and managing migrations. Sequelize is an ORM (Object-Relational Mapping) tool that simplifies database operations by allowing developers to use JavaScript to interact with the database.

1. Initialize Sequelize in your project:

   Note: This step will generate `config`, `models`, `migrations`, and `seeders` folders in your project. If these folders are already present, you may skip this step.

```
npx sequelize-cli init
```

2. Create migrations:

   Note: Execute this command for creating or modifying tables.

```
npx sequelize-cli migration:generate --name create-name-table
```

3. Run the migration:

```
npx sequelize-cli db:migrate
```

4.3.1.3 Client Setup

The frontend environment is created using Create React App, a tool that sets up a modern React application with a sensible default configuration. Vite is used as the build tool to enhance development speed and efficiency. Initialize the React project (command *npm create vite@latest purrfectmatch –template react*), install necessary dependencies, and configure Vite for optimal performance.

1. Navigate to the client directory:

```
cd Project/client
```

2. Install the dependencies:

```
npm install
```

3. Start the client:

```
npm run dev
```

4.3.2 Developing the Backend

The backend of PurrfectMatch is responsible for handling HTTP requests, processing business logic, managing user authentication, and interacting with the database. The key components of the backend include the authentication system, cat profiles management, favorites list, adoption request system, care guide, and user profiles.

4.3.2.1 Authentication System

Implemented using JSON Web Tokens (JWT) for secure and stateless authentication, PurrfectMatch allows users to register and log in using their email addresses. During the registration process, a new user record is created in the database. The login process generates a JWT token for session management. To enhance security and user experience, a refresh token is also implemented, ensuring that users remain logged in until they choose to log out by visiting the */logout* page. Middleware is employed to protect routes that require authentication, verifying the presence and validity of JWT tokens.

1. Generate a secret key for JWT:

```
openssl rand -base64 32
```

4.3.2.2 Cat Profiles Management

Users can create, view, edit, and delete cat profiles. Each profile includes comprehensive details like age, health status, breed, and more. The backend provides API endpoints to handle CRUD operations for cat profiles, ensuring that users can manage their listings effectively.

4.3.2.3 Adoption Request System

Users can submit requests to adopt a cat directly from the cat's profile page. This system includes form submission and status tracking. The backend processes adoption requests, updating their status and notifying the relevant users. Users can also adjust their address information to ensure accurate delivery of adoption-related documents and communications.

4.3.2.4 User Profiles

Users can manage their personal information and view their adoption history. The backend provides endpoints to update user profiles, manage privacy settings, and retrieve adoption records.

4.3.2.5 Chat Box

The chat box enables real-time communication between users. Implemented using WebSockets, it allows users to send and receive messages instantly. This feature is crucial for facilitating interactions between adopters and guardians, as well as for community discussions and support.

4.3.3 Developing the Frontend

The frontend of PurrfectMatch is structured into multiple components, each responsible for a specific part of the application. This modular approach ensures that each component can be developed, tested, and maintained independently.

4.3.3.1 Authentication Components

The authentication system includes components for registration, login, and password recovery. The registration form collects user information and performs client-side validation. The login form allows users to authenticate with their username/e-mail and password. The forgot password form enables users to request a password reset link based on their email.

4.3.3.2 Favorites List

Users can add cats to their favorites list from the cat's profile page, allowing them to easily access the profiles of cats they are considering for adoption. The favorites list component allows users to view and manage their favorite cats.

4.3.3.3 Cat Catalog

Displays a list of cats available for adoption. Users can filter, sort, and paginate through the list to find cats that meet their preferences. The catalog is designed to provide an intuitive browsing experience.

The filtering options include criteria such as age, breed, gender, and health status. One of the most significant filters is the location feature. This filter prioritizes cats based on the user's geographical location. If the user has provided a location address in their profile, the system uses the latitude and longitude of that address to sort the list. If the user has not provided an address or is not logged in, the system falls back on the browser's location data to determine the sorting order. This ensures that users can easily find cats that are nearby, enhancing the chances of successful adoptions. The geographical sorting is managed through a combination of geolocation services and database queries optimized for handling spatial data.

4.3.3.4 Navigation Bar

Includes links to different sections of the application, such as the cat catalog, favorites list, care guide, and user profile. The navigation bar also features a *"Give a Cat a Home"* button for guardians to add new cats to the platform.

Additionally, there is a search input bar that allows users to access the cat list by searching with specific criteria from any page in the app. This ensures that users can quickly and easily find the information or cats they are looking for, enhancing overall usability and user experience.

