Project report

on

“Pet Heaven”

DAV UNIVERSITY

For the partial fulfilment of the requirement

For the award of degree of

“BACHELOR OF COMPUTER APPLICATIONS”

Submitted to:

(Name)

Department:

BCA (4th sem)

Submitted from:

Jasmeet kaur

Decalaration

I, **Jasmeet kaur**, hereby declare that the project report entitled **"Pet Heaven"** submitted by me to DAV University for the degree of **Bachelor of Computer Science and Applications (Semester IV)** is an original piece of work and has not been submitted to any other university for the award of any degree.

I also undertake that any quotation or idea from the published or unpublished work of another person has been duly acknowledged in this project report.

Place: DAV University

Signature of the Student: _____

Acknowledgement

I have received invaluable support and assistance in preparing this project. While a brief acknowledgment here in no way writes them off, it is a small courtesy whose sentiments are sincere. I would like to extend my sincere thanks to all the individuals who helped me in different ways with the development of this project report. Without their continuous support and guidance, the completion of my project would have been impossible.

I extend my gratitude to Mr.Arvind Mahindru, Head of the Department, School of Computational Science, for his continuous support and encouragement throughout the degree. I also wish to express my most sincere thanks to my supervisor [Supervisor's Name] for her invaluable guidance, advice, support, and encouragement. I will carry out her guidance throughout my life.

I shall be falling behind in my duties if I do not place on record my sincere thanks to all those writers and authors from whose writings I have benefited.

I would like to express my special gratitude and thanks to the O7 Services Staff for giving me such attention and time.

In the end, I would also like to mention that this project would not have been possible but for the continuous support and guidance of my parents, who gave me the strength and will to succeed. My thanks and appreciations also extend to the people who have willingly helped me out with their abilities.

Jasmeet kaur

CHAPTER 1 – INTRODUCTION

1.1 INTRODUCTION:

In today's world, where compassion and digital innovation go hand-in-hand, **pet adoption** has emerged as a beautiful act of kindness. Many pets, whether rescued, abandoned, or born in shelters, deserve a safe and loving home. However, traditional pet adoption methods are often slow, manual, and lack accessibility.

To solve these issues, we present **"Pet Heaven"**, a modern pet adoption web-based system that connects users with verified NGOs and animal shelters. This platform acts as a **bridge** between potential pet parents and organizations working for animal welfare.

"Pet Heaven" is not just a pet adoption portal; it is an effort to make pet finding, bonding, and adopting smoother and more responsible. The platform provides 3 role-based modules:

- **Admin**
- **User**
- **NGO / Shelter Partner**

Each module has specific features to simplify operations, ensure security, and support a real-time, transparent, and emotionally fulfilling adoption process.

This system also aims to raise awareness about pet welfare, promote adoption over buying, and encourage responsible animal care through technology.

1.2 PROJECT DESCRIPTION

The **Pet Heaven** website is a full-stack web application built using **React, Firebase, HTML, CSS, and Bootstrap**. It supports user registration, browsing of pets by category (dogs, cats, rabbits, etc.), viewing pet details, applying for adoption, and monitoring adoption history. NGOs can list pets for adoption and manage applications.

MODULES OF THE PROJECT:

1. ADMIN MODULE:

The **Admin** is the central controller of the Pet Heaven platform. The admin panel ensures smooth functioning by managing users, NGOs, and pet listings. It helps maintain data security, monitors adoption activity, and ensures only verified NGOs and pet details are available on the platform.

- **Admin Login:** Secure access to the backend dashboard.
- **Manage Users and NGOs:** View, verify, or block suspicious users/NGOs.
- **Approve Pet Listings:** Check pet details uploaded by NGOs before they go live.
- **View Adoption Requests:** Monitor all adoption activity to avoid abuse or fake entries.
- **Remove False Listings:** Delete spam, irrelevant, or duplicate pet profiles.
- **Platform Control:** Handle complaints or reports from users and NGOs.

2. USER MODULE:

The **User Module** is designed for individuals or families who wish to adopt a pet. It provides a user-friendly interface that allows browsing, shortlisting, and applying for adoption of pets.

- **Registration/Login:** Users can create their accounts securely.
- **Browse Pets:** Explore pets by type (dog, cat, etc.), breed, age, or location.
- **View Pet Details:** Check pet photos, vaccination info, medical history, and NGO contact.
- **Apply for Adoption:** Fill an adoption form and submit it to the NGO.
- **Track Application Status:** Get real-time updates on approval or rejection.
- **Provide Feedback:** Rate NGOs or adoption experience.
- **Manage Profile:** Update personal details, past adoptions, etc.

3. NGO MODULE:

This module is for **NGOs, animal shelters, or foster homes** that rescue and care for animals. It enables them to upload pet profiles, manage requests, and find loving homes for animals.

- **Register/Login as NGO:** Authenticate via email or admin approval.
- **List Pets:** Add new pets with name, age, category, medical and vaccination status.
- **Upload Documents:** Attach medical records, images, and care history.
- **Manage Applications:** View applications from users, accept/reject with comments.
- **Track Adoption History:** Keep a record of which pets have been adopted, pending, or returned.
- **Edit or Remove Listings:** Update pet status once adopted or unavailable.

1.3 PROBLEM DEFINITION:

Current pet adoption processes, especially in smaller cities and NGOs, are often offline and fragmented. These challenges include:

- Manual and time-consuming processes for registration and approvals.
- Lack of real-time updates on pet availability or adoption status.
- NGOs struggling to maintain digital pet profiles and medical records.
- Difficulty in verifying users or screening genuine adopters.
- No centralized system to track feedback, history, and communication.

These problems reduce the adoption rate, increase the burden on shelters, and lead to poor user experience.

1.4 EXISTING SYSTEM:

The traditional pet adoption system is primarily manual, relying on paperwork, physical visits, or random WhatsApp/Facebook posts by NGOs. These systems face:

- Poor visibility for pets available for adoption.

