Title of the project

"Zoo Management System"

Course: **Project – Fall'24**

Course Code: CIS499

Department of Computing and Information System (CIS)

Submitted By:

Tahmid Jhauhar Adnin

ID: 182-16-323

Supervised By:

Md. Mehedi Hassan

Lecturer (Senior Scale) of department CIS

Daffodil International University

Approval

This Project titled "Zoo Management System", Submitted by Tahmid Jhauhar Adnin, ID No. 182-16-323 to the Department of Computing and Information Systems, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information Systems and approved as to its style and contents. The presentation has been held on 13-01-2025.

BOARD OF EXAMINERS

Md Sarwar Hossain Mollah — **Chairman**
Associate Professor and Head
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Md. Nasimul Kader — **Internal Examiner**
Assistant Professor
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

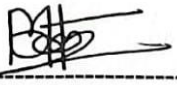
Md. Mehedi Hassan — **Internal Examiner**
Lecturer (Senior Scale)
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Ahmed Saif Reza — **External Examiner**
Managing Director & Chief Technology Officer
Medico Bio Limited

Declaration

I hereby declare that; this project has been done by me under supervision of Md. Mehedi Hassan, Lecturer (Senior Scale), department of Computing and Information System (CIS) of Daffodil International University. I am also declaring that this project or any part of there has never been submitted anywhere else for the award of any educational degree like, B.Sc., M.Sc., Diploma or other qualifications.

Supervised By



Md. Mehedi Hassan
Lecturer
Department of CIS
Daffodil International University

Submitted By

Name: Tahmid Jhauhar Adnin
ID: 182-16-323
Department of CIS
Daffodil International University

Acknowledgement

First and foremost, I would like to express my gratitude to Almighty Allah for providing me with this great opportunity to learn and grow. Without His advice and assistance, I would not have been able to complete this project and all the connected tasks. I have learned a variety of knowledge and abilities, especially in respect to emerging technology and industry practices. I am immensely indebted to my friend who went with me zoo for several times and Md. Mehedi Hassan, My academic supervisor. Their unfailing support, guidance, and encouragement have been vital in my career progress. They have always been there to provide the proper guidance, assist me make informed decisions, and motivate me to conquer problems. I am really grateful to them for their politeness, assistance, and belief in my ability. Their Assistance has not only helped me navigate through difficult assignments but has also taught me how to flourish in demanding conditions and handle large-scale projects in the future.

Abstract

The Zoo Management System will revolutionize how zoos manage their operations, providing improved care for animals, maintain a strategic way to handle database for food supply, better experiences for visitors, and more efficient operations for staff. Through this proposal, my aim is to store data that can be fetched easily in terms of need. Because in our country, there is a lack of justification as to whether one animal receives the required food or not. Because there are only manual data sheets, which are so time-consuming to locate any specific data. And sometimes they are lost. On the other hand, there is a delay in enlisting illness reports to the authority. Or I can say there is so much negligence to fulfill in this criteria. So there needed to be at least a specific data sheet where all the ill animals would be enlisted with adequate information. Along with this, either they are attended by someone or not; that is marked by authority. I believe both these fundamental criteria are met in the "Zoo Management System" successfully. I seek approval for the ZMS project, which will deliver substantial benefits.

Table of Contents

Approval i

Declaration ii

Acknowledgement iii

Abstract iv

1 Chapter: Introduction 1

1.1 Introduction 1

1.2 Purpose of Project 1

2 Chapter: Initial Study 3

2.1 Project Purpose: 3

2.2 Project Scope 4

2.3 Background of the project 4

2.4 Problem Area 5

2.5 Possible Solution 5

3 Chapter: Literature Review 7

3.1 Discussion on problem domain based on available solution: 7

3.2 Discussion on problem based on available solution: 8

3.3 Comparison of leading Solution: 9

4 Chapter: Methodology 10

4.1 What to use 10

4.1.1 Agile Model: 10

4.1.2 Rapid Application Development (RAD) Model: 11

4.2 Why to use 12

4.3 Sections of Methodology 13

4.4 Implementation Plan 14

5 Chapter: Project Plan 15

5.1 Work Break Down structure 15

5.2 Resource Allocation 16

5.3 Time Boxing 16

5.4 Gantt chart 18

6 Chapter: Feasibility Study 19

6.1 Every viable types of feasibility study: 19

6.1.1 Operational Feasibility: 19

6.1.2 Technical Feasibility: 19
6.1.3 Economic Feasibility: 19
6.1.4 Market analysis driven by feasibility determinants: 19
7 Chapter: Foundation 20
7.1 Problem area identification: 20
7.2 Interview: 21
7.3 Questionnaire: 21
7.4 Requirement Specification: 22
7.4.1 Functional Requirements: 22
7.4.2 Non-Functional Requirements: 23
7.5 Technology to be implemented: 23
8 Chapter: Exploration 24
8.1 Activity diagram: 24
8.1.1 Activity Diagram (1) (Admin process): 24
8.1.2 Full Activity Diagram: 25
8.2 Full system use case: 25
8.3 Requirement Catalog: 26
8.4 Prioritized Requirement List (PRL): 27
9 Chapter: Exploration 28
9.1 Module of the system: 28
9.2 Class diagram: 28
9.3 Sequence diagram of the system: 29
9.4 Prototype of the system: 29
10 Chapter: Development 32
10.1 Existing Folder structure of the system: 32
10.2 Core Module Coding Samples-1: 33
10.3 Core Module Coding Outcome-1: 37
10.4 Core Module Coding Samples-2: 38
10.5 Core Module Coding Outcome-2: 42
11 Chapter: Testing 44
11.1 Test Case: 44
11.2 Unit Testing: 45
11.2.1 Unit Test -1 45
11.2.2 Unit Test -1-Output Visualization: 46

11.2.3 Unit Test -2 47
11.2.4 Unit Test -2-Output Visualization: 48
11.2.5 Module Test -1 49
11.2.6 Unit Test -2-Output Visualization: 50
11.3 Integration Testing: 52
11.4 Security Testing: 52
12 Chapter: Implementation 55
12.1 Training: 55
12.2 Scaling: 56
12.3 Load Balancing: 56
13 Chapter: Critical Appraisal and Evaluation 57
13.1 Objective that could be met: 57
13.2 How much better could have been done: 57
13.3 Which features could not be touched 57
14 Chapter: Lesson Learned 58
14.1 Pre Project – Review – Closing: 58
14.2 The Problem I Have Faced: 58
14.3 What Solutions Occurred: 58
15 Chapter: Lesson Learned 59
15.1 Summary of the project: 59
15.2 Goal of the project: 59
15.3 What I have done in the documentation 59
15.4 My Experience: 60
Works Cited 61

List of Figures:

Figure 4.1.1-1 Agile Model 11
Figure 4.1.2-1 RAD Model 12
Figure 4.1.2-1 Work Break Down Structure 15
Figure 4.1.2-1 Resource Allocation 16
Figure 4.1.2-1 Time Boxing 17
Figure 4.1.2-1 Gantt Chart 18
Figure 8.1.1-1 Admin Activity Diagram 24
Figure 8.1.2-1 Full System Panel Activity Diagram 25
Figure 8.1.2-1 Full System Use Case 26
Figure 8.1.2-1 Class Diagram 28
Figure 8.1.2-1 Sequence Diagram 29
Figure 8.1.2-1 Prototype of Home Page 29
Figure 8.1.2-2 Prototype of Home Page(Hover) 30
Figure 8.1.2-3 Prototype of Location 30
Figure 8.1.2-4 Prototype of Admin Log In 31
Figure 8.1.2-5 Prototype of Animal Feed list 31
Figure 8.1.2-1 Actual Folder Structure Sample 32
Figure 8.1.2-1 Core Module Coding Sample for Adding Animal, Classification wise 33
Figure 8.1.2-2Core module Coding Sample for Animal Classification wise 33
Figure 8.1.2-3 Core Module Coding Sample for Adding Animal Classification wise 34
Figure 8.1.2-4 Core Module Coding Sample for Adding Animal Classification wise 34
Figure 8.1.2-5 Core Module Coding Sample for Adding Animal Classification wise 35
Figure 8.1.2-6 Core Module Coding Sample for Adding Animal Classification wise 35
Figure 8.1.2-7 Core Module Coding sample For Adding Animal Classification Wise 36
Figure 8.1.2-8 Core Module Coding Sample for Adding Animal Classification wise 36
Figure 8.1.2-1 Core Module Coding Outcome for Adding Animal Classification wise 37
Figure 8.1.2-2 Core Module Coding Outcome for Adding Animal Classification wise 37
Figure 8.1.2-1 Core Module Coding Sample for Adding Animal Sick List to Take Care 38
Figure 8.1.2-2 Core Module Coding Sample for Adding animal Sick list to Take Care 38
Figure 8.1.2-3 Core Module Coding Sample for Adding Animal Sick List to Take Care 39
Figure 8.1.2-4 Core Module Coding Sample for Adding Animal Sick List to Take Care 39
Figure 8.1.2-5 Core Module Coding Sample for Adding Sick List to Take Care 40
Figure 8.1.2-6 Core Module Coding Sample for Adding Animal Sick List to Take Care 40
Figure 8.1.2-7 Core Module Coding Sample for Adding Animal Sick list to Take Care 41
Figure 8.1.2-8 Core Module Coding Sample for Adding Animal Sick List To Take Care 41
Figure 8.1.2-9 Core Module Coding Sample for Adding Animal Sick List to Take Care 42
Figure 8.1.2-1 Core Module Coding Outcome for Adding Animal into Watch List Due to Sick List 42
Figure 8.1.2-2 Core Module Coding Outcome for Adding Animal into Watch List due to Sick List 43
Figure 8.1.2-3 Core Module Coding Outcome for Adding Animal into Watch List Due to Sickness 43
Figure 11.2.2-1 Successfully Storing Data with Message Status 46
Figure 11.2.2-2Viewing Data from Database into Feed List 46
Figure 11.2.2-3 Showing a Warning Message Due To Fill up The Required Field 47
Figure 11.2.4-1 Viewing Data from Database before Any Action Performed 48

Figure 11.2.4-2 After Performing Action such as Mark Attended Showing Data with Successful Status 48
Figure 11.2.4-3 Viewing Data from Database in View Attended Feed List 49
Figure 11.2.4-4 After Performing Action such as Mark Unattended Restore Data 49
Figure 11.2.6-1Take Input For Booking Ticket 50
Figure 11.2.6-2 Calculation Total Price Successfully 51
Figure 11.2.6-3 Showing Alert Message To Book Ticket 51
Figure 11.2.6-1 Giving Wrong account data for log in 53
Figure 11.2.6-2Showing Alert Message 54
Figure 11.2.6-3 Showing Alert Message For Giving Wrong Data 54

List of Tables:

Table 3.3-1 Comparison of Leading Solution 9
Table 8.4-1 PRL 27
Table 9.1-1 Module of the System 28
Table 11.2-1 Unit test-1 45
Table 11.2-2 Unit test-2 48
Table 11.2-3 Module test-1 50
Table 11.3-1 Integration Testing 52
Table 11.4-1 Security Testing 53

1 Chapter: Introduction

1.1 Introduction

Zoo Management System (ZMS) is a comprehensive software application that is intended to improve the operational efficacy and management of zoos. The system is designed to optimize a variety of operations, including animal management, animal food distribution information, animal care identification stage, personnel management, visitor interactions, financial operations and many more. Our objective is to offer a contemporary solution that improves the overall visitor experience, while simultaneously guaranteeing the optimal care of animals and the efficient management of resources through the implementation of ZMS.

The primary objectives of the Bangladesh National Zoo are the conservation of wildlife through the acquisition and breeding of uncommon and endangered wild animal species, as well as research, education, and recreation. The promotion of public awareness about these species of animals, education and research on wild animals, and conservation of the diversity of wild animals. The names and descriptions of all the exhibits are accurate. Accompanying the illustrations are detailed descriptions of endangered and extinct species (About: bnzoo, n.d.). This zoo management system will satisfy all the requirements of the Bangladesh National Zoo, in addition to providing additional benefits, while considering all of its objectives. Because in terms of animal food supplement, there was shown a lot of problem due to lacking of adequate explanation to the superior. If one wished to inquiry about this, he or she couldn't due to insufficient data. Besides this, there should a systematic plan for care unit. By which an accountable person easily finds out which birds or animals are ill and needed to cure them according to their severity. That's the essential object to establish the system.

1.2 Purpose of Project

- **Animal Management:** To build a centralized database for monitoring animal information, including species, health status, feeding schedules, and breeding programs.
- **Visitor Management:** To develop a user-friendly interface for ticketing, visitor feedback, and special events, enhancing visitor engagement and satisfaction.
- **Staff Management:** To implement tools for human resource management, including staff scheduling.

- **Financial Management:** To incorporate budgeting, accounting, and reporting features for improved financial oversight and decision-making.
- **Reporting and Analytics:** To facilitate data-driven decision-making through comprehensive reporting and analysis of zoo operations.
- **Event Management:** To schedule or organize event for different purpose.

2 Chapter: Initial Study

2.1 Project Purpose:

This project "Zoo Management System", will be an organized user-friendly application that will satisfy all the consumer along with authorities can easily manage their duties precisely as well as simply. It's our duty to support other creatures to live a better life. For this reason, all the animals have rights to be treated equitably. Not like that, which cages or which parts are nearest, they only received the best service and who are staying far treated neglected. And no one takes liabilities for their loss. Hence, there must be need appropriate documentation about each animal health and nutritional status. Besides this, children or young generation can know about appropriate information about specific animal which are in the zoo. So that they will more aware about leveraging them. Event or workshop will also be an excellent concept for this purpose.

- **Objective:** Develop an application that to build a centralized database for monitoring animal information, including species, health status, feeding schedules, and storing data in organize way into database.
- **Features:** This system will offer an intuitive interface for the user to perform all the action so smoothly. Like, adding animal, update, delete, feed list, watch list to take care of animal health and food. Maintain job vacancies, ticket booking, organize event, employee data handling, sponsorship data handling etc. will be also included.
- **Benefits:** This zoo management system will satisfy all the requirements of the Bangladesh National Zoo, in addition to providing additional benefits, while considering all of its objectives. Because in terms of animal food supplement, there was shown a lot of problem due to lacking of adequate explanation to the superior. If one wished to inquiry about this, he or she couldn't due to insufficient data. Besides this, there should a systematic plan for care unit. By which an accountable person easily finds out which birds or animals are ill and needed to cure them according to their severity. That's the essential object to establish the system.
- **Timeline:** Within 5 to 6-month time duration, this proposed system presumed to be finished, incorporating phases like, collecting requirements, building application, evaluation, and implementation.

- **Resources:** To accomplish this project successfully there require a skilled team of developers, designers, with necessary hardware as well as software resources.
- **Conclusion:** The system is designed to optimize a variety of operations, including animal management, animal food distribution information, animal care identification stage, personnel management, visitor interactions, financial operations and many more. Our objective is to offer a contemporary solution that improves the overall visitor experience, while simultaneously guaranteeing the optimal care of animals and the efficient management of resources through the implementation of ZMS.

2.2 Project Scope

The project will include the following components:

- **Animal Database:** A module to record data about each animal, including species, habitat, health history, feeding routines, and reproductive information.
- **Visitor Management:** A ticketing system that allows online purchase and reservation of tickets, monitoring visitor numbers, and generating feedback forms.
- **Staff Management:** A comprehensive management tool to handle staff schedules, roles, payroll, and incident reporting.
- **Financial Management:** An accounting module to monitor income from ticket sales, donations, gifts, and expenditures on animal care and management.
- **Reporting Tools:** Customizable reporting capabilities to analyze visitor demographics, animal health trends, and financial performance.
- **Mobile Accessibility:** A mobile-friendly interface for staff and visitors to access information conveniently.
- **Integration with Existing Systems:** Ensure that the system can be integrated with any current databases or third-party applications in use.

2.3 Background of the project

The Bangladesh National Zoo, located in Dhaka, is one of the most prominent zoological institutions in the country. Founded in 1974, this expansive zoo encompasses 186 acres and

is home to a diverse collection of animal species from Bangladesh and around the globe. As a center for wildlife conservation, research, and education, the zoo plays a crucial role in raising awareness about biodiversity and fostering a connection between the public and the natural world (About: bnzoo, n.d.).

2.4 Problem Area

Despite its significance, the Bangladesh National Zoo confronts several management challenges that impact its operational efficacy and the quality of care provided to the animals. Key areas that necessitate improvement include:

- **Animal Management:** Maintaining accurate records of animal health, reproduction, and habitats is vital, but current manual monitoring methods can lead to inconsistencies and data loss.

- **Visitor Engagement:** Enhancing the visitor experience is essential for promoting wildlife conservation, but issues such as lengthy wait times, lack of information on exhibits.
- **Staff Coordination:** Effective staff scheduling and communication are critical for zoo operations, yet the absence of a structured management system can lead to inefficiencies and negligence.
- **Financial Administration:** Accurate administration of ticket sales, memberships, and other financial transactions is necessary to ensure the zoo's sustainability, but extant manual processes can complicate financial oversight.

2.5 Possible Solution

To address these challenges, the introduction of an efficient Zoo Management System (ZMS) is essential. A comprehensive software solution would optimize various operations within the Bangladesh National Zoo, providing benefits such as:

- **Enhanced Animal Care:** A comprehensive animal management module would enable staff to monitor health records, breeding programs, and nutritional information with simplicity, ensuring the welfare of the animals.

- **Improved Visitor Experience**: The ZMS would facilitate online ticket registration, provide real-time information about animal exhibits and schedules, and enhance visitor interactions, leading to increased engagement and satisfaction.
- **Efficient Staff Management:** A tool for staff scheduling, task assignments, and internal communication would facilitate greater coordination, allowing the zoo personnel to work more effectively.
- **Financial Efficiency:** Automated financial monitoring of ticket sales, memberships, and donations would enhance the zoo's financial health and accountability.
- **Impact on Conservation Efforts**: By implementing a modern Zoo Management System tailored to the requirements of the Bangladesh National Zoo, the institution can enhance its operational efficacy and focus more on its core mission of education and conservation. The integration of technology will provide valuable insights that can be used for research and conservation initiatives, ultimately contributing to the preservation of Bangladesh's abundant biodiversity

The context of this Zoo Management Project for the Bangladesh National Zoo emphasizes the imperative need for a digital transformation to combat existing challenges while promoting sustainability and conservation. The proposed ZMS stands to modernize zoo operations, elevate the visitor experience, and reinforce the zoo’s commitment to wildlife preservation and education in Bangladesh.

3 Chapter: Literature Review

3.1 Discussion on problem domain based on available solution:

Zoo Information Management Systems (ZIMS) have emerged as essential instruments for enhancing the management and operational efficiency of zoological institutions. With the increasing concentration on biodiversity conservation, animal welfare, and efficient zoo operations, the implementation of sophisticated information management systems has become critical. This literature review synthesizes findings from relevant scholarly works, including the document titled "Zoo Information Management System ZIMS" and the IEEE paper “Use of a Web-Based Zoo Animal Information Management System in the Philippines” (IEEE, 2017), to explore the critical features, applications, and benefits of ZIMS. The development of Zoo Information Management Systems is anchored in the prerequisite for effective record-keeping and information dissemination within zoological institutions. According to the ZIMS documentation, these systems are designed to facilitate the administration of animal records, including health monitoring, breeding histories, and enclosure management. These systems enable zoos to maintain accurate and real-time data at their fingertips, facilitating better decision-making and enhanced animal care. The IEEE paper elaborates on the implementation of web-based ZIMS in the Philippines, demonstrating how such systems can enhance data accessibility and facilitate communication among zoo personnel. By leveraging technology, zoos can ensure that critical information regarding animal health and welfare is readily available to veterinarians and zookeepers, thus enhancing the overall care provided to the animals.

