PetManipal- Animal Shelter Website

Mini Project Report – Web Technologies Lab
Department of Data Science & Computer Applications

B. Tech. Data Science & Engineering
Vth Semester

Submitted By

Niyati Kotian	200968018
Shiv Pratap Singh	200968038
Khushee Kapoor	200968052
Sanjana Arun	200968054
Sahaj Jaggi	200968076

Mentored By

Tojo Thomas
Assistant Professor – Senior Scale
Department of Data Science & Computer Applications
Manipal Institute of Technology

Akshay Bhat
Assistant Professor
Department of Data Science & Computer Applications
Manipal Institute of Technology

CERTIFICATE

This is to certify that the students Niyati Kotian (200968018), Shiv Pratap Singh (200968038), Khushee Kapoor (200968052), Sanjana Akhila Arun (200968054) and Sahaj Jaggi (200968076) have successfully executed a mini project titled "PetManipal" rightly bringing forth the competencies and skill sets they have gained during the course – Web Technologies Lab (DSE 3163 & DSE), thereby resulting in the culmination of this project.

Mr. Tojo Thomas
Assistant Professor – Senior Scale
Department of Data Science & Computer Applications
Manipal Institute of Technology

Mr. Akshay Bhat
Assistant Professor
Department of Data Science & Computer Applications
Manipal Institute of Technology

Place: MIT, Manipal
Date: 4th November 2022

Acknowledgements

es us great pleasure to present the Report of our Project – PetManipal undertaken in the Vth Semester
r B. Tech. We would like to express our gratitude to **Mr. Tojo Thomas**, Assistant Professor – Senior
, Department of Data Science & Computer Applications, and **Mr. Akshay Bhat**, Assistant Professor,
rtment of Data Science & Computer Applications, for their constant guidance and support throughout
ourse of our work. Our deepest thanks also go to **Dr. Radhika M Pai**, Head of the Department,
rtment of Data Science & Computer Applications for her supervision.

would also like to acknowledge with much appreciation the efforts of the Management, Administration
Lab Staff, who gave us the necessary support to use the laboratory facilities.

ly, we pray thanks to God, our parents, and friends, who contributed to some extent to complete the
.

Niyati Kotian

Shiv Pratap Singh

Khushee Kapoor

Sanjana Arun

Sahaj Jaggi

Place: MIT, Manipal
Date: 4th November 2022

Abstract

Animal shelters play an important role in our communities because they work tirelessly to reunite pets with their owners, shelter those in need, and find new homes for animals that are lost, without a permanent home, or that should not be roaming our streets for our own safety. Many animals on the streets require the assistance of these shelters but are unable to access them.

PetManipal is a searchable online database of animals in need of shelter and care. The goal of this project is to create a website for an animal shelter that is responsive, interactive, and visually appealing.

The web application links the animals to the closest animal shelter. When the user sees an injured animal, he or she uploads a photo of it along with its contact information and location. A rescue operation is launched by the nearest shelter. The user can set up an account and create and track numerous initiatives. The initiative is deactivated once an operation is completed. The project's technology stack includes HTML, CSS, and EJS for the front end, Figma for design and UI/UX, NodeJS, and MongoDB for the backend, and Git & GitHub for Version Control.

The web application is responsive and meets the project objectives. The application is currently deployed locally, but it can be deployed on the cloud for commercial use and/or future expansion.

Table of Contents

List of Figures

List of Tables

Chapter 1. Synopsis

1.1 Introduction

PetManipal is a searchable online database of animals in need of homes. The goal of this project is to create a responsive, engaging, and attractive website for an animal sanctuary. We assist in the rescue and protection of suffering animals in need as a neighborhood animal welfare organization for creatures directly influenced by humans.

In the future we envision, people treat animals with empathy, respect, and understanding. With this initiative, we want to do that. Pet enthusiasts can find the pet that best suits their needs while relaxing in front of their laptops. They can then look up a shelter's website to learn more about the services it provides. To assist pet owners in keeping their pets in their homes, PetManipal also offers discussion forums, a directory of pet-care resources, and a library of free pet-care articles.

For these operations, a database and backend are planned. The database will have all the information for an animal shelter, and the website won't remain static.

1.2 Motivation

Dogs, cats, and other domesticated animals rely on human caregivers to keep them safe, happy, and healthy. Some animals are fortunate enough to be adopted into loving, permanent homes. Unfortunately, there are far more animals in need of a loving family than there are people willing or able to give them a lifetime of love and support. An animal shelter is a staffed facility where homeless animals, as well as animals seized by authorities in cruelty cases, can find safety and comfort, be cared for, and be adopted. Sheltering animals temporarily keeps them from roaming the streets, where they may struggle to find clean food and water, be hit by cars, be attacked by other animals or cruel humans, or face other potential dangers.
We see many animals in our community that could benefit from shelters but do not have the means to get to them. This necessitates a call to action by students. A web application that allows students to assist these animals in need by connecting them to the nearest animal shelter is urgently needed. We came up with the idea for a user-friendly application that allows users to upload photos of animals in need as well as their location. The nearest shelter's rescuer would be notified, and the rescue operation would begin. This enables prompt intervention. We believe that a portal connecting animal shelters and Samaritans is a good use of technology.

1.3 Objectives

- To create a website that is completely responsive so that it can be read on laptop displays and mobile devices
- To create a strong frontend for with desirable User Interface and Experience
- To establish a database and API design for adding and retrieving data
- To formulate a proper code documentation so that any new coder can read and comprehend the project

1.4 Technical Stack

Table 1.4. Technical Stack

UI/UX	
Figma	Adobe Photoshop
Frontend	
HTML [2]	CSS [3]
ReactJS [4]	EJS [5]
Backend	
NodeJS [6]	MongoDB [7]
Version Control	
Git	GitHub

Table 1.4 refers to the Technical Stack used for developing PetManipal. Figma and Adobe Photoshop were used to develop the UI/UX; HTML, CSS, ReactJS and EJS were used to work on the Frontend of the web application; NodeJS was used to develop the backend connected with MongoDB to store the unstructured data; and Git and GitHub were used as Version Control System to ease the development process.