- No structured way to apply or manage applications.
- No secure database of adopters or shelters.
- High chances of losing track of records or adopting pets into unsafe homes.

1.5 PROPOSED SYSTEM:

Pet Heaven aims to digitalize and simplify the entire adoption journey. It offers a real-time, secure, and responsive platform for users, NGOs, and administrators to manage everything in one place.

Key features of the **proposed system**:

➢ **Admin:**

- Centralized monitoring and approval of users, NGOs, and pet listings.
- Ensures all adoptions go through verified accounts.

➢ **Users:**

- Explore pets by breed, location, size, age.
- Apply online, track status, communicate with NGOs.
- Access guidance on adoption and care.

➢ **NGOs:**

- Add new pets with details, vaccination proof, and photos.
- Accept/reject adoption requests with reasons.
- Maintain real-time pet database with adoption status.

This system not only solves the major problems of the existing manual system but also builds **trust**, **speed**, and **transparency** in the adoption ecosystem.

CHAPTER 2 – HARDWARE AND SOFTWARE REQUIREMENTS

For smooth functioning and development of **Pet Heaven**, both hardware and software components must meet specific standards. This chapter outlines the minimum configuration required to develop, test, and run the system efficiently.

HARDWARE REQUIREMENTS:

- Processor: Intel Core i3 or higher (e.g., Intel Core i5, Intel Core i7)
- Ram: 8 GB
- SSD:256GB

SOFTWARE REQUIREMENTS:

- Front End: HTML, CSS, Bootstrap, JavaScript, ECMA Script, React JS
- DB Tool: Firebase Fire store
- Browser: Mozilla Firefox/Chrome/Edge or any other relevant browser
- OS: Windows operating system/Linux
- Text Editor: Visual Studio

Chapter 3 FEASIBILITY STUDY

Before developing a full-fledged web application like **Pet Heaven**, it is important to evaluate the **feasibility** of the system from different perspectives. This ensures that the proposed system is not only achievable but also sustainable, cost-effective, and user-friendly.

Pet Heaven is built with the objective of improving the adoption rate of pets by providing a secure, fast, and centralized digital solution for animal lovers and NGOs.

3.1 ECONOMIC FEASIBILITY:

Economic feasibility means checking whether the project is **cost-effective** or not. It helps us understand if the system can be built and maintained without spending too much money.

In the case of **Pet Heaven**, there is no need to purchase expensive software or servers. It uses **free and open-source tools** like HTML, CSS, JavaScript, React, and Firebase. Firebase offers a **free tier** for hosting, authentication, and database services which is enough for small and medium-sized projects.

Since everything runs online, even the NGOs and users can access the platform using a regular computer or mobile phone with internet. Therefore, the overall development and running cost is **very low**, and it brings **high value** to the community by simplifying the adoption process.

3.2 TECHNICAL FEASIBILITY:

Technical feasibility checks whether the required **technology, tools, and skills** are available to complete the project.

For **Pet Heaven**, all the technologies used like **ReactJS for frontend**, **Firebase for backend**, and **Visual Studio Code as code editor** are easily available and well-documented. These technologies are also **scalable**, meaning the system can grow in the future without starting over.

Also, all team members and developers are familiar with these tools, which reduces technical errors. The real-time features of Firebase allow instant updates, making it perfect for a live adoption tracking system.

Thus, from a technical point of view, the project is completely **feasible and reliable**.

3.3 BEHAVIOURAL FEASIBILITY:

Behavioral feasibility checks whether **people will accept and use** the new system or not.

Pet Heaven is designed with a **simple and user-friendly interface**, so that people from all backgrounds (even without technical knowledge) can use it easily. Users can find pets, apply for adoption, and track their status — all from their home.

NGOs will also find it useful as it reduces paperwork and allows them to reach more adopters digitally. The admin has full control and can monitor everything easily.

Since the platform helps all three groups (users, NGOs, and admin), the chances of acceptance and satisfaction are **very high**.

3.4 METHODOLOGY / PLANNING OF WORK:

The development of Pet Heaven follows a systematic approach:

1. **Requirement Gathering** – Understanding adoption flow, NGO needs, and technical tools.
2. **System Design** – Planning modules, UI layout, and database structure.
3. **Development** – Using React for frontend and Firebase for backend.
4. **Testing** – Validating user registration, listing, adoption process, etc.
5. **Deployment** – Hosting the application using Firebase Hosting.
6. **Feedback & Maintenance** – Collecting feedback from users/NGOs for further improvement.

3.5 USE CASE DIAGRAM:

A **Use Case Diagram** is a visual representation of how different users interact with a system and what functions or actions they can perform. It helps in understanding the system's **functional requirements** and user roles clearly.

In **Pet Heaven**, there are three main types of users: **Admin**, **User**, and **NGO**. Each of these roles has different responsibilities and access within the system.

- The **Admin** manages the entire platform. Admin can log in securely, verify or block NGOs and users, approve pet listings, monitor adoption activity, and remove inappropriate content.
- The **User** is someone who wants to adopt a pet. Users can register/login, browse pets, view pet details, apply for adoption, track application status, and give feedback or ratings.
- The **NGO** or shelter is responsible for adding pets to the system. NGOs can log in, add or update pet profiles, upload vaccination or medical documents, manage adoption applications, and track which pets are adopted.

The use case diagram helps in **visualizing all the actions** each role can perform. It also ensures that all the necessary features are included during development. This diagram plays an important role in planning the structure of the website and assigning roles and permissions correctly.

CHAPTER 4 – SYSTEM ANALYSIS

4.1 DATA ANALYSIS:

Before developing the **Pet Heaven** pet adoption platform, a detailed analysis of the current adoption process was conducted. In most cases, adoption is done manually—through phone calls, social media posts, or in-person visits to shelters. This often leads to **confusion**, **duplicate information**, **delayed responses**, and even **incomplete tracking** of adopted animals.

