Project Report

On

Farmlog : An Insightful App For Modern Farming Practices

Submitted to D Y Patil International University, Akurdi, Pune in partial fulfilment of full-time degree

Bachelor of Computer Applications

Submitted By:

Name: Shekhar Patil

PRN: 20210801123

Name: Manish Parihar

PRN: 20210801124

Name: Rohit Patil

PRN: 20210801128

Under the Guidance of

Dr. Sarika Jadhav

School of Computer Science, Engineering and Applications

D Y Patil International University, Akurdi,Pune, INDIA, 411044

[Session 2023-24]

D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

CERTIFICATE

This report on AWS BlogWiz: Unleashing the Power of Django in the Cloud is submitted for the partial fulfillment of project,which is part of Bachelor of Computer Applications curriculum, under my supervision and guidance.

Ms. Hetal Thaker
(DYPIU Guide)

Dr. Swapnil Waghmare
(Project Coordinator)

Dr. Maheshwari Biradar
(HOD BCA & MCA)

Dr. Bahubali Shiragapur
Director
School of Computer Science Engineering & Applications
D Y Patil International University, Akurdi
Pune, 411044, Maharashtra, INDIA

DECLARATION

I, hereby declare that the following Project which is being presented in the Project entitled as AWS BlogWiz: Unleashing the Power of Django in the Cloud is an authentic documentation of my own original work to the best of my knowledge. The following Project and its report in part or whole, has not been presented or submitted by me for any purpose in any other institute or organization. Any contribution made to my work, with whom i have worked at D Y Patil International University, Akurdi, Pune, is explicitly acknowledged in the report.

Name: Chaitanya Mule
PRN No: 20210801057 Signature :

Name: Aditya Gaonkar
PRN No:20210801058 Signature :

Name: Bushra Tamboli
PRN No:20210801033 Signature :

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide Ms. Hetal Thaker, for her valuable help and guidance. We are thankful for the encouragement that she has given us in completing this Project successfully.

It is imperative for us to mention the fact that the report of project could not have been accomplished without the periodic suggestions and advice of our project supervisor Dr. Swapnil Waghmare.

We are also grateful to our respected, Dr. Bahubali Shiragapur(Director), Dr. Maheshwari Biradar (HOD, BCA & MCA) and Hon'ble Vice Chancellor, DYPIU, Akurdi, Prof. Prabhat Ranjan for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing support and encouragement.

Name: Chaitanya Mule
PRN No: 20210801057

Name: Aditya Gaonkar
PRN No: 20210801058

Name: Bushra Tamboli
PRN No: 20210801033

Abstract

Building an online presence through blogging is not only a common pastime in the current digital era, but it's also essential for individuals and companies to share their knowledge, insights, and experiences with the world. It is imperative to guarantee that your blog platform is both secure and scalable in order to handle growing traffic and safeguard confidential information. In this project, we set out to install a stable blog application built using Django on Amazon Web Services (AWS). Through the use of several AWS services, including ec2, RDS, s3, DynamoDB, CloudFront, and route 53, we hope to establish a web application environment that is highly secure, scalable, and available.

keywords: Django Framework,Amazon Elastic computation Cloud (EC2),Relational Database Service (Amazon),Amazon S3 (Simple Storage Service),Amazon DynamoDB,Amazon CloudFront,Amazon Route 53

TABLE OF CONTENTS

List of Figures

1. INTRODUCTION

1.1. Background

a vital component of today's internet environment, blogging gives people and businesses a forum to communicate with a worldwide audience about their ideas, experiences, and knowledge. the requirement for safe and scalable web apps to support blogging platforms is growing along with their demand. in the past, configuring and managing a blog application required large expenditures in infrastructure and resources, such as database administration, server management, and making sure that strong security measures were in place. but thanks to cloud computing and platforms like amazon web platforms (aws), developers now have resources and tools at their disposal to make the development and deployment process go more quickly.

1.2. Objectives

A document wallet application serves as a digital hub for users to efficiently manage their documents. Its primary objectives encompass organizing and securely storing documents, enabling easy accessibility from anywhere, ensuring robust security measures, facilitating quick search and retrieval, promoting collaboration (if applicable), automating tasks, integrating with other software, implementing backup and recovery measures, adhering to compliance standards (for regulated industries), optimizing user experience, and catering to mobile accessibility. User feedback and continuous improvement are integral to achieving these objectives, ensuring that the application remains a reliable and user-friendly tool for document management.

1.3. Purpose

The purpose of a document wallet app is to provide users with a digital solution for effectively managing, organizing, and securing their important documents. This type of application serves as a convenient and centralized platform for individuals or organizations to store, access, and interact with various types of documents, such as PDFs, images, spreadsheets, and more. munity.

1.4. Scope

The scope of a document wallet application is multifaceted, ranging from personal document management for individuals to comprehensive document handling solutions for businesses and organizations. Such an application can expand its reach by integrating with popular cloud storage services and ensuring cross-platform compatibility.

1.5. Applicability

A document wallet application demonstrates its wide applicability across diverse domains and industries, serving as a versatile tool for effective document management and organization. From individuals seeking to streamline their personal document storage to businesses in need of secure collaboration and data management, this application finds utility. It proves invaluable in sectors with stringent data privacy regulations like healthcare and legal, where sensitive information must be handled with care. Educational institutions, government agencies, and financial services can optimize their document workflows, while freelancers, entrepreneurs, and researchers benefit from its organizational capabilities. Whether in construction, real estate, travel, or non-profit sectors, the adaptability of this application facilitates better document handling, compliance, and efficiency, making it a valuable asset in today's digitally-driven world.

2. PROJECT PLAN

2.1. Problem Statement

- **Security:** Setting up secure access controls and encryption measures can be challenging.
- **Scalability:** Configuring auto-scaling and load balancing for optimal performance can be complex.
- **Integration:**Seamless communication between services and handling dependencies may pose challenges.
- **Performance optimization:** Identifying and addressing performance bottlenecks require thorough testing.
- **Deployment automation:** Setting up automated deployment pipelines and managing multiple environments can be complex.
- **Monitoring and debugging:** Monitoring health, detecting errors, and debugging issues in a distributed environment can be challenging.

2.2. Requirement Specification

- **Software Requirements:**
  ”AWS BlogWiz: Unleashing the Power of Django in the Cloud” requires Android 8 and above as the operating system, the Dart programming language (Flutter), and Firebase for the backend infrastructure.
- **Hardware Requirements:**
  Users can download the application with a download size of approximately 40MB. There are no specific hardware requirements beyond a compatible Android device.