1.5 Role Distribution

Table 1.5. Role Distribution

Khushee Kapoor	**Documenter**
The documentation will follow IEEE standards and will include methodology, modules, software, process model, code explanation, utility, development approach, ER model, schema diagram, data dictionary, sequence activity, state transition diagram, use case diagram, general flow diagram, scrum/ agile/ waterfall iterative process explanation, etc.	
Sanjana Arun	**Design & UI/UX Developer**
To ensure that our goal is reached, we need to enhance the design, user interface and user experience. This will be done using Figma.	
Niyati Kotian	**Frontend Developer**

Apart from design, the responsiveness, implementation of the design, and the way the various components of the website work as seen by the user come under this area. It will be enabled with the use of HTML, CSS, ReactJS, and EJS Frontend Framework.	
Shiv Pratap Singh	**Databases & API**
To enable smooth functioning of pet adoption, we will use databases. The interaction with the database to retrieve relevant data and display it to the frontend when the user visits a particular pet page is achieved through APIs. For Databases, we will use MongoDB.	
Sahaj Jaggi	**Backend Developer**
The backbone of the project, the backend will be implemented in NodeJS and will ensure that the various components of the website coordinate with each other and make the entire process easy.	

Table 1.5 illustrates the distribution of roles and responsibilities for effective team work and development of the Web Application.

Chapter 2. Software Requirements Analysis

2.1 Introduction

2.1.1 Purpose

The purpose of this chapter is to present a detailed description of the open-source online searchable database PetManipal. It is four-fold:

- To explain the process by which the preliminary requirements from PetManipal were analyzed and refined.
- To state the requirements for the PetManipal Web-App System in a way that allows tracing from the original (ambiguous, unrefined) form to the analyzed and refined form.
- To show the analysis of requirements (dependencies, issues resolved, analysis of requirements, etc.)
- To explore the purpose and features of the software, the interfaces – frontend and backend, the intended use of the software, and what constraints it must operate under.

This chapter is also a statement of how the requirements were derived from the input and their relationship with each other.

2.1.2 Product Scope

PetManipal is a searchable online database of animals in need of homes. The goal of this project is to create a responsive, engaging, and attractive website for an animal sanctuary, to assist in the rescue and protection of suffering animals in need as a neighborhood animal welfare organization for creatures directly influenced by humans. Objectives of this project include creating a completely responsive website so that it can be read on laptop displays and mobile devices, which are typical viewing platforms for this kind of a website; creating a strong frontend for with desirable User Interface and Experience; establish a database and API design for adding and retrieving data; formulate a proper code documentation so that any new coder can read and comprehend the project.

2.2 Overall Description

2.2.1 Product Perspective

PetManipal is an open-source project designed to be virtually self-contained online searchable database that is developed to run on Windows, Mac OS, Linux, Android, and iOS, and will require users to have access to a web browser on their workstation computer or mobile. The user does not need to invest in any other software to get most of the web application and can be run on any device that has a web browser installed. The system will also have the ability to take image, text, and numerical input from the user using the web interface.

2.2.2 Product Functions

The user will be able to avail the following functionalities basis their need:

- If the user identifies or locates an animal in need, they will be able to upload the animal's picture and upload it on the web-application along with its location and the user's contact details, thereby initiating the rescue operation. After the animal has been rescued, the operation can be shut down and the catalogue item can be removed.
- If the user or organization wants to rescue a suffering animal, they can browse through the catalogue and identify the animals nearest to their base and retrieve the contact details of the person who has initiated the rescue operation.

2.2.3 User Classes and Characteristics

The web-application is intended for the general population of Manipal including but not limited to MAHE students, faculty, staff, and Manipal natives who empathize with treat animals with empathy, respect, and understanding and assist in the rescue and protection of suffering animals in need as a neighborhood animal welfare organization for creatures directly influenced by humans.

2.2.4 Operating Environment

The web application is developed to work on any operating system including but not limited to Windows, Mac OS, Linux, Android, and iOS, which has access to a web browser including but not limited to Microsoft Edge, Google Chrome, and Mozilla Firefox. The web application will be completely responsive and so that it can be read on a laptop, tablet, mobile, etc. Since it is self-contained, there is no need for any other software to be installed.

2.2.5 Design and Implementation Constraints

The user must give consent before entering private sensitive data such as contact details and location which are necessary for the web-application. If such data is not provided, the intended purpose will not be satisfied, and the web-application will not be able to execute its functionalities.

2.2.6 User Documentation

A user-manual covering tutorials, feature explorations, troubleshooting, helplines, etc. will be released along with the release of the web application.

2.2.7 Assumptions and Dependencies

The PetManipal project works on the assumption that the users have a steady access to internet and web browsers and are willing to provide their private sensitive information such as contact details and location. We also hope that there are individuals and organizations in and around Manipal who will have adequate logistic measures in place to rescue animals. Assumptions are also made that the users have sufficient knowledge on the use of technology, and the care of animals.

2.3 External Requirements

2.3.1 User Interfaces

The User Interface is compatible with different viewing platforms – on a larger device (width: 992px and above), the standard User Interface is reflected with maximum possible information at the landing pages, on a medium range device (768px to 991px), some features are hidden under buttons and menus for effective screen usage, and for smaller devices (767px and below), minimal information is displayed with most information hidden under buttons and menus.

The UI/UX is sports a beige background – signifying closeness to the ground and nature, and easing accessibility for people with color blindness, according to the Web Content Accessibility Guidelines. [8]

2.3.2 Hardware Interfaces

The Web Application can be used on larger devices (width: 992px and above), medium range devices (768px to 991px) and smaller devices (767px and below).

2.4 System Features [1]

2.4.1 Initiator Module

2.4.1.1 Description

The initiator module allows the user to initiate a rescue operation by uploading the animal's images, their contact details and location. The user can also deactivate a successful operation.

2.4.1.2 Stimulus/ Response Sequences

After logging in, the user must select the initiator option, wherein they can view their previous operation initiatives, create a new initiative, and deactivate an initiative after a successful rescue operation. While creating a new initiative, the user will be prompted to upload images of the animal in need, provide their contact details and location, and publish the initiative. While deactivating initiatives, the user will be

prompted to select an active initiative, and then deactivate it. After the operations of both these buttons, the user will land on the initiator landing page of the web application.