4.3.3.5 User Profile

The user profile component allows users to manage their personal information, view their adoption history, adjust privacy settings, and update their address. It provides a comprehensive overview of the user's activities and preferences, ensuring all personal data is accurate and up-to-date.

4.3.3.6 Chat Box

The chat box component facilitates real-time communication within the application. It allows users to send and receive messages, participate in private discussions, and seek advice from the community. The chat box is designed to provide a smooth and responsive messaging experience, enhancing user interaction and engagement.

4.4 Testing the Application

Testing is a crucial phase in the development of PurrfectMatch, ensuring the application functions correctly, remains secure, and delivers a seamless user experience. The testing process involves multiple levels, including unit testing, integration testing, and system testing. Each level focuses on different aspects of the application to ensure comprehensive coverage and robustness.

4.4.1 Unit Testing

Unit testing focuses on validating individual components and functions of the application in isolation. For PurrfectMatch, Jest is used as the primary testing framework due to its versatility and compatibility with both Node.js and React.

For Node.js Jest is used to write and run tests that assert the expected behavior of backend functions. For instance, tests can verify that a user registration function correctly hashes passwords, creates user records, and handles errors for invalid inputs.

For React form handling tests ensure that the user registration form correctly captures input data, updates state, and triggers appropriate actions upon submission. These tests help identify issues with form validation, data handling, and user feedback mechanisms.

4.4.2 Integration Testing

Integration testing examines the interactions between different modules of the application, ensuring that they work together as intended. Jest, along with Supertest, is used to handle integration testing for PurrfectMatch. Supertest is utilized to test HTTP APIs, verifying that routes and middleware function correctly together. For example, tests can validate the user authentication process by ensuring that login endpoints correctly interact with the database and handle sessions.

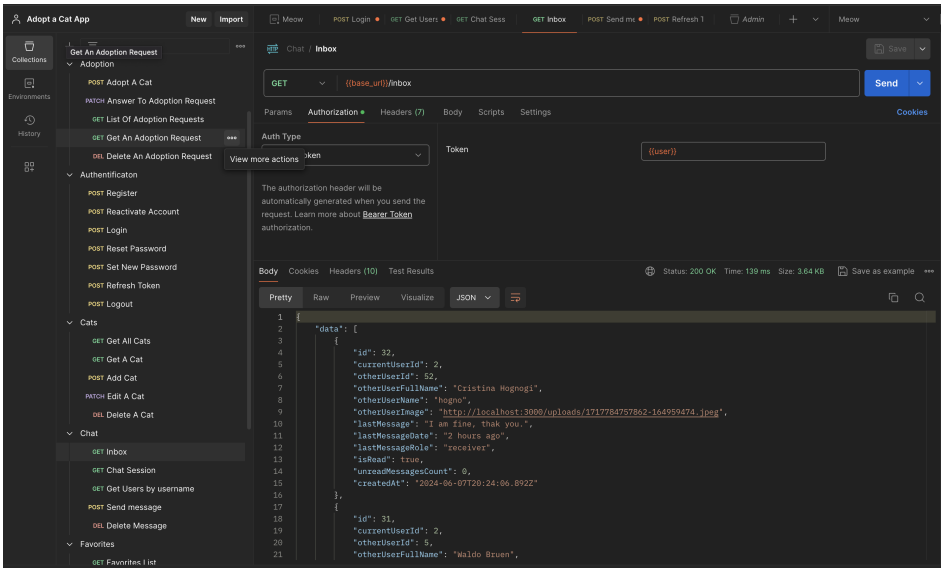

Figure 4.4: RESTful API endpoint with Postman

In addition to Jest and Supertest, Postman is also used for integration testing, particularly for testing RESTful API endpoints. Postman allows developers to create and run automated tests, verify responses, and ensure that the API endpoints behave as expected. By leveraging Postman's comprehensive testing features, the development team can easily test and document APIs, streamline the testing process, and improve collaboration.

4.5 Functionalities

4.5.1 User Registration

The user registration process *(/register)* for PurrfectMatch involves several steps to ensure a secure and verified on boarding of new users. Users receive a verification email to confirm their email address. Once verified, they can complete their profiles. This process ensures only legitimate, verified users access the platform.

- **Input Fields**: Users must provide the following information: First Name, Last Name, Username *(must be unique)*, Email *(must be a valid email address)*, Password *must follow criteria*, Birthday
- **Validation**: The input data undergoes client-side and server-side validation to ensure all required fields are filled correctly. The email address must be unique and properly formatted, and the password must meet security criteria.