There is no centralized system where users can see available pets in real-time, check their health or vaccination status, or apply online with clear tracking. Similarly, NGOs have no simple way to manage pet listings, documents, or communicate with adopters without switching between multiple platforms (WhatsApp, Email, Excel, etc.).

With this background, Pet Heaven was designed to solve these problems by **digitizing** and **centralizing** the pet adoption process. It stores and organizes data in a structured format using Firebase, ensuring that pet listings, user information, application records, and health documents are properly maintained and can be accessed securely and instantly.

4.4 TYPES OF ANALYSIS:

To ensure the success of **Pet Heaven**, it is essential to analyze the system from different angles. The three main types of analysis that help evaluate the practicality and reliability of the system are:

- **Operational Analysis:**It focuses on how well the system integrates into the day-to-day workflows of users, NGOs, and admins. The traditional adoption process is often slow, paper-based, and scattered across multiple communication channels like calls, emails, and WhatsApp. Pet Heaven eliminates this mess by offering a single online platform where users can apply for pet adoption, NGOs can manage pet records, and admins can supervise everything from a central dashboard. This results in better time management, improved coordination, and reduced errors, making operations smooth and efficient.
- **Technical Analysis:** It evaluates whether the required technology and tools are available, and if the system can be built and maintained with those technologies. Pet Heaven is developed using **React JS** for frontend and **Firebase** for backend services like authentication, database, and hosting. These technologies are open-source, widely supported, and ideal for web applications. The system runs well on modern browsers and devices, requires no physical servers, and can scale as the user base grows. From a technical point of view, the system is strong, secure, and highly feasible.
- **Economic Analysis**: It deals with cost-effectiveness. Pet Heaven is developed using **free or low-cost tools**, such as Visual Studio Code, Firebase (free tier), and open-source

libraries. There’s no need for expensive infrastructure, licenses, or hardware. NGOs and users can use the platform via any smartphone or computer with internet access. In the long run, it reduces operational costs by removing paperwork, manual tracking, and repetitive communication. This makes the system economically practical and affordable.

CHAPTER 5

Technology used

5.1 HTML

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content

HyperText Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance. HTML elements are the building blocks of HTML pages.

Advantages:

- HTML helps to build structure of a website and is a widely used Markup language.
- It is easy to learn.
- Every browser supports HTML Language.
- HTML is light weighted and fast to load.

Disadvantages:

- It cannot produce dynamic output alone, since it’s a static language.
- Making the structure of HTML documents becomes tough to understand.
- Errors can be costly.
- It is the time consuming as the time it consumes to maintain on the color scheme of a
- page and to make lists, tables and forms.

5.2 CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other
- media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Advantages of CSS:

- CSS plays an important role, by using CSS you simply got to specify a repeated style for element once & use it multiple times as because CSS will automatically apply the required styles.
- The main advantage of CSS is that style is applied consistently across variety of sites. One instruction can control several areas which is advantageous.
- Web designers needs to use few lines of programming for every page improving site speed.
- Cascading sheet not only simplifies website development, but also simplifies the maintenance as a change of one line of code affects the whole web site and maintenance time..

Disadvantages of CSS:

- Browser compatibility (some styles sheet are supported and some are not).
- CSS works differently on different browsers. IE and Opera supports CSS as different
- logic.
- There might be cross-browser issues while using CSS.
- There are multiple levels which creates confusion for non-developers and beginners.

5.3 JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming

language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity. JavaScript enables developers to create complete solutions for various problems. JavaScript can be used to develop websites, games, mobile apps, and more.

JavaScript can be added to your HTML file in two ways:

- Internal JS: We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- External JS: We can write JavaScript code in another file having an extension .js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

Applications of JavaScript:

JavaScript is used to create interactive websites. It is mainly used for:

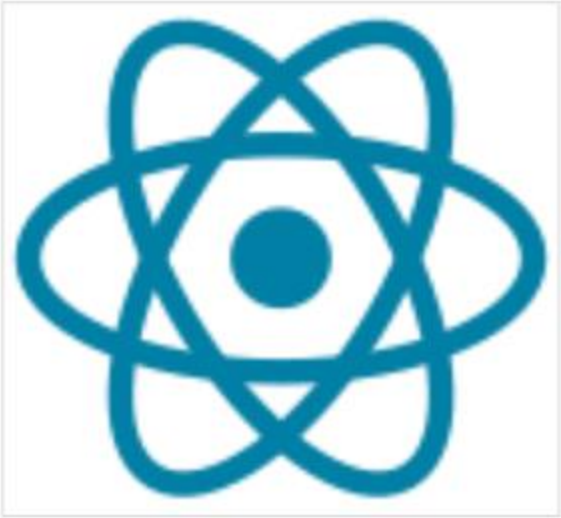
- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

5.4 Bootstrap

Bootstrap is an HTML, CSS and JS library that focuses on simplifying the development of informative web pages (as opposed to web applications). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. Bootstrap also comes with several JavaScript components which do not require other libraries like jQuery. They provide additional user interface elements such as dialog boxes, tooltips, progress bars, navigation drop-downs, and carousels.

5.5 React

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on components by Facebook Inc. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React is only concerned with the user interface and rendering components to the DOM, react applications often rely on libraries for routing and other client-side functionality. A key advantage of React is that it only renders those parts of the page that have changed, avoiding unnecessary rendering of unchanged DOM elements. React code is made of entities called components. These components are modular and reusable. React applications typically consist of many layers of components. The components are rendered to a root element in the DOM using the React DOM library. When rendering a component, values are passed between components through props (short for "properties"). Values internal to a component are called its state. The two primary ways of declaring components in React are through function components and class components.