2.4.1.3 Functional Requirements

Table 2.4.1.3 Functional Requirements of Initiator Module

Input	User Contact Details and Location, Images of Animals
Processing	If Create New Initiative: Store Input User Contact Details and Location, Images of Animals If Deactivate Initiative: Delete the data related to the Initiative
Output	If Create New Initiative: Display the Created Initiative If Deactivate Initiative: Display Confirmation Message

Table 2.4.1.3 refers to the processing module of the Initiator Module. The Input is the User's Contact Details, Location, and Images of Animals, which are stored in case of Creating an Initiative and Deleted in case of Deactivating an Initiative, with appropriate messages being displayed for each case.

2.4.2 Rescuer Module

2.4.2.1 Description

The rescuer module allows the user or organization to view the active rescue operations and retrieve the contact details and location of the initiator.

2.4.2.2 Stimulus/ Response Sequences

After logging in, the user must select the rescuer option, they can view the active rescue operations and retrieve the location and contact details of the initiator.

2.4.2.3 Functional Requirements

Table 2.4.2.3 Functional Requirements of Rescuer Module

Input	Interested Initiative Details
Processing	Retrieve Data of Interested Initiative
Output	Display Contact Details and Location of Initiator of Interested Initiative

Table 2.4.2.3 refers to the processing module of the Rescuer Module. The Input is the Interested Initiative Details, which are retrieved and displayed.

Chapter 3. Process Model

3.1 System Design

3.1.1 Use Case Diagram [1]

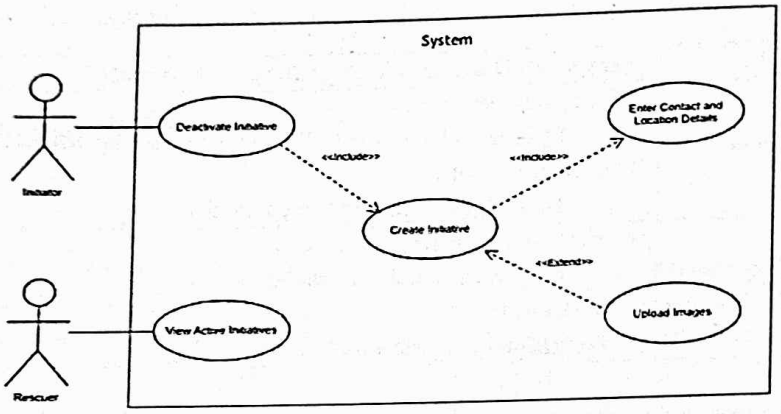


Fig 3.1.1 Use Case Diagram

In Fig. 3.1.1, we see the Use Case Diagram of PetManipal. As we can see, there are two actors – Initiator and Rescuer. The Initiator can Deactivate Initiatives, for which they must Create Initiative, for which they must Enter Contact and Location Details. They may or may not choose to Upload Images. The Rescuer can View Active Initiatives, for which the Initiator must have Created Initiatives.

3.1.2 Sequence Diagrams [1]

3.1.2.1 Create Initiative Use Case

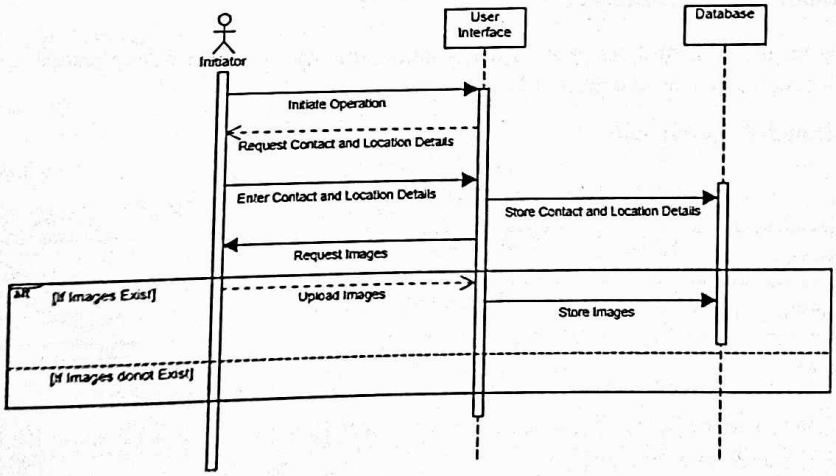


Fig. 3.1.2.1 Sequence Diagram for Create Initiative Use Case

In Fig. 3.1.2.1, we see the sequence of the Create Initiative Use Case. The Initiator sends a message to the User Interface to initiate an operation. In return, the User Interface asks the Initiator for the Contact and Location Details. After the Initiator enters the Contact and Location Details, the User Interface sends them as a message to store in the Database. The User Interface then requests to Upload Images. If the Initiator wishes to upload the Images, they are sent as a return message and are then forwarded to be stored in the database. Else, no action is taken.