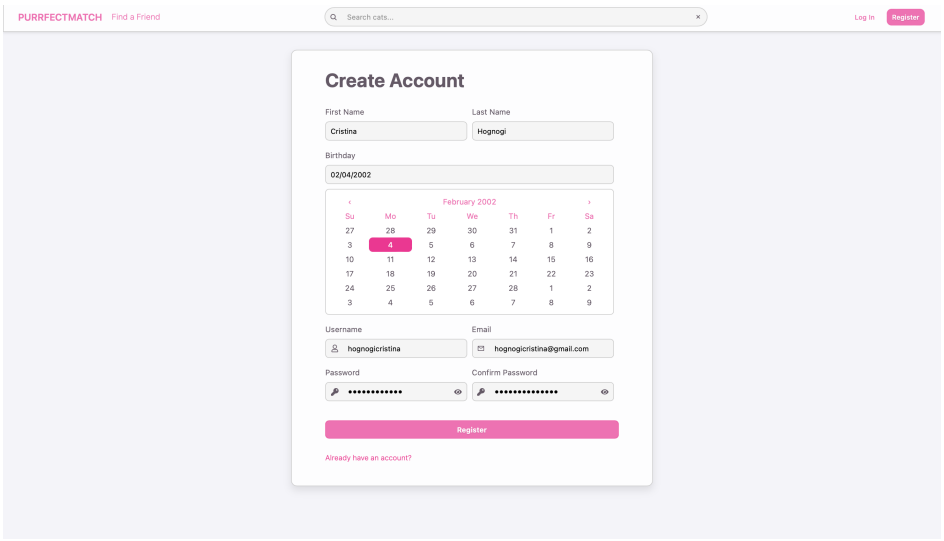

Figure 4.5: Register Form

- **Email Verification**: After submitting the registration form, an email containing a verification link is sent to the user's provided email address. The user must click on this link to verify their email and activate their account before the link expires. If the token expires, the user can request another verification email link to be sent, ensuring they can complete the registration process and access their account.
- **Additional Features**: The registration system also includes a *"Forgot Password"* feature *(/reset)*, allowing users to reset their password if forgotten. Users can log in *(/login)* using their username or email and password after successful registration and email verification.

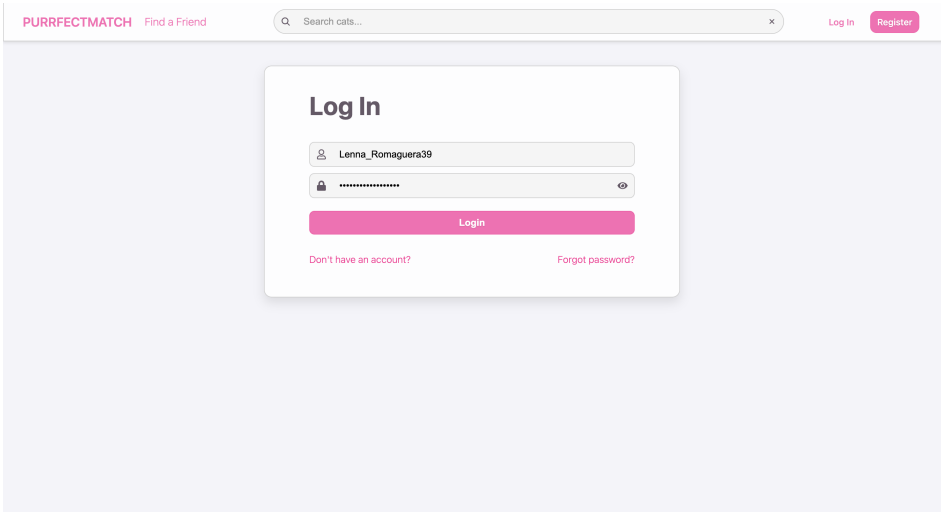

Figure 4.6: Login Form

4.5.2 Adoption Process

The adoption process in PurrfectMatch is designed to facilitate a smooth and informed adoption experience for both users and cat guardians. Users can browse detailed cat profiles, submit adoption requests directly from the cat's profile page, and track the status of their requests in real-time.

- **Cats Catalog**: Users can browse through a catalog of cats available for adoption. The catalog offers various filtering options to help users find cats that match their preferences. Filtering options may include age, breed, health status, location, etc.