- **Virtual DOM:** React uses a virtual DOM to efficiently update the browser DOM. It compares the virtual DOM with the real DOM and only applies the necessary changes, which improves performance.
- **JSX (JavaScript XML):** JSX is a syntax extension in React that allows you to write HTML-like code directly in JavaScript. This makes it easier to create and manage React components within your codebase.
- **State Management:** React components can manage their own state using useState hook (for functional components) or class-based state (for class components). This allows components to store and update data internally, facilitating dynamic user interfaces.
- **Lifecycle Methods:** Class components in React have lifecycle methods such as componentDidMount, componentDidUpdate, and componentWillUnmount, which allow developers to execute code at specific points during a component's lifecycle.
- **React Hooks:** Introduced in React 16.8, hooks are functions that let you use state and other React features without writing a class. Hooks provide a more functional approach to state management and side-effects in functional components.
- **Component Reusability:** React promotes component reusability and modularity. Components can be composed together to build complex UIs, and they can be reused across different parts of the application, leading to cleaner and more maintainable code.

- **Community and Ecosystem:** React has a vibrant community and a rich ecosystem of libraries, tools, and extensions (like Redux for state management, React Router for routing, and Axios for HTTP requests) that enhance its capabilities and support development across various use cases.

5.6 Firebase

Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++. Firebase is a comprehensive platform developed by Google for creating mobile and web applications. It provides a variety of tools and services that help developers build high-quality apps, grow their user base, and earn more profit.

Here are the main components and features of Firebase in detail:

1. Firebase Realtime Database:

Firebase Realtime Database is a NoSQL cloud database that stores data in JSON format and synchronizes data in real-time across all connected clients. It supports offline usage, allowing data to be stored locally on the device, and then synchronized when the client regains connectivity. With its ability to handle real-time updates, it is particularly useful for applications where data needs to be instantly shared between users.

2. Cloud Firestore:

Cloud Firestore is a flexible, scalable database designed for mobile, web, and server development. It supports real-time updates and complex querying, allowing developers to structure data hierarchically. Firestore ensures data synchronization even when the client is offline, and is built to scale as your app grows, making it suitable for large datasets.

3. Firebase Authentication:

Firebase Authentication provides backend services, easy-to-use SDKs, and pre-built UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, and popular federated identity providers like Google, Facebook, and Twitter. This service ensures secure user management and simplifies the implementation of authentication features.

4. Firebase Cloud Messaging (FCM):

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that allows you to send messages and notifications to your adopters at no cost. You can send messages to specific devices, groups of devices, or subscribed topics, making it easy to target your audience and engage users with timely information.

5. Firebase Analytics:

Firebase Analytics is a free app measurement solution that provides insights into app usage and user engagement. It offers unlimited reporting and integrates with other Firebase features,

enabling detailed analysis and user segmentation for targeting specific audiences. This helps developers understand user behavior and improve the app experience**.**

6. Firebase Cloud Storage:

Firebase Cloud Storage is designed to help developers store and serve user-generated content like photos and videos. It offers scalable and secure storage solutions, easily integrates with Firebase Authentication for security, and supports large file uploads and downloads, making it ideal for media-heavy applications.

7. Firebase Hosting:

Firebase Hosting is a fast and secure web hosting service for static and dynamic content. It features a global content delivery network (CDN) for rapid content delivery, free SSL certificates for secure connections, and a simple deployment process with a single command, ensuring that your content is quickly and securely accessible.

8. Firebase Functions:

Firebase Functions is a serverless framework that enables developers to run backend code in response to events triggered by Firebase features or HTTPS requests. It allows for scalable backend logic and reduces operational overhead by automatically managing server infrastructure, making it easier to extend app functionality without managing servers.

9. Firebase Performance Monitoring:

Firebase Performance Monitoring helps developers gain insight into the performance characteristics of their iOS, Android, and web apps. It automatically collects performance data and allows for custom performance traces, providing detailed information on network request performance and helping identify and fix performance bottlenecks.

10. Firebase Remote Config:

Firebase Remote Config allows developers to change the behavior and appearance of their app without requiring users to download an update. It supports A/B testing and

personalization, enabling real-time updates and feature toggling, which helps in delivering a dynamic and personalized user experience.

11. Firebase Test Lab:

Firebase Test Lab is a cloud-based app-testing infrastructure that allows developers to test their apps on a wide range of physical and virtual devices. It integrates with CI/CD pipelines and provides automated testing, helping to ensure that apps perform well across different devices and environments.

12. Firebase Crashlytics:

Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize, and fix stability issues. It provides insights into the causes of crashes and offers detailed crash reports, including breadcrumb logging, which helps in identifying the sequence of events leading to a crash, thus aiding in faster resolution of issues.

Benefits of Using Firebase:

Firebase offers an integrated platform with a suite of products that work seamlessly together. It is scalable, supporting apps of all sizes, from small projects to large-scale applications. Firebase is known for its ease of use, with comprehensive documentation and easy-to-use SDKs, speeding up the development process. Real-time capabilities, such as those provided by Realtime Database and Firestore, make it ideal for applications requiring instant data synchronization.

Use Cases:

Firebase is well-suited for a variety of applications, including chat applications that benefit from real-time data syncing, e-commerce platforms that require scalability and analytics, gaming apps needing high user engagement and real-time data handling, and social media

applications that leverage user authentication, real-time data, and performance monitoring features.