3.1.2.2 Deactivate Initiative Use Case

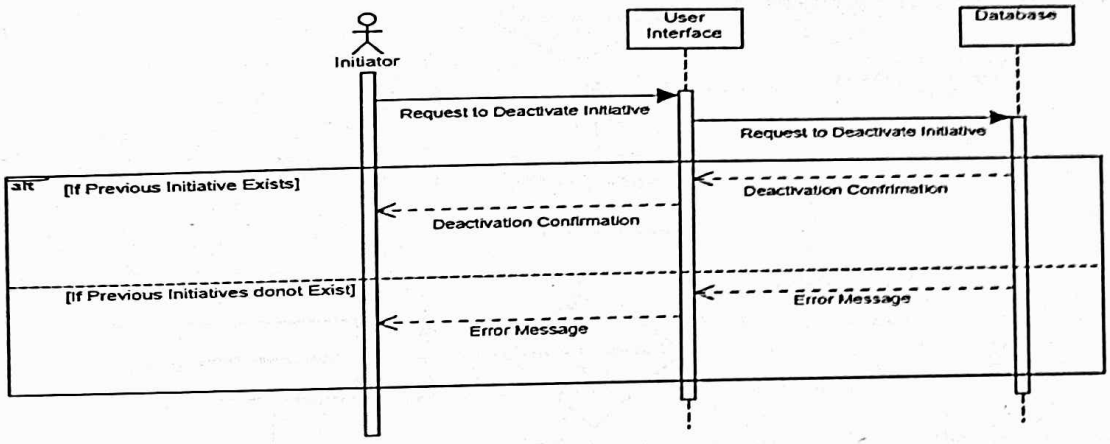


Fig 3.1.2.2 Sequence Diagram for Deactivate Initiative Use Case

In Fig. 3.1.2.2, we see the sequence of the Deactivate Initiative Use Case. The Initiator sends a Request to Deactivate an Initiative. The User Interface then forwards the message to Database. If Previous Initiatives Exist, then the Initiative is Deactivated. Else, an Error Message is displayed.

3.1.2.3 Rescuer Module Use Case

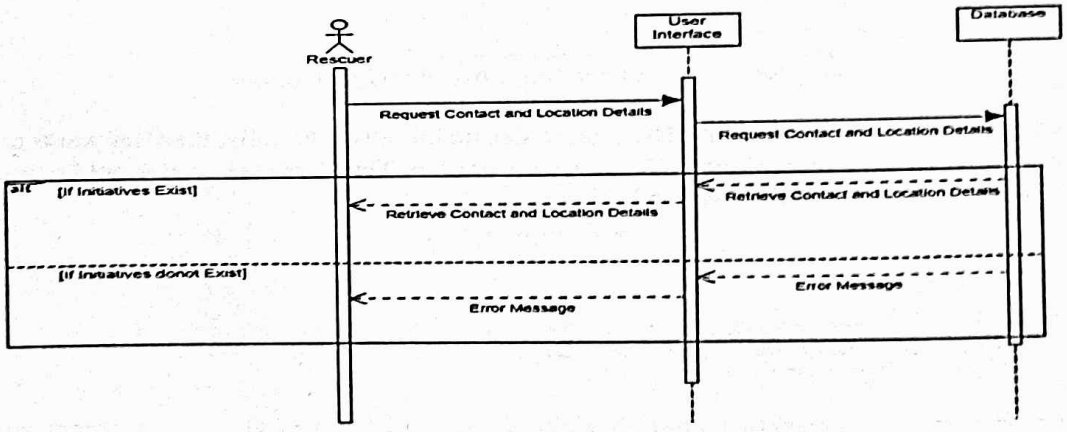


Fig 3.1.2.3 Sequence Diagram for Rescuer Module

In Fig. 3.1.2.3, we see the sequence for the Rescuer Module. The Rescuer requests for Contact and Location Details of the Interested Initiative. This message is then forwarded to the Database. If Previous Initiatives Exist, then the Contact Details and Location are retrieved from the database and displayed. Else, an Error Message is displayed.

3.1.3 Activity Diagrams [1]

3.1.3.1 Create Initiative Activity

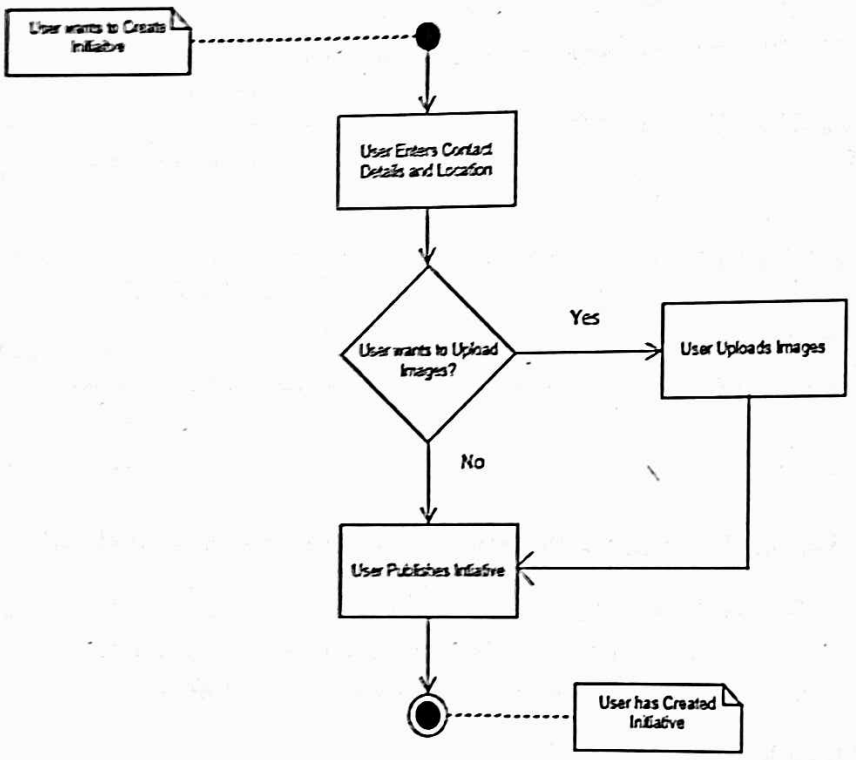


Fig 3.1.3.1 Create Initiative Activity Diagram

In Fig. 3.1.3.1, we can see the Activity Diagram to Create Initiative. Initially, the User wants to Create Initiative. Then the User Enters Contact Details and Location. The User may or may not Upload the Images. After that, the User Publishes the Initiative.