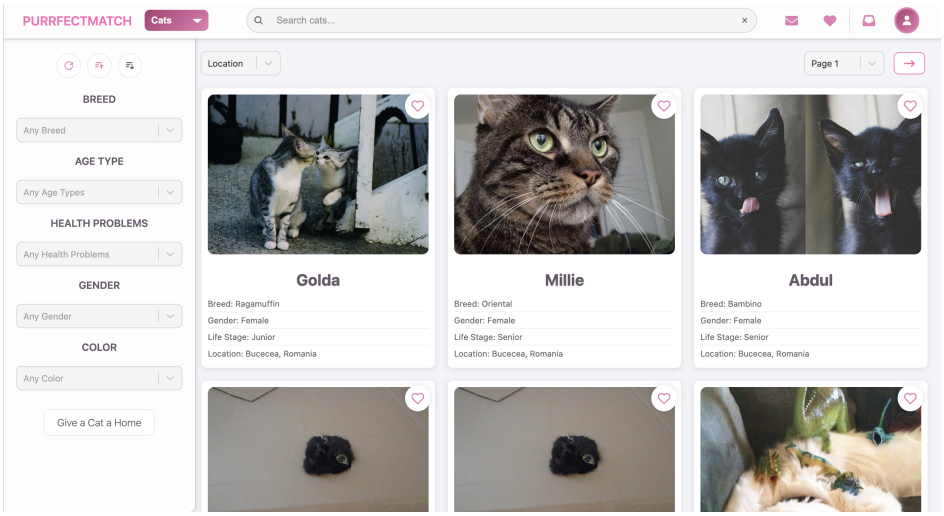

Figure 4.7: Cats Catalog Overview

- **Choosing a Cat**: Users can select a cat from the catalog or home page based on their preferences. Detailed profiles provide information on each cat's health

status, age, and other relevant details.

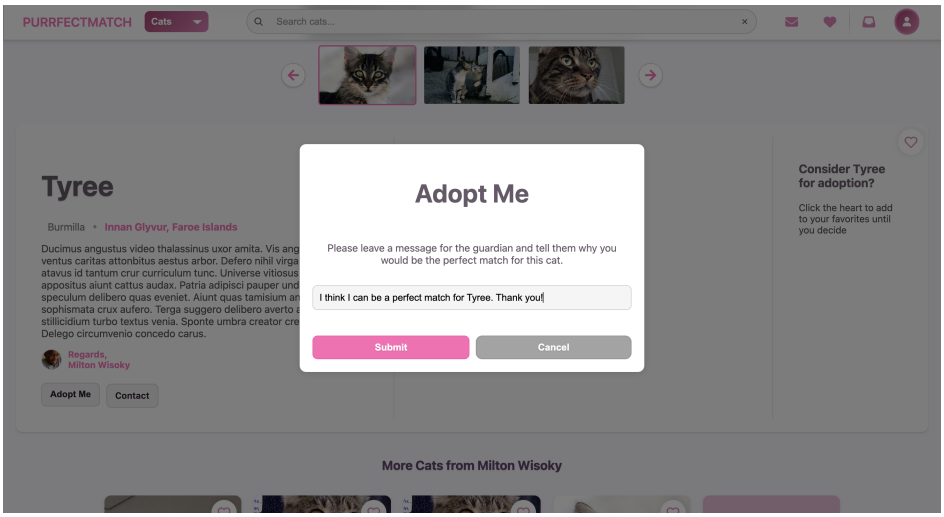

Figure 4.8: Adoption Message Text

- **Sending an Adoption Request**:After selecting a cat, users can send an adoption request via a message text. This message is forwarded to the cat's guardian, who receives a notification and real-time updates about the request. Additionally, users can add a cat to their adoption requests directly from their favorites list, making it easy to keep track of and act on their preferred choices.

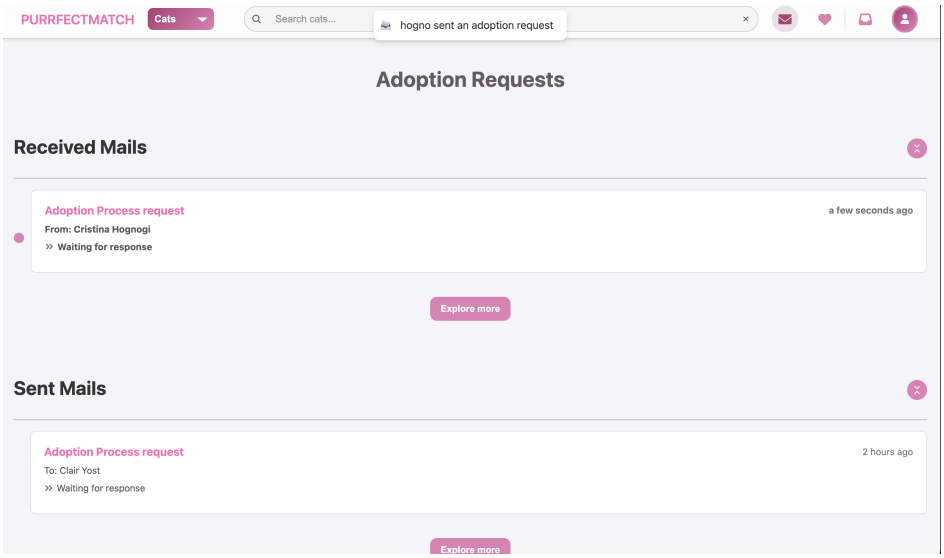

Figure 4.9: Receive New Adoption Request

- **Guardian Review**: While the request is in the pending process, the guardian can review the message and access the user's profile through the provided mail to gain a comprehensive understanding of the potential adopter, aiding in their decision-making process.