CHAPTER 6

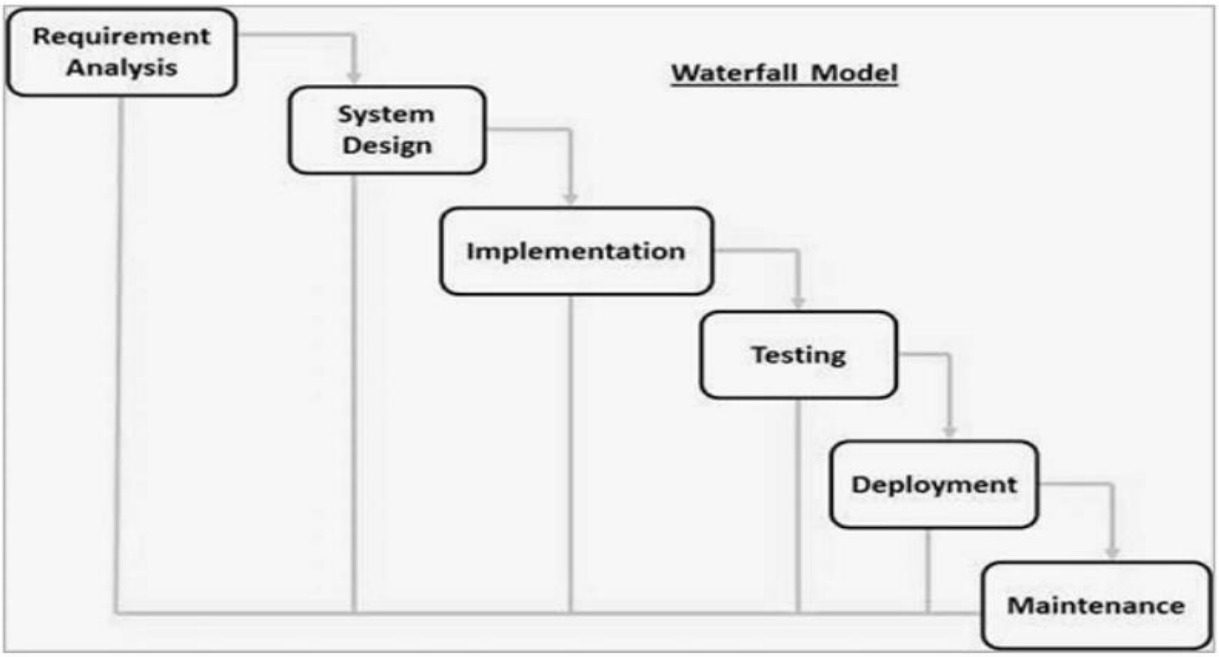
SOFTWARE PROCESS MODEL

6.1 WATERFALL MODEL

The **Waterfall Model** is a **sequential development process** where each phase must be completed before the next one begins. This model is easy to understand and manage, making it a good choice for academic and small-scale projects like Pet Heaven.

Each phase has clear deliverables and documentation, and it ensures that the development process moves in a **step-by-step flow** — just like a waterfall.

The following illustration is a representation of the different phases of the Waterfall Model.



1. **Requirement Gathering and Analysis:**

In this phase, all the functional and non-functional requirements of Pet Heaven were collected. This includes user needs, NGO workflows, admin controls, and data management. These were documented and finalized before starting the system design.

2. **System Design:**

Based on the requirements, the system's architecture was designed. This includes deciding the modules (Admin, User, NGO), data flow, UI layout, Firebase database structure, and navigation logic. It acts as a blueprint for developers.

3. **Implementation:**

In this phase, actual coding was done using HTML, CSS, Bootstrap, JavaScript, ReactJS, and Firebase. Each module was developed independently using component-based design.

4. **Integration and Testing:**

Once individual modules were complete, they were integrated together. The entire system was tested to ensure it works smoothly. Errors and bugs were fixed, and user authentication, pet listing, adoption flow, and profile management were tested thoroughly.

5. **Deployment:**

After successful testing, the system was deployed using **Firebase Hosting**, making it accessible over the internet to real users and NGOs.

6. **Maintenance:**

After deployment, any feedback or reported issues are handled through updates. Future changes like new filters, chat support, or pet training tips can be added easily due to the scalable structure.

6.3 WHY WATERFALL FOR PETJOYLANDS?

- The requirements of the system were **clearly defined** in the beginning.
- The project followed a **linear structure** — plan, design, code, test, and deploy.
- It is **easy to manage and document**, making it ideal for academic submission.
- Each phase had **clear goals**, and review was done before moving forward.

6.4 ADVANTAGES OF WATERFALL MODEL

- Simple and easy to understand and use.
- Each phase has specific deliverables and clear documentation.
- Easy to manage due to the model's rigidity.
- Phases are completed one at a time in a strict sequence.
- Works well for smaller or academic projects with fixed requirements.
- Clearly defined start and end for each phase.
- Allows for departmentalization and structured workflow.
- Easy to arrange and allocate tasks based on phase.
- Results and progress are well documented.
- Testing and implementation happen only after full development.
- Less scope for confusion due to linear approach.
- Ensures good control over development lifecycle.
- Project schedule and deadlines can be clearly set in advance.

6.5 LIMITATIONS OF WATERFALL MODEL

- No working software is produced until the last phase.
- Difficult to go back and change something once a phase is completed.
- Not suitable for complex or object-oriented projects.
- High risk if requirements change mid-project.
- Testing is delayed and happens only after full development.
- Not ideal for long-term or ongoing projects.

- Requires clear and fixed requirements from the beginning.
- High chances of discovering bugs late in the process.
- No user feedback during development stages.
- Integration happens at the end, causing "big bang" risk.
- Real progress is hard to measure in early stages.
- Cannot handle flexible or evolving needs efficiently.
- Not suited for projects that demand continuous iteration or updates.

CHAPTER 7 – SYSTEM DESIGN

System Design is the process of planning the **architecture, layout, flow, and functionality** of the project before actual development begins. It is one of the most important phases of software development as it defines how the system will work, how users will interact with it, and how data will flow from one part of the system to another.

The design phase of **Pet Heaven** focuses on breaking down the system into various **modules** like Admin, User, and NGO, and clearly defining their **roles, functionalities, and connections**.

7.1 OBJECTIVE OF DESIGN:

The main objective of system design in Pet Heaven is to ensure that:

- The system is **modular**, meaning each section (admin/user/NGO) works independently but is still connected to the overall platform.
- The design reflects the **real-life process** of pet adoption — right from viewing pet details to submitting and processing adoption requests.
- Every action performed by the user or NGO is **accurately reflected** in the backend (Firebase), ensuring data consistency.
- The interface should be **clean, intuitive, and mobile-friendly** so that people with less technical knowledge (like NGO staff or pet adopters) can also use it easily.

- The system can be **maintained and scaled** in the future by adding new features without breaking the existing structure.

