One-Stop Pet Adoption Platform
FINAL YEAR DESIGN PROJECT REPORT

By
Adnan Ashkari
Student ID: 191-15-12710
Tahmid Islam
Student ID: 191-15-12899

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements for the **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Sharun Akter Khushbu
Assistant Professor
Department of Computer Science and EngineeringDaffodil International University

Co-Supervised by

Mr. Amir Sohel
Assistant Professor
Department of Computer Science and EngineeringDaffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh
September 17, 2025

APPROVAL

This Project titled **"One-Stop Pet Adoption Platform"**, submitted by Adnan Ashkari, ID No: **191-15-12710 &** by Tahmid Islam, ID No: **191-15-12899** to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **17 September, 2025**.

BOARD OF EXAMINERS

Dr. Arif Mahmud (AM) — **Chairman**
Associate Professor and Associate Head
Chairman, Department of CSE, DIU

Dr. Fizar Ahmed (FZA) — **Internal Examiner**
Associate Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mushfiqur Rahman (MUR) — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Md. Manowarul Islam (DMMI) — **External Examiner**
Professor
Department of Computer Science and Engineering
Jagannath University

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Sharun Akter Khushbu, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

17.9.2025

Sharun Akter Khushbu
Assistant Professor
Department of Computer Science and Engineering Daffodil International University

Co-Supervised by:

17.9.2025

Mr. Amir Sohel
Assistant Professor
Department of Computer Science and Engineering Daffodil International University

Submitted by:

17.09.2025

Adnan Ashkari
Student ID: 191-15-12710
Department of Computer Science and Engineering Daffodil International University

Tahmid 17.9.2025

Tahmid Islam
Student ID: 191-15-12899
Department of Computer Science and Engineering Daffodil University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has as- sisted us in one way or another.

First, we express our heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for us to complete the **Final Year Design Project(FYDP)** successfully.

We are grateful and wish our profound indebtedness to **Sharun Akter Khushbu**, **Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of our supervisor in the field of **Computer Science and Information Technology** to carry out this project. His endless pa- tience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting themat all stages have made it possible to complete this project.

We would like to express our heartfelt gratitude to the Head of the Department of Com- puter Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

We would like to thank our entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

The One-Stop Pet Adoption Platform is a web-based application that consolidates pet discovery and adoption with post-adoption services in a single cohesive system. Traditional processes are fragmented across different websites and providers, which discourages adoptions and complicates aftercare. Our platform integrates searchable adoption listings with rich pet profiles, an e-commerce module for essential supplies, and veterinary appointment booking. The frontend is implemented with **React** (Vite and React Router) and **TailwindCSS** for responsive UI; the backend uses **Node.js/Express**, while **MySQL** provides persistent storage. **JWT** and **bcrypt** enforce secure authentication. Testing spans unit, integration, system, usability, performance, and security validations. Results indicate that unifying adoption and aftercare reduces friction for users and improves operational oversight for administrators via role-based tools. The solution establishes a robust foundation for future enhancements such as secure online payments, cloud deployment, AI-based recommendations, and mobile application.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Companion animals constitute a significant factor in enhancing psychological well-being, emotional resilience, and social companionship, yet millions of pets continue to remain unhomed within shelters on a global scale. International welfare assessments consistently demonstrate a disproportionate imbalance between intake and adoption levels, thereby exacerbating conditions of overcrowding and chronic underfunding in existing facilities. Conventional adoption mechanisms—distributed across localized shelter websites, fragmented e-commerce environments, and offline veterinary practices—are inherently disjointed, operationally inefficient, and largely inaccessible to a considerable segment of potential adopters. This systemic fragmentation functions as a deterrent to adoption, ultimately resulting in diminished placement efficacy and reduced sustainability of long-term human–animal relationships.

The rapid evolution of advanced web technologies has created an opportunity to consolidate fragmented adoption processes into a singular, scalable digital infrastructure. Modern frameworks such as **React**, **Node.js**, and **MySQL** provide the architectural foundation for constructing secure, responsive, and data-intensive platforms that simultaneously enhance user interaction and optimize administrative oversight. The proposed **One-Stop Pet Adoption Platform** is conceptualized as an integrated system designed to centralize adoption listings, embed e-commerce functionality for essential supplies, and streamline veterinary appointment scheduling through intelligent automation. By addressing the deficiencies inherent in existing disjointed solutions, the platform seeks to establish a holistic, technology-driven ecosystem that strengthens operational efficiency while fostering sustainable connections between adopters, shelters, and companion animals.

1.2 Motivation

This project is driven by necessity in the society and technological feasibility. Stray and abandoned animals are becoming a serious issue in most urban centers including Dhaka that has few resources in sheltering and low adoption turnover. The challenge of locating trustworthy information, submitting applications via separate systems, and securing post-adoption services is deterring many of its prospective adopters.

Simultaneously, the fast implementation of digital platforms has altered the sector of retail, healthcare, and education. This shows that technology when used wisely can remove the inefficiencies and provide an increased user engagement. The vision of this project is to bring these benefits to the pet adoption ecosystem.

Through all these efforts to consolidate pet discovery, adoption applications, supply ordering, and veterinary services under a single digital roof, we will stimulate responsible ownership, minimizing adoption obstacles, and aid animal welfare organizations in their cause. Finally, we are driven not only by academic but humanitarian reasons as well, by using our technical abilities to make a physical social change.

1.3 Objectives

The goals of the **One-Stop Pet Adoption Platform** are designed to be precise, measurable, and relevant. They extend operational functionality to achieve long-term benefits for both adopters and administrators.

1. User-Centric Services

- Provide secure registration and authentication using industry-standard protocols (JWT, bcrypt).
- Empower users with intuitive navigation, advanced search, filtering options, and detailed pet profiles (including age, breed, health, and history).

2. Simplification of Adoption Process

- Enable users to submit adoption applications digitally and monitor their status.
- Provide administrators with decision dashboards to review, approve, or reject applications transparently.

3. Post-Adoption Support

- Incorporate an online shopping feature for purchasing pet products, ensuring adopters can access essential supplies conveniently.
- Enhance veterinary appointment booking and record-keeping to support long-term pet health and wellbeing.

4. Administrative Oversight

- Equip administrators and shelters with role-based tools to manage pets, products, orders, and user data.
- Provide dashboards for monitoring system performance and adoption trends.

5. Scalability and Security

- Ensure the system can handle increased loads with minimal architectural adjustments.
- Guarantee strong data protection, secure authentication, and safe user interactions.

1.4 Methodology

A balanced approach towards development was employed as a hybrid in order to strike a balance between the rigidity of the Waterfall model and the adaptability of iterative improvement. Waterfall stages facilitated thorough documentation and clear milestones that is very vital in an academic environment. Simultaneously, the possibility of iterative cycles permitted making changes based on the feedback, testing outcomes, and changing needs. This combination allowed reducing risks without making the development process rigid.

The methodology was designed in the following key stages:

Requirement Analysis

- Collecting functional as well as non-functional requirements, as seen by end-users (adopters) and administrators (shelters, system managers).

- It involved defining the modules that are critical and those that need to be built and recorded like adoption module, e-commerce module, and veterinary bookings and some of the limitations that should be recorded including security, scalability, and usability.

System Design

- Creation of architectural diagrams that can demonstrate how frontend, backend and database layers interact.

- Formalizing data models and workflows with the help of creating Entity-Relationship (ER) diagrams and use-case diagrams.

- Trace process flows (adoption, checkout, and booking) to provide the consistency of these processes across modules.

Implementation

- **Frontend**: React, Vite, and React Router were used to develop this frontend modularly and using components. Responsiveness and accessibility have been upheld using TailwindCSS.

- **Backend**: Scalable APIs and middleware support was selected to be based on Node.js with Express.

- **Database**: MySQL used as the persistent storage facility with normalized data schema, foreign key, and performance optimization index.

Testing

- **Unit Testing**: A single component and API endpoint.
- **Integration Testing**: Proven end-to-end processes like adoption application subjection and cart-to-order processing.
- **System Testing**: Confirmed overall platform stability and properness.
- **Usability Testing**: Measurement of mobile navigation, mobile responsibility, and accessibility.
- **Performance/Security Testing**: Secured optimized database queries, JWT authentication, and implemented OWASP-compatible security measures.

Evaluation & Documentation

- Evaluated the module effectiveness against the requirements.
- Recorded findings, known weaknesses, and suggested improvements of future versions.
- Easy-to-follow figures, tables and appendices to give clarity and professionalism of the report.

Overall, this mixed approach enabled to create a rigorous account and at the same time provide flexibility. The Waterfall model architecture gave a roadmap to follow with the definite deliverables and iterative improvements were used to respond to the arising issues to guarantee a maintainable, scalable, and secure system.

1.5 Project Outcome

The adopted system proves the effectiveness and importance of the combination of adoption discovery and post-adoption services into one unified platform. In contrast to the traditional fragmented solutions where the adopters have to use various systems to carry out various activities; this platform offers a centralized platform that simplifies the whole process of adoption.

On the part of the user, the platform makes life easier since one can filter pets by advanced features and browse profiles, place adoption applications, order supplies, and make appointments at the veterinary facility- all without leaving the same space. Not only does this enhance convenience but it also brings about responsible ownership because it is simpler to remain engaged once adopted.

On behalf of the administrator, the system provides role-gated controls and dashboards enabling shelters and managers to monitor pets, users, orders and applications. This enhances the management of operations, decision-making processes, and efficiency in dealing with adoption requests and supply management.

The project also creates a good basis of improvement in future. Its modular structure ensures that one can add more features like:

- Online payment processing of real time transactions.
- Mobility applications to reach more and be convenient.
- Recommendations systems based on preferences and history through AI to pair adopters with pets.
- Deployment on the cloud to be able to achieve scalability, resilience, and greater accessibility.

Overall, the end product of this project is not just a demonstration of concept but a practical policy that will cope with the issue of real-world adoption. It promotes animal welfare by enhancing access to adoption and long-term follow-ups as well as delivering a framework of scalable scholarly, technical as well as community-based extensions.

1.6 Organization of the Report

- **Chapter 2** presents background knowledge, the literature review, similar applications, and a gap analysis.

- **Chapter 3** details the methodology and design, including requirements, data flow, UI design, and the project plan.

- **Chapter 4** covers implementation details, testing and evaluation, and the results.

- **Chapter 5** discusses relevant standards, societal/ethical impact, sustainability, project management/financials, and engineering mappings.

Chapter 6 concludes with limitations and future work. References and appendices follow.

Chapter 2

Background

Table 2.1: Summary of Literature Reviewed

2.3 Similar Applications

- **Adoption discovery**: feature-rich search and filtering across shelters are common.
- **E-commerce for pet supplies**: comprehensive catalogs and logistics exist but with no linkage to adoption workflows.
- **Healthcare bookings**: standalone systems for clinics, not integrated with adoption or commerce.

2.4 Gap Analysis

Users must juggle multiple systems: adoption portals, shopping sites, and clinic booking pages. This creates friction, increases drop-off, and disconnects post-adoption support. A unified platform can centralize discovery, application, product ordering, and veterinary bookings while providing administrators a normalized data model and role-based tools.

Features	Petfinder	Adopt-a-Pet	Local Shelter Site	E-commerce (Amazon/Daraz)	Proposed System
Adoption Listing	Yes	Yes	Yes		Yes
Rich Pet Profiles	Yes	Yes	Variable	No	Yes
Search & Filters	Yes	Yes	Variable	No	Yes
New Pet Alerts/Recommendations	Yes (alerts)	Yes (alerts)	No	No	Yes
Veterinary Booking	No	No	No	No	Yes
Pet Supplies/E-commerce	No	No	No	Yes	Yes

2.5 Summary

The ecosystem is fragmented. Our system addresses this by integrating core adoption workflows with supplies and healthcare under a single, scalable architecture.

Chapter 3

Research Methodology

3.1 Methodology/Requirement Analysis & Design specification

We used a **hybrid Waterfall + Iterative** model to balance documentation and flexibility. Each stage produced tangible artifacts (requirements, architecture/ER/use-case diagrams, implementation, tests, and evaluation) and allowed targeted refinements.

3.2 System Design (Architecture)

The platform adopts a **client–server** architecture: a **React** single-page application communicates with an **Express/Node.js** backend over **RESTful** APIs. **MySQL** stores persistent data using a normalized relational schema with foreign keys and indexes. This separation of concerns improves maintainability and enables independent scaling of tiers.
Core entities: User, Pet, Adoption Application, Product, Cart, Cart Item, Order, Order Item, and Appointment. Status fields capture lifecycle states (e.g., application pending/approved/rejected; order pending/paid/cancelled).

3.3 Functional Requirements

- **Authentication & Profiles:** register, login (JWT), view/update profile.
- **Pet Discovery:** browse, search, filter, sort; view details with images and attributes (type, breed, age, gender, description, status).
- **Adoption Applications:** submit, view status; admin review (approve/reject).
- **Commerce:** add to cart, update quantities, place orders; manage products and stocks (admin).
- **Appointments:** request veterinary appointments; store records for review.
- **Administration:** manage pets, applications, products, orders, users/roles; dashboards.

3.4 Non-Functional Requirements

- **Performance:** responsive UI; optimized DB queries and indexes.
- **Scalability:** stateless APIs; horizontal scaling of web tier; normalized schema.
- **Security:** bcrypt-hashed passwords; JWT; input validation; CORS; prepared statements.
- **Usability:** responsive, accessible UI with TailwindCSS.
- **Reliability:** referential integrity; defensive programming; clear status lifecycles.

3.5 Data Flow & Process Models

- **Adoption flow:** discover → view details → submit application → admin review → decision notification.

- **Order flow:** compose cart → checkout → create order & items → confirmation.

- **Appointment flow:** request slot → admin/vet review → confirmation. Sequence diagrams clarify client–server–database interactions for order placement and booking.

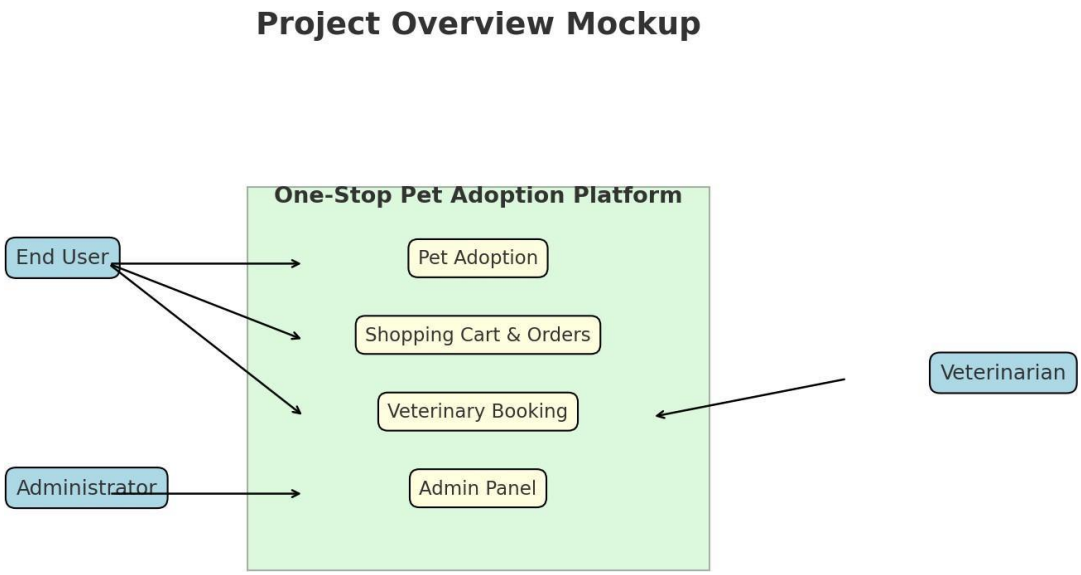

Figure 3.1: Project Overview Mockup

This platform connects **end users**, **admins**, and **veterinarians** in one place. Users can adopt pets, place orders, and book vet appointments, while admins manage everything through the panel. Vets confirm bookings, making the whole adoption and care process smooth and centralized.