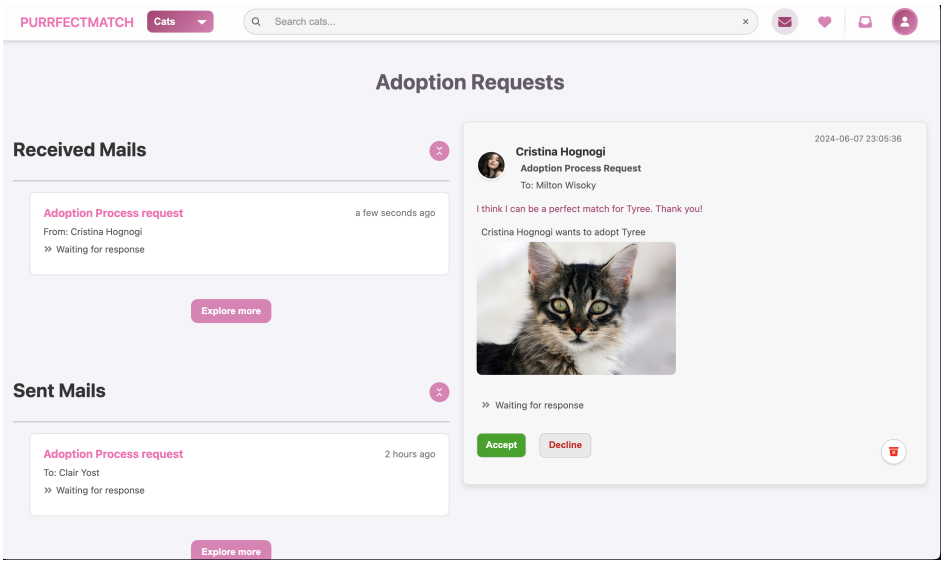

Figure 4.10: Answer Request Inbox

- **Completion of Adoption**: If the guardian accepts the request, the user becomes the official owner of the cat and gains all associated rights and responsibilities, the status of the cat being updated to reflect the new ownership.

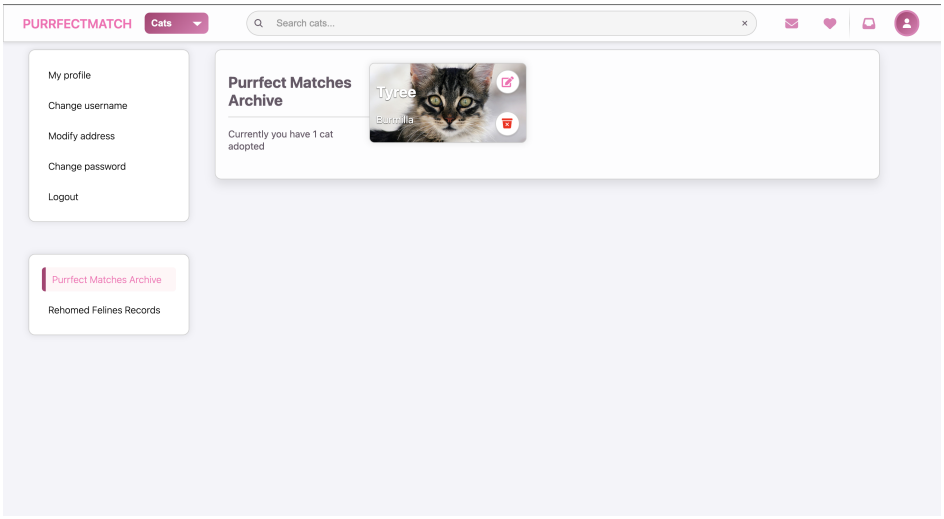

Figure 4.11: List of Ownes cats for a User

4.5.3 Chat Box

This chat box platform is designed for smooth and fast communication, allowing users to share or receive any kind of information quickly and efficiently, providing them with the answers they seek. Users can initiate a chat session directly from another user's profile, making it easy to ask questions or offer advice. The platform encourages users to communicate and offer guidance, fostering a supportive community that enhances the adoption process.