7.2 SYSTEM ARCHITECTURE OVERVIEW:

Pet Heaven follows a **three-tier architecture**, which separates the user interface, business logic, and data management into different layers. This helps maintain clean code, better organization, and simplifies testing and debugging.

1. Presentation Layer:

This layer consists of all the frontend components built using **React JS**, HTML5, CSS3, and Bootstrap. It handles how the user sees and interacts with the system — pet listing, navigation bar, registration/login forms, adoption forms, etc. The use of **React components** ensures modular and reusable UI elements.

2. Application Layer:

Also known as the logic layer, it consists of React's internal state management, functions, event handling, and conditional rendering. It processes all user inputs, manages user state, and controls routing/navigation between pages.

3. Data Layer:

This includes **Firebase Firestore**, which stores and retrieves real-time data such as pet records, adoption requests, and user profiles. **Firebase Authentication** handles secure login and role-based access, while **Firebase Storage** is used to upload and store pet images and medical documents.

7.3 MODULAR DESIGN

To keep the system clean, flexible, and easy to test, Pet Heavenis divided into **three main functional modules** based on user roles:

Admin Module:

- Admin logs in through a secure login portal.
- Can **monitor all users and NGOs**, including approval or blocking.
- Has authority to **approve pet listings**, delete false data, or remove inactive accounts.
- Can view **real-time data** on adoption requests, pet statuses, and site activity.
- Ensures the system remains safe and free from unauthorized activities.

User Module:

- A user can register/login using email credentials.
- After login, they can **browse pets**, view details like age, breed, health, and vaccination status.
- Users can **apply for adoption** by filling a detailed online form.
- They can also **track the status** of their request — whether accepted, pending, or rejected.
- After successful adoption, users can give feedback or rating to the NGO.

NGO Module:

- NGOs register through an NGO-specific registration portal and get access upon admin approval.
- Once logged in, NGOs can **upload pet information**, including images, age, medical history, and current status (available/adopted).
- They can **review adoption applications** and approve/reject them based on suitability.
- NGOs also manage their profile and view adoption history for record-keeping.

7.4 DATABASE STRUCTURE:

Pet Heaven uses **Firebase Firestore**, a NoSQL document-based cloud database, for all backend storage. It allows fast, secure, and real-time access to data across devices.

Key collections used in the database:

- **Users:** Stores user profile, contact info, and past adoption history.
- **NGOs:** Stores NGO details, registered pets, and verification status.
- **Pets:** Stores individual pet records including name, breed, category, image, and medical documents.
- **Applications:** Stores adoption request forms with user ID, pet ID, application status, and timestamps.

Each document inside these collections is identified by a unique ID, making data access fast and efficient.

7.5 DATA FLOW EXPLANATION:

Pet Heaven ensures **smooth and structured data flow** between users, NGOs, and admins. Here's how the data flows through the system:

1. **User** registers and logs in → frontend sends request to Firebase Auth.
2. User browses pets (data fetched from **Pets** collection in Firestore).
3. User fills **adoption form** → saved in **Applications** collection.
4. **NGO** views the application → updates its status (Accepted/Rejected).
5. Status change reflects instantly for the user due to **real-time database sync**.
6. **Admin** monitors and manages all user, pet, and adoption data centrally.

This flow is **efficient, transparent**, and ensures every action is traceable and logged.

CHAPTER 8 – DATA FLOW DIAGRAM (DFD)

Data Flow Diagrams (DFDs) were first developed by Larry Constantine as a method of expressing system requirements in a graphical form. Also known as bubble charts, DFDs aim to clarify system requirements and identify major transformations, ultimately aiding in the

system design process.

DFD is a means of representing a system at any level of detail through a graphic network of symbols that illustrate data flows, data stores, data processes, and data sources/destinations.

Purpose:

- The primary purpose of data flow diagrams is to provide a semantic bridge between users and systems developers. They offer several key benefits:
- Graphical Representation: DFDs are graphical, which eliminates the need for lengthy textual descriptions and makes complex systems easier to understand.
- Logical Representation: They model what a system does, focusing on the logical aspect rather than the physical implementation. This helps in understanding the functionality without getting bogged down by technical details.
- Hierarchical Structure: DFDs are hierarchical, allowing systems to be shown at any level of detail. This makes it easier to break down complex processes into simpler, manageable components.
- User Understanding and Review: By providing a clear, visual representation of the system, DFDs facilitate user understanding and make it easier for users to review and provide feedback.

DFD Symbols are as follows:

- The External Entity symbol represents sources of data to the system or destinations of data from the system.

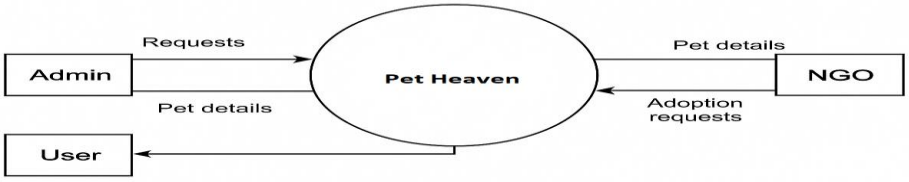
- The Data Flow symbol represents the movement of data.

- The Data Store symbol represents data that is not moving (delayed data at rest).

- The Process symbol represents an activity that transforms or manipulates the data.