A comprehensive ZIMS typically includes several important features that address the multifaceted requirements of zoos:

- **Animal Record Management:** A core feature of any ZIMS is its ability to maintain detailed records for each animal, including species information, health histories, nutrition schedules, and breeding data. The ZIMS documentation emphasizes the significance of accurate data entry and management in ensuring animal welfare.
- **Health Record Management:** A core feature of any ZIMS is its ability to maintain detailed

records for each animal, including species information, health histories, nutrition schedules, and breeding data. The ZIMS documentation emphasizes the significance of accurate data entry and management in ensuring animal welfare.

- **Visitor Engagement Tools:** Modern ZIMS also incorporate visitor administration functionalities, such as ticketing and scheduling, which contribute to an enhanced visitor experience. By offering interactive elements, zoos can engage visitors in conservation efforts and increase awareness about wildlife issues.
- **Data Analytics:** The ability to analyze data trends is a significant advantage provided by ZIMS. The ZIMS documentation notes that through data analytics, zoos can gain insights into animal population dynamics, habitat requirements, and reproductive success rates, informing conservation strategies (Venkataraman & Shaji, 2017).

3.2 Discussion on problem based on available solution:

The implementation of Zoo Information Management Systems presents several benefits:

- **Improve Animal Welfare:** By ensuring that health records are scrupulously ZIMS contributes to improved overall care for zoo animals. Staff can swiftly resolve health issues and optimize care routines.
- **Enhanced Operational Efficiency:** The automation of administrative tasks reduces the burden on staff and minimizes the possibilities of errors associated with manual data entry. This efficacy is crucial in bustling zoo environments where timely information can influence care and management strategies.
- **Support for Conservation Efforts:** By providing robust data on animal populations and health, ZIMS can support in-situ and ex-situ conservation efforts.
- **Increased Public Awareness:** Interactive features that engage the public can enhance awareness about wildlife conservation, thereby promoting responsible stewardship of natural resources.

3.3 Comparison of leading Solution:

Bnzoo	Proposed ZMS
No justification of providing food	Has justification of providing food
No online ticket facility	Will provide online ticket purchasing system
No navigation system within the zoo	Will be provide the location with specific animal name.
No event management facility	Will provide event management facility
Few news shows that different kinds of animal are died because of starvation and for some common diseases but the site is not helpful solving these problem.	Could be solve the problem of death because of starvation and some common diseases.
Not enough profitable	Will be profitable

Table 3.3-1 Comparison of Leading Solution

4 Chapter: Methodology

The framework and procedure employed in the development of software applications are referred to as software development methodology. It incorporates Agile as well as RAD methodologies, that intensify sequential planning as well as flexibility, correspondingly. The methodology selected is dependent upon the complexity of the undertaking and the size of the team. The development process becomes easier by the implementation of an appropriate methodology, which promotes efficiency, collaboration, and structure. It optimizes team work, alleviates risks, improves project management, and maximize customer contentment.

4.1 What to use

To build the system "Zoo Management System", I have to choose a strategy to incorporate our entire operation flow in a systematic way. Proper methodology helps a lot to build the system and adapt it according to necessity. Agile, Rapid application development (RAD) both are very popular methodology to implement it. But to find out the best suitable methodology we need to know how they work in which scale and then we will choose which one is right according to our system.

4.1.1 Agile Model:

The model of agile posits that each project requires a unique approach, and that the current methods must be customized in order to satisfy the project's demands. In Agile, the tasks are divided into time boxes, which are small time frames that are used to deliver specific features for a release. A functioning software build is delivered after each iteration, and an iterative approach is employed. The final build contains all the features that the customer requires, while each succeeding build is incremental in terms of features (sdlc:sdlc_agile_model, n.d.).

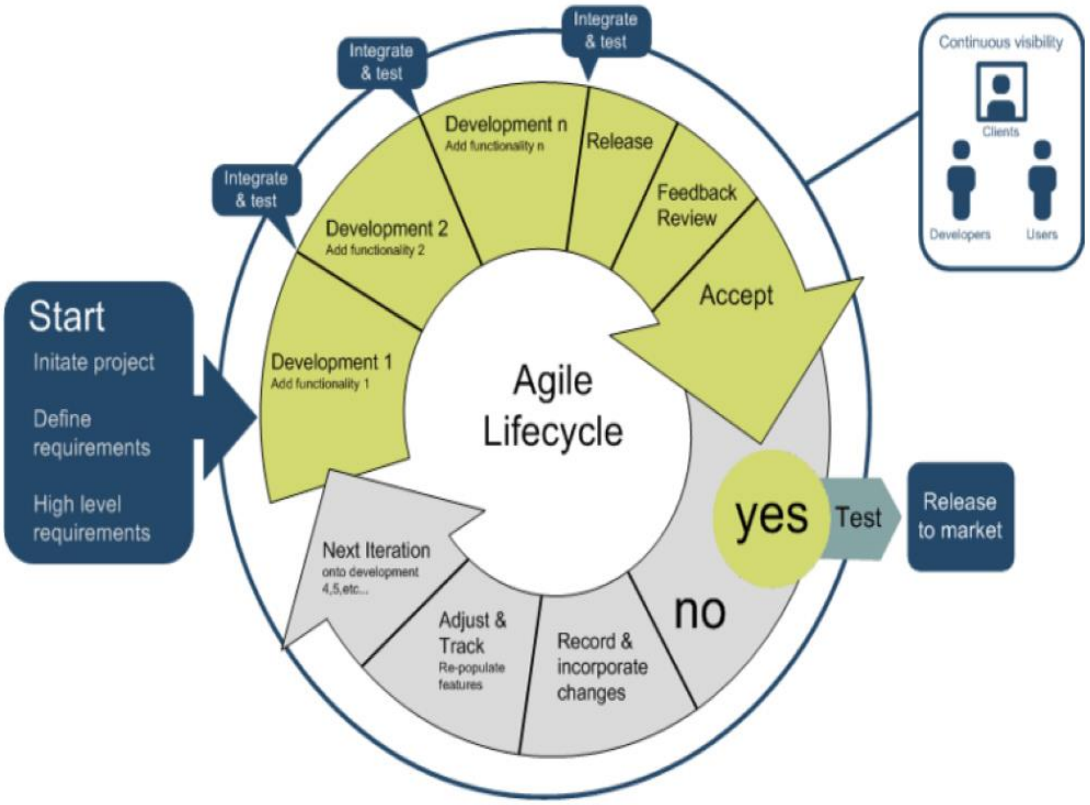


Figure 4.1.1-1 Agile Model

4.1.2 Rapid Application Development (RAD) Model:

RAD model represents a software development framework which emphasizes on speedy creation as well as iteration. It is particularly appropriate for projects that require flexibility, speed, and active user interaction. RAD permits adjustments to be made to the system depending on user feedback and growing requirements, making it flexible. The RAD model is suitable for smaller, less complex projects that require rapid development and continual user participation. However, it might be not the ideal choice for wide-ranging or highly organized projects because to its flexibility and propensity for scope creep. RAD's focus on speed over meticulous planning could lead to challenges in the long run, especially if the system matures and grows. A zoo might eventually need more extensive data management, security capabilities, or scalability that RAD might not be able to properly offer.

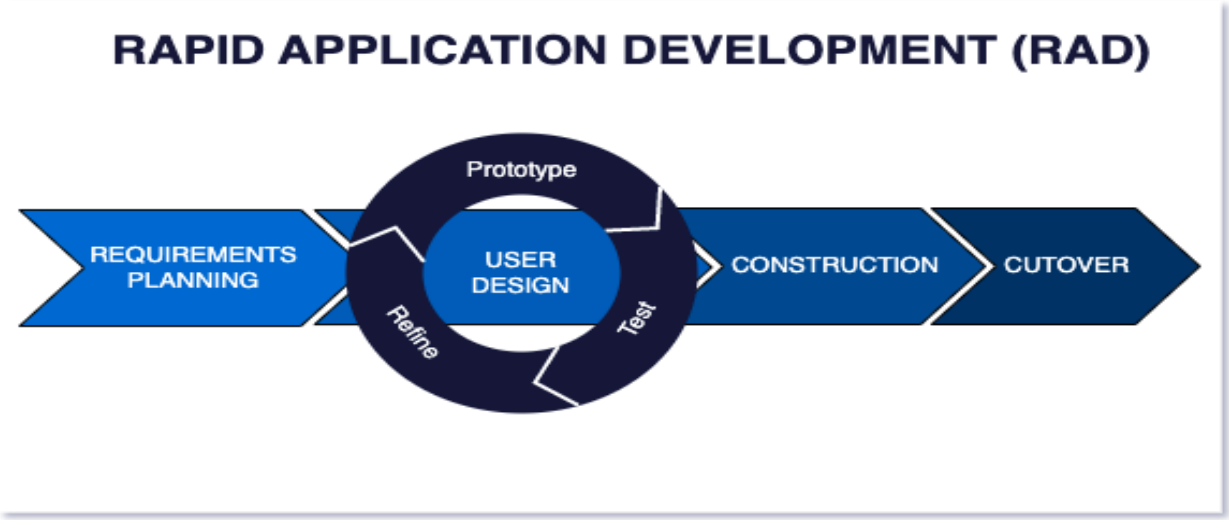


Figure 4.1.2-1 RAD Model

4.2 Why to use

After having a short discussion of two different methodologies every approach has some pros as well cons. Besides this, there are another criterion like feasibility to implement the methodology. Our project, "Zoo Management System" and its functionalities. Agile method is suitable for the system. Here's written the method suitable for the system is below:

- **Agile methods are adaptable:** Since phases in the traditional waterfall method flow into one another, shifting strategies is challenging and can disrupt the rest of the project roadmap. Since software development is a much more adaptable field, project managing rapid changes in the traditional sense can be challenging (resources:agile-methodology, n.d.). This is part of the reason why Agile project management is favored in software development. Our project is for vast number of user. So It's common that, there will be needed to change something as per requirement rapidly.
- **Agile method focuses on customer needs:** One of the unique aspects of software development is that teams can focus on customer requirements much more closely than other industries. With the rise of cloud-based software, teams can get feedback from their actual customers rapidly. Since customer satisfaction is a key driver for software development, it's simple to see why it was included in the Agile process. By interacting with customers, Agile teams can prioritize features that focus on customer requirements. When those needs change, teams can adopt an Agile approach and shift to a different

project (resources:agile-methodology, n.d.). It is the best part of agile methodology which is essential for zoo management system. Because this project is created for user who will get highest benefits by using it.

- **Functional Software:** Working software is the primary measure of progress. The most important thing that teams should strive for with the Agile framework is the product. The goal here is to prioritize functional software over everything else (resources:agile-methodology, n.d.). By collaborating face-to-face, every iteration can play a vital role on appropriate functionality rather than anything else, which I thing is needed in zoo management system.

4.3 Sections of Methodology

The Agile development methodology is a dynamic and repetitive approach to software creation, focusing on teamwork, customer input, and flexibility. It divides the development process into multiple phases, each serving a vital function in delivering high-quality software promptly. These phases are structured to ensure the development team can swiftly and effectively adapt to changes, while consistently enhancing the product.

- Iterative Development: Agile development is built on short, recurring cycles known as sprints. At the end of each sprint, a working version of the product is delivered.
- Collaboration and Communication: Agile promotes teamwork and communication among all stakeholders (developers, clients, and users). Routine check-ins, encourage continual alignment as well as collaboration.
- Customer Involvement: - Continuous involvement of consumers and stakeholders is vital for ensuring the product meets their needs. Feedback is collected after each sprint, allowing for modifications based on user input.
- Flexibility and Adaptability: - Agile approach is highly flexible, allowing teams to make adjustments depending on evolving requirements or feedback. The method embraces change, even late in the development process.
- Focus on Individuals and Interactions: - Agile favors individuals and interactions over processes and tools, stressing personal communication within the development team. It supports self-organizing, cross-functional teams that can adjust to change fast.
- Continuous Improvement: - Agile emphasizes continuous improvement at every level.

Teams reflect on their performance and identify methods to improve both the product and their processes. Regular retrospectives enable teams review what went well and what may be improved.

- Transparency: - Agile emphasizes transparency within the development process, ensuring that everyone involved has visibility into progress, issues, and impending tasks. Tools like burndown charts and task boards assist preserve this transparency.

In simple terms, Agile approach emphasizes on flexibility, cooperation, and iterative development to create functioning software quickly, allowing for regular upgrades and ongoing user feedback.

4.4 Implementation Plan

Here's an overview of how we can utilize the agile model for our project:

- **Establish Project Overview:** The Zoo Management System (ZMS) is designed to help manage various aspects of a zoo, including animal care, control food-supply chain, visitor management, staff management, inventory, and finances. The system will be implemented using Agile methodology to ensure flexibility, regular feedback, and incremental progress.
- **Generate a Product Backlog:** Gather and arrange the project's features and requirements in order of importance. This backlog will provide a detailed list of tasks for execution according to priority.
- **Sprint Planning and set-up:** Determine a list of tasks from the product backlog to address in each sprint and subdivide them into smaller development actions.
- **Organize Daily Briefings:** Hold brief daily sessions with the development team to review progress, address any obstacles, and ensure alignment among all members.
- **Feedback-Driven Development:** Execute the defined requirements in brief cycles or sprints, usually spanning a few weeks. Focus on delivering functional software increments by the end of each cycle after adaption according to the feedback of the stakeholders.
- **Ongoing Testing and Integration:** Consistently assess and incorporate new code updates as integration to ensure the software's quality and functionality. This allows for early detection and resolution of issues.

5 Chapter: Project Plan

Resource management is the process of assigning and leveraging resources efficiently to achieve project tasks and meet project goals. For a successful project, it's crucial to delegate appropriate tasks to the group and prepare optimal use of resources. This project imitates the resource management approach outlined below, with clear roles assigned to all stakeholders, from users to project management, at each stage.

5.1 Work Break Down structure

Breaking work into smaller assignments is a common productivity technique employed to make the work more manageable and accessible. For projects, the Work Breakdown Structure (WBS) is the tool that implements this technique and is one of the most essential project management documents. It solely incorporates scope, cost and schedule baselines ensuring that project plans are in alignment (workbreakdownstructure, n.d.).

Task Name	Duration(Days)	Start	End
Introduction	5	5-Jun-24	9-Jun-24
Initial Study	8	10-Jun-24	18-Jun-24
Literature Review	3	19-Jun-24	21-Jun-24
Mthodology	7	22-Jun-24	28-Jun-24
Project Plan	8	29-Jun-24	6-Jul-24
Feasibility Studies	8	7-Jul-24	14-Jul-24
Foundation	6	15-Jul-24	20-Jul-24
Exploration	5	21-Jul-24	25-Jul-24
Engineering	7	26-Jul-24	1-Aug-24
UI/UX development	7	2-Aug-24	8-Aug-24
Development	35	9-Aug-24	13-Sep-24
Testing	7	14-Sep-24	20-Sep-24
Implementation	8	21-Sep-24	28-Sep-24
Critical Epraisal & Evolution	3	29-Sep-24	1-Oct-24
Lesson Learned	2	2-Oct-24	3-Oct-24
Conclution	3	4-Oct-24	6-Oct-24
	122		

Figure 4.1.2-1 Work Break Down Structure

5.2 Resource Allocation

Resource allocation is the process of assigning and scheduling the resources accessible in the most effective and economical way feasible. Projects will always need resources but they can often be scarce. The task, thus, lies with the project manager to determine the appropriate timing and allocation of those resources within the project schedule. In project management, resources are often in high demand but low in availability. This fact puts project managers in a position where they must strategize the best methods to use what they have. They need to determine who does what, when, and with which resources or support. It's a balancing act that requires keen insight into the project's requirements and the capabilities of the resources at on hand. This guide will discuss the positive impacts of effective resource allocation, such as enhanced team performance and better project outcomes. We'll address prospective challenges, which includes shortages of resources and the complexities of people management. We'll also cover the best practices for resource assignment and provide insights on how to effectively manage and deploy resources throughout a project's life cycle (project-management-guide:FAQ, n.d.).

Task Name	Duration(Days)	Resource
Introduction	5	Analyst, user
Initial Study	8	Analyst
Literature Review	3	Analyst, Team Leader
Mthodology	7	Analyst, Developer, Project manager
Project Plan	8	Analyst, Team Leader, Project manager
Feasibility Studies	8	Analyst, Team Leader, Project manager, user
Foundation	6	Analyst, Team Leader
Exploration	5	Analyst, Team Leader, Developer, Designer
Engineering	7	Project Manager, Team Leader
UI/UX development	7	User, Team Leader, Developer, Designer
Development	35	Analyst, Developer, Tester
Testing	7	User, Team Leader, Developer, Tester
Implementation	8	Project Manager, Developer, Tester
Critical Epraisal & Evolution	3	Analyst, Developer, User
Lesson Learned	2	Analyst, Developer
Conclution	3	Analyst
	122	

Figure 4.1.2-1 Resource Allocation

5.3 Time Boxing

In time boxing model, development is done iteratively as in the iterative improving model.

Nevertheless, in time boxing methodology, each iteration occurs in a timebox of fixed duration. The functionality to be developed is adjusted to suit the duration of the timebox. Additionally, each timebox is divided into a sequence of fixed phases where each stage executes a clearly defined task (analysis, implementation, and deploy) which can be done separately. This approach also requires that the time duration of each phase is approximately equal so that pipelining concept is employed to have the reduction in development time and product releases. There is a dedicated team for each stage in order that the work can be done in pipelining. Consequently, stages should be chosen in such a manner that each stage accomplish some logical unit of work that turns into the input for next stage (software-engineerin:timeboxing-model, n.d.).

Time Boxing	Task Name	Duration(Days)	Resource
TB1	Introduction	5	Analyst, user
	Initial Study	8	Analyst
TB2	Literature Review	3	Analyst, Team Leader
	Mthodology	7	Analyst, Developer, Project manager
TB3	Project Plan	8	Analyst, Team Leader, Project manager
	Feasibility Studies	8	Analyst, Team Leader, Project manager, user
	Foundation	6	Analyst, Team Leader
TB4	Exploration	5	Analyst, Team Leader, Developer, Designer
	Engineering	7	Project Manager, Team Leader
	UI/UX development	7	User, Team Leader, Developer, Designer
TB5	Development	35	Analyst, Developer, Tester
TB6	Testing	7	User, Team Leader, Developer, Tester
	Implementation	8	Project Manager, Developer, Tester
TB7	Critical Epraisal & Evolution	3	Analyst, Developer, User
	Lesson Learned	2	Analyst, Developer
	Conclution	3	Analyst
		122	

Figure 4.1.2-1 Time Boxing

5.4 Gantt chart

A Gantt chart is a powerful visual tool designed to simplify project management by presenting tasks, schedules, and timelines in a clear chronological format. It provides an organized, easy-to-understand graphical representation of project activities, their durations, and how they relate to one another. By illustrating task dependencies and overlapping timelines, a Gantt chart helps teams identify potential bottlenecks and streamline workflows. This tool not only facilitates effective planning and coordination but also enhances tracking and monitoring of a project's progress over time. Stakeholders can quickly see which tasks are on track, delayed, or upcoming, enabling proactive decision-making and adjustments as needed. Whether used for small projects or complex, multi-phase initiatives, Gantt charts are invaluable for ensuring that deadlines are met, resources are allocated efficiently, and the overall project remains aligned with its goals.