- **Start Session**: The first step in using the chat box is to start a session. Users can browse through their conversations list to see all ongoing chats. To connect with a specific user, they can use the search feature by entering the user's first name or last name. If there isn't already an existing conversation with the selected user, the platform allows users to initiate a new chat session effortlessly, making it easy to begin a new conversation.

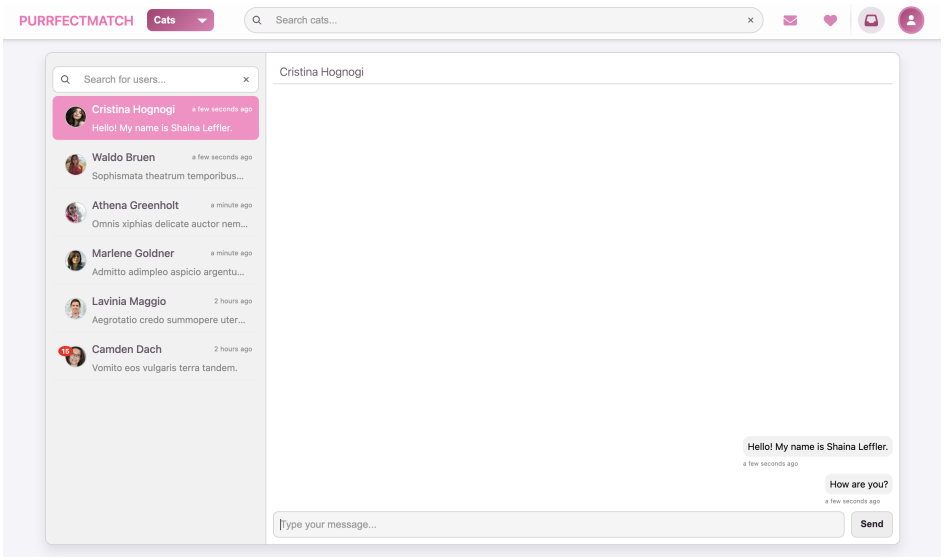

Figure 4.12: Start A New Chat Session

- **Receive New Message**: Once a session has started, users can receive new messages in real time. The chat box is designed to notify users promptly when a new message arrives, ensuring that they do not miss any important communication. Messages are displayed in a clear, easy-to-read format, enabling users to quickly understand and respond to the content.

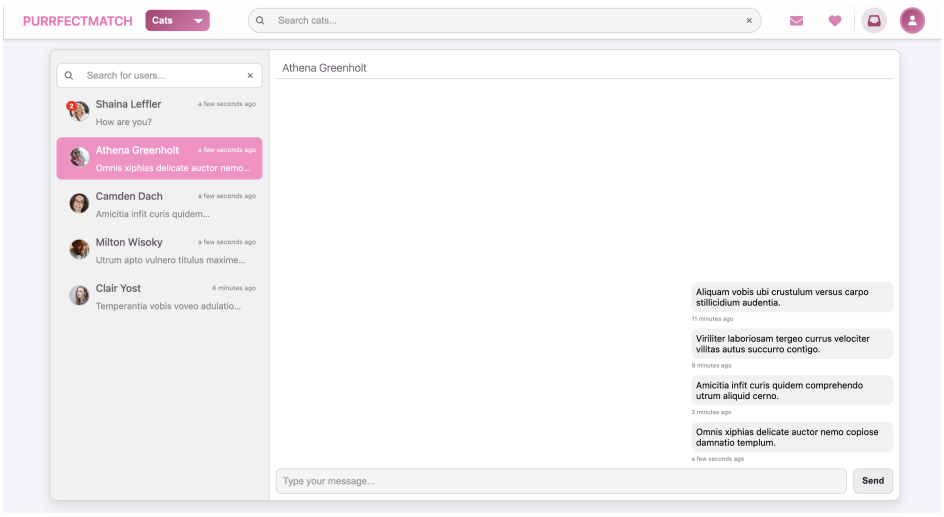

Figure 4.13: Receive The New Chat Session With Messages

- **Receive Message Back**: The chat box supports continuous, real-time communication, allowing users to receive and respond to messages instantly. This facilitates a smooth and engaging conversation flow, ensuring efficient information exchange and advice sharing. The platform's design encourages users to communicate more frequently and offer support to one another, fostering a strong, connected community.