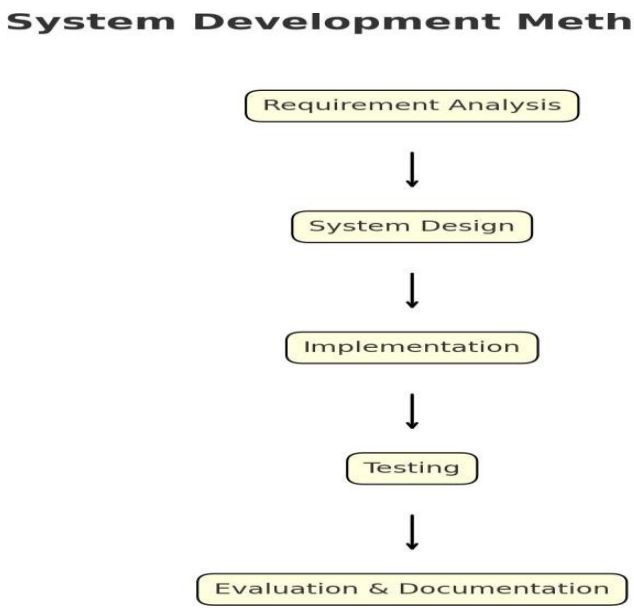

Figure 3.2: System Development Methodology of the Platform

This diagram shows the **system development lifecycle**, moving step by step from requirement analysis to design, implementation, testing, and finally evaluation with documentation

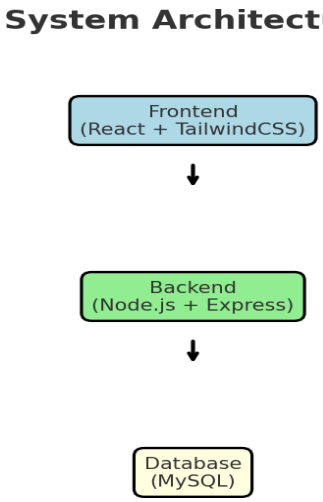

Figure 3.3: System Architecture Diagram of the Platform

This architecture shows a **React + TailwindCSS frontend**, a **Node.js + Express backend**, and a **MySQL database** working together to power the system

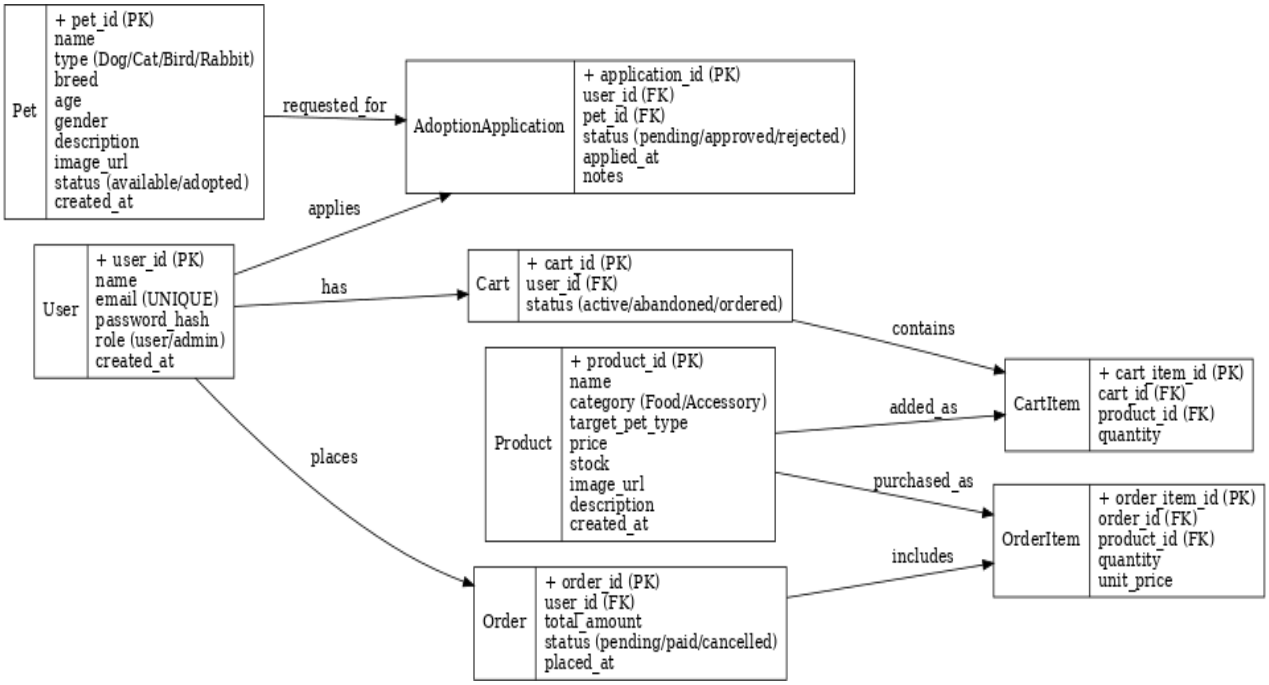

Figure 3.4: ER Diagram of the One-Stop Pet Adoption Platform

The system connects **users, pets, carts, products, and orders** into one flow. Users can apply for **pet adoption**, with applications tracked by status (pending, approved, rejected). They can also shop for **products**, adding items to a cart and placing orders, which record quantities, prices, and payment status. **Pets** have detailed profiles, while **products** store stock and category info. **Orders** and **order items** ensure transactions are clear and traceable. Altogether, the design keeps adoption and shopping processes well-organized for both users and admins.

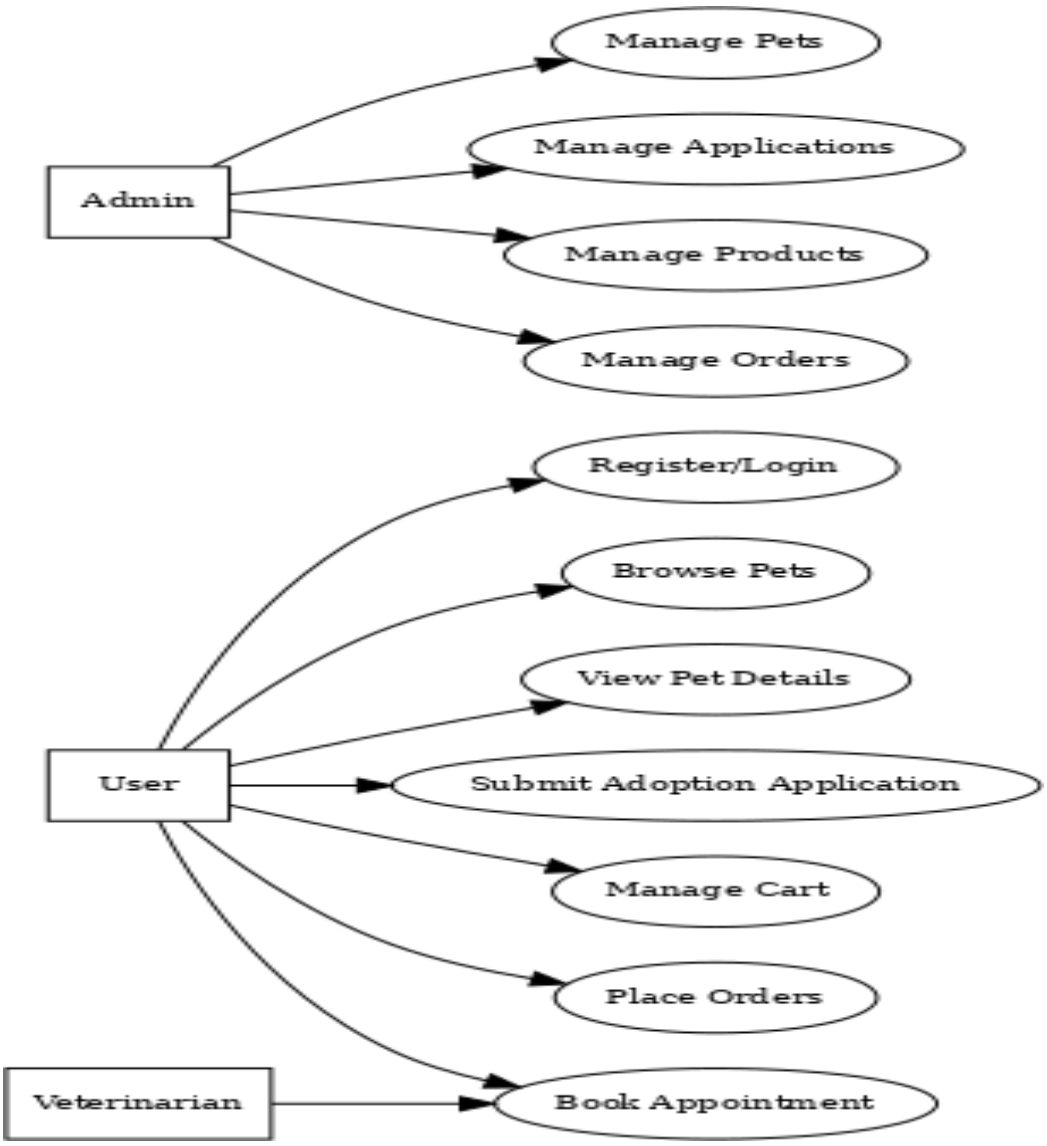

Figure 3.5: Use Case Diagram of the One-Stop Pet Adoption Platform

This diagram illustrates the main roles in the system. **Admins** manage pets, products, applications, and orders. **Users** can register, browse pets, submit adoption applications, manage carts, and place orders. **Veterinarians** focus on booking and managing appointments. It shows how each role has clear responsibilities within the platform.

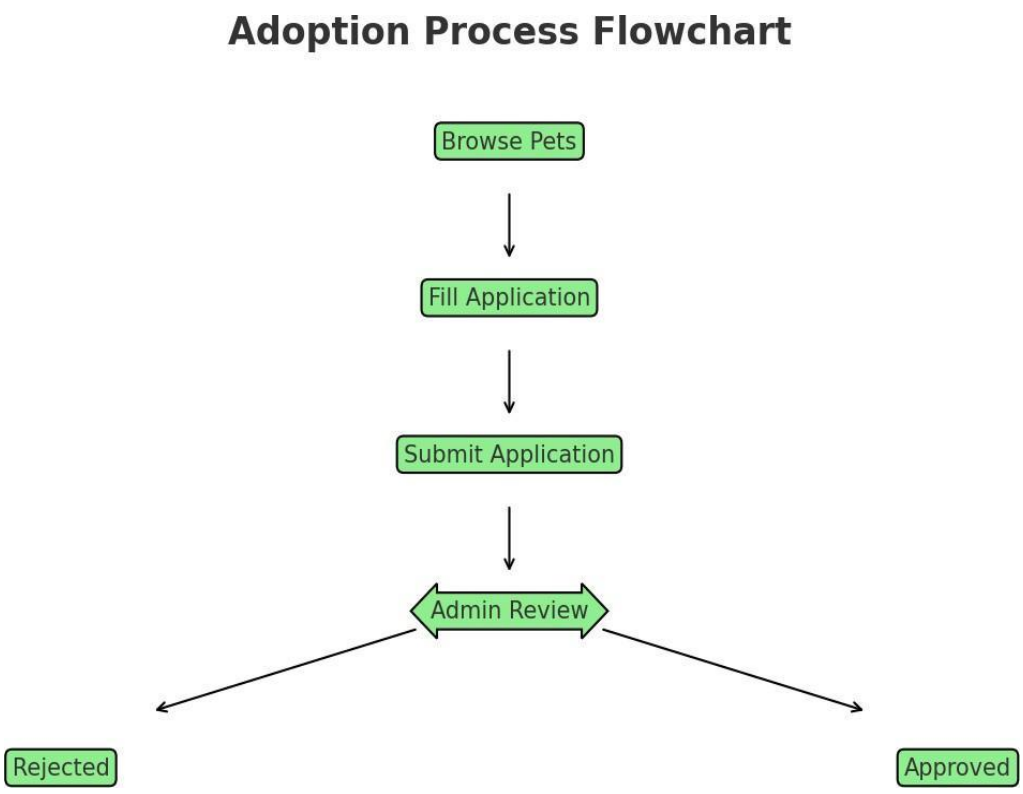

Figure 3.6: Flowchart of Adoption Process

The adoption process begins when a user **browses available pets** on the platform. Once they find a pet they are interested in, they **fill out an application form** with the necessary details. After completing it, they **submit the application** through the system. The request then goes to the **admin review stage**, where administrators carefully evaluate the application. Based on their decision, the outcome can go in two directions: the application is either **approved**, allowing the user to proceed with adoption, or it is **rejected**, ending the process. This flow ensures that every adoption is properly reviewed, making the process transparent and well-managed.

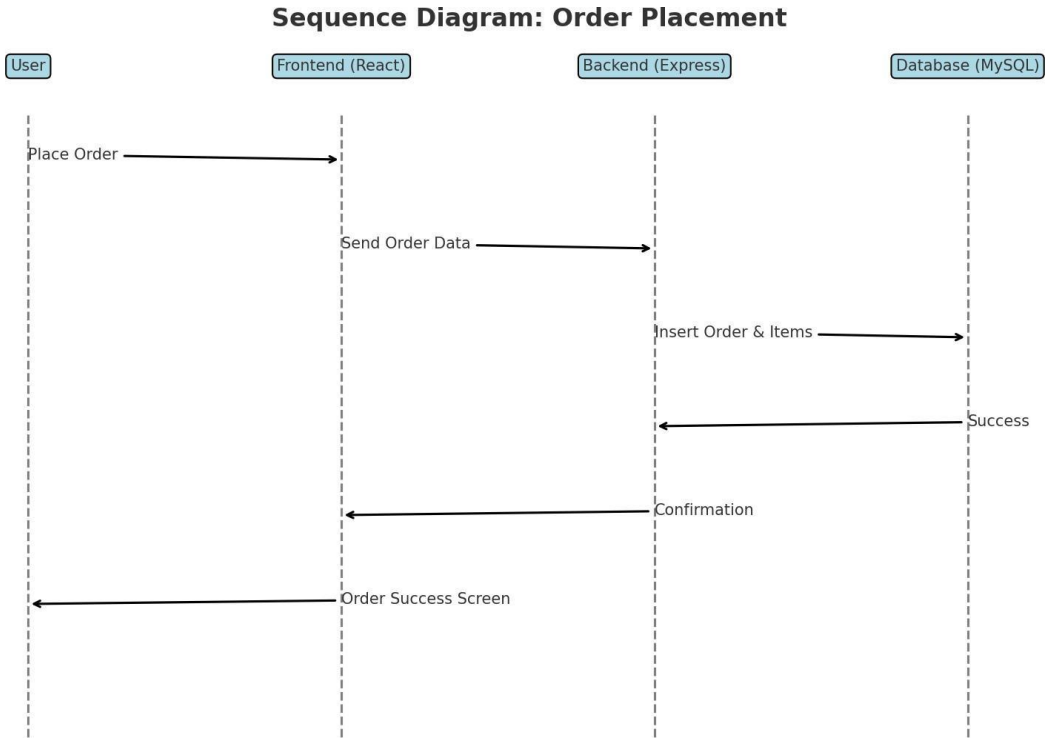

Figure 3.7: Sequence Diagram for Order Placement

The process starts when a **user places an order** through the frontend (React). The frontend then **sends the order data** to the backend (Express). The backend processes this by **inserting the order and items into the database (MySQL)**. Once the database confirms the operation was successful, the backend sends back a **success response**. The frontend then displays a **confirmation and success screen** to the user, completing the order flow.

3.6 UI Design

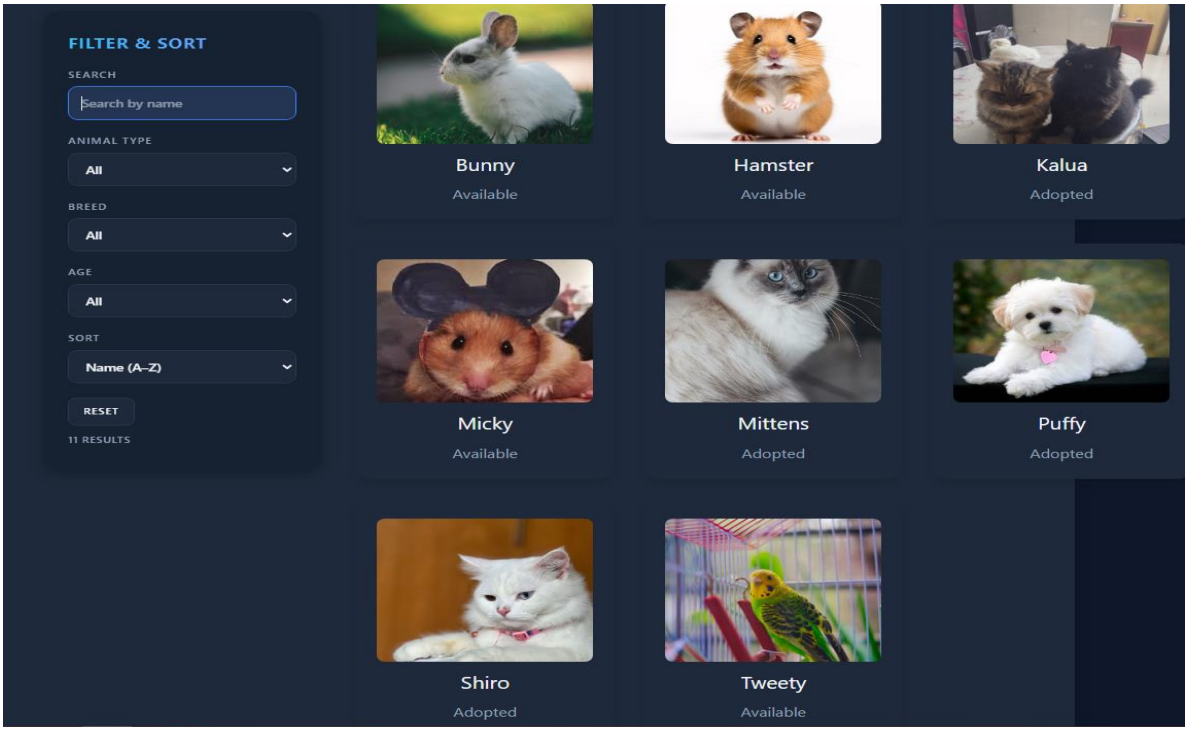

Figure B1: Screenshot: Homepage/Landing Page

This interface lets users **browse, filter, and sort pets** to see which are available for adoption and which have already been adopted.

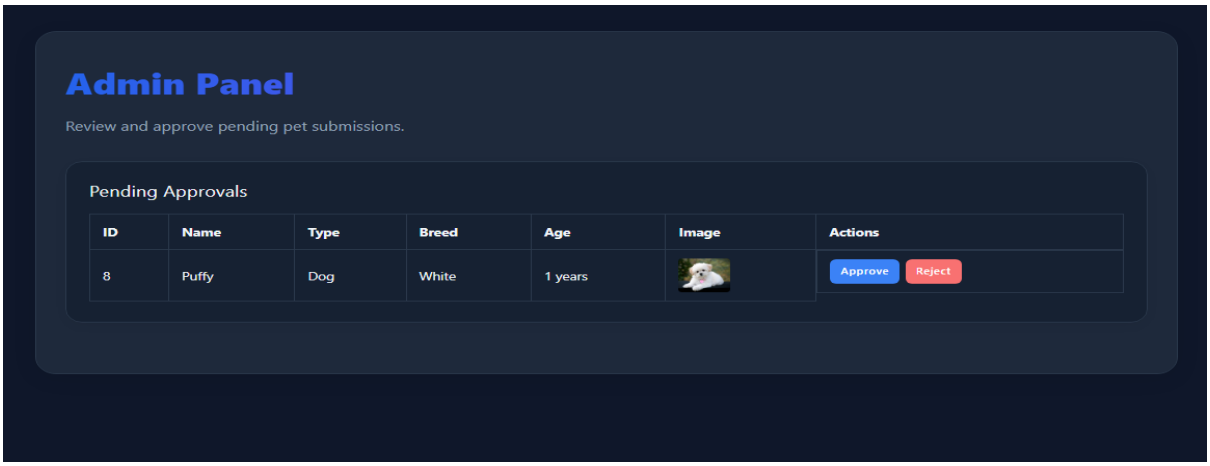

Figure B.2: Screenshot: Admin Panel

This Admin Panel allows administrators to review pet submissions and either approve or reject them before they are listed for adoption.

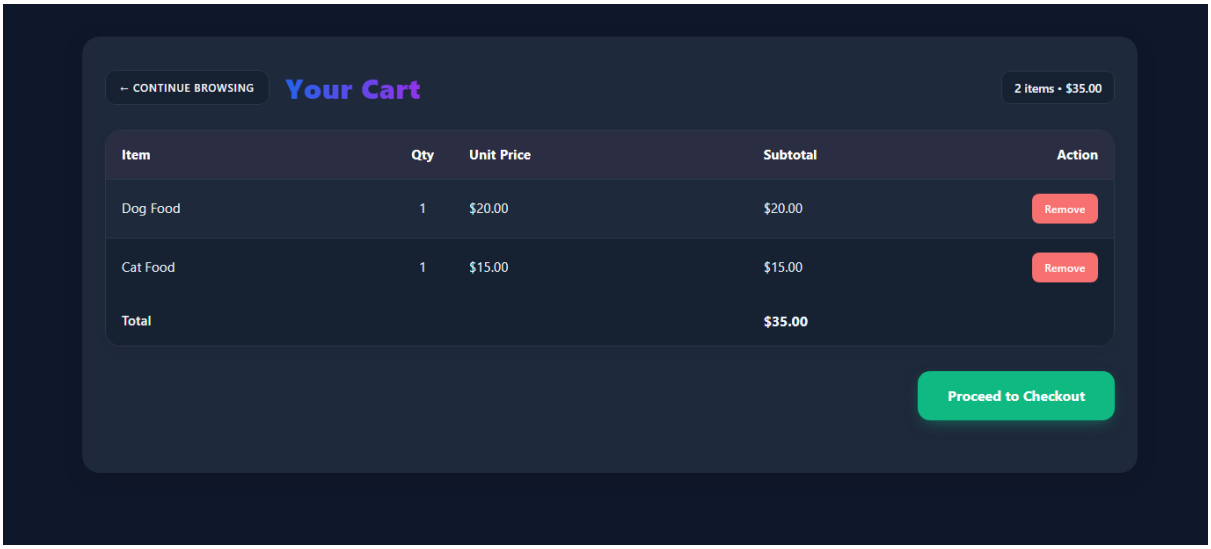

Figure B.3: Screenshot: Shopping Cart

This **Cart page** shows selected items with quantity, price, and total cost, allowing users to remove products or **proceed to checkout** for placing their order.

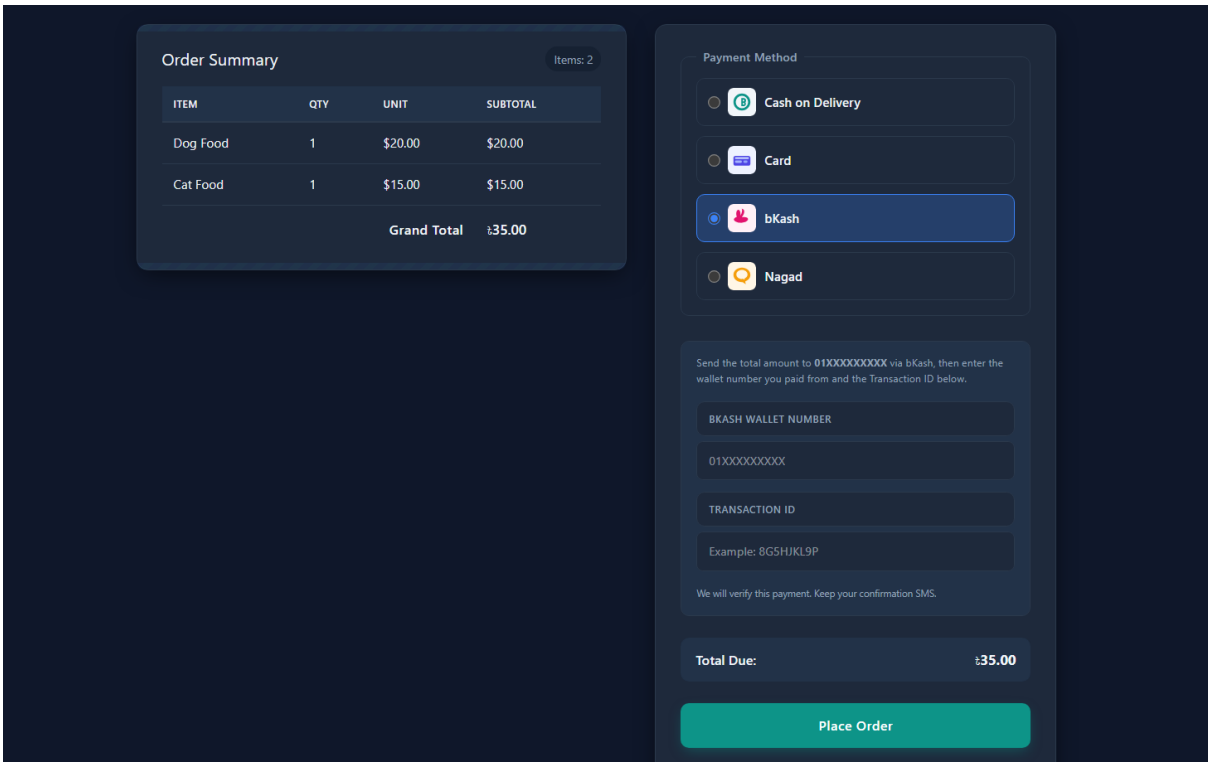

Figure B.4: Screenshot: Checkout Page

This checkout page gives users a clear order summary and multiple payment options like cash on delivery, card, bKash, or Nagad. Customers can enter payment details, review the total amount, and then place the order securely.

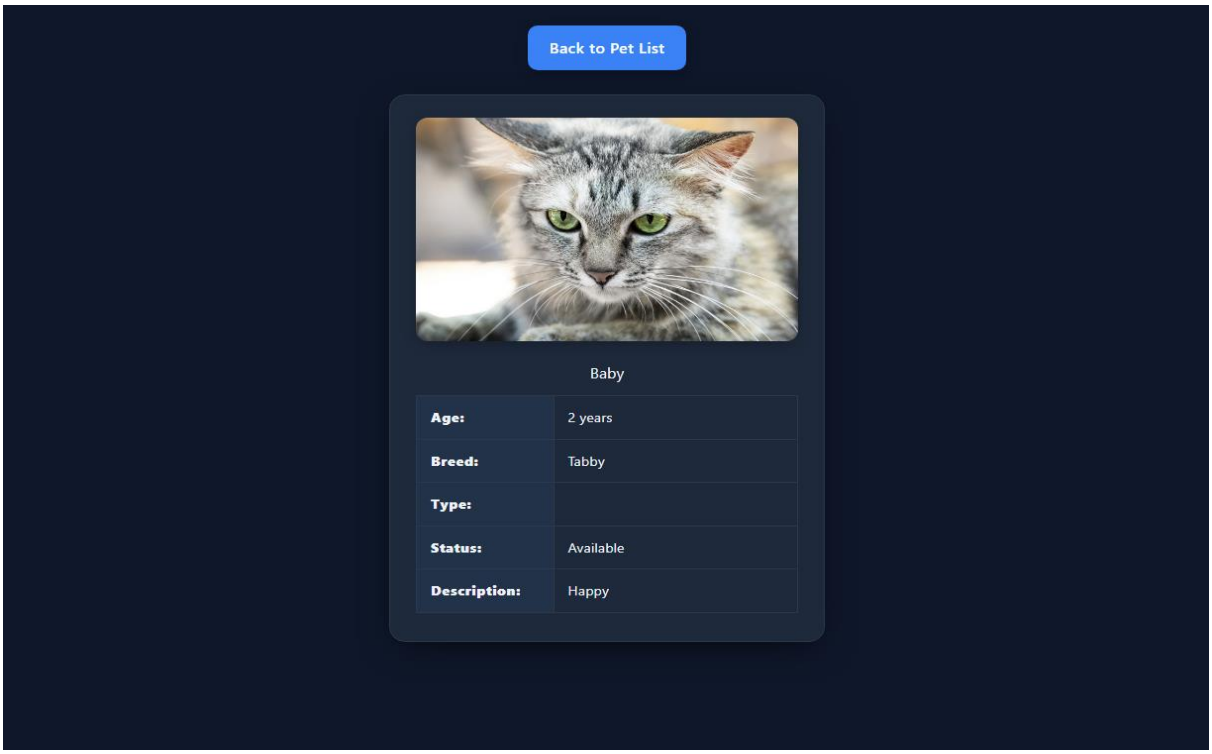

Figure B.5: Screenshot: Pet Details Page

3.7 Detailed Methodology & Design Choices

- **Frontend:** React + React Router + Vite for modular components and fast builds; TailwindCSS for utility-first responsive styling.

- **Backend:** Express/Node.js with middleware (CORS, cookie parsing); non-blocking I/O suits concurrent public usage.

- **Database:** MySQL with normalized schema, foreign keys, indexes on frequent filters (type, breed, status).

- **Security & Config:** bcrypt for hashing; JWT for stateless auth; environment variables via dotenv; multer for images (where applicable).

- **Testing:** unit tests (components/endpoints), integration (end-to-end flows), system/usability/performance/security validations.

3.8 Project Plan

3.8.1 Overview

The project runs from **Week 12 to Week 48** with overlapping workstreams to reduce risk and keep momentum. Streams reflect your modules: **Authentication**, **Pets & Adoption Applications**, **Products/Cart/Orders**, **Appointments**, **Admin**, and **Frontend UX**. Each phase below lists the **objective**, **key activities**, **deliverables**, and **exit criteria**. Milestones M1–M6 mark the review gates used for viva/demo readiness.

3.8.2 Timeline by Phase

Week 12–18 — Requirements & Design Stabilization (M1)

Objective: lock functional scope and system design.

Key activities:

- Capture/confirm functional & non-functional requirements; finalize user roles (user/admin) and permissions.
- Produce **architecture view** (React SPA + Express/Node.js + MySQL), **ER diagram**, **use-case diagram**, **adoption & checkout flows**, and **sequence** for order placement
- Draft API contracts (Auth, Pets, Applications, Products, Cart/Orders, Appointments) with request/response schemas and status codes.

Week 18–26 — Frontend Foundation & Auth (M2)

Objective: stand up the SPA, navigation, and authentication.

Key activities:

- Scaffold React app (Vite, React Router), route layout, protected routes.
- Build **Registration/Login** UIs with client-side validation; integrate **JWT** handling and guard admin routes.
- Implement **Pet listing & details** screens with filters/sort; adoption application form UI (client only).

Week 20–30 — Backend Core APIs & Data Layer (overlaps with Week 18–26) (M3)

Objective: **make core endpoints production-worthy.**

Key activities:

- Implement **Auth** (bcrypt hashing, JWT issuance/verification), user profile retrieval.
- Implement **Pets** (CRUD for admin; public browse/search/filter), and **Adoption Application** endpoints with status transitions (pending/approved/rejected).
- Finalize **MySQL schema** with foreign keys and indices (type, breed, status); add seed data and image field strategy.
- Input validation, error handling, and **CORS** configuration.

Week 26–34 — Commerce: Cart & Orders + Admin Consoles (M4)

Objective: complete shopping workflow and administrative tools.

Key activities:

- Frontend **Cart** state (add/update/remove), **Checkout** UI; server-side **Orders** (create order + line items)
- **Products** management (price, stock) and **Order** tracking for admins.
- Payment gateway stays out-of-scope for this phase; leave a payment stub and status transitions.

Week 30–36 — Appointments & Post-Adoption Support

Objective: enable users to book **veterinary appointments**.

Key activities:

- Appointment request UI with slot selection; endpoints to create/list/review appointments.
- Basic status lifecycle (requested/confirmed/cancelled); notifications stub (e.g., toast/email placeholder).

Week 26–34 — Test Wave 1; Week 34–38 — Usability & Performance (M5)

Objective: validate correctness, security, and UX

Key activities:

- **Unit & integration tests** for components & endpoints (status/payload/DB effects).
- **Scenario tests:** adoption pipeline; cart→order; appointment request→confirmation.
- **Security checks:** bcrypt configuration, JWT expiry/refresh rules, CORS tightening, input sanitization, prepared statements.

- **Performance:** verify API latency under typical load; index tuning on frequent queries.

- **Usability:** refine filters, forms, error messaging, and mobile breakpoints.

Week 34–48 — Documentation, Figures & Submission (M6)

Objective: produce the final FYDP report and demo package.

Key activities:

- Capture **screenshots** (Homepage, Listing, Details, Cart, Checkout, Admin) and export **diagrams** (ERD, Use-case, Architecture, Flows, Sequence).

- Author chapters (Implementation, Results, Standards, Impact, Management, EP/KP/EA mappings), format **ToC/LoF/LoT**, references (IEEE).

- Polish wording; proofread; ensure consistency of terminology and figure/table captions.

3.9 Task Allocation

Member A — Adnan (Frontend & UX lead)

- React (Vite, React Router), **pet discovery UI** (search, filters, sort), **pet profile view**, adoption form UI.
- **Cart & Checkout** pages (totals, validation, order review), JWT handling on client, protected routes.
- **UI/UX polish & responsiveness** (TailwindCSS), accessibility checks.
- **Frontend testing**: component/unit tests; integration tests for cart totals, filter logic, and application submission.
- Contributes to screenshots and Appendix B, and Chapter 4 write-ups.

Member B — Tahmid (Backend & Data lead)

- **Express/Node** REST APIs: Auth (bcrypt/JWT), Pets CRUD, AdoptionApplication review (approve/reject), Products & Orders, Appointments.
- **MySQL schema design** (User, Pet, AdoptionApplication, Product, Cart/CartItem, Order/OrderItem, Appointment), FKs and **indices** (type, breed, status).
- **Security**: input validation, prepared statements, CORS; role-based controls (isAdmin).
- **Backend testing**: endpoint status/payloads, DB effects; seed scripts.
- Contributes to ERD/Use Case diagrams (Appendix A) and Chapter 5 standards/mappings.

Shared work (both members)

- Requirements & design artifacts (architecture diagram, use-case, flows).
- End-to-end scenario tests (adoption pipeline; cart→order).
- Documentation pass (proofreading, figures, ToC/LoF/LoT updates).
- Presentation rehearsal and submission checklist.

3.10 Summary

The project approach was a hybrid model of the structure planning approach integrated with flexibility of iterative development. The design process that started with the requirements gathering involved the architectural diagrams, ER diagrams, and process models to ensure that the workflows were defined appropriately. The implementation was structured into separate layers as frontend, backend, and database using technologies that were chosen on the basis of their scalability and efficiency.

Through various types of testing, such as unit, integration, system, and usability testing, validation was done to ensure that the system was working well on an individual module and end-to-end workflow. The staged methodology was used to control risks and make sure that every milestone was made. This moderate approach ensured a solid grounding of providing a platform which is feasible, safe and capable of further development.

Chapter 4
Implementation and Results

4.1 Environment Setup

The environment was designed well so that it could be stable, have good performance and be scalable. React was implemented on the frontend alongside Vite and React Router, which allows modular development, easy routing, and speedy builds. TailwindCSS has been used to ensure that it is responsive to use across devices as well as enhance accessibility. The back-end was developed using Node.js (Express), with the following middleware being used: cors to allow secure cross-origin access and cookie-parser to handle the sessions. Dotenv was used to manage configuration values and provide some flexibility on the various environments.

The persistent storage layer of the system was based on MySQL relational database. The schema was also normalized using the foreign key, the indexed columns, especially on the most commonly accessed columns like type, breed and status. This was to guarantee rapid query performance and overall responsiveness. Security was designed through consideration of the initial stages that included the use of bcrypt password hashing, JWT tokens to perform stateless authentication, input validation, and CORS limitation. Also, Multer was incorporated to facilitate the effective management of image uploads of pet profiles and products. The combination of this environment provided a solid ground that is able to support the existing functionality, and at the same time, enable future expansion.

4.2 Testing & Evaluation / Performance

Functionality and performance were tested on various levels to ensure the validation of the tests. Unit testing was focused on specific components and API endpoints, and each of the building blocks was tested to operate as expected. Integration testing was aimed at testing end-to-end workflows like adoption applications and checkout procedures and ensuring that the modules communicated properly.

Extended system testing was used to simulate actual user steps such as discovery to application submission or cart creation to order confirmation. Usability tests were conducted to assess the ease of navigation, responsiveness of the layout and accessibility across different devices and resulted in the improvement of the error messages and mobile adaptations.

Performance testing indicated that the system was responsive to UI rendering and could acceptably render API latency in general use. Lastly, security test verification proved that the authentication, password hashing, sanitization of inputs, and protections only provided by OWASP were functional, which minimized the chances of data breaches or unauthorized access.

4.3 Results & Discussion

These findings had shown that a combined attitude towards adoption and aftercare significantly enhanced the total experience. The adopters enjoyed discovery, application, shopping, and veterinary booking within a single workflow, thus, causing fewer frictions and making it more convenient. Simultaneously, administrators were provided with the clear tools of decision making and working dashboards, which enabled them to control processes more efficiently.

This single platform was both more efficient and visible than the fragmented solutions available before the new product. The preliminary experiments showed that the workflow consistency, the user satisfaction rate, and the administrative control were improved in a measurable way, which validates the feasibility of the design.

4.4 Summary

The implementation process was able to confirm the soundness of the system architecture and show the effectiveness of the selected technologies. The development environment was carefully configured with the help of systematic testing, guaranteeing that every single detail (frontend user interface to backend API and database manipulation) performed its specific task and functions in harmony with the rest of the system.

The findings demonstrated the great enhancement in workflow consistency, transparency of decision making, and user engagement in contrast to fragmented solutions. Although such drawbacks as the absence of payment integration and mass implementation are still present, the platform manages to demonstrate its viability and create a channel through which positive changes can occur in the future.

Chapter 5

Engineering Standards and Design Challenges

5.1 Compliance with the Standards

5.1.1 Software Standards

- **Security practices:** OWASP-aligned considerations for authentication, session handling, and input validation.

- **API design:** RESTful conventions, consistent status codes, and documented request/response shapes.

- **Code quality:** ESLint/Prettier (frontend) and conventional Node.js structure (backend).

5.1.2 Hardware Standards

As a web application, no specialized hardware standards are required beyond typical server specifications. Hosting options include shared VPS, managed PaaS, or container-based cloud deployments.

5.1.3 Communication Standards

- **HTTP/HTTPS** for client–server communication.
- **JWT** for authentication tokens in headers.
- **CORS** for controlled cross-origin access

5.2 Impact on Society, Environment and Sustainability

5.2.1 Impact on Life

Simplifies adoption and supports responsible ownership, contributing to better animal welfare outcomes.

5.2.2 Impact on Society & Environment

Encourages adoption over commercial breeding, potentially reducing stray populations and shelter overcrowding; promotes a culture of compassion and accountability.

5.2.3 Ethical Aspects

Focus on data privacy and secure handling of user/application data; fair access; transparent criteria for application approval.

5.2.4 Sustainability Plan

Modular architecture, cloud-ready deployment, and scalable storage for images; roadmap includes automation for backups, monitoring, and cost optimization.

5.3 Project Management and Financial Analysis

- **Costs:** hosting, domain, storage/CDN for media, transactional email/SMS, and monitoring.
- **Alternatives:** low-cost VPS vs. managed cloud; open-source tools vs. paid services.
- **Revenue models:** donations and sponsorships; affiliate links for supplies; optional premium dashboards for shelters/clinics; advertising with strict privacy guardrails.

5.4 Complex Engineering Problem Mapping (EP1–EP7)

Code	Dimension	Rationale in this Project
EP1	Depth of Knowledge	Combines web engineering, security, and data modeling to deliver integrated workflows.
EP2	Range of Conflicting Requirements	Balances usability, security, and performance for public-facing adoption flows.
EP3	Depth of Analysis	Normalization, indexing, and API design to ensure data integrity and responsiveness.
EP4	Familiarity of Issues	Mix of standard web issues and domain-specific constraints (adoption vetting).
EP5	Applicable Codes	Follows web best practices and privacy considerations; no special regulatory codes.
EP6	Stakeholder Involvement	End users, shelter admins, and vets with distinct goals and access levels.
EP7	Interdependence	Interconnected modules (applications, orders, appointments) require coordinated design.

Table 5.1: Mapping with Complex Engineering Problem.

5.5 Knowledge Profile Mapping (K1–K8)

Code	Knowledge Area	Application to Project
K3	Engineering Fundamentals	Relational schema design, indexing, and API structuring.
K4	Specialist Knowledge	Web frameworks (React, Node.js), authentication (JWT), hashing (bcrypt).
K5	Engineering Design	UI/UX with React + Tailwind, modular components, and navigational flow.
K6	Engineering Practice	Secure coding, defensive programming, CI-ready structure.
K8	Research Literature	Survey and comparative analysis of existing platforms and gaps.

Table 5.2: Mapping with Knowledge Profile

5.6 Engineering Activities Mapping (EA1–EA5)

Code	Activity	Rationale
EA1	Range of Resources	Open-source frameworks, relational DBMS, and cloud-ready deployment.
EA2	Level of Interaction	Multi-role system (users, admins, vets) with coordinated interactions.
EA3	Innovation	Unified adoption+commerce+healthcare workflow in one platform.
EA4	Societal/Environmental Consequences	Potential reduction in strays and improved welfare.
EA5	Familiarity	Common web stack with domain-specific review and approval flows.

Table 5.3: Mapping with Complex Engineering Activities

5.7 Summary

Standards compliance, impact analysis, and engineering mappings demonstrate rigor and responsible design aligned with the project's goals.

Chapter 6

Conclusion

6.1 Summary

This project demonstrates that the current web technologies can be implemented to address some significant challenges in the society. The system was able to resolve the inefficiency of the fragmented services by integrating the pet supplies and veterinary booking services into one digital platform; adoption discovery. To adopters, this made the experience more fluid and reliable, whereas to administrators, it provided them with the means to monitor and manage in addition to a clearer way of making decisions. In general, the project confirmed the possibility of such an integrated model and emphasized its perspective on a more significant benefit to society.

6.2 Limitation

Regardless of these successes, there are a number of constraints. The new version is still not ready to assist any real time payment integration thus limiting its capacity to deal with real transactions. Though useful, veterinary bookings do not have any connections with the outside clinic systems, which restricts their usefulness in the real-world setting. Besides this, there has not been any large-scale stress testing, which casts doubts on how the platform will respond to such heavy traffic. Resource availability also limited hosting and integration with more advanced third party services.

6.3 Future Work

The next feature of development will be the work on overcoming these shortcomings and increasing the scope of the platform. Real-life transactions could be made possible by the introduction of secure payment gateways like Stripe or bKash. This role-based access to the veterinarians, shelter managers, and volunteers should be extended to enhance the administrative ecosystem. A react-native mobile application would also make it more accessible and available to more people, whereas the use of the cloud would offer elasticity, resilience, and more effective media storage. Medical records, reminding, and partner clinic cooperation also could be included in the veterinary module. Lastly, recommendation features powered by AI might be implemented to recommend pets to adopters to enhance the success rate of the matches and the successful adoption process.

References (IEEE style)

[1] “React Documentation,” https://react.dev/
[2] “Express.js Documentation,” https://expressjs.com/
[3] “MySQL Documentation,” https://dev.mysql.com/doc/
[4] “TailwindCSS Documentation,” https://tailwindcss.com/docs
[5] “JSON Web Tokens,” https://jwt.io/
[6] “bcrypt Library,” https://github.com/pyca/bcrypt

Verified 17.9.25

paper

ORIGINALITY REPORT

8%	6%	4%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	2%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
3	arxiv.org Internet Source	1%
4	www.arxiv-vanity.com Internet Source	<1%
5	Submitted to United International University Student Paper	<1%
6	Jun Bai, Chuantao Yin, Jianfei Zhang, Yanmeng Wang, Yi Dong, Wenge Rong, Zhang Xiong. "Adversarial Knowledge Distillation Based Biomedical Factoid Question Answering", IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2022 Publication	<1%
7	Yu-Jia Zhou, Jing Yao, Zhi-Cheng Dou, Ledell Wu, Ji-Rong Wen. "DynamicRetriever: A Pre-trained Model-based IR System Without an Explicit Index", Machine Intelligence Research, 2023 Publication	<1%
8	Submitted to University of York Student Paper	<1%

22% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

18 AI-generated only 22%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.