2.3. Time Line chart

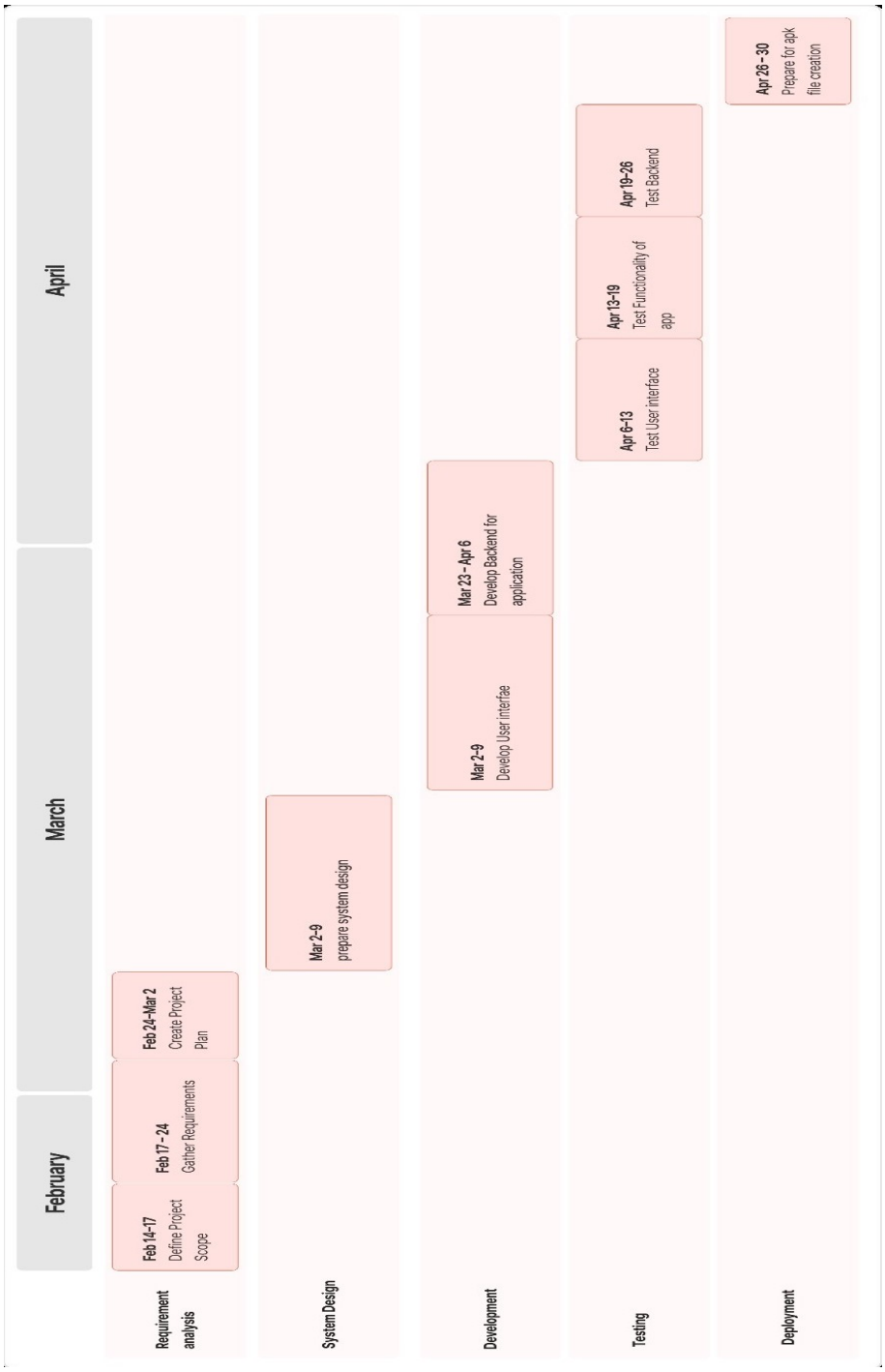

Fig. 1: Time Line Chart

3. PROPOSED SYSTEM AND METHODOLOGY

3.1. System Architecture

1. **ERD Diagram**