Project Lead: Tahmid

WBS	Task	Priority	Resource	Start	Finish	Duration
1	Introduction	NORMAL		Wed 05-Jun-24	Sun 09-Jun-24	3
2	Initial Study	NORMAL		Mon 10-Jun-24	Mon 17-Jun-24	6
3	Literature Review	NORMAL		Wed 19-Jun-24	Fri 21-Jun-24	3
4	Methodology	HIGH		Sat 22-Jun-24	Tue 02-Jul-24	7
5	Project Plan	HIGH		Sat 29-Jun-24	Wed 10-Jul-24	8
6	Feasibility Studies	NORMAL		Sun 07-Jul-24	Tue 16-Jul-24	7
7	Foundation	HIGH		Mon 15-Jul-24	Mon 22-Jul-24	6
8	Exploration	NORMAL		Sun 21-Jul-24	Fri 26-Jul-24	5
9	Engineering	HIGH		Fri 26-Jul-24	Mon 05-Aug-24	7
10	UI/UX Development	HIGH		Fri 02-Aug-24	Mon 12-Aug-24	7
11	Development	HIGH		Fri 09-Aug-24	Thu 26-Sep-24	35
12	Testing	HIGH		Sat 14-Sep-24	Fri 20-Sep-24	5
13	Implementation	HIGH		Sat 21-Sep-24	Sat 28-Sep-24	5
14	Critical Epraisal & Evolution	NORMAL		Sun 29-Sep-24	Tue 01-Oct-24	2
15	Lesson Learned	NORMAL		Wed 02-Oct-24	Thu 03-Oct-24	2

Figure 4.1.2-1 Gantt Chart

6 Chapter: Feasibility Study

6.1 Every viable types of feasibility study:

6.1.1 Operational Feasibility:

The project is operationally feasible because by this the efficiency of the zoo management system will be increased without doing any kind of hamper or loss. If we can collect the daily data of the food supply in by enlist it in the database, it will be less time consuming as well as can provide clear report to the superior. Also, disease can be handled instantly by allotting an attended doctor.

6.1.2 Technical Feasibility:

For achieving the operational things there will be needed phone camera which is available to everyone. To perform this kind of activities I will use web based platform rather than desktop or apps. Because it will be cost effective as well as can be operate easily. To collect data for the disease handling and food supply there are two ways. One is directly from the zoo and another one is self-basis authentic data collection. As, there are communication gap so self-basis authentic data collection is more feasible than collecting data from zoo. But to give more accurate result, the data from zoo will be more preferable.

6.1.3 Economic Feasibility:

Web based solution is more feasible than hybrid app based solution. On the other hand, for the phone camera, primarily not much cost effective but in terms of clear report and ensuring food it will be cost effective than man power. And if the animals of zoo are healthy, and facilities can be provided properly then more visitors will visit which will bring more profit. Why not approximately 4 million people visits every year in the national zoo. So it will be also feasible economically.

6.1.4 Market analysis driven by feasibility determinants:

There is one website for the national zoo of Bangladesh. Hence there are many lacking for providing facilities. For instance, No online ticket facility, No navigation system within the zoo etc. Few news shows that different kinds of animal are died because of starvation and for some common diseases but the site is not helpful solving these problem. On the other hand, by this project all the problem can be solved for the market.

7 Chapter: Foundation

7.1 Problem area identification:

To establish a successful system there is an essential point to identify problem area in project management which will assist to face the key challenges or issues that must be resolved. It is a systematic path which help manager to overcome the challenges with best solution to them. Here are given some steps that often engaged in the solution process to identify problematic areas:

- **Collecting Data and Information:** Collecting data is very important step in software development for gathering requirements and gaining important insights into user behavior and demands. Developers can better understand consumer preferences, problems, and expectations by turning it into information. By keeping in mind all the preferences, developers can effectively build the system according to user needs, leading to improved user experiences and higher levels of user satisfaction (About:goodworklabs, n.d.). Project managers play a crucial role in collecting information from stakeholders and analyzing environmental or outside variables influencing project performance. This process involves identifying key inputs that influence decision-making and project outcomes. To achieve this, project managers employ various techniques such as conducting surveys to gather feedback, holding interviews for in-depth insights, performing detailed research to explore relevant trends, and reviewing existing documents to understand historical data and context. These methods ensure a comprehensive understanding of the project environment, enabling better planning, risk assessment, and successful execution.
- **Analyzing Data:** After collecting all the data, project manager evaluates user behavior patterns, trends, sentiments, etc., based on detecting crucial areas of the system. By this process, the project manager can understand the vital sectors that need to be handled cautiously and observe their consequences on the project's objectives.
- **Priorities the Problematic Areas:** Based on data analyzation, project manager find out severity of the problem. Like which area must be built at first and which one should be built less priorities. Based on the priority list developer can decide the key functionality

of the system. Priorities the functionality, has the capacity to impact project success and the probability of resolution within the project's constraints.

- **Outlining Problem Statements:** To enlist the identified problems clearly, there are some techniques for developer. Specific, achievable, quantifiable, pertinent and time-specific statements of issues (SMART) enable focused efforts for resolving challenges.
- **Collaboration with Stakeholder:** To validate the requirement of the functionalities, project managers collaborate with stakeholders. And add or adapt functionality according to feedback from stakeholder. Stakeholder perspectives are crucial for comprehensively understanding the issues and obtaining their endorsement for proposed solutions.
- **Developing Solutions:** By determining as well as assessing trouble areas, managers of projects can commence designing possible approaches. Conversations, feasibility assessments, different strategies, and advice from experts may all be part of this process.

7.2 Interview:

A project manager can perform this most popular data collecting strategy with a stakeholder, either through direct one-on-one interactions with people or through group interviews. For this, the project manager asks questions based on pre-defined questions for detailed responses or just asks questions about what they desire. This allows for further investigation of topics along with the capacity to disclose any ambiguity. Interviews may occur in individual, via phone call, or by webinars.

7.3 Questionnaire:

Questionnaires consist of sending structured questions to stakeholders through various channels including internet surveys, paper forms, or electronic questionnaires. They provide a systematic approach to data collecting, which aids in effective data processing and analysis. Questionnaires are especially beneficial when working with large numbers of stakeholders or when closed-ended or scaled questions may effectively capture the essential information.

7.4 Requirement Specification:

The specification of requirements plays an important role for the success of a project, as it guarantees clear communication, aligns expectations, and offers a framework for assessing and validating progress throughout the project lifecycle. A well-defined specification highlights the project objectives, outlining both the desired outcomes and the overall purpose of the project. It also describes the scope, specifying what is included and what is excluded, helping to manage stakeholder expectations and avoid scope creep. In addition to outlining project goals, the specification of requirements plays a crucial role in increasing communication among team members and stakeholders. It provides a shared reference point, assuring alignment across all parties engaged. This document also supports the tracking and validation of progress, allowing a mechanism to determine if the project is accomplishing its objectives and staying within scope. Furthermore, the formulation of requirements drives the testing phase, since it provides the criteria for evaluating the deliverables and determining if they fulfill the established needs. Lastly, it acts as a crucial textual reference for future use, delivering valuable insights for ongoing maintenance, updates, or comparable projects down the line.

7.4.1 Functional Requirements:

- Build user log-in system for different user.
- Authenticate and validate for the user
- Able to book ticket throw online by user.
- Perform CRUD functionality for animal adding by admin
- Create event on different occasion by admin
- Manage job vacancies by admin.
- Adding animal to ensure feed list by zookeeper.
- Check animal to ensure feed list by admin.
- Create animal classification by admin.
- View system functionality by manager.
- Create top breeding list of the weak by admin.
- Manage feedback form by admin

7.4.2 Non-Functional Requirements:

- Rapid response time for an impeccable client engagement.
- Facilitate multi-user access simultaneously excluding compromising functioning by system
- User interface should be user-friendly and simple.
- Confidential user information, such as admin and animal records, must be reliably handled.
- The application must ensure high uptime and availability, with reduced risk of downtime or delay.
- It should incorporate measures for data safeguarding and retrieval to protect against loss and sustain operational flow.
- The website has to cross-platform, supporting various web browsers, operating systems, and devices for broader user access.
- System should be responsive to different display dimensions and resolutions to ensure optimal user interaction across all devices.

7.5 Technology to be implemented:

- Frontend: HTML, CSS, JavaScript
- Backend: Node.js with Python
- Database: MySQL
- Hosting: AWS, Azure, or Google Cloud Platform

8 Chapter: Exploration

8.1 Activity diagram:

Activity diagrams assist understanding of the control flow by illustrating the stages that comprise a system's operation. They indicate the sequence in which activities occur and whether they occur simultaneously (concurrent) or sequentially. These diagrams assist in revealing the mechanisms that initiate specific actions or events within a system. An activity diagram began at an initial point and concludes at a final point, representing various decision paths along the way. They are frequently employed in business and process modeling to show the behavior of a system over time (unified-modeling-language-uml-activity-diagrams, n.d.).

8.1.1 Activity Diagram (1) (Admin process):

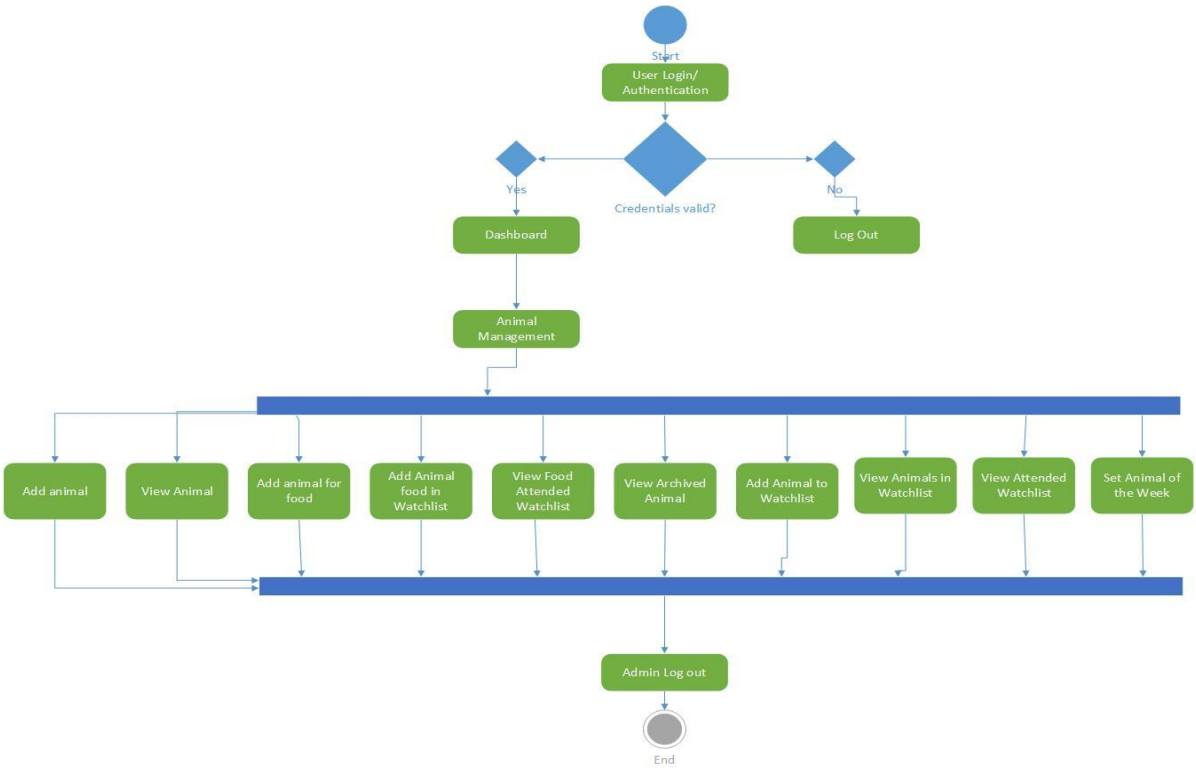


Figure 8.1.1-1 Admin Activity Diagram

8.1.2 Full Activity Diagram:

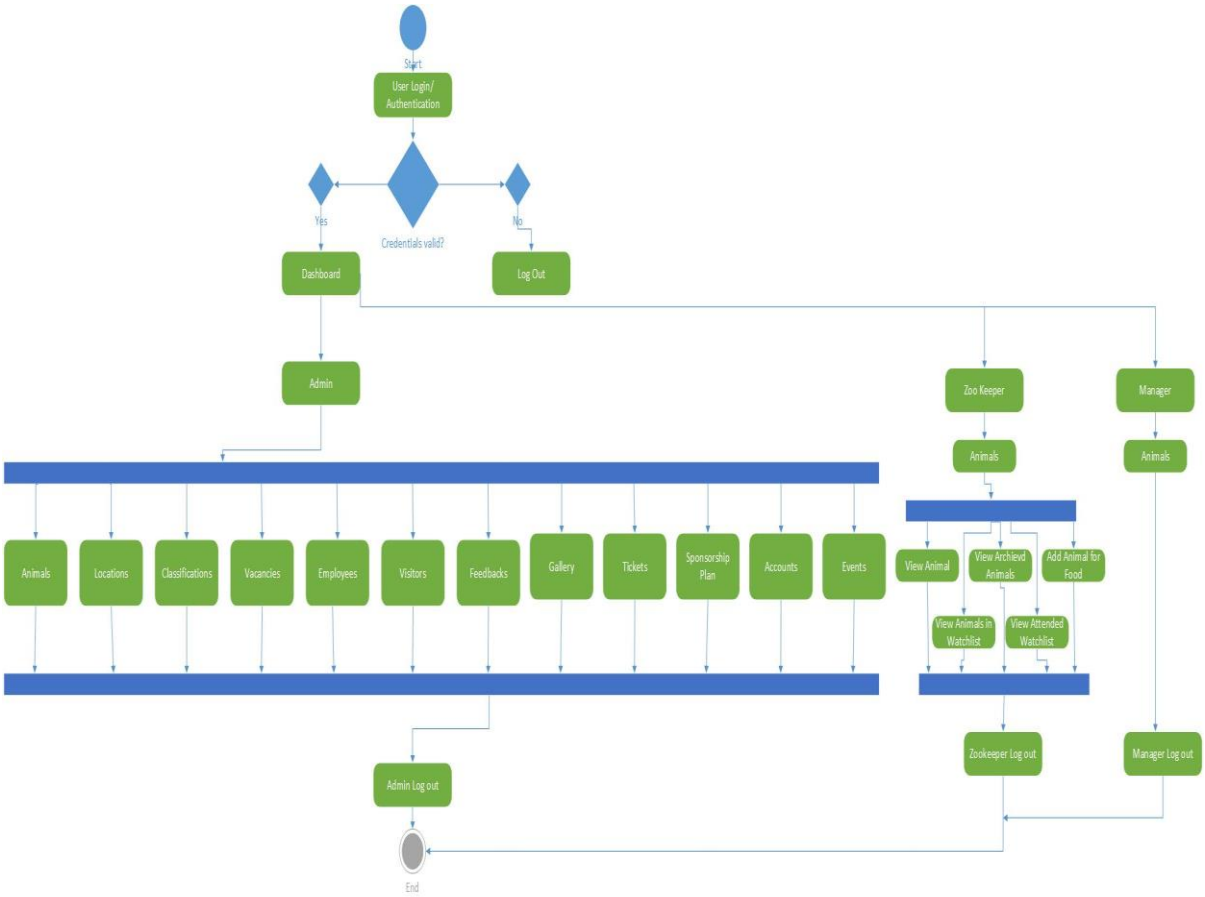


Figure 8.1.2-1 Full System Panel Activity Diagram

8.2 Full system use case:

In the Unified Modeling Language (UML), a Use Case Diagram is a visual representation that demonstrates the interactions between a system and consumers (actors). It defines the operational needs of a system-software. by demonstrating how different users interact with specific functionalities or use cases within the system. Stakeholders, developers, and analysts can benefit from use case diagrams, which offer a comprehensive overview of a system's behavior. This information is essential for comprehending the user's perspective and the interrelationships between various processes. They are essential for the establishment of the system's scope and requirements.

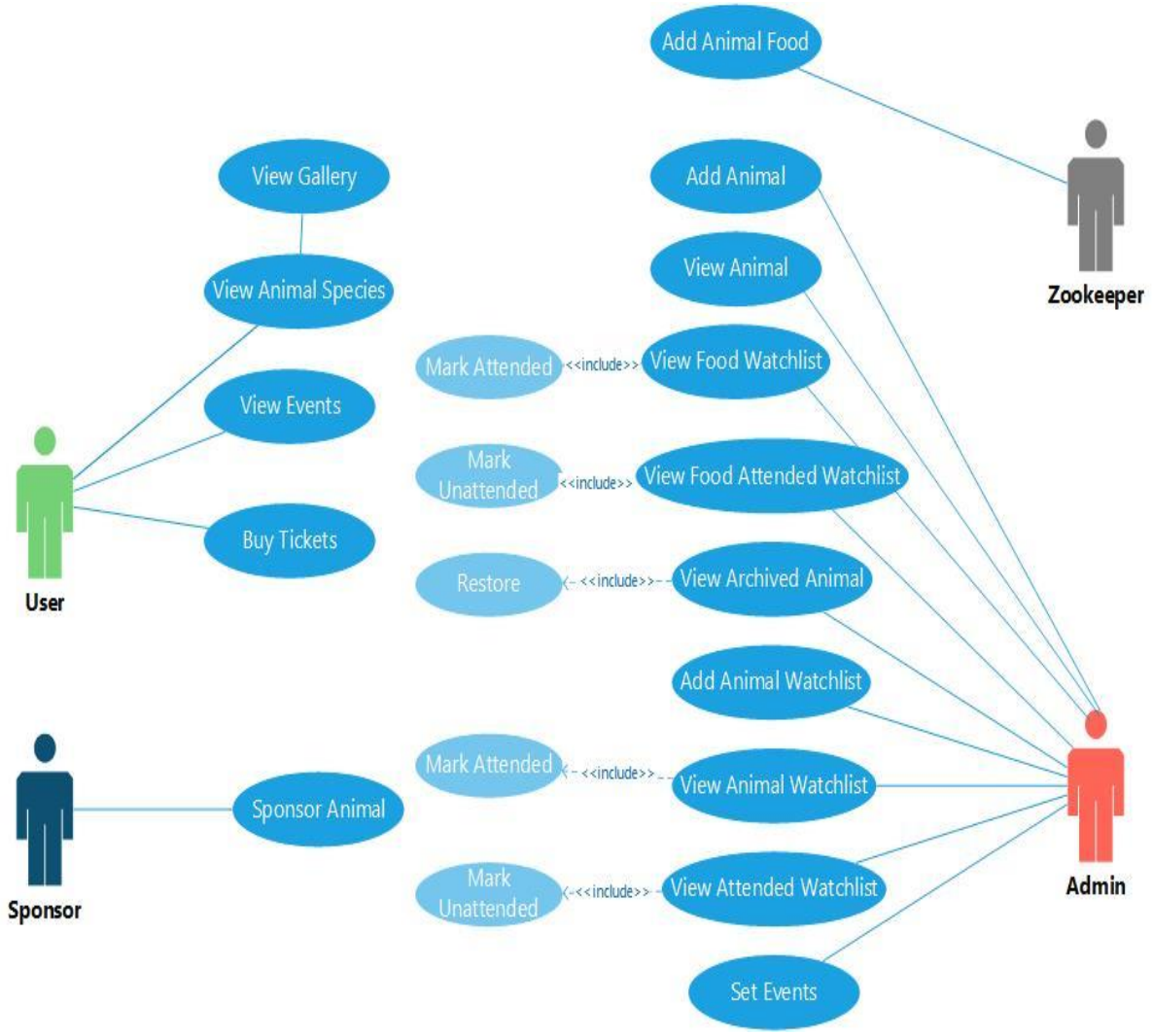


Figure 8.1.2-1 Full System Use Case

8.3 Requirement Catalog:

- Admin can supervise the data of supplying food
- View watch list of animal sickness by admin
- Mark attended to the animal who are taken care by admin
- Tickets and gallery can be categorized by the admin.
- Tickets can be managed by the visitors.
- Event can be managed by the admin
- Add food supply data by Zookeeper
- Manager can supervise all data related to animal care and food supply
- Animal classification can be crud by Admin

- Sponsor can sponsor animal by applying through Admin approval
- Generate invoice of transaction to the buyer after payment.
- Vacancies, employee can be managed by Admin
- Admin can update notice.
- Build this site for mobile app.

8.4 Prioritized Requirement List (PRL):

Functional Requirement No.	**Functional Requirement**	**Priority**
FR1	Admin can supervise the data of supplying food	Must Have
FR2	View watch list of animal sickness by admin	Must Have
FR3	Mark attended to the animal who are taken care by admin	Must Have
FR4	Tickets and gallery can be categorized by the admin.	Must Have
FR5	Tickets can be managed by the visitors.	Must Have
FR6	Event can be managed by the admin	Must Have
FR7	Add food supply data by Zookeeper	Must Have
FR8	Manager can supervise all data related to animal care and food supply	Must Have
FR9	Animal classification can be crud by Admin	Must Have
FR10	Sponsor can sponsor animal by applying through Admin approval	Must Have
FR11	Generate invoice of transaction to the buyer after payment.	Should Have
FR12	Vacancies, employee can be managed by Admin	Should Have
FR13	Admin can update notice.	Could Have
FR14	Build this site for mobile app.	Won't Have

Table 8.4-1 PRL

9 Chapter: Exploration

9.1 Module of the system:

Serial of Action	User Action	System Interaction
User Registration:	Create a new account by providing necessary information.	Authenticate user entries, protect user data, and assign a distinct user ID.
Input Animal data:	Add data into animal section according to classification	Store data into the database and view data from database.
Input food data:	Add food data for specific animal. Mark it approval or not.	Store data into the database and view data
Input care unit:	Add animal in sick list and mark it either the animal is taken care of or not.	Store data into the database and view data

Table 9.1-1 Module of the System

9.2 Class diagram:

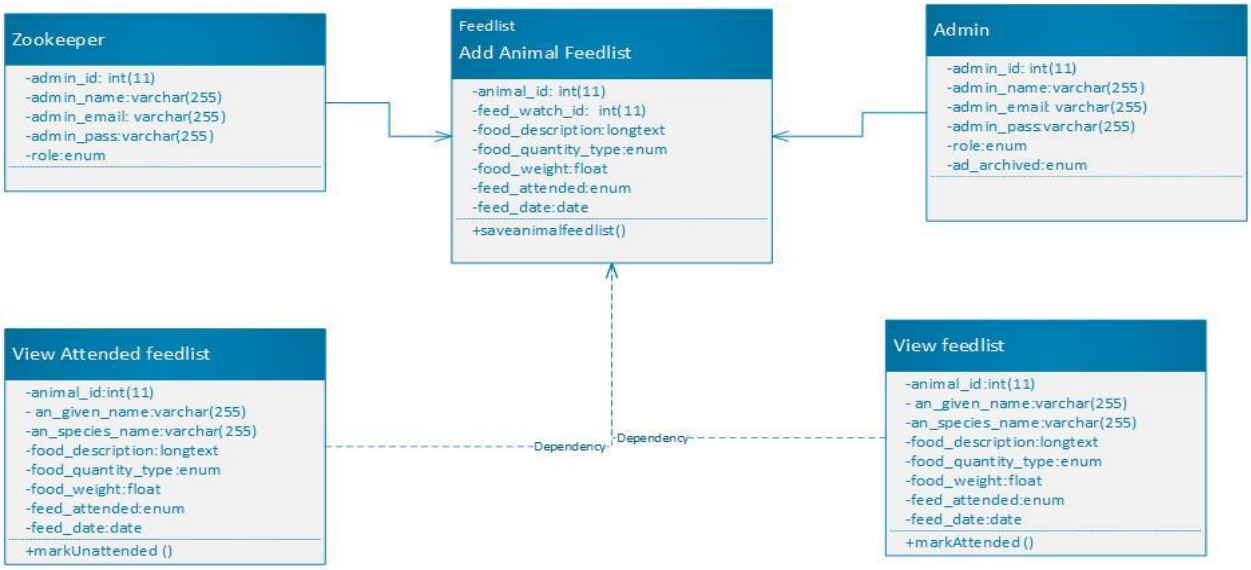


Figure 8.1.2-1 Class Diagram

9.3 Sequence diagram of the system:

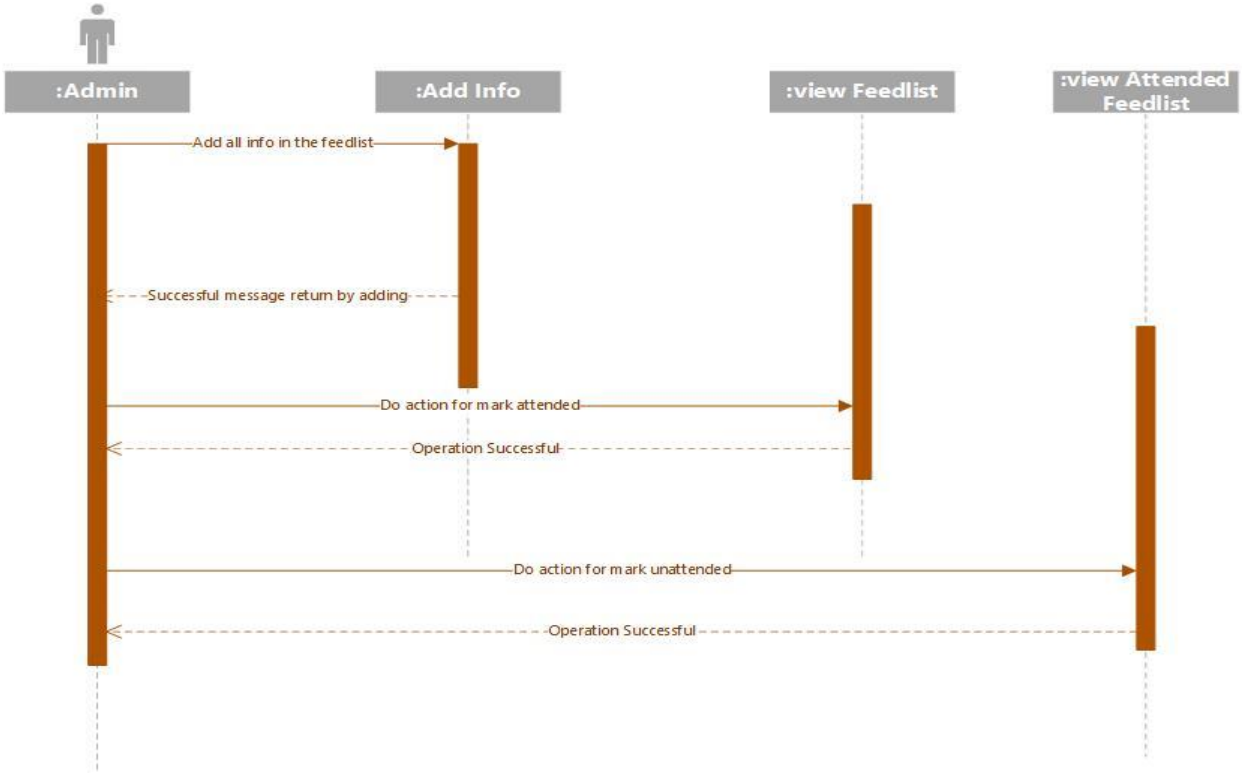


Figure 8.1.2-1 Sequence Diagram

9.4 Prototype of the system:

Figure 8.1.2-1 Prototype of Home Page

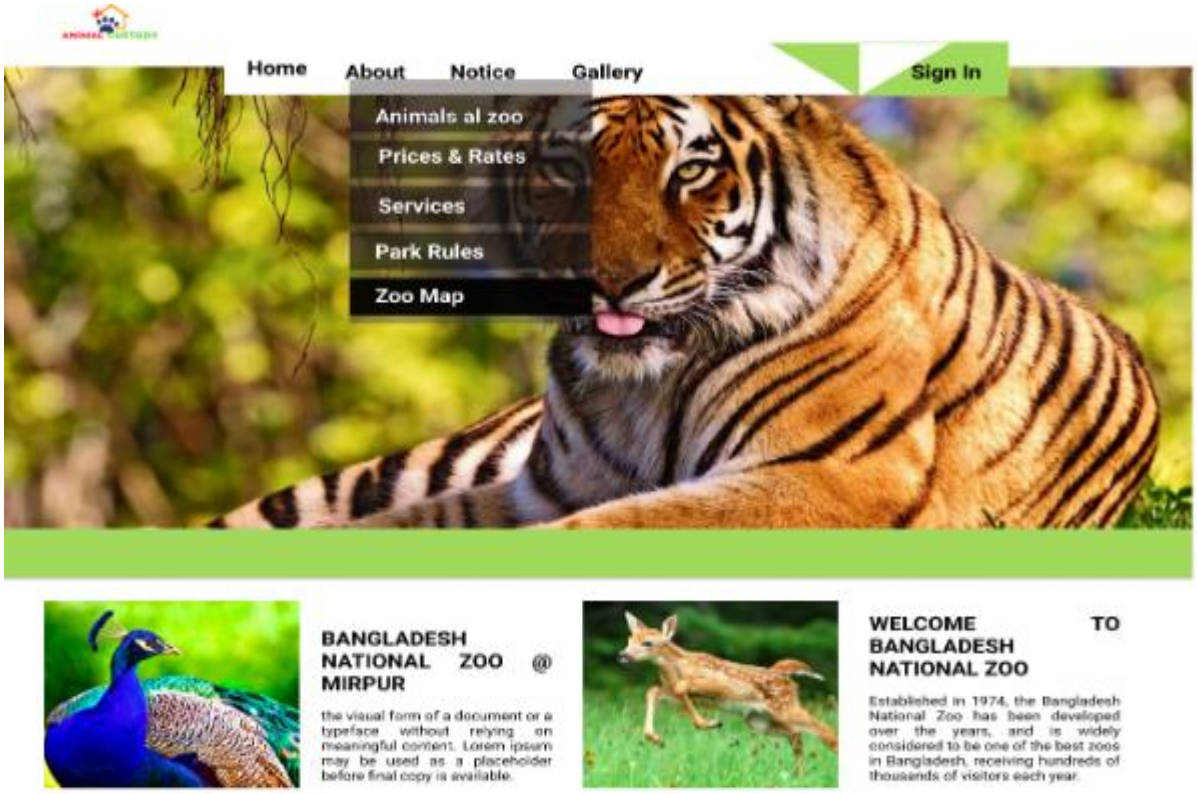


Figure 8.1.2-2 Prototype of Home Page(Hover)

Figure 8.1.2-3 Prototype of Location

Figure 8.1.2-4 Prototype of Admin Log In

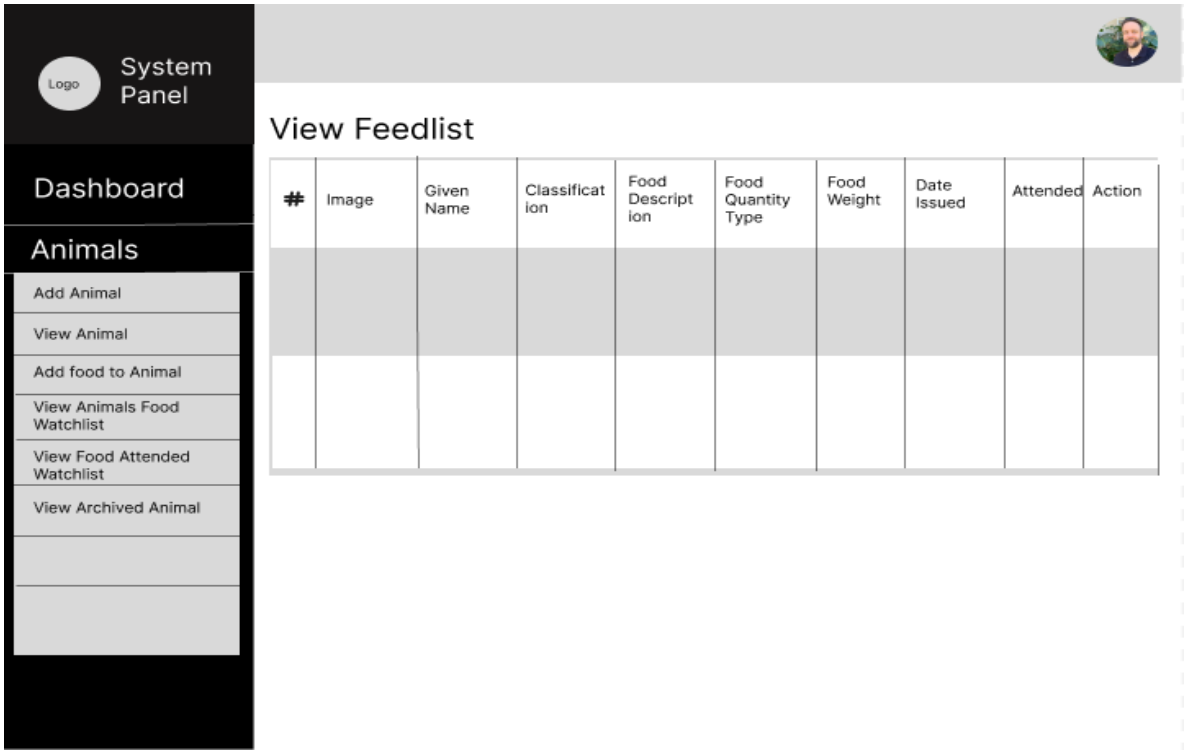


Figure 8.1.2-5 Prototype of Animal Feed list

10 Chapter: Development

10.1 Existing Folder structure of the system:

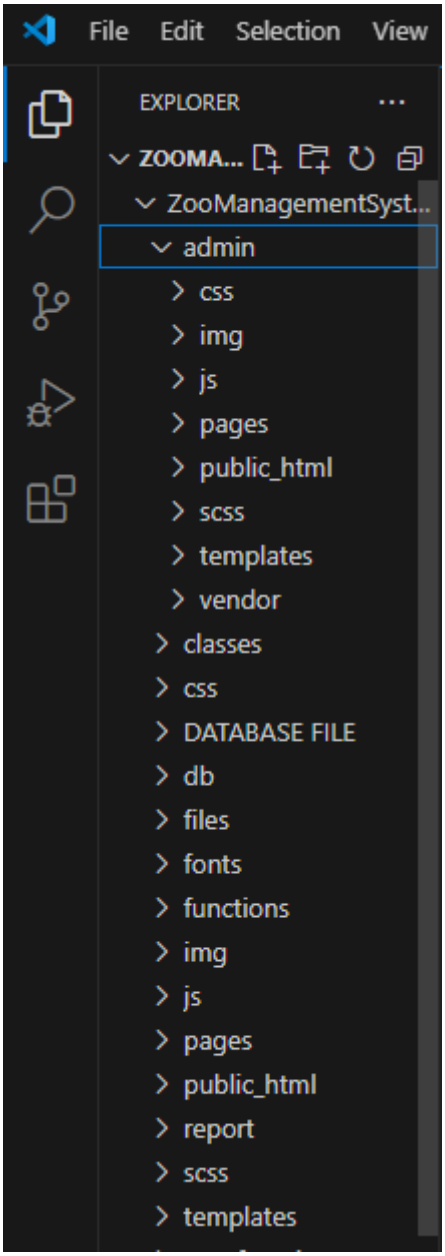


Figure 8.1.2-1 Actual Folder Structure Sample

10.2Core Module Coding Samples-1:

save_animal.php

ZooManagementSystem > admin > pages > save_animal.php

```php
<?php
if (!isset($_SESSION['authenticated'])) {
    header("Location:admin_login");
}

if ($_SESSION['role'] == "zookeeper") {
    header('Location:admin-home');
}

$classTable = new Table('classifications');
$classifications = $classTable->findInDatabase('class_archived', 'false');

$locationTable = new Table('locations');
$locations = $locationTable->findInDatabase('location_archived', 'false');

$mammalTable = new Table('mammals');
$birdTable = new Table('birds');
$fishTable = new Table('fish');
$reptileTable = new Table('reptiles');
$amphibianTable = new Table('amphibians');

$animalImageTable = new Table('animal_images');

//form submit event
if (isset($_POST['submit'])) {
    $archived;
    if (!isset($_POST['an_archived']))
        $archived = 'false';
    else
        $archived = $_POST['an_archived'];

    $common = [
        'animal_id' => $_POST['animal_id']
```

Figure 8.1.2-1 Core Module Coding Sample for Adding Animal, Classification wise

save_animal.php

ZooManagementSystem > admin > pages > save_animal.php

```php
$animalImageTable = new Table('animal_images');

//form submit event
if (isset($_POST['submit'])) {
    $archived;
    if (!isset($_POST['an_archived']))
        $archived = 'false';
    else
        $archived = $_POST['an_archived'];

    $common = [
        'animal_id' => $_POST['animal_id'],
        'an_given_name' => $_POST['an_given_name'],
        'an_species_name' => $_POST['an_species_name'],
        'an_dob' => $_POST['an_dob'],
        'an_gender' => $_POST['an_gender'],
        'an_avg_lifespan' => $_POST['an_avg_lifespan'],
        'location_id' => $_POST['location_id'],
        'an_dietary_req' => $_POST['an_dietary_req'],
        'an_natural_habitat' => $_POST['an_natural_habitat'],
        'an_pop_dist' => $_POST['an_pop_dist'],
        'an_joindate' => $_POST['an_joindate'],
        'an_height' => $_POST['an_height'],
        'an_weight' => $_POST['an_weight'],
        'an_description' => $_POST['an_description'],
        'class_id' => $_POST['class_id'],
        'an_med_record'=>$_POST['an_med_record'],
        'an_transfer'=>$_POST['an_transfer'],
        'an_transfer_reason'=>$_POST['an_transfer_reason'],
        'an_death_date'=>$_POST['an_death_date'],
        'an_death_cause'=>$_POST['an_death_cause'],
        'an_incineration'=>$_POST['an_incineration'],
        'an_archived' => $archived
```

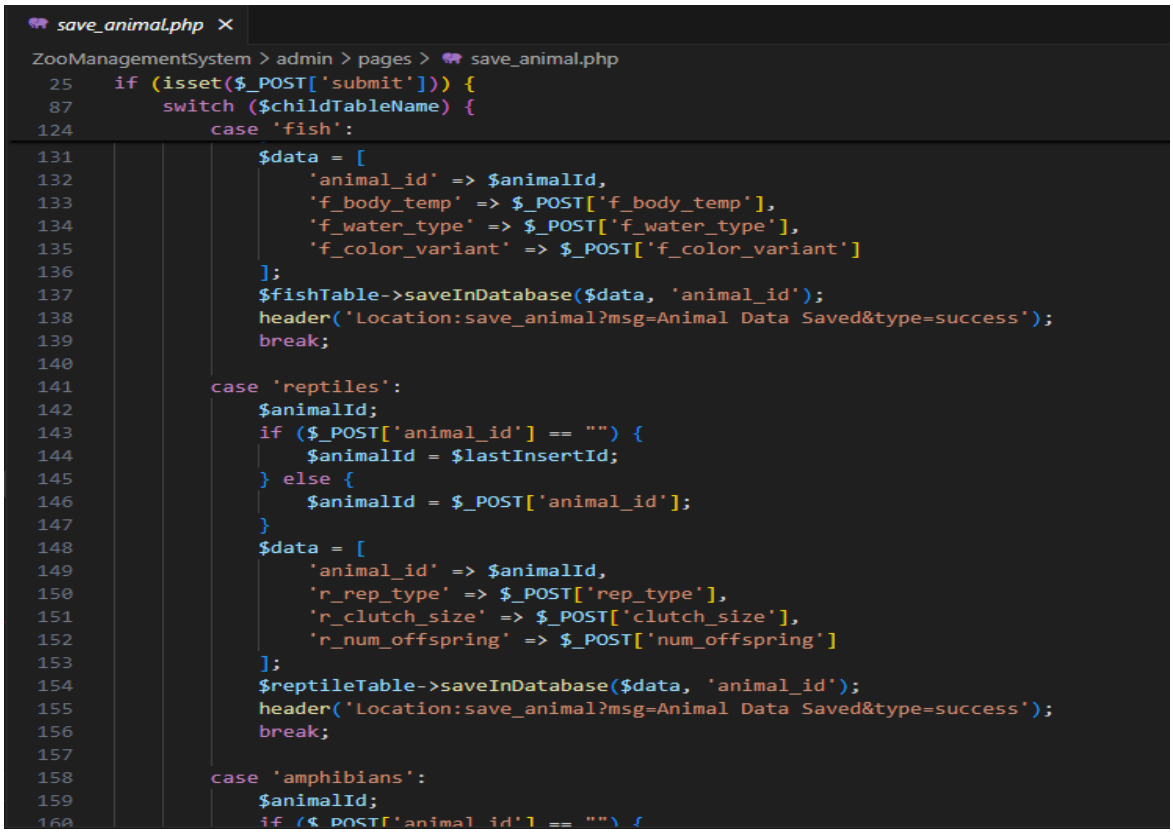
Figure 8.1.2-2Core module Coding Sample for Animal Classification wise

Figure 8.1.2-3 Core Module Coding Sample for Adding Animal Classification wise

Figure 8.1.2-4 Core Module Coding Sample for Adding Animal Classification wise

```
save_animal.php
ZooManagementSystem > admin > pages > save_animal.php
25    if (isset($_POST['submit'])) {
87        switch ($childTableName) {
88            case 'mammals':
105           case 'birds':
106               $animalId;
107               if ($_POST['animal_id'] == "") {
108                   $animalId = $lastInsertId;
109               } else {
110                   $animalId = $_POST['animal_id'];
111               }
112               $data = [
113                   'animal_id' => $animalId,
114                   'b_nest_const' => $_POST['b_nest_const'],
115                   'b_clutch_size' => $_POST['b_clutch_size'],
116                   'b_wingspan' => $_POST['b_wingspan'],
117                   'b_ability_fly' => $_POST['b_ability_fly'],
118                   'b_color_variant' => $_POST['b_color_variant']
119               ];
120               $birdTable->saveInDatabase($data, 'animal_id');
121               header('Location:save_animal?msg=Animal Data Saved&type=success');
122               break;
123
124           case 'fish':
125               $animalId;
126               if ($_POST['animal_id'] == "") {
127                   $animalId = $lastInsertId;
128               } else {
129                   $animalId = $_POST['animal_id'];
130               }
131               $data = [
132                   'animal_id' => $animalId,
133                   'f_body_temp' => $_POST['f_body_temp'],
134                   'f_water_type' => $_POST['f_water_type']
```

Figure 8.1.2-5 Core Module Coding Sample for Adding Animal Classification wise



```
save_animal.php
ZooManagementSystem > admin > pages > save_animal.php
25    if (isset($_POST['submit'])) {
87        switch ($childTableName) {
124           case 'fish':
131               $data = [
132                   'animal_id' => $animalId,
133                   'f_body_temp' => $_POST['f_body_temp'],
134                   'f_water_type' => $_POST['f_water_type'],
135                   'f_color_variant' => $_POST['f_color_variant']
136               ];
137               $fishTable->saveInDatabase($data, 'animal_id');
138               header('Location:save_animal?msg=Animal Data Saved&type=success');
139               break;
140
141           case 'reptiles':
142               $animalId;
143               if ($_POST['animal_id'] == "") {
144                   $animalId = $lastInsertId;
145               } else {
146                   $animalId = $_POST['animal_id'];
147               }
148               $data = [
149                   'animal_id' => $animalId,
150                   'r_rep_type' => $_POST['rep_type'],
151                   'r_clutch_size' => $_POST['clutch_size'],
152                   'r_num_offspring' => $_POST['num_offspring']
153               ];
154               $reptileTable->saveInDatabase($data, 'animal_id');
155               header('Location:save_animal?msg=Animal Data Saved&type=success');
156               break;
157
158           case 'amphibians':
159               $animalId;
```

Figure 8.1.2-6 Core Module Coding Sample for Adding Animal Classification wise

save_animal.php

ZooManagementSystem > admin > pages > save_animal.php

```
25  if (isset($_POST['submit'])) {
87      switch ($childTableName) {
141         case 'reptiles':
158         case 'amphibians':
159             $animalId;
160             if ($_POST['animal_id'] == "") {
161                 $animalId = $lastInsertId;
162             } else {
163                 $animalId = $_POST['animal_id'];
164             }
165             $data = [
166                 'animal_id' => $animalId,
167                 'am_rep_type' => $_POST['rep_type'],
168                 'am_clutch_size' => $_POST['clutch_size'],
169                 'am_num_offspring' => $_POST['num_offspring']
170             ];
171             $amphibianTable->saveInDatabase($data, 'animal_id');
172             header('Location:save_animal?msg=Animal Data Saved&type=success');
173             break;
174         default:
175             header('Location:save_animal?msg=Please create table for the selected classification&type=danger');
176             break;
177     }
178 }
179 //end of form submit
180
181 if (isset($_GET['an_id'])) {
182     $animalTable = new Table('animals');
183     $animalQ = $animalTable->findInDatabase('animal_id', $_GET['an_id']);
184     $animal = $animalQ->fetch();
185
186     $classQ = $classTable->findInDatabase('class_id', $animal['class_id']);
```

Figure 8.1.2-7 Core Module Coding sample For Adding Animal Classification Wise

save_animal.php

ZooManagementSystem > admin > pages > save_animal.php

```
25  if (isset($_POST['submit'])) {
178 }
179 //end of form submit
180
181 if (isset($_GET['an_id'])) {
182     $animalTable = new Table('animals');
183     $animalQ = $animalTable->findInDatabase('animal_id', $_GET['an_id']);
184     $animal = $animalQ->fetch();
185
186     $classQ = $classTable->findInDatabase('class_id', $animal['class_id']);
187     $classData = $classQ->fetch();
188
189     $childTable = new Table($classData['class_table_name']);
190     $childQ = $childTable->findInDatabase('animal_id', $animal['animal_id']);
191     $child = $childQ->fetch();
192 } else {
193     $animal = [];
194     $child = [];
195 }
196
197
198 $title = "Zoo System - Save Animal";
199 $content = loadTemplate('../templates/animal_form_template.php', [
200     'classifications' => $classifications, 'locations' => $locations,
201     'animal' => $animal, 'child' => $child
202 ]);
203
```

Figure 8.1.2-8 Core Module Coding Sample for Adding Animal Classification wise

10.3Core Module Coding Outcome-1:

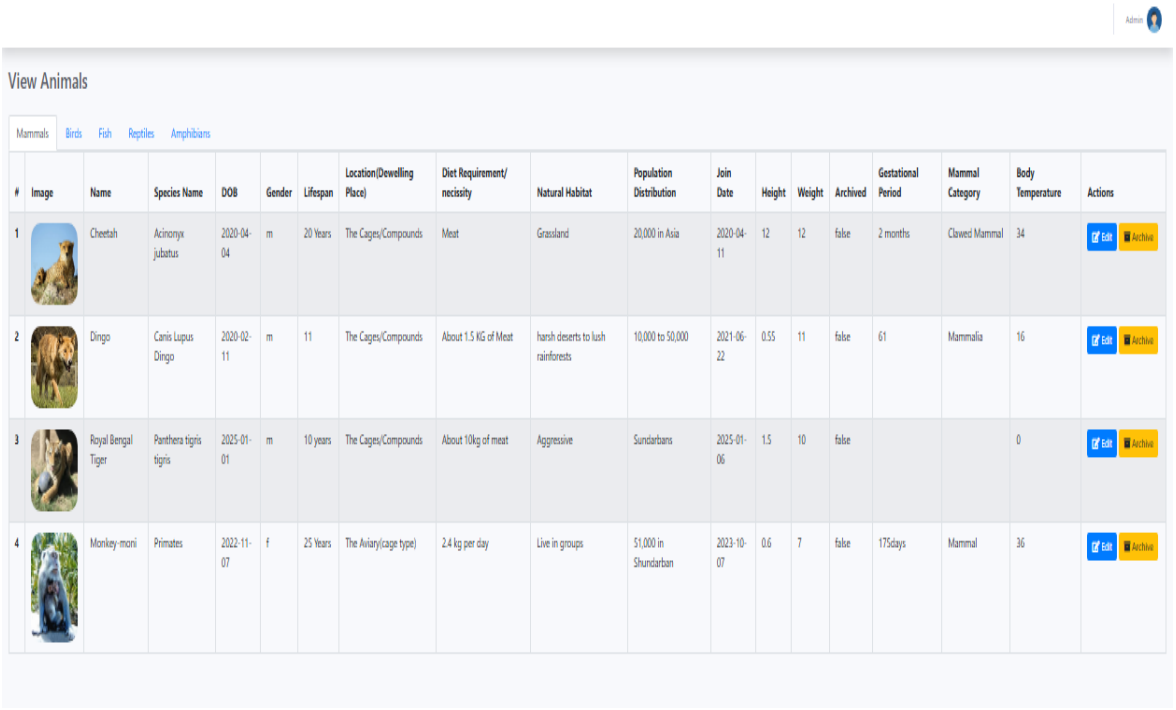


Figure 8.1.2-1 Core Module Coding Outcome for Adding Animal Classification wise

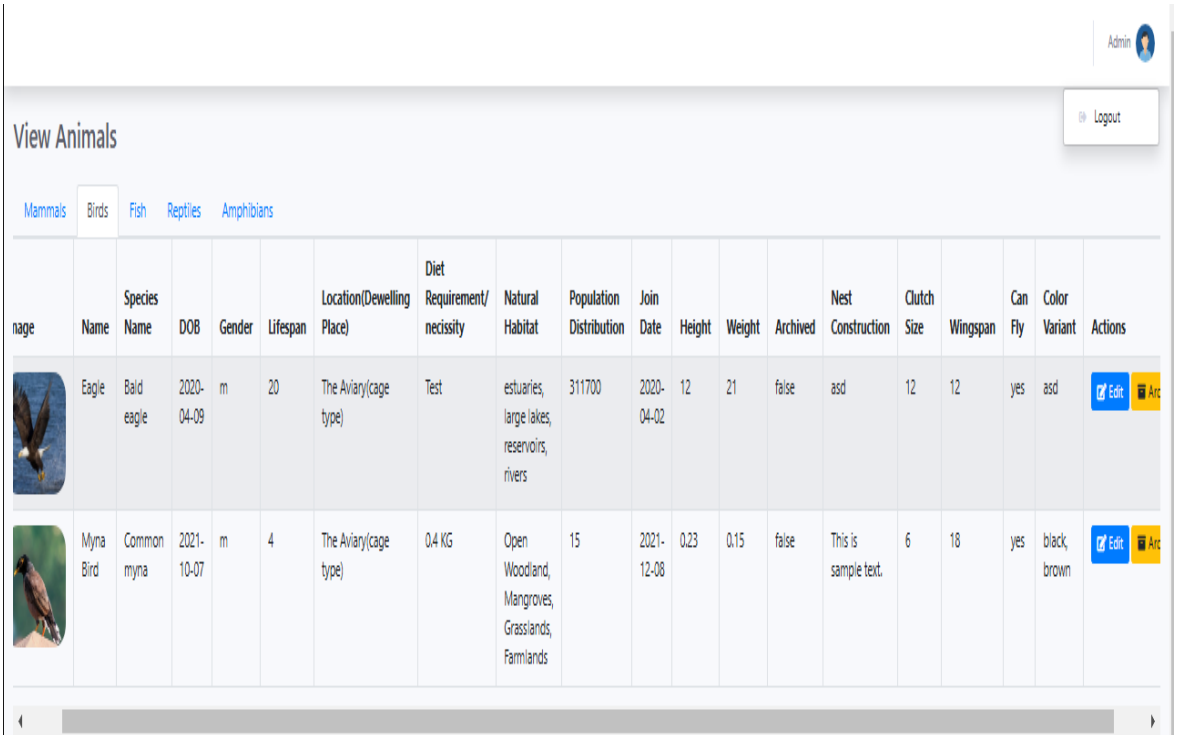


Figure 8.1.2-2 Core Module Coding Outcome for Adding Animal Classification wise

10.4Core Module Coding Samples-2:

animal_watchlist_form.php

ZooManagementSystem > admin > templates > animal_watchlist_form.php

```
<div class="container">
    <div class="row pb-5">
        <div class="col-md-8 order-md-1">
            <?php
            if (isset($_GET['msg'])) {
                echo '<div class="alert alert-success alert-dismissible fade show" role="alert">' . $_GET['msg'] . '
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
                </div>';
            }
            ?>
            <h1 class="h3 mb-4 text-gray-800">Add Animal to Watchlist</h1>
            <form action="" method="post">
                <div class="mb-4 field-required">
                    <label for="select_animal">Select Animal</label>
                    <select class="custom-select d-block w-100" name="animal_id" id="select_animal" required>
                        <?php foreach ($animals as $animal) { ?>
                            <option value="<?= $animal['animal_id'] ?>">
                                <?= $animal['an_given_name'] . ' - ' . $getClassName($animal['class_id']) ?>
                            </option>
                        <?php } ?>
                    </select>
                </div>
                <div class="mb-4 field-required">
                    <label for="issue">Describe the Issue</label>
                    <textarea name="watch_description" placeholder="...." class="form-control" id="issue" cols="30" rows
                </div>
                <div class="mb-4 field-required">
                    <label for="severity">Select Severity</label>
                    <select class="custom-select d-block w-100" name="watch_severity" id="severity" required>
                        <option value="high">High</option>
                        <option value="medium">Medium</option>
```

Figure 8.1.2-1 Core Module Coding Sample for Adding Animal Sick List to Take Care

animal_watchlist_form.php

ZooManagementSystem > admin > templates > animal_watchlist_form.php

```
<div class="container">
    <div class="row pb-5">
        <div class="col-md-8 order-md-1">
            <form action="" method="post">
                <div class="mb-4 field-required">
                    <?php foreach ($animals as $animal) { ?>
                        <option value="<?= $animal['animal_id'] ?>">
                            <?= $animal['an_given_name'] . ' - ' . $getClassName($animal['class_id']) ?>
                        </option>
                    <?php } ?>
                </select>
            </div>
            <div class="mb-4 field-required">
                <label for="issue">Describe the Issue</label>
                <textarea name="watch_description" placeholder="...." class="form-control" id="issue" cols="30" rows
            </div>
            <div class="mb-4 field-required">
                <label for="severity">Select Severity</label>
                <select class="custom-select d-block w-100" name="watch_severity" id="severity" required>
                    <option value="high">High</option>
                    <option value="medium">Medium</option>
                    <option value="low">Low</option>
                </select>
            </div>
            <button class="btn btn-primary btn-lg btn-block" type="submit" name="submit">Add to Watchlist</button>
        </form>
    </div>
</div>
```

Figure 8.1.2-2 Core Module Coding Sample for Adding animal Sick list to Take Care

save_animal_watchlist.php

ZooManagementSystem > admin > pages > save_animal_watchlist.php

```php
<?php
if (!isset($_SESSION['authenticated'])) {
    header("Location:admin_login");
}

$animalTable = new Table('animals');
$animals = $animalTable->findInDatabase('an_archived', 'false');

$classTable = new Table('classifications');

$getClassName = function ($classId) {
    global $classTable;
    $classQ = $classTable->findInDatabase('class_id', $classId);
    $class = $classQ->fetch();
    return $class['class_display_name'];
};

$watchlistTable = new Table('watchlist');

if (isset($_POST['submit'])) {
    unset($_POST['submit']);
    $_POST['watch_attended'] = 'false';
    $_POST['watch_date'] = date("Y-m-d");

    $watchlistTable->insertInDatabase($_POST);
    header('Location:save_animal_watchlist?msg=Animal added to watchlist');
}

$title = "Zoo System - Watchlist";
$content = loadTemplate('../templates/animal_watchlist_form.php', [
    'animals' => $animals,
    'getClassName' => $getClassName
]);
```

Figure 8.1.2-3 Core Module Coding Sample for Adding Animal Sick List to Take Care

view_animal_watchlist.php

ZooManagementSystem > admin > pages > view_animal_watchlist.php

```php
<?php
if (!isset($_SESSION['authenticated'])) {
    header("Location:admin_login");
}
$animalTable = new Table('animals');
$watchlistTable = new Table('watchlist');
if (isset($_GET['attended'])) {
    $watchlists = $watchlistTable->findInDatabase('watch_attended', 'true');
} else {
    $watchlists = $watchlistTable->findInDatabase('watch_attended', 'false');
}

$getAnimalData = function ($animalId) {
    global $animalTable;
    $animalQ = $animalTable->findInDatabase('animal_id', $animalId);
    $animal = $animalQ->fetch();
    return $animal;
};

$classTable = new Table('classifications');

$getClassName = function ($classId) {
    global $classTable;
    $classQ = $classTable->findInDatabase('class_id', $classId);
    $class = $classQ->fetch();
    return $class['class_display_name'];
};

$animalImageTable = new Table('animal_images');
$getImageName = function ($animalId) {
    global $animalImageTable;
    $imageQ = $animalImageTable->findInDatabase('animal_id', $animalId);
```

Figure 8.1.2-4 Core Module Coding Sample for Adding Animal Sick List to Take Care

view_animal_watchlist.php

ZooManagementSystem > admin > pages > view_animal_watchlist.php

```php
$classTable = new Table('classifications');

$getClassName = function ($classId) {
    global $classTable;
    $classQ = $classTable->findInDatabase('class_id', $classId);
    $class = $classQ->fetch();
    return $class['class_display_name'];
};

$animalImageTable = new Table('animal_images');
$getImageName = function ($animalId) {
    global $animalImageTable;
    $imageQ = $animalImageTable->findInDatabase('animal_id', $animalId);
    $images = $imageQ->fetchAll();
    $oneImageName = $images[0]['image_name'];
    return $oneImageName;
};

$title = "Zoo System - View Watchlist";
$content = loadTemplate('../templates/view_watchlist_template.php', [
    'watchlists' => $watchlists,
    'getAnimalData' => $getAnimalData,
    'getClassName' => $getClassName,
    'getImageName'=>$getImageName
]);
```

Figure 8.1.2-5 Core Module Coding Sample for Adding Sick List to Take Care

view_watchlist_template.php

ZooManagementSystem > admin > templates > view_watchlist_template.php

```php
<?php
$badge = [
    'high' => 'danger',
    'medium' => 'warning',
    'low' => 'success'
];
?>
<div class="container-fluid">
    <div class="row">
        <div class="col">
            <?php
            if (isset($_GET['msg'])) {
                echo '<div class="alert alert-success alert-dismissible fade show" role="alert">' . $_GET['msg'] . '
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
                </div>';
            } ?>
            <h1 class="h3 mb-4 text-gray-800">View <?= isset($_GET['attended']) ? "Attended" : "" ?> Watchlist</h1>
            <div class="table-responsive">
                <table class="table table-hover table-striped table-bordered" style="min-width: 600px;">
                    <thead>
                        <tr>
                            <th scope="col">#</th>
                            <th scope="col">Image</th>
                            <th scope="col">Given Name</th>
                            <th scope="col">Classification</th>
                            <th scope="col">Issue</th>
                            <th scope="col">Severity</th>
                            <th scope="col">Date Issued</th>
                            <th scope="col">Attended</th>
                            <?php if ($_SESSION['role'] != "zookeeper") {  ?>
                                <th scope="col">Action</th>
```

Figure 8.1.2-6 Core Module Coding Sample for Adding Animal Sick List to Take Care

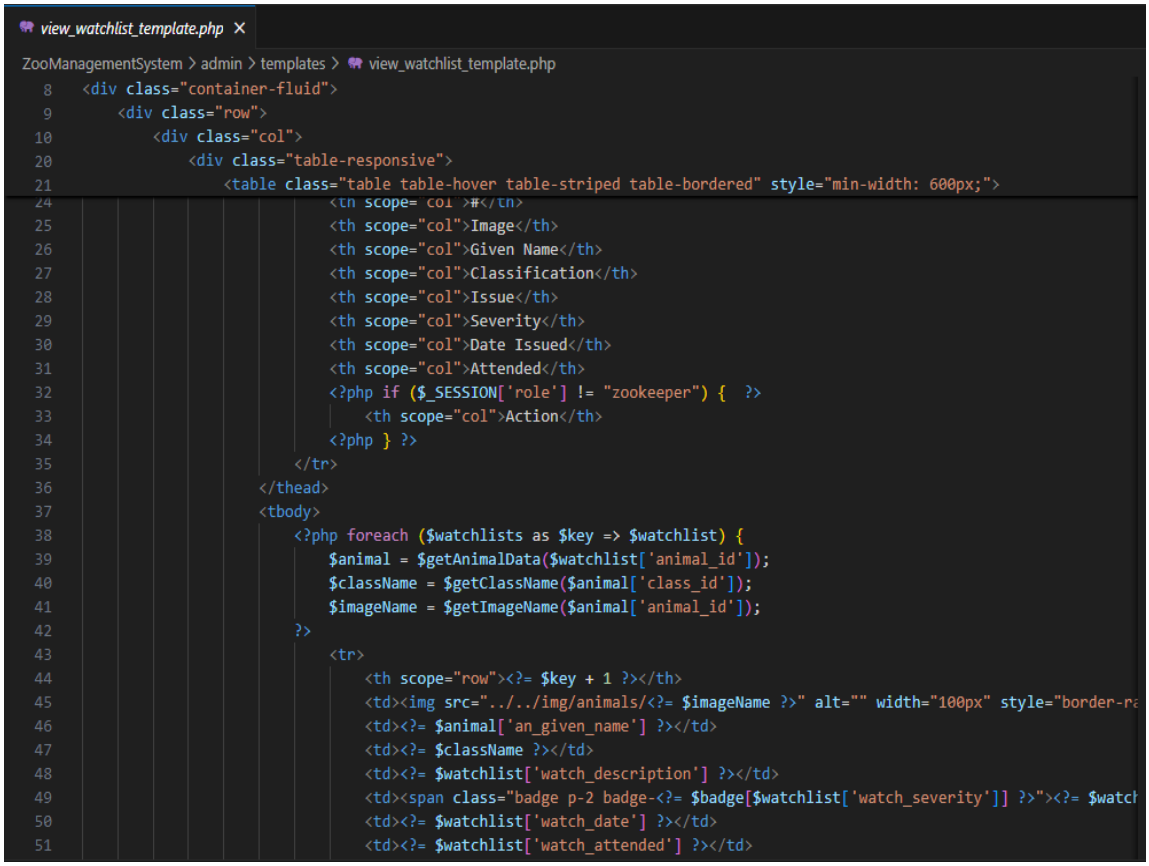


Figure 8.1.2-7 Core Module Coding Sample for Adding Animal Sick list to Take Care

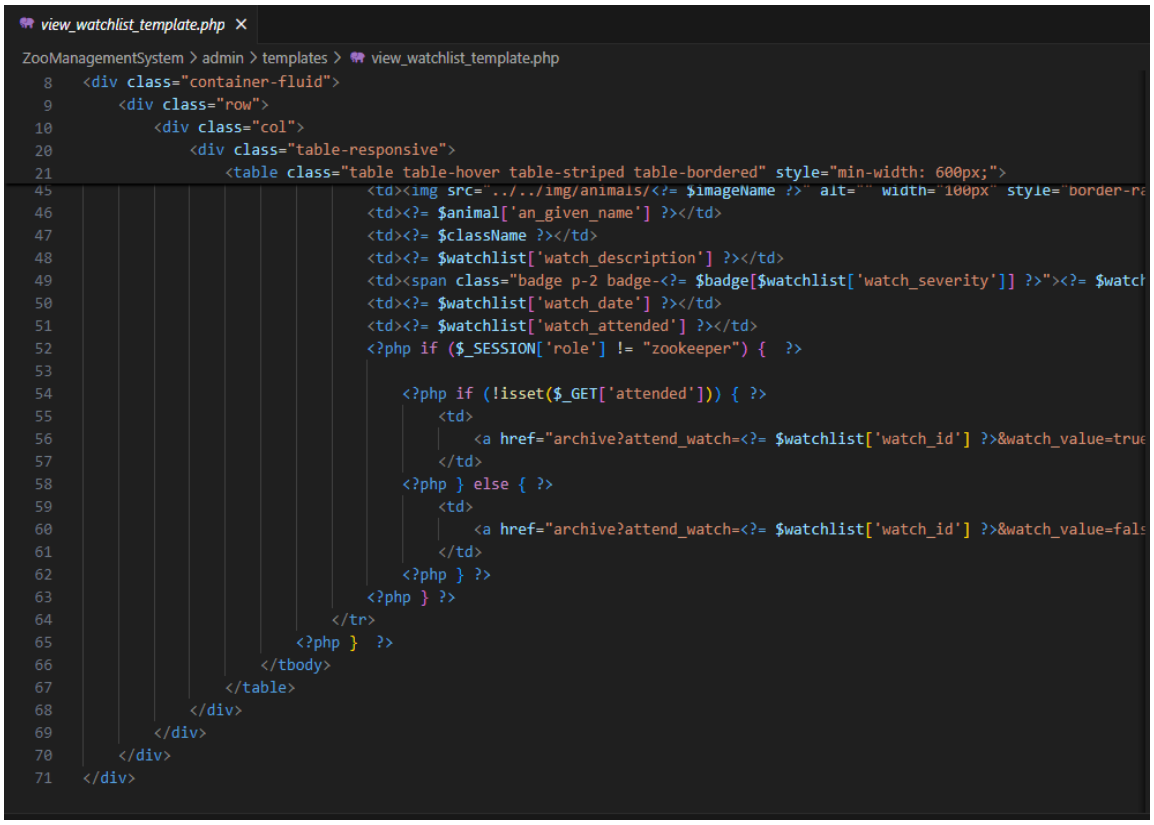


Figure 8.1.2-8 Core Module Coding Sample for Adding Animal Sick List To Take Care

archive.php

ZooManagementSystem > admin > pages > archive.php

```php
if (isset($_GET['set_an_week'])) {
    $data = [
        'animal_id' => $_GET['set_an_week']
    ];
    $animalWeekTable->insertInDatabase($data);
    header('Location:set_animal_of_the_week?msg=Animal of the week set');
}

//attending watchlist
if (isset($_GET['attend_watch'])) {
    $watchlistTable = new Table('watchlist');
    $data = [
        'watch_id' => $_GET['attend_watch'],
        'watch_attended' => $_GET['watch_value']
    ];
    $watchlistTable->saveInDatabase($data, 'watch_id');
    $attendText = $_GET['watch_value'] == "false" ? "&attended=true" : "";
    header('Location:view_animal_watchlist?msg=Operation Successful' . $attendText);
}

//attending feedlist
if (isset($_GET['attend_feed'])) {
    $feedlistTable = new Table('feedlist');
    $data = [
        'feed_watch_id' => $_GET['attend_feed'],
        'feed_attended' => $_GET['feed_value']
    ];
    $feedlistTable->saveInDatabase($data, 'feed_watch_id');
    $attendText = $_GET['feed_value'] == "false" ? "&attended=true" : "";
    header('Location:view_animal_feedlist?msg=Operation Successful' . $attendText);
}
```

Figure 8.1.2-9 Core Module Coding Sample for Adding Animal Sick List to Take Care

10.5Core Module Coding Outcome-2:

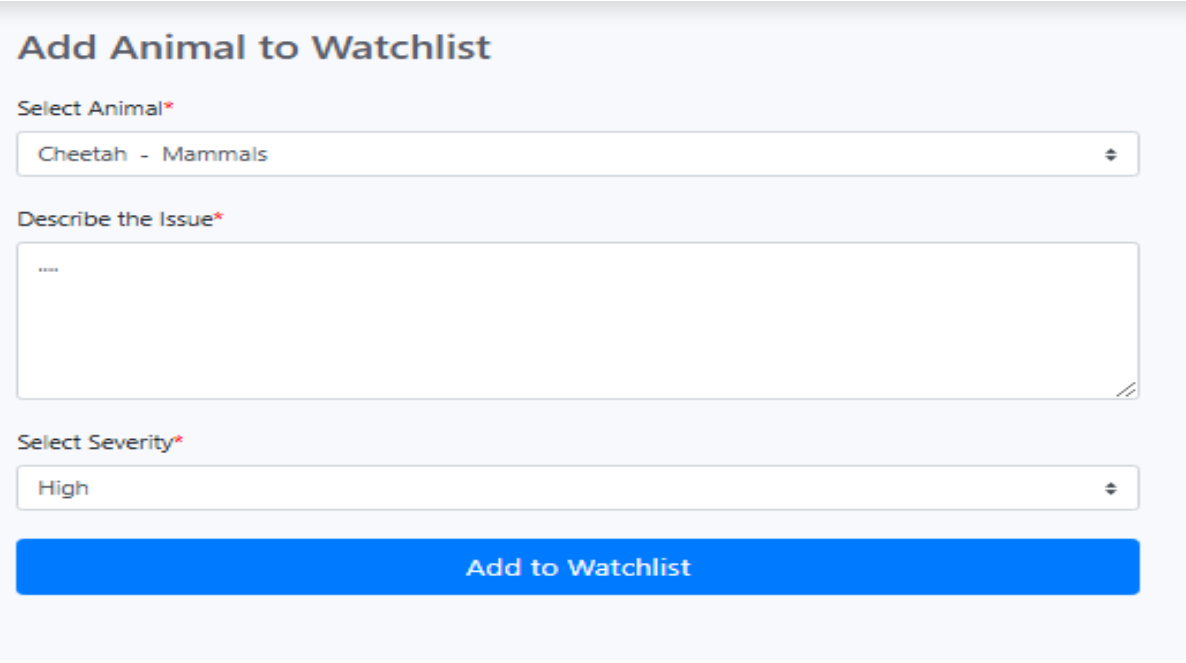


Figure 8.1.2-1 Core Module Coding Outcome for Adding Animal into Watch List Due to Sick List

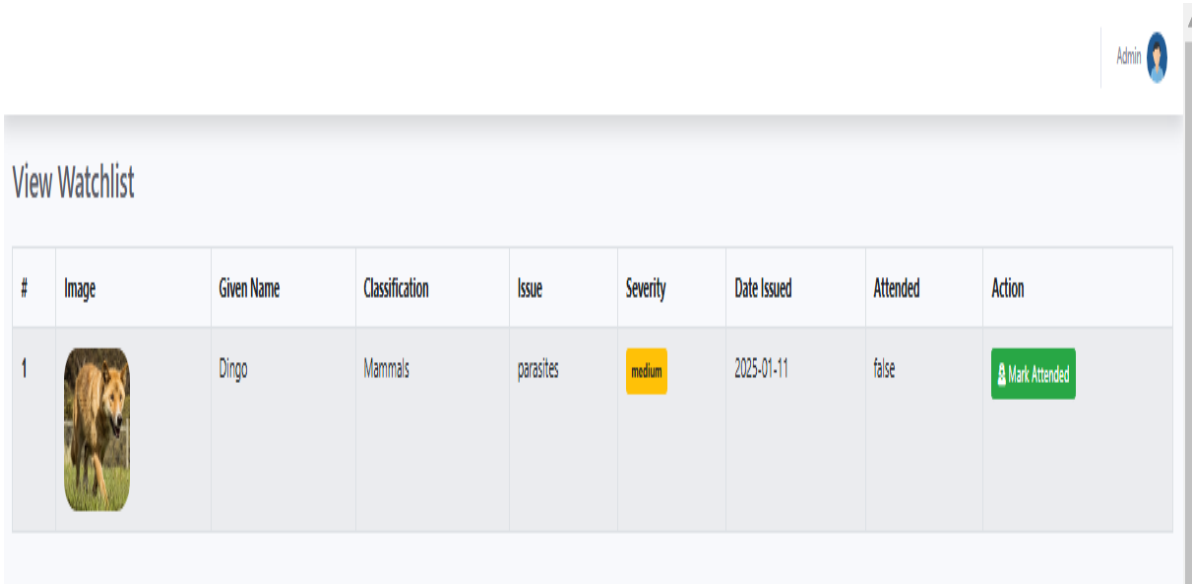


Figure 8.1.2-2 Core Module Coding Outcome for Adding Animal into Watch List due to Sick List

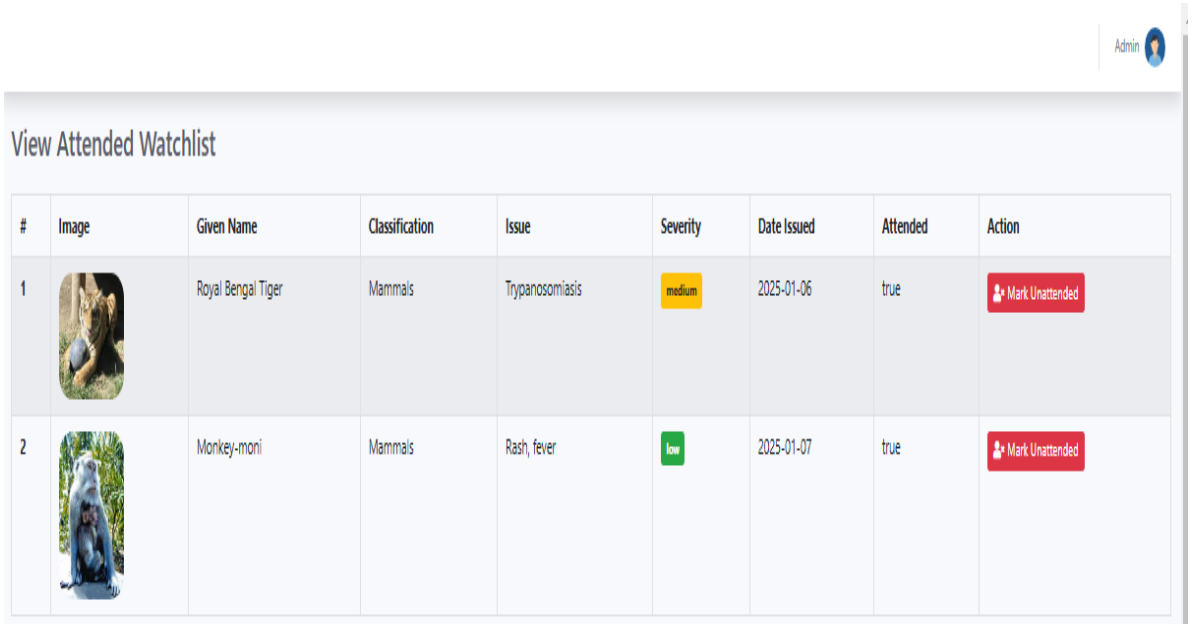


Figure 8.1.2-3 Core Module Coding Outcome for Adding Animal into Watch List Due to Sickness

11 Chapter: Testing

11.1Test Case:

Testing is an essential component of the development of software, aimed at detecting and rectifying faults to guarantee that the application functions as intended. It entails evaluating the system against defined criteria and confirming that all functionalities function as anticipated. Testing will be essential for our project to guarantee effective data storage and retrieval, as well as to validate the performance of authentication elements. Furthermore, testing will encompass verifying the flawless interface between the frontend and backend components, assuring all system elements function cohesively with minimal complications. This thorough procedure will ensure that the system is trustworthy secure, and operates efficiently across all functions.

A test case usually comprises multiple elements that simultaneously constitute an extensive testing strategy. The following elements include:

- **Outline of a test:** A short specific description about the key objectives and achievable goal of the test case.
- **Prerequisite:** Some key conditions must be enlisted and set up prior to performing the test procedure which will help to validate the test based on these.
- **Actions:** Throughout the test phase, a set of operations are performed for the justification of specific function.
- **Presumed Outcome:** The intended reaction or result is observed if the system behaves as expected.
- **Actual Outcome:** Actual result is observed as is it similar with the expected output across the full span of test case execution.
- **Performance Thresholds:** The factors accustomed to evaluate whether the functionality of the test case has succeeded or not according to the presumed outcome.
- **Test Environment:** The particular environment or system setup used for testing, including the browser and operating system involved.
- **Test Data:** Input values, parameter or data required for test case execution.

To create concise and accessible test case documentation, I will primarily outline the test scenario, procedures, actual results, presumed outcomes, and success or failure benchmarks for each test case. Our testing efforts included unit testing, module testing, integration testing, and security testing on various scenarios to ensure the system's features are performing effectively.