3.1.3.2 Deactivate Initiative Activity

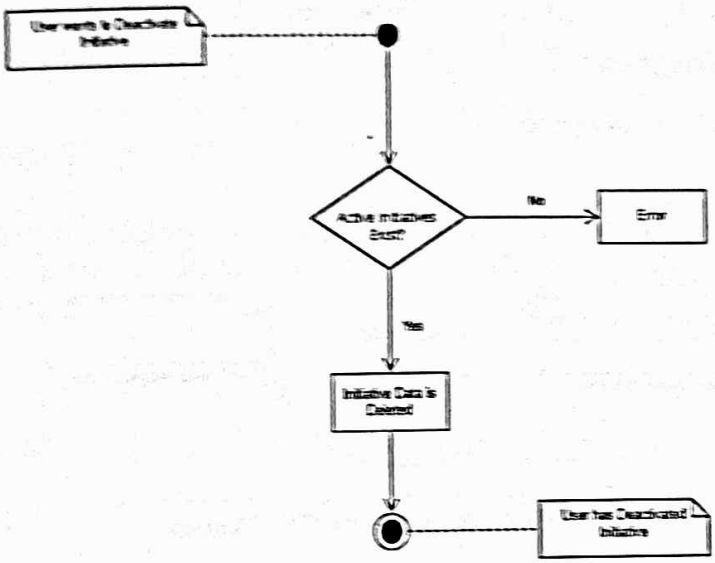


Fig 3.1.3.2 Deactivate Initiative Activity Diagram

In Fig. 3.1.3.2, we can see the Activity Diagram to Deactivate Initiative. Initially, the User wants to Deactivate Initiative. If Active Initiatives do not exist, then an Error is generated. Else, Initiative Data is Deleted. Thus, the User has Deactivated Initiative.

3.1.3.3 Rescuer Module Activity

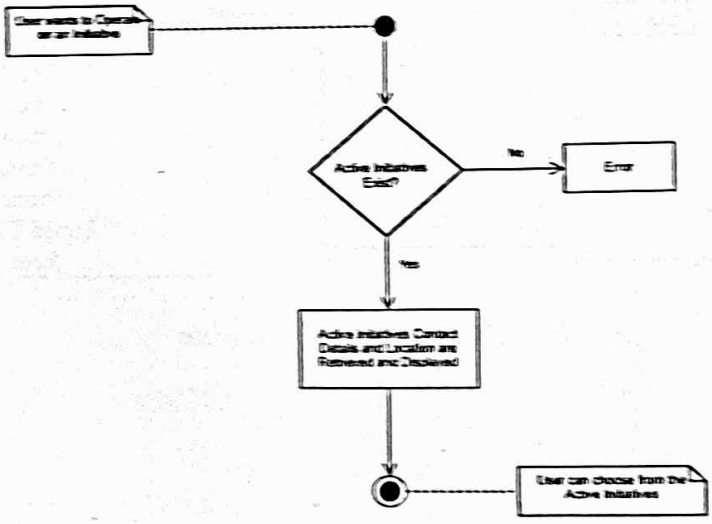


Fig. 3.1.3.3 Rescuer Module Activity Diagram

3.1.3.2 Deactivate Initiative Activity

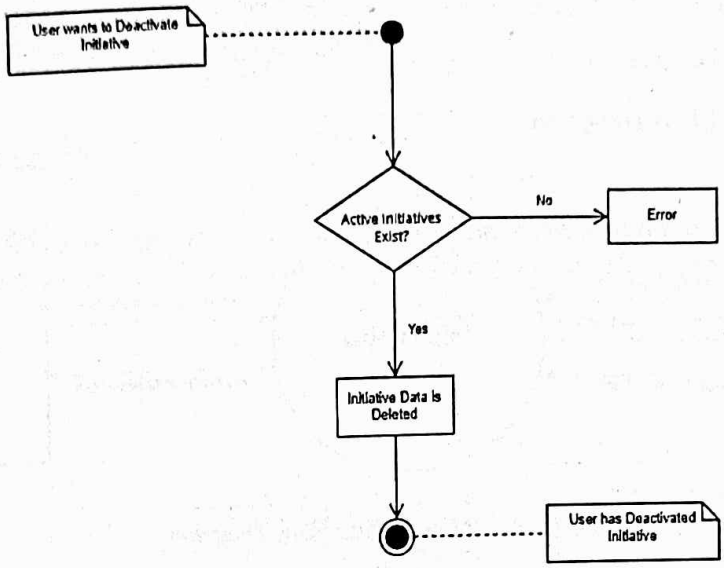


Fig 3.1.3.2 Deactivate Initiative Activity Diagram

In Fig. 3.1.3.2, we can see the Activity Diagram to Deactivate Initiative. Initially, the User wants to Deactivate Initiative. If Active Initiatives do not exist, then an Error is generated. Else, Initiative Data is Deleted. Thus, the User has Deactivated Initiative.

3.1.3.3 Rescuer Module Activity

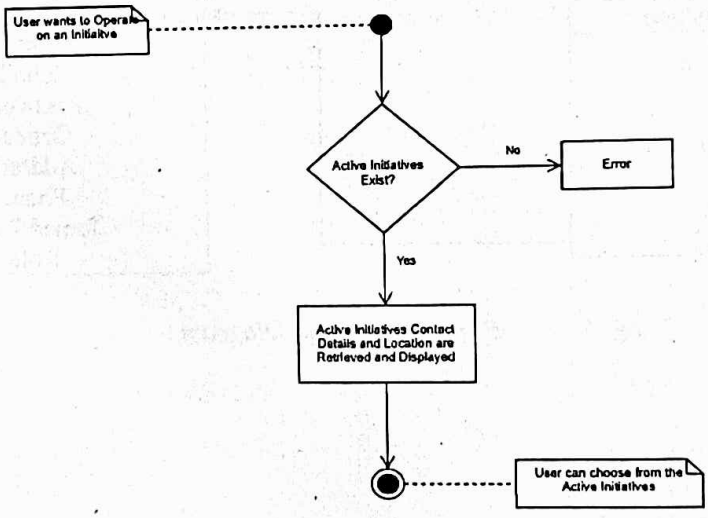


Fig. 3.1.3.3 Rescuer Module Activity Diagram

In Fig. 3.1.3.3, we can see the Activity Diagram for the Rescuer Module. Initially, the User wants to operate on an Active Initiative. If Active Initiatives do not exist, then an Error is generated. Else, the Active Initiatives' data is Retrieved and Displayed. The User can then choose from the Active Initiatives.