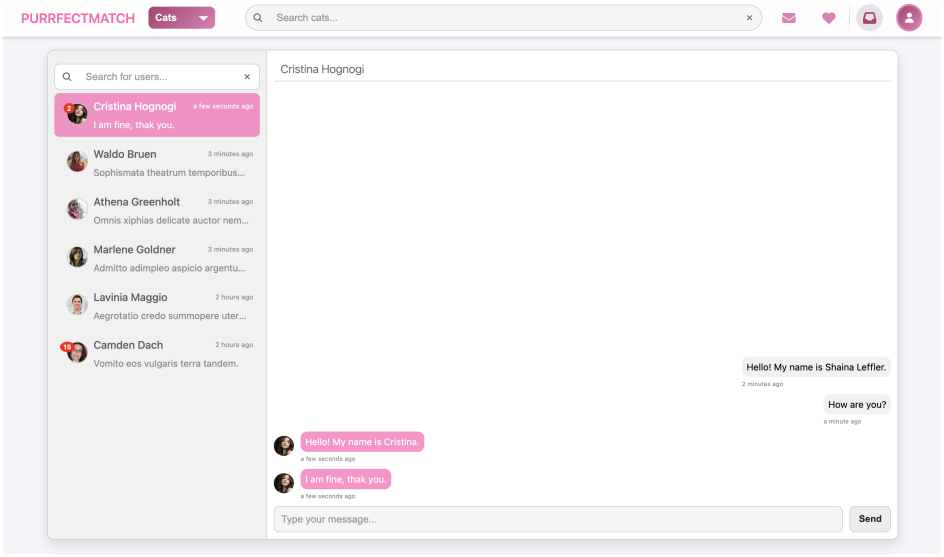

Figure 4.14: Receive Message Back

Chapter 5

Conclusion

The PurrfectMatch web application has been meticulously designed and developed to address the needs of individuals looking to adopt cats and those needing to rehome their pets. By utilizing modern web technologies and following the principles of Human-Computer Interaction (HCI), the application offers a user-friendly platform that facilitates seamless cat adoptions. This thesis has provided a comprehensive overview of the system architecture, the development process, and the implementation details, highlighting the key features and functionality of the application.

The application employs a decoupled architecture, ensuring a clear separation of layers and enhancing maintainability. The frontend is developed using React with Vite, providing a dynamic and responsive user interface, while the backend, built with Node.js and Express, handles HTTP requests, processes business logic, and interacts with the SQLite database through Sequelize ORM. The database schema is designed to efficiently manage users, cats, favorites, and adoption requests, ensuring organized storage and retrieval of data. The development process involved setting up a robust development environment, structuring the frontend and backend into modular components, and implementing key features.

To enhance the functionality and user experience of PurrfectMatch, several future improvements can be considered: implementing a matching algorithm that suggests potential cats to users based on their preferences and responses to a set of questions, thereby improving the likelihood of successful adoptions by aligning the needs and expectations of both adopters and cats; introducing a feature that allows users to send their cats to someone for a period of time, useful for cat owners who need temporary care for their pets due to travel, illness, or other circumstances; and adding social features that enable users to follow each other, share updates, and build a community around cat adoption, fostering a sense of community and support among users and encouraging more active engagement with the platform.

In conclusion, PurrfectMatch marks a major advancement in utilizing technol-

ogy to streamline cat adoptions. By integrating modern web development technologies with a user-centered design approach, the application provides a valuable service to both cat owners and adopters. The proposed future improvements have the potential to further enhance the functionality and user experience, making PurrfectMatch an even more effective and comprehensive platform for cat adoption. Through continuous development and user feedback, PurrfectMatch can evolve to meet the changing needs of its users and make a positive impact on the lives of cats and their owners.

Bibliography

[aca23] Keystone academy. Web development basics: An introduction to building dynamic websites, mar 2023. Accessed: May 2024 https://www.keystonesubic.com/en/web-development-basics-an-introduction-to-building-dynamic-websites/.

[Ade23] Ibrahim Lanre Adedimeji. Node.js architecture: Understanding node.js architecture, Sep 2023. Accessed: March 2024 https://medium.com/@ibrahimlanre1890/node-js-architecture-understanding-node-js-architecture-5fb32879b994.

[And] Dan Anderson. Conscious decoupling: The future of web development. Accessed: June 2024 https://www.vshift.com/ideas/conscious-decoupling-the-future-of-web-development.

[Bec20] Christopher Reid Becker. *Learn Human-Computer Interaction: Solve human problems and focus on rapid prototyping and validating solutions through user testing*. Packt Publishing Ltd, 2020.

[Bes24] Mukhadin Beschokov. What is websocket and how it works?, feb 2024. Accessed: June 2024 https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is.

[Bie22] Cezary Biele. *Human Movements in Human-Computer Interaction (HCI)*. Springer, 2022.

[BP20] Alex Banks and Eve Porcello. *Learning React: modern patterns for developing React apps*. O'Reilly Media, Inc., 2nd edition, 2020.

[Bra19] Bramus. React hook flow diagram, Mar 2019. Accessed: March 2024 https://www.bram.us/2019/03/11/react-hook-flow-diagram/.

[Bro19] Ethan Brown. *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media, Inc., 2019.

[Car23] Rafael Carvalho. What is a single-page application (spa)? pros & cons with examples, jan 2023. Accessed: June 2024 https://www.scalablepath.com/front-end/single-page-applications.

[CM20a] Mario Casciaro and Luciano Mammino. *Node. js Design Patterns*. Packt Publishing Ltd, 3rd edition, 2020.

[CM20b] Mario Casciaro and Luciano Mammino. *Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques*. Packt Publishing, third edition, 2020.

[Cza18] Aleksandra Czajka. Front-end, back-end and database-side - the structure of an app, mar 2018. Accessed: May 2024 https://www.linkedin.com/pulse/front-end-back-end-database-side-structure-app-aleksandra-czajka.

[DFAB03] Alan Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction*. Pearson Education, 3rd edition, 2003.

[Fou16] Interaction Design Foundation. What is user centered design (ucd)?, June 2016. Accessed: May 2024 https://www.interaction-design.org/literature/topics/user-centered-design.

[Gor21] Eresh Gorantla. Db migration in go lang, jul 2021. Accessed: April 2024 https://medium.com/geekculture/db-migration-in-go-lang-d325effc55de.

[Gup23] Amar Gupta. Building a real-time chat application with react and websocket, sep 2023. Accessed: April 2024 https://dev.to/amarondev/building-a-real-time-chat-application-with-react-and-websocket-3138.

[Hav24] Marijn Haverbeke. *Eloquent javascript: A modern introduction to programming*. No Starch Press, 4th edition, 2024.

[Hof20] Andrew Hoffman. *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O'Reilly Media, Inc., 1st edition, 2020.

[Kan22] Vijay Kanade. Human-computer interaction (hci) meaning, importance, and examples, Jul 2022. Accessed: March 2024 https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-hci/.

[Kat21] Jozsef Katona. A review of human-computer interaction and virtual reality research fields in cognitive infocommunications. 2021.

[Kee22] Lance Keene. What is UI/UX design and why it matters for software development, apr 2022. Accessed: March 2024 https://www.keenesystems.com/blog/what-is-ui/ux-design-and-why-it-matters-for-software-development.

[Kle17] Martin Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media, Inc., 1st edition, 2017.

[Kru13] Steve Krug. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3rd edition, 2013.

[L.21] Rajesh L. Simple REST API using node.js, express and sequelize, oct 2021. Accessed: April 2024 https://www.linkedin.com/pulse/simple-rest-api-using-nodejs-express-sequelize-rajesh-lakshmanamurthy.

[Mar08] Robert C Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, 1st edition, 2008.

[McF09] David Sawyer McFarland. *CSS: the missing manual*. O'Reilly Media, Inc., 2009.

[Mea18] Andrew Mead. *Learning Node. js Development: Learn the fundamentals of Node. js, and deploy and test Node. js applications on the web*. Packt Publishing Ltd, 2018.

[MJ13] Mikowski Michael and Powell Josh. *Single Page Web Applications: JavaScript end-to-end*. Manning, 1st edition, 2013.

[Nie93] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1993.

[Nor13] Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[Owe06] Michael Owens. *The definitive guide to SQLite*. Springer, 2006.

[Ram23] Ramotion. Human-computer interaction: Significance for UX designers, apr 2023. Accessed: March 2024 https://www.ramotion.com/blog/human-computer-interaction-for-ux-designers/.

[Rob07] Jennifer Niederst Robbins. *Learning Web Design: A Beginner's Guide to (X) HTML, StyleSheets, and Web Graphics*. O'Reilly Media, Inc., 3rd edition, 2007.

[Sim15] Kyle Simpson. *You Don't Know JS: ES6 & Beyond*. O'Reilly Media, Inc., 2015.

[Sto16] Stefanov Stoyan. *React Up & Running: Building web Applications*. O'Reilly Media, Inc., 1st edition, 2016.

[Uni16] Stanford University. Human-computer interaction research. *CS376 Course Report*, 2016.

[Wik] Wikitechy. What is ExpressJS - ExpressJS tutorial. Accessed: May 2024 https://www.wikitechy.com/tutorial/expressjs/what-is-expressjs.