11.2 Unit Testing:

11.2.1 Unit Test -1

Test Case Name	Unit Test-1
Test Case	Take Input and storing food data in the database

Test Scenario	Expected Result	Actual Result	Remarks
Storing data in database	Store data with successful message status and view all the data in respect of animal food supply	Store data with successful message status and view all the data in respect of animal food supply	Passed
Required data missing message	Showing a warning message due to fill up the required field	Showing a warning message due to fill up the required field	Passed

Table 11.2-1 Unit test-1

11.2.2 Unit Test -1-Output Visualization:

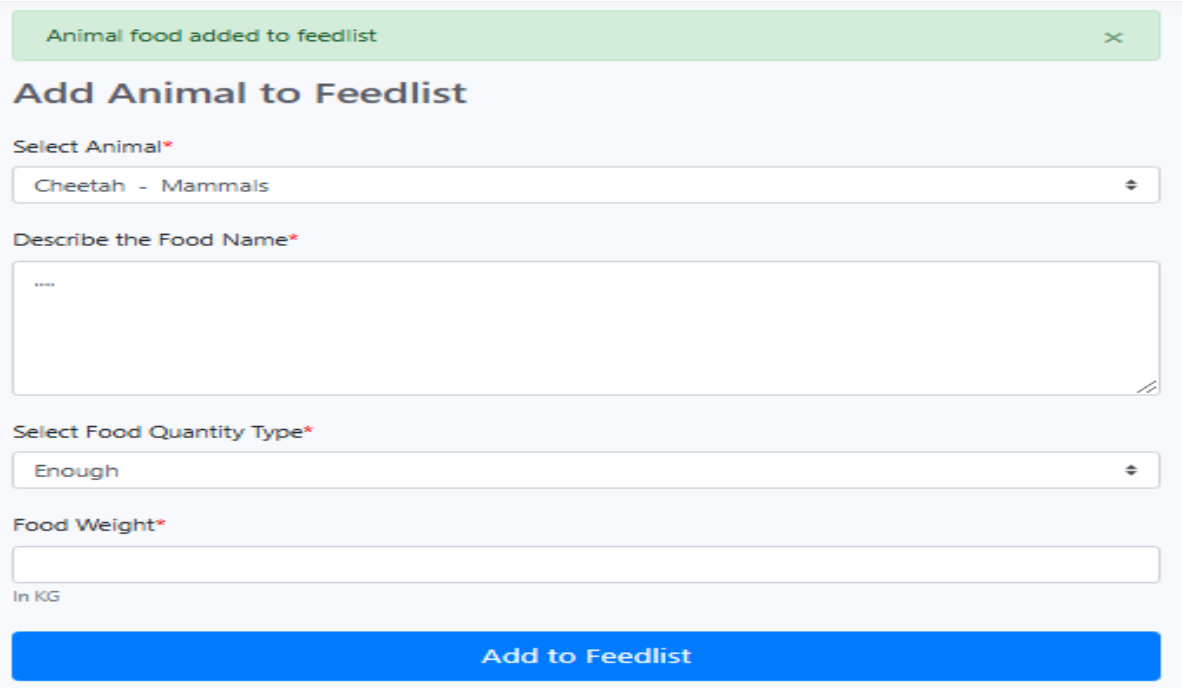


Figure 11.2.2-1 Successfully Storing Data with Message Status

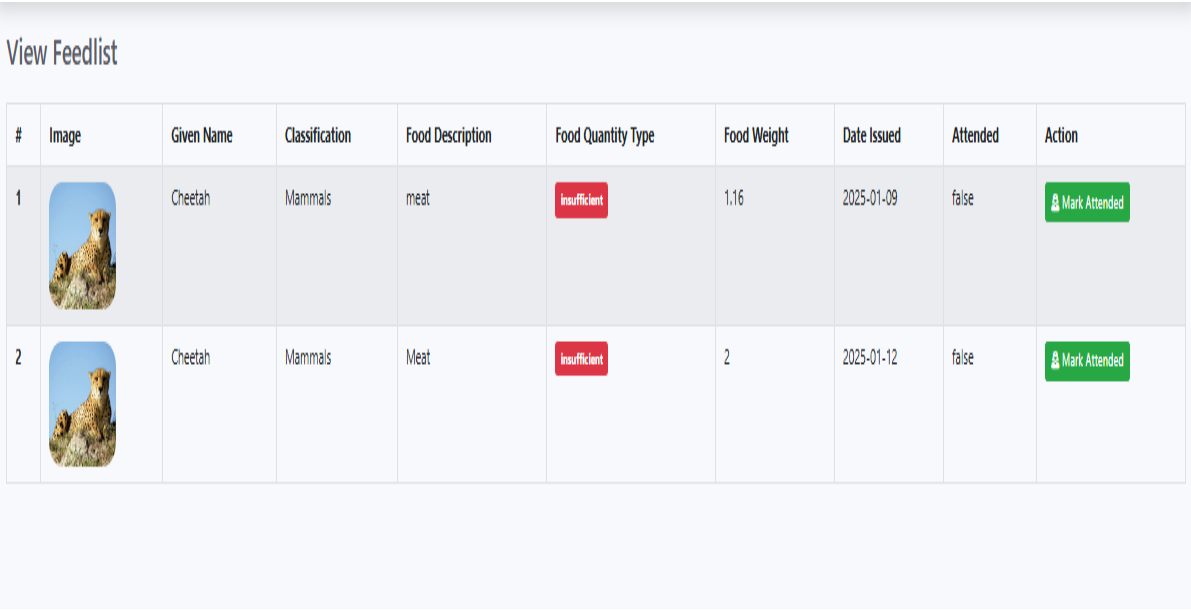


Figure 11.2.2-2Viewing Data from Database into Feed List

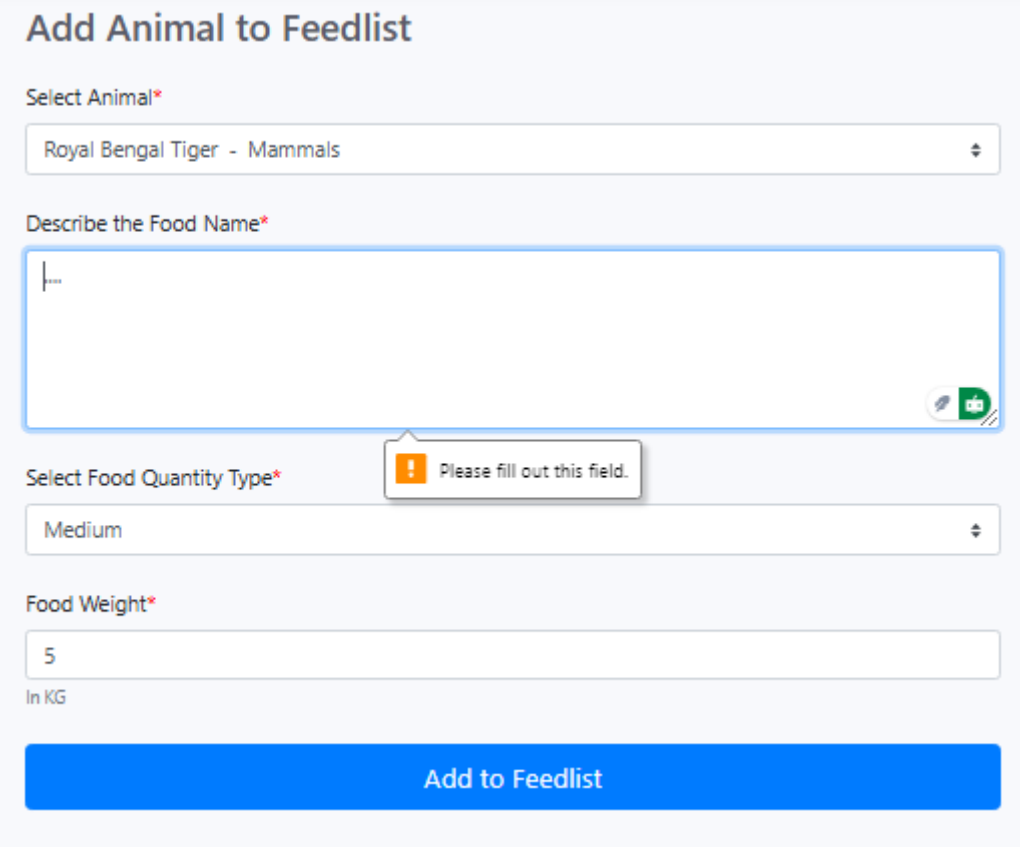


Figure 11.2.2-3 Showing a Warning Message Due To Fill up The Required Field

11.2.3 Unit Test -2

Test Case Name	Unit Test-2
Test Case	Perform action successfully with message status

Test Scenario	Expected Result	Actual Result	Remarks
Mark attended of the food list	Action will be performed by mark attended to the food list with successful message status	Action performed by mark attended to the food list with successful message status	Passed

View mark attended data in the view attended watch list	Showing mark data from database which can be also restored	Showing mark data from database which can be also restored	Passed

Table 11.2-2 Unit test-2

11.2.4 Unit Test -2-Output Visualization:

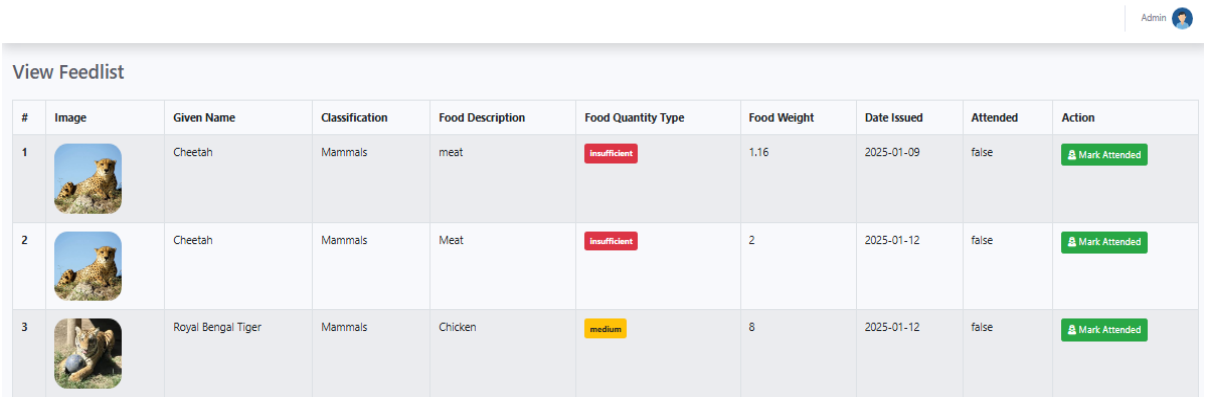


Figure 11.2.4-1 Viewing Data from Database before Any Action Performed

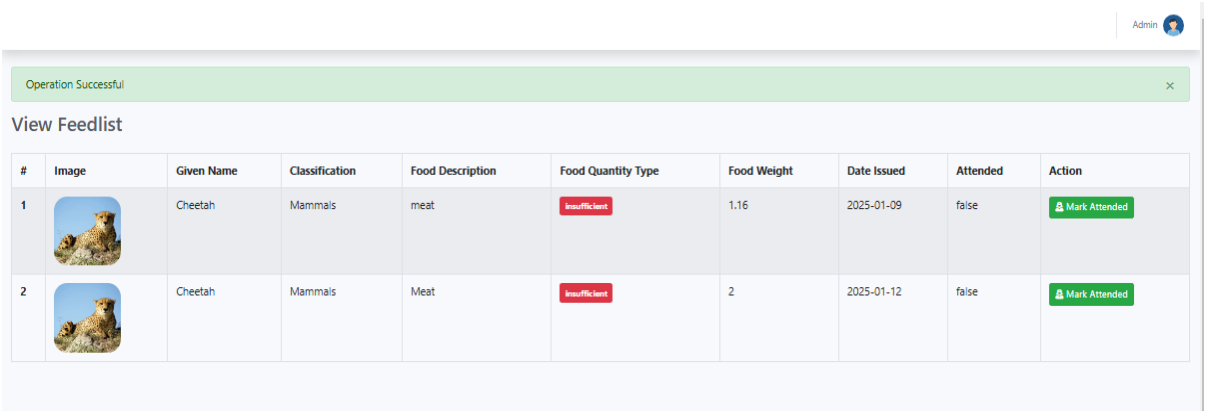


Figure 11.2.4-2 After Performing Action such as Mark Attended Showing Data with Successful Status

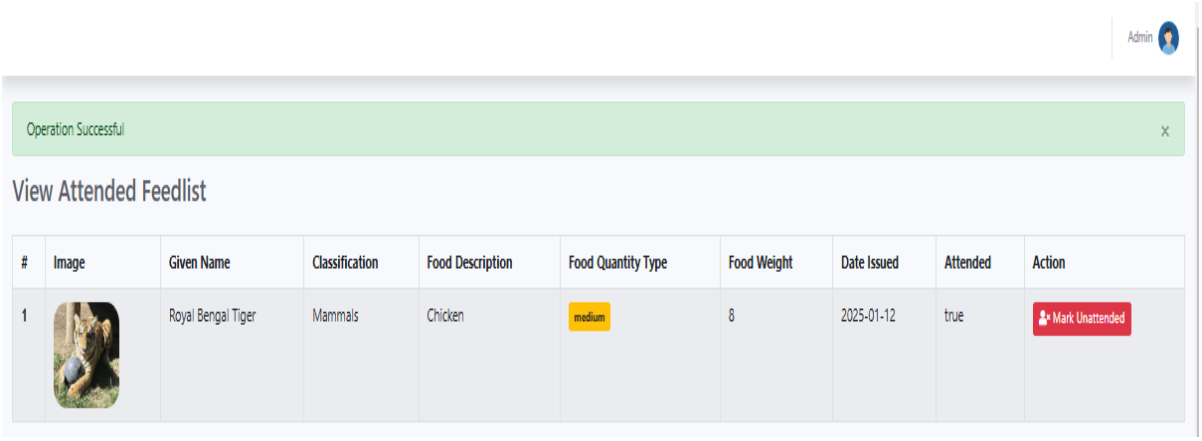


Figure 11.2.4-3 Viewing Data from Database in View Attended Feed List

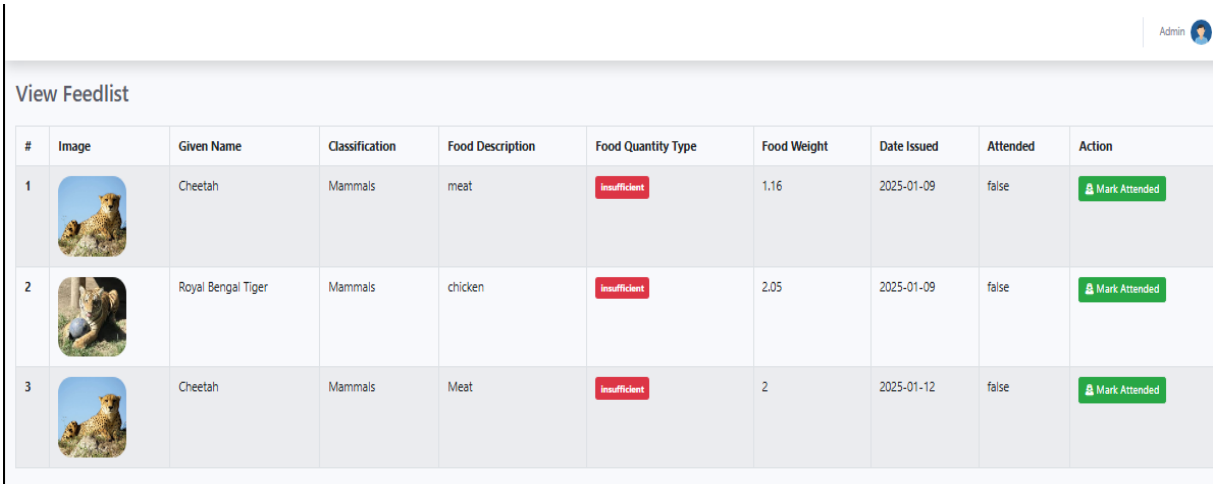


Figure 11.2.4-4 After Performing Action such as Mark Unattended Restore Data

11.2.5 Module Test -1

Test Case Name	Module Test-1
Test Case	Ticket Purchasing Module Testing

Test Scenario	Expected Result	Actual Result	Remarks

Validate the accurate computation of totals of booking tickets	Total amount of taka in respect of ticket quantity calculated accurately with result	Total amount of taka in respect of ticket quantity calculated accurately with result	Passed
Verifying user authentication to book tickets	Showing warning message due to log in for booking tickets	Showing warning message due to log in for booking tickets	Passed