3.1.4 Data Flow Diagrams [1]

3.1.4.1 0 Level Data Flow Diagram

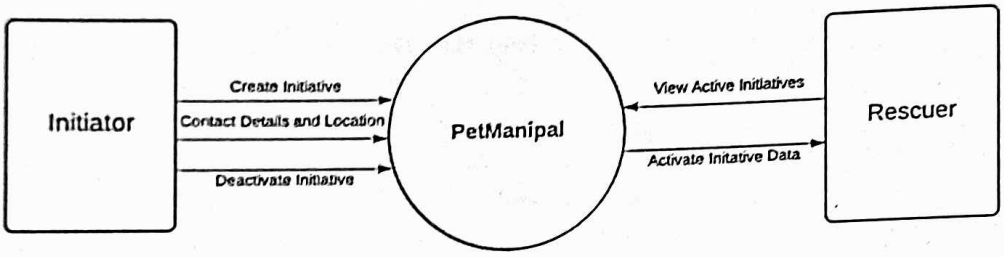


Fig 3.1.4.1 0 Level Data Flow Diagram

In Fig. 3.1.4.1, we can see the High-Level Design of the Software. There are two External Entities – Initiator, which interacts with the Software to Create Initiatives, Deactivate Initiatives, and feed Contact Details and Location, and the Rescuer, which interacts with the Software to View Active Initiatives and Retrieve the Active Initiative Data.

3.1.5 Schema Diagram [1]

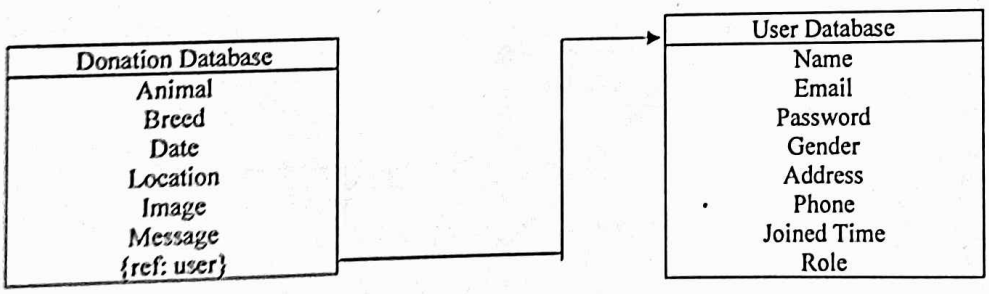


Fig 3.1.5 Schema Diagram

3.2 Development Approach [1]

A Top-Down Approach is followed to develop the software. The problem statement is broken down into various modules and they are individually implemented. The modules in this software include the Initiator and Rescuer modules as described in section 2.4.

3.3 Methodology [1]

The Agile Methodology is implemented to develop the software. This allows smooth introduction of new features in the future and make fixing bugs easier, since the requirements, design and system functionalities are dynamic in nature.

Chapter 4. Conclusion

4.1 Conclusion

PetManipal is a locally deployed web application that connects animals in need of assistance with the services that provide it. It is responsive, strong, and easy to use. There are issues with data concurrency and security, but these will be addressed. The project's objectives have been met. Migrating the deployment to the cloud provides opportunities for improvement and expansion. The project can be commercialized through collaboration with similar organizations that have the means but not the technology. This is something that can be done not only locally, but also globally. Our team has successfully used modern technologies to solve real-world problems through this project.

4.2 Learning Curve

This project has been a very rewarding experience for all of us. We were introduced to a prevalent problem in our society during the ideation phase, which taught us innovation skills. We came across technologies that were beyond the scope of our curriculum while developing the application. This provided us with the opportunity to learn something new. The documentation process also taught us about the various software engineering standards and formats. The design, documentation, and construction processes, as well as the continuous feedback from our facilitators, enabled us to correct flaws in our product and strive for excellence.

4.3 Future Work

In the future, we plan to introduce new features and modules are per the feedback provided by the users. We also plan to introduce a Chatbot facility to ease the admin-user communication to identify bugs and introduce new features. The data collected viz. Image and Location can also be used for data analysis purposes. An Image Classification module will be implemented for Image Tagging. A Natural Language Processing Module for Reviews and Ratings is also in the pipeline.

PROJECT SYNOPSIS

Lyrical

(Improving upon the spotify song search algorithm)

Mini-project for Web Technologies Lab 5th Semester

By-:

ARCHIT AGARWAL (Project Lead/Lead Backend Developer)
AKRITI GHOSH (Frontend Developer)
VEDANT BAKSHI (Frontend Developer)
KADAM TRIPATHI (Designer)
DINAKAR (Documentation incharge)

BTech Data Science, Section-A, Lab Batch-1
Vth Semester,3rd Year,

Introduction And Motivation

Lyrical is a proposed music inspired web app to be built on top of the Spotify API with certain 'Quality Of Life' improvements to the search and Filtration process of the various available songs. It will allow for users to register their account and search for their music in new and more robust ways and save their search history for future reference.The Web app is supposed to simulate the addition of these features to the actual Spotify App.

The motivation for the project comes from the mutual interest and love for music in the team. It's quite common that a particular song lyric is stuck in our heads and we just can't figure out the name of the song. There may also be moments where we want the songs from a particular year. So the motivation for the project lies within building small Quality of Life improvements within the Spotify app providing a more easily accessible and robust Searching and Filtration system for Your music.

Objectives

As stated above the main purpose of Lyrical is "Quality of life" improvements for the users. Although the problem we're looking at is not groundbreaking, It is an extra tool at the user's disposal to make finding their Music just that little bit easier..

Lyrical isn't "Reinventing the wheel" in the music apps world nor seeks to be a new competitor to giants like Apple Music or Spotify. It's merely a set of extra features built on top of those Apps to solve simple headscratchers for music lovers in their time of need.

It aims to provide a fully functional User Registration Login system and allow for users to organize and save their searches for future references.

It aims to add multiple search and filtration options like "Search By Lyric" and "Search By Year" and many other such relevant Filters and Searches to the pool of ones already existing in spotify.