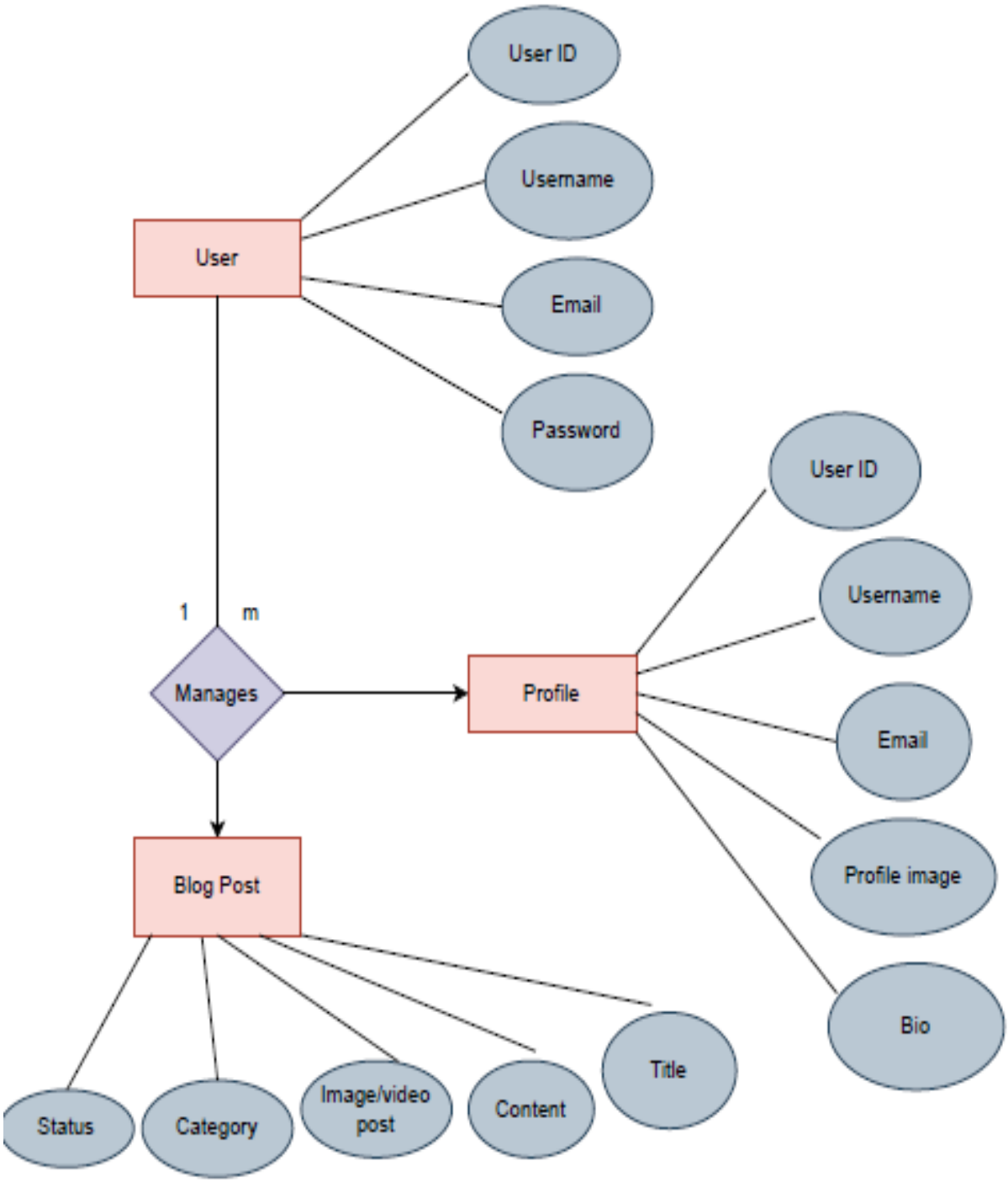

Fig. 2: ERD Diagram

2. **Flow Chart**

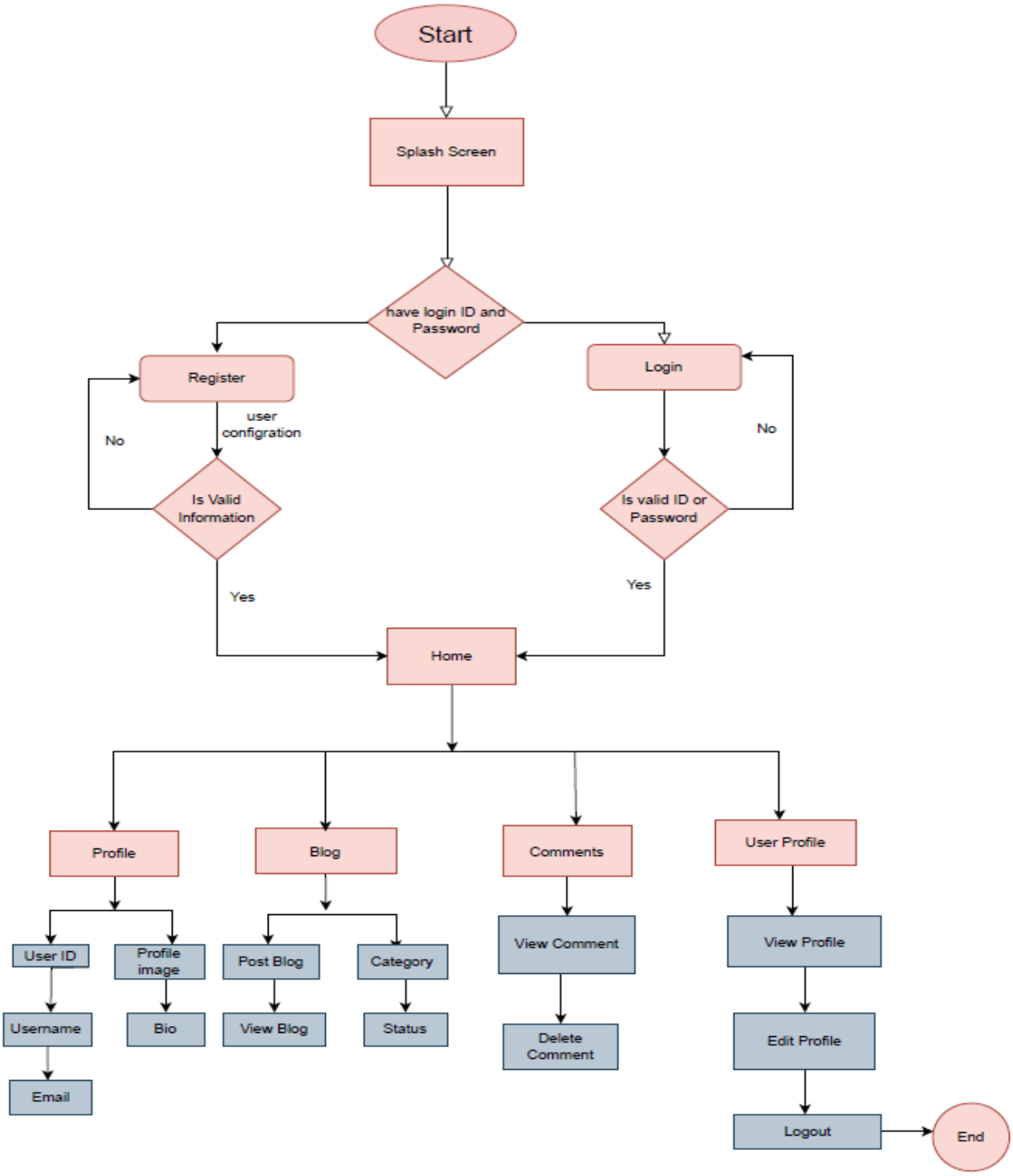

Fig. 3: Flow Chart

3.2. Implementation

3.2.1. Data Flow Diagrams

1. **Level 0 DFD**