Table 11.2-3 Module test-1

11.2.6 Unit Test -2-Output Visualization:

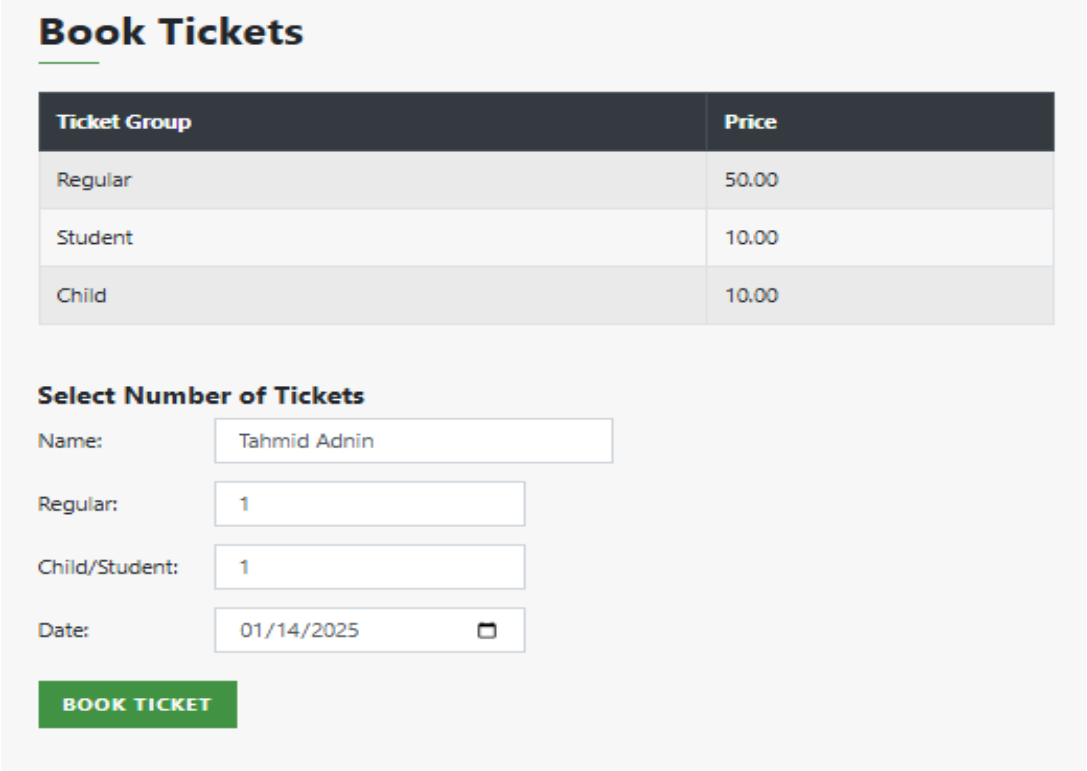


Figure 11.2.6-1Take Input For Booking Ticket

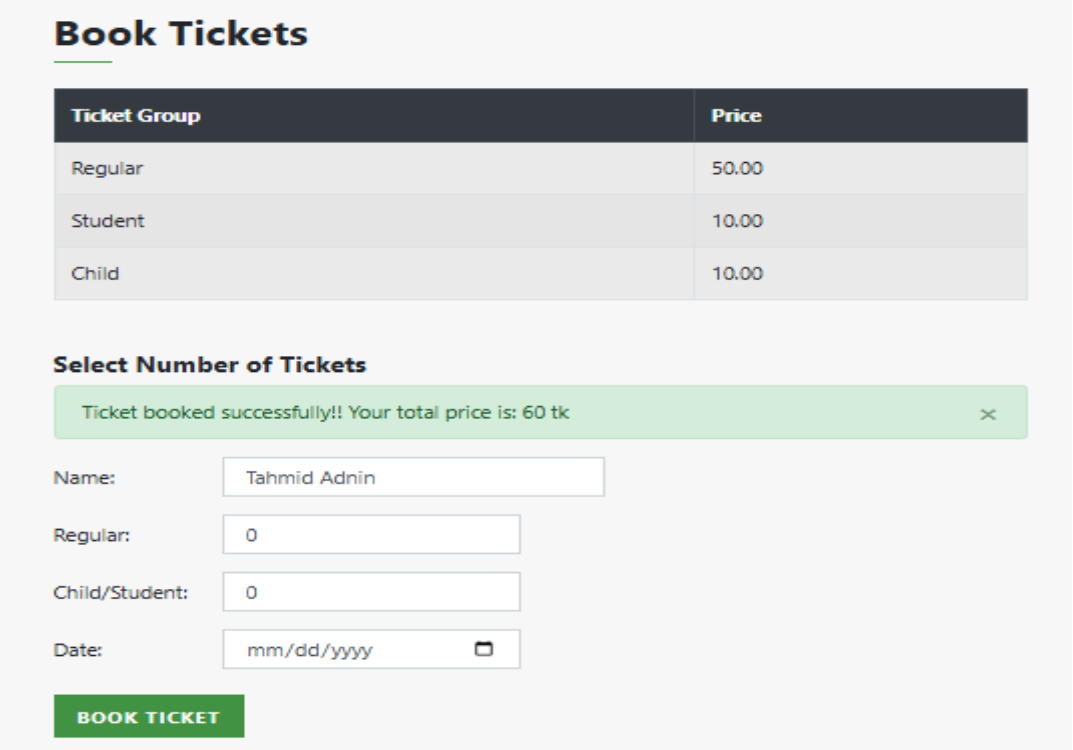


Figure 11.2.6-2 Calculation Total Price Successfully

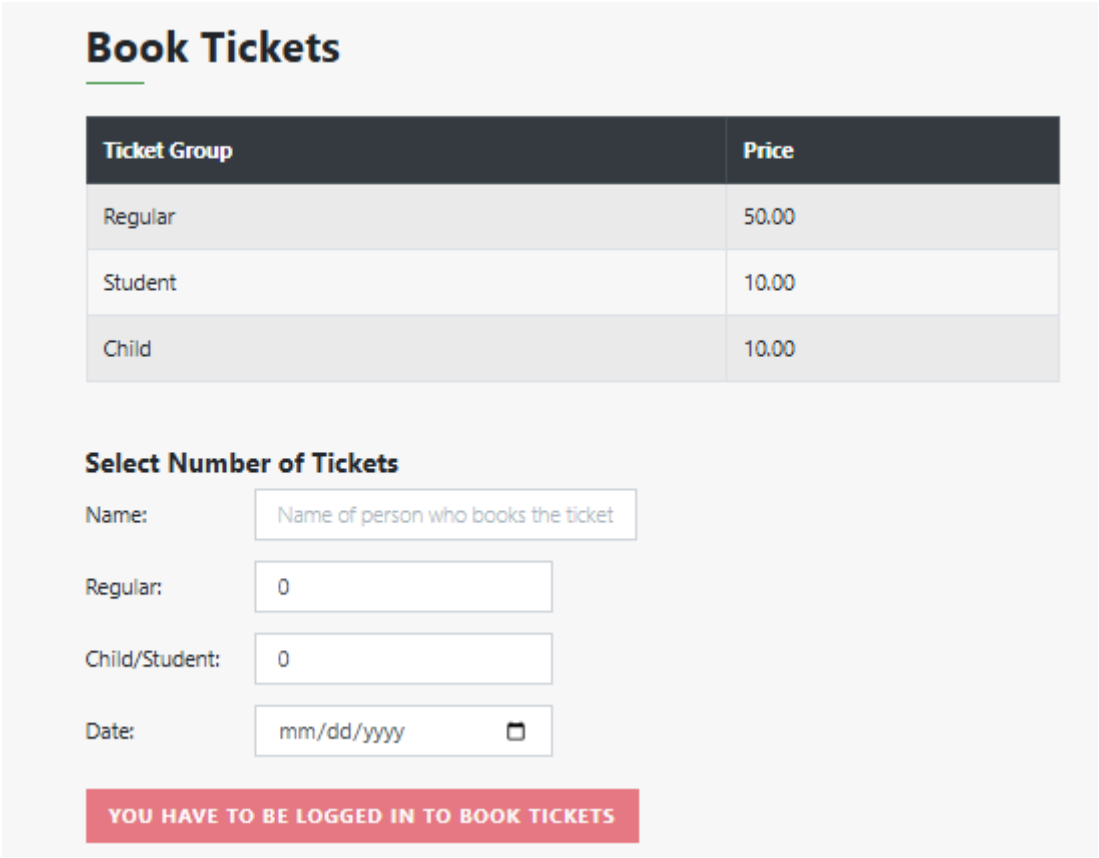


Figure 11.2.6-3 Showing Alert Message To Book Ticket

11.3 Integration Testing:

Test Case Name	Integration Test
Test Case	Integration Test

Test Scenario	Expected Result	Actual Result	Remarks
Verify the interaction between the frontend and backend systems to ensure smooth communication and data synchronization.	Frontend and backend interface are working properly	Frontend and backend interface are working properly	passed
Verify the integration of external packages and libraries uses.	Every outsourced entity package is working simultaneously	Every outsourced entity is installed properly and Version are handled	passed

Table 11.3-1 Integration Testing

11.4Security Testing:

Test Case Name	Security Test
Test Case	Security Test

Test Scenario	Expected Result	Actual Result	Remarks
Test admin log in panel security	No entry without authentication by showing an alert message	No entry without authentication by showing an alert message	passed
Test for giving wrong data	Showing an alert message with no entry	Showing an alert message with no entry	passed

Table 11.4-1 Security Testing

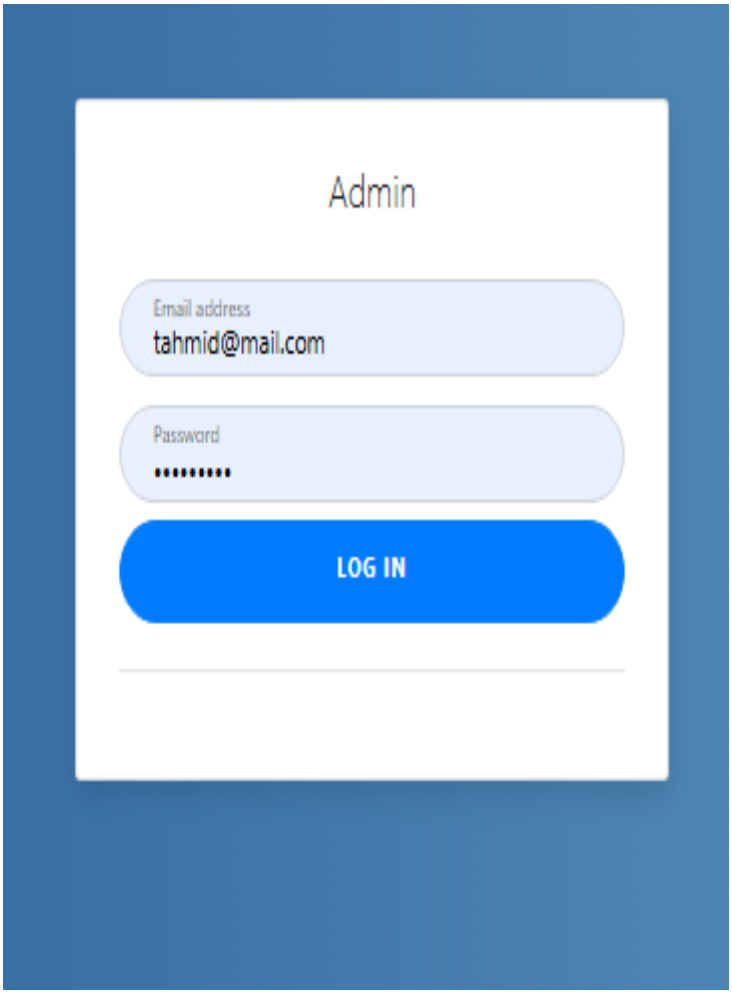


Figure 11.2.6-1 Giving Wrong account data for log in

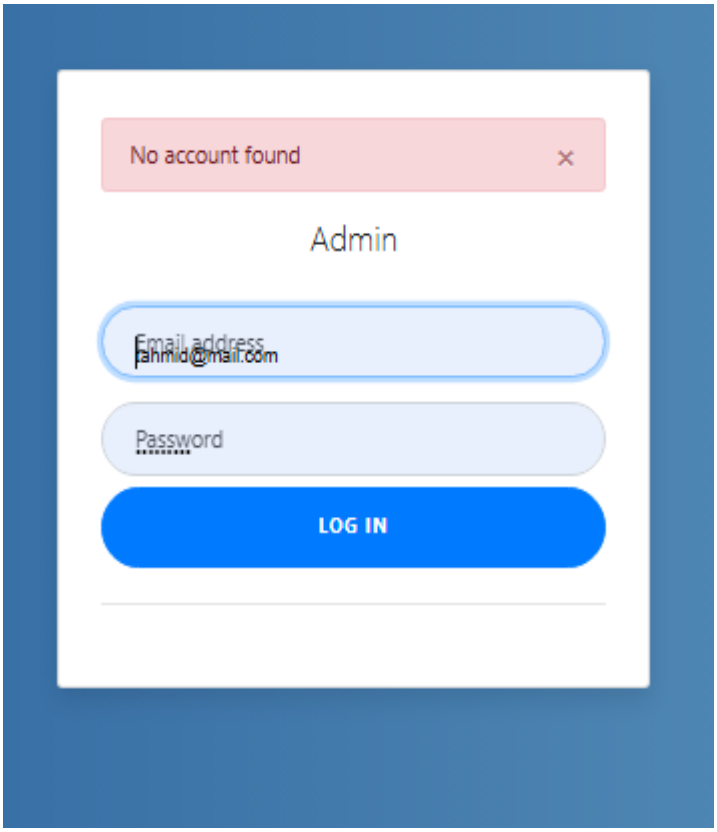


Figure 11.2.6-2Showing Alert Message

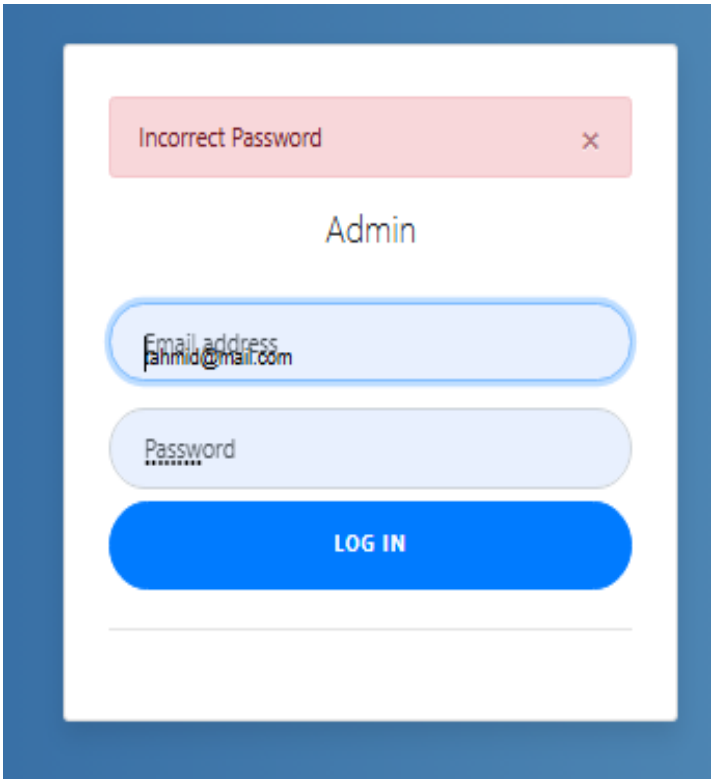


Figure 11.2.6-3 Showing Alert Message For Giving Wrong Data

12 Chapter: Implementation

12.1 Training:

Conducting a training session enables new users to rapidly acquire expertise in utilizing the application efficiently. The system augments customer interaction, promotes system integration, promotes effective utilization, diminishes sustenance inquiries, as well as offers seamless integration procedure.

- **Seamless Adoption:** A comprehensive training session enables inexperienced user to comprehend to travers as well as operate the system interface proficiently which mitigates the adjustment period, facilitating their adoption and utilization of the system without experiencing overwhelm or confusion.
- **Enhanced User Experience:** Providing training improves the entire user journey. System end-user develop assurance in utilizing the application's functionalities, hence diminishing disappointment as well as enhancing enjoyment. These results in increased system user interaction, stickiness, and devotion.
- **Optimized System Use:** Training makes certainty that users are familiar with all the operations as well as features of the system. User can discover ideal procedures, abbreviations, and successful processes, assisting them in using the system proficiently and achieving what they want.
- **Decreased Customer Inquiries:** A proficient user group is crucial for minimizing issues and reducing the need for extensive support. When users are equipped with clear instructions and guidance during training sessions, they are more likely to troubleshoot problems on their own. This empowerment not only boosts user confidence but also leads to fewer support requests. As a result, the frequency of support inquiries decreases, allowing support teams to focus on more complex issues while saving valuable time for both users and support personnel. By investing in comprehensive training, organizations can create a smoother, more efficient experience for everyone involved.
- **Seamless Training:** Training sessions offer a systematic familiarization procedure, directing consumer by the application’s functionalities as well as facilitating rapid productivity. It enables them to grasp how the system fulfills their requirements and how to utilize its functionalities.

12.2 Scaling:

Scaling is the process of adjusting a zoo management system to handle increased demands without compromising performance or reliability. As the zoo grows, it may need to accommodate more animals, manage larger volumes of visitor data, handle increased operational tasks, and ensure seamless integration with different systems (e.g., ticketing, animal tracking, or inventory management). This involves adding more servers or upgrading existing ones to accommodate more animals, handle visitor data, handle operational tasks, and ensure seamless integration with other systems. This approach maintains optimal performance, ensuring smooth operations, rapid response times for staff and visitors, and the ability to expand capabilities as the zoo evolves. This ultimately enhances the visitor experience and supports the zoo's future growth.

12.3Load Balancing:

Load balancing for a zoo management system involves distributing incoming traffic or tasks across multiple servers or resources to ensure the system performs efficiently, even during periods of high demand. In the context of a zoo, this could include managing a large number of visitors accessing the website or ticketing system, balancing the load of data processing for animal tracking, or distributing tasks between staff management and inventory systems. By using load balancing techniques, such as round-robin, least connections, or dynamic load balancing, the zoo management system can maintain optimal performance, reduce server downtime, and prevent system overloads. This ensures that even as the zoo experiences spikes in traffic or operational demand, the system remains responsive, fast, and reliable. Load balancing improves the user experience for both visitors and staff, ensuring smooth, uninterrupted operations, especially during peak times or special events.

13 Chapter: Critical Appraisal and Evaluation

13.1Objective that could be met:

Each application, despite the quality of design, opportunities for optimization and upgrades. During the building of the system, I have endeavored to achieve all the important Intends to maintain its efficiency installation as well as seamless functioning. Yet, there are some lacking which we have mentioned some objectives that could be met below; they are:

- **Disease Prediction by using AI:** In this project, here could be integrated AI for predicting and detecting diseases of animals. Therefore, it will be for useful site for zoo management system. In the next update version, it will be integrated to the system.
- **Streamline Ticket Purchasing and Booking Processing:** Implement a streamlined ticketing and booking platform that automates order handling, monitors transactions, and delivers real-time updates for a seamless customer experience from selection to fulfillment.
- **Boost Performance:** Ensure the application functions fluidly and effectively, achieving prompt load times and intuitive user interactions, even for sophisticated data processing for diseases of animal.

13.2 How much better could have been done:

As per requirement, the site is quite suitable for handling user efficiently. But If there is a way to take data from picture of animal food then it will be more automated site which could have really useful in terms of time efficiency. Additionally, AI disease detection and prediction feature could help to take care of animal health.

13.3Which features could not be touched

Though in this project, there are implemented different kinds of features perfectly, still some aspects are needed to be updated or integrated with the system. This untouched functionalities or features might consist of enhanced refinement possibilities, syncing with the network of social media, and applying machine learning (ML) approaches for disease prediction.

14 Chapter: Lesson Learned

14.1 Pre Project – Review – Closing:

The project named "Zoo Management System" is inspired from Bangladesh National Zoo. Because in the Bangladesh National Zoo website there are missing some feature which I develop in my site. When I wanted to survey their site deeply, I went to the zoo to the data center. I saw that their data center was manual and to search or find any kind of data, that was too time consuming. So I wanted to make an automated database, where all data will be stored systematically. If one need to find any kind of data, then admin can help them immediately.

In closing, I can say our study illustrates a detailed view of effective establishment of the system with proper attachment, diagram etc.

14.2 The Problem I Have Faced:

- To gather requirements
- To survey real life website for more effective features
- To fetch data from database for viewing
- To add validation

In spite of these challenges, I have tried my best to overcome all the obstacle by utilizing my problem solving skills for improvement. By tackling these issues directly, I achieved the successful completion of the project.

14.3 What Solutions Occurred:

I visited several times to the Mirpur zoo for requirements purpose. And successfully I obtained enough. Then, in terms of features I survey a lot of real time website. For example, indian website of zoo, helped me a lot in this purpose. After that fetching data from database, there I faced some problem. For this reason, I search and gain some additional knowledge to fetch and view data from database perfectly. Sometimes, there are not working alert message properly to validate the data. So later I worked on it and successfully apply all the validation.

15 Chapter: Lesson Learned

15.1 Summary of the project:

My goal is to store data which can be fetched easily in terms of needed, across the whole project. Because in our country, there are lacking of justification either one animal gets the required food or not. Because there are only use manual data sheet, which are so time consuming to find any specific data. And sometimes they are lost. On the other hand, there are seen delay to enlist sick report to the authority. Or I can say there are so negligence to fulfill in this criteria. So there needed at least a specific data sheet where all the sick animal will be enlisted with adequate information. Along with this, either they are attended by someone or not that is marked by authority. I think both these fundamental criteria are met in the "Zoo Management System" successfully.

15.2 Goal of the project:

- Improved management of animal health and records.
- Ensuring management of animal food and records.
- Enhanced visitor experience and increased ticket sales.
- Streamlined staff management with better resource allocation.
- Improved financial oversight leading to better funding and resource utilization.
- Data-driven insights leading to better strategic decisions.

15.3What I have done in the documentation

The key objective of the entire documentation is to offer a detailed review of our project. For Instance, in each segment, I have broken out each procedure which are taken during the project as well as provided an explanation of what I accomplished in each stage with adequate prove from fundamental Insights to create and build the system. Hence, I think that, by reviewing the entire material, anybody could comprehend my aim, approaches clearly and easily. I have provided a complete figure, an output snapshot, and a short description of the component as well as functionality along with all the materials which I utilized into my website.

15.4My Experience:

Building this project taught me how the company manages real-life projects by adhering to SDLC. Throughout the process, perquisite data collecting and research for designing, implementing, testing and developing the system, I monitored the structured method for establishing a user-friendly, effective and scalable software solution which match with the industries standardization. Overall, to work on this project, it broadens my technical skills along with furnishing me real life experience of building industry oriented application. It shows how important it is to be aware of the latest technologies; road map work is key to gaining project success.

Works Cited

About: bnzoo. (n.d.). Retrieved from Dhaka Mirpur Zoo: https://bnzoo.org/

About:goodworklabs. (n.d.). Retrieved from goodworklabs: https://www.goodworklabs.com/the-role-of-data-in-software-development/

knowledge-base: rapid-application-development. (n.d.). Retrieved from agilelonestar.com: https://www.agilelonestar.com/knowledge-base/rapid-application-development

project-management-guide:FAQ. (n.d.). Retrieved from wrike.com: https://www.wrike.com/project-management-guide/faq/what-is-resource-allocation-in-project-management/

resources:agile-methodology. (n.d.). Retrieved from asana.com: https://asana.com/resources/agile-methodology

saigontechnology. (n.d.). Retrieved from saigontechnology.com: https://saigontechnology.com/blog/agile-development-process-in-software-outsourcing/

sdlc:sdlc_agile_model. (n.d.). Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

software-engineerin:timeboxing-model. (n.d.). Retrieved from ecomputernotes.com: https://ecomputernotes.com/software-engineering/timeboxing-model#google_vignette

unified-modeling-language-uml-activity-diagrams. (n.d.). Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

Venkataraman, D., & Shaji, R. (2017). Improvising knowledge based acquisition on Zoo Management system. *Zoo Management system*. doi:10.1109/ICACCS.2017.8014697

workbreakdownstructure. (n.d.). Retrieved from workbreakdownstructure.com: https://www.workbreakdownstructure.com/

182-16-323

ORIGINALITY REPORT

15%	14%	0%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	5%
2	Submitted to Daffodil International University Student Paper	2%
3	listens.online Internet Source	2%
4	www.wrike.com Internet Source	2%
5	www.pvpsiddhartha.ac.in Internet Source	1%
6	Submitted to University of Ulster Student Paper	1%
7	www.goodworklabs.com Internet Source	<1%
8	Submitted to University of Greenwich Student Paper	<1%
9	Submitted to Asia e University Student Paper	<1%