Technologies

The Web App will be built using using the following technologies-:

- Frontend: HTML, CSS, Javascript
- Backend: NodeJS,ExpressJS
- Database: MongoDB

Modules

1.) Fully functional user Login/Registration system.
2.) Ability to save their music info and organize it

3.) Searching music By Song Name, Artist, genre, Lyrics, Year etc
4.) Allows for viewing and selecting of lyrics of a song

Methodology

The idea behind the web app is quite simple- act like an extensible feature for a music application. As such the methodology is quite simple-:

1.) The User can come in and register themselves on the web app and then login.
2.) They can then search for their music using all the filters like-:
    - Lyrics-: All songs with the entered lyrics are displayed
    - Year of release-: Songs from the entered year are displayed
    - Artist-: Songs are filtered by their artist
    - Genre-: Search Songs from a particular genre

    The data will be fetched from the spotify API and filters will be applied to it and the filters will apply to it within the fetch call itself so that only the required amount of data is brought in. Everything will be displayed to the user in a carefully designed card model.
3.) If they find a song and they wish to save it, they'll be able to do so easily with the click of a button that maps to an endpoint to send a request to the Backend of the Web app itself.
    The data will be stored and attached to their user profiles in MongoDB.
4.) Users can easily manage their account details and organize their playlists.

Conclusion

As a team this project helps us to learn the various complexities that come with building a full web app like this and understanding the nuances of Web Development with the modern industry standards. Picking a music related project is part of the team synergy building as the love faor music that all members can find common ground on and hence the motivation to make

this a solid learning opportunity. As stated above, the idea and website are not meant to be revolutionary and groundbreaking solutions rather just ease a minor inconvenience of the everyday music listener.

LYRICAL: SRS

Index

5.3 : Security Requirements

1: Introduction

1.1: Purpose-:

This Document lays down and highlights the Development Plan and details for Our Project Lyrical. The purpose is to lay out a detailed description of the project's intent,target audience, functional requirements,overview etc.

1.2: Intended Audience-:

The document is meant to introduce and explain 'Lyrical' to all the readers. The audience intended is the respected faculty, the team behind Lyrical and everyone who wishes to understand the working of the project.

1.3: Project Scope-:

Lyrical is meant to be a self-sustained extension for already existing popular music services like Spotify, Amazon prime music, Soundcloud etc. Ease of use and quality of life for users are probably the most important segments of any product. Every design, every advancement is made to make it easy for the general public.

The current music providers have done an excellent job making it easy for users to find and save their favourite songs. However, with lyrical we intend on building upon this and making it further easier for users to find their favourite songs.

That being said, we aren't looking to or claiming to be the next Big Music provider. We just wish to build something to improve ease of access to music.

२: Overall Description

२.१: Product Perspective-:

Lyrical targets all those who are fond of music and pay attention to lyrics. The team behind lyrical are all music enthusiasts and the idea was solidified when we realized that although available there is not really a set in stone way to search for music via lyrics. How many times has it happened that there are these lyrics stuck in your head and you just can't figure out which song they are from? This is the primary concern addressed by Lyrical.

Thus Lyrical isn't changing the world. It's just a humble attempt to improve everyday music browsing experience.

२.२: Functional Requirements-:

1.) Users should be able to Register and create their accounts. They should be allowed to login with the same account.

2.) Users should be able search and view songs based on lyrics that they enter.

User should also be able to search music By Song Name, Artist, genre, Year etc

3.) Users should be able to save said songs to their accounts in terms of "Playlists"

२.३: User Classes-:

For Lyrical in its current State, there is only One class of user i.e., all music lovers. There is no segmentation there. The only other "class" could be the development team of lyrical.

2.4: Design and Implementations constraints-:

The constraints we are currently working under is the fact the user can only browse the song info, not actually listen to the song i.e, there is no song player.

The Constraint on us is also economic. We are using all technologies that are free to use-:

For deployment purposes we'll also be using Heroku as it is free and hence the extension will work much slower.

Also, the database will be on a shared cluster of MongoDB Atlas again as that is free.

2.5: Platform-:

Lyrical Provides a user interface through Simple HTML, CSS and JavaScript. The Backend API is built in NodeJS and ExpressJS. Database is MongoDB Atlas.

The app will be deployed using Heroku as a Monolithic application. It will be freely accessible as a website on the internet

3: External Interfaces

- Cloudinary as the cloud to store User photos
- MongoDB Atlas as our database
- Mailtrap for sending mails to the user

4: System Features

4.1: User Interface-:

The Logic is all implemented at the backend which will provide a RESTful API for all the functionality.Hence we need a proper User Interface to access all the features of Lyrical. The User Interface will require-:

- Login Form
- Register Form
- Lyrics Search Bar
- Interface to view and edit playlists
- Search bar for other filters as required
- Interface to view and update account details

4.2: Communications Interface-:

To get all the data about songs we're using a spotify API from RapidAPI. Thus this API communicates to us all the info about the songs. In our implementation, We'll implement the API in such a way that with each request we get only the data that is required and not the entire data. It will be implemented in the Client Side JavaScript.

On the server side we have our own API that provides functionality for User Account and playlists management. User can-:

i. Login
ii. Register
iii. Update Account
iv. Update Password
v. Forget Password
vi. Logout
vii. Add to playlist

viii. Remove from playlist
ix. Delete playlist
x. Make a new playlist.

5: Other non-functional requirements

5.1: Performance Requirements-:

- The application should load and be usable within a few seconds
- The application should update the interface on interaction within few seconds

5.2: Safety Requirements-:

- There should be no data leak through the frontend console.
- There should be no data leak via querying.