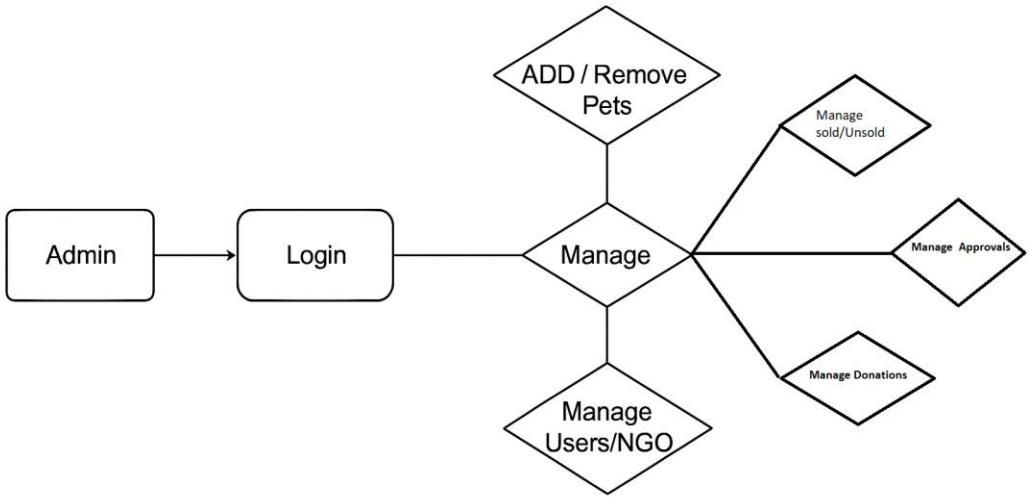
DFD Level 0

Context Level Diagram



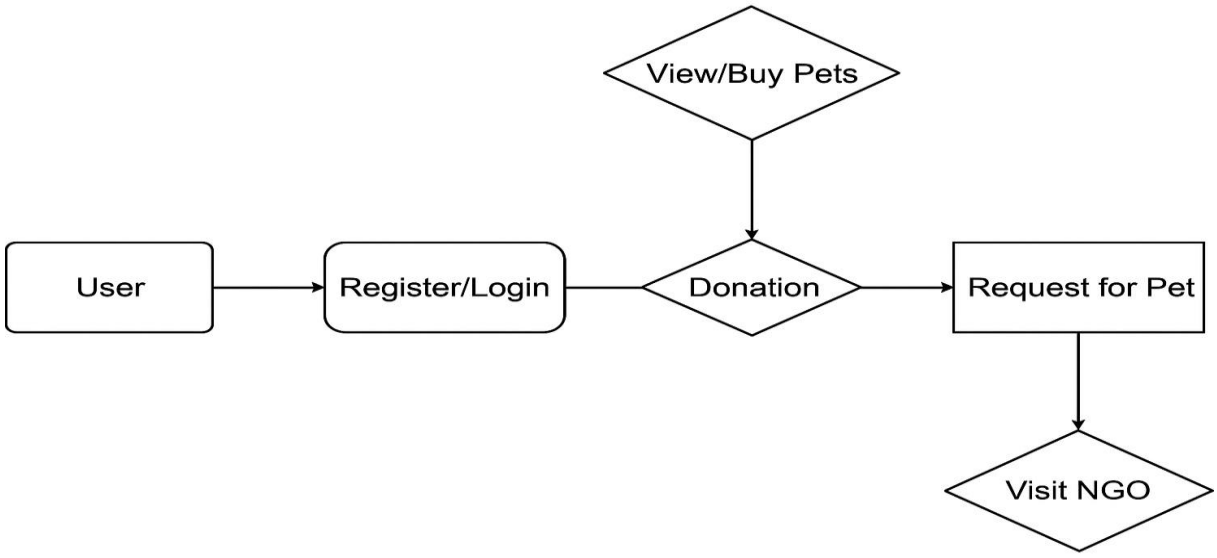
Level 1 DFD:

DFD for Admin



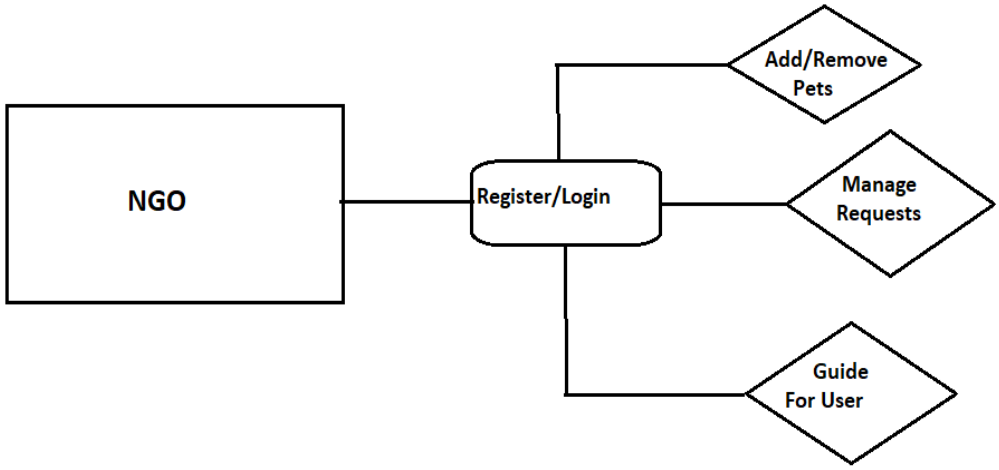
Level 2 DFD:

DFD for User



Level 3 DFD:

DFD for NGO



CHAPTER 10 TESTING

10.1 Introduction to Testing

Testing is a vital and unavoidable phase in the software development life cycle. It ensures that the software performs as per the requirements, delivers the expected results, and is free from any major errors or bugs. A well-tested system not only improves reliability and performance but also boosts user confidence and system security.

In the **Pet Heaven** project, where multiple users such as NGOs, admins, and adopters interact with the platform, it was essential to thoroughly test each module for correctness, responsiveness, and smooth functionality. All the user input forms,

Firebase database interactions, UI components, and navigation logic were carefully tested to detect any possible flaws before final deployment.

The main goal of testing this system was to check whether the pet listing, user registration, adoption process, and admin functionalities are working correctly across different use cases and devices. It also ensured that invalid or incomplete inputs are properly handled and that only verified users can access the system.

10.2 TYPES OF TESTING USED:

To achieve reliable results, the following types of testing techniques were used during the development of Pet Heaven:

1. Unit Testing:

Unit testing is done to check individual components or modules of the system separately. For example, the login form, registration form, pet upload module, and application status panel were each tested on their own. This helped in detecting any issues at the core logic level before connecting them to other modules.

2. Integration Testing:

Once all modules were unit-tested, they were combined and tested together as one system. Integration testing ensured that the system behaves correctly when different modules work in coordination. For instance, if a user submits an adoption request, it should be visible to the NGO and also trackable by the admin. This type of testing helped in validating the interaction between frontend (React) and backend (Firebase).

3. Manual Testing:

The entire system was manually tested by simulating different real-life user

scenarios. All forms were filled with valid and invalid inputs to check how the system responds. Buttons, navigation links, alert messages, and loading behavior were tested thoroughly across devices and browsers. Manual testing confirmed that the UI was responsive, all features were working properly, and that the system was ready for deployment.

10.3 TEST CASES:

Test Case 1: User Login Functionality

This test case was performed to verify whether a user can successfully log in to the system using valid credentials. When the correct email and password were entered, the user was directed to their dashboard. In case of incorrect login details, the system displayed an appropriate error message. This confirms that the login module is functioning correctly and secure access is maintained.

Test Case 2: Pet Upload by NGO

In this test, an NGO logged into their account and added a new pet using the upload form. Required fields such as pet name, breed, age, and vaccination info were filled, and an image was uploaded. Upon submission, the pet was successfully added and displayed on the user side. This test verified that the pet listing feature is working as expected.

Test Case 3: Adoption Request Submission

A registered user selected a pet and submitted an adoption request through the

online form. The submitted request was stored in the database and became visible to the respective NGO. The NGO was able to view the request and take action (approve/reject). This confirms that the adoption process is functioning smoothly between user and NGO panels.

CHAPTER 11 – IMPLEMENTATION

Implementation is the stage where the actual code is deployed, and the system is made live for use. After designing and testing all modules of the **Pet Heaven** system, the final application was implemented using real-time cloud integration through Firebase. The website was hosted online so that users, NGOs, and the admin can access their respective panels from anywhere.

All user roles — **User, NGO, and Admin** — were successfully integrated into a single platform. The interface is fully responsive and accessible on both desktop and mobile devices. Implementation also involved **connecting frontend pages** with the **Firebase backend**, so that any action performed by the user reflects instantly across the platform.

11.2 DESIGNER CONCERNS:

From the designer's perspective, the system needed to be **modular**, **scalable**, and **easy to maintain**. Each module (Admin, NGO, User) was designed separately using **React components** to maintain separation of concerns.

Firebase was selected as the backend due to its real-time data sync, built-in authentication, and cloud hosting capabilities. This eliminated the need for managing complex servers and databases.

The UI/UX was kept clean using **Bootstrap**, and React Router was used for smooth page transitions. The design ensures future features (like vet panel, reminders, notifications) can be added without disturbing existing code.

Security and performance were also important concerns — which were managed by using Firebase security rules and lightweight frontend logic.

11.1 USER CONCERNS

The system is designed with the **end user in mind**. Since the platform is used by general public users, NGOs, and admin staff, the interface is kept **simple, clear, and responsive**. All pages are mobile-friendly and easy to navigate.

The user is not required to have any technical knowledge to use the system. From browsing pets to applying for adoption, every action is simplified through clear forms and easy buttons. Feedback and confirmation messages are shown after each action to keep the user informed.

Special attention was given to users with limited digital experience (such as NGO workers), ensuring that even basic users can operate the platform without training.

CHAPTER 12 – MAINTENANCE

12.1 INTRODUCTION

Maintenance is the final but **continuous phase** of the software development life cycle. Once a system is deployed and being used in the real world, it is important to monitor its performance, fix any unexpected issues, and make necessary improvements over time. In the case of **Pet Heaven**, maintenance ensures that the website continues to run smoothly for users, NGOs, and admins without any interruptions or outdated data.

As users interact with the system, feedback is collected, and based on that, updates and improvements are planned. Also, technical aspects like hosting, database performance, and user authentication are regularly checked. Maintenance helps keep the system **reliable, up-to-date, and error-free** even after deployment.

12.2 TYPES OF MAINTENANCE

Software maintenance is generally divided into four types:

---

1. Corrective Maintenance

This type of maintenance is done to **fix bugs or errors** that are found after the system is deployed. If users report login issues, form problems, or data mismatches, corrective maintenance is applied immediately.

---

2. Adaptive Maintenance

As technology changes, the software also needs to be **updated to adapt**. For example, if Firebase updates its rules or browsers change behavior, Pet Heaven may need to adapt to stay compatible.

---

3. Perfective Maintenance

This refers to **improving the system's performance** or adding new features based on user feedback. In Pet Heaven, this may include adding search filters, chat features, or improved mobile layout.

---

4. Preventive Maintenance

This type of maintenance is performed to **prevent future issues** by optimizing code, updating security settings, or cleaning up unused data. It helps increase system life and reduce risks.

CHAPTER 13 – CONCLUSION

The development of **Pet Heaven** was a valuable and practical learning experience. The project aimed to simplify the pet adoption process by connecting adopters, NGOs, and admins on a single digital platform. It helped us understand the complete software development life cycle — from planning and design to implementation and testing.

Through this project, we explored frontend technologies like **HTML, CSS, Bootstrap, ReactJS**, and backend services like **Firebase Firestore and Authentication**. We successfully created a responsive, real-time, and user-friendly system that allows secure login, pet listing, adoption requests, and admin control.

The system solves many problems that exist in traditional adoption processes such as manual tracking, delays, and lack of visibility. With Pet Heaven, adoption becomes **faster, easier, and more transparent**.

In future iterations, the platform can be enhanced by integrating **chat support between NGOs and users**, **SMS/email notifications for updates**, **medical or vaccination reminders for adopted pets**, and even **AI-based pet match recommendations** to connect users with pets based on their preferences and lifestyle.

Overall, the successful implementation of Pet Heaven reflects our ability to work as a team, apply technical knowledge to real problems, and build a socially impactful solution. It also motivates us to explore further innovations in the area of **animal welfare and digital transformation**.