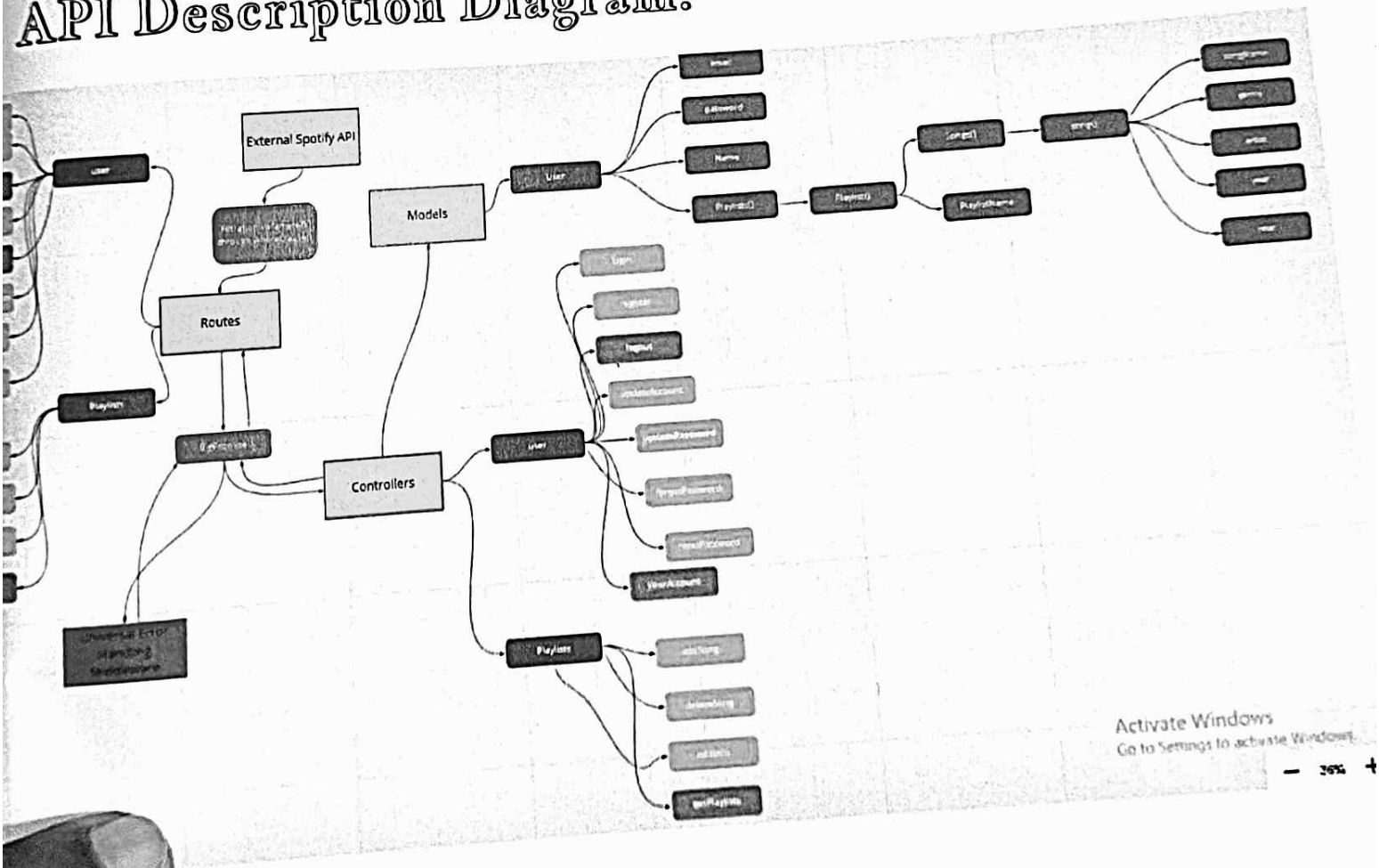
5.3: Security Requirements-:

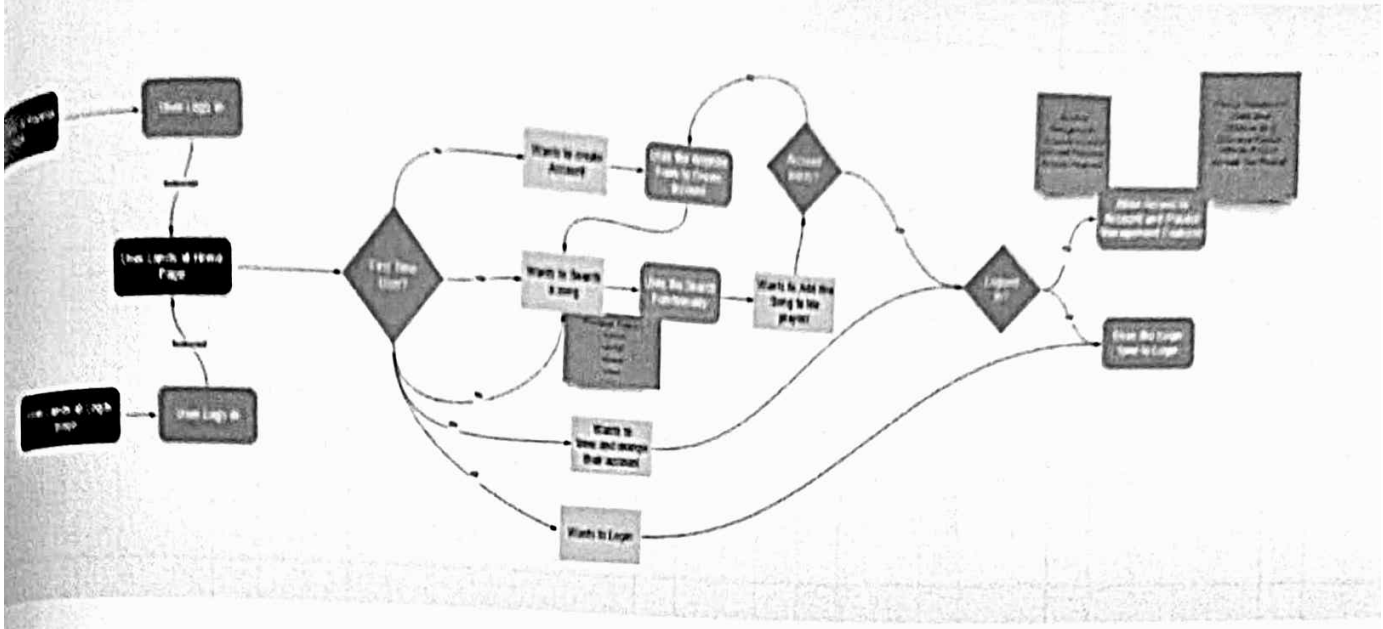
- Password should be encrypted
- User data shouldn't be shared.

LYRICAL:

Software Design

API Description Diagram:





Use Case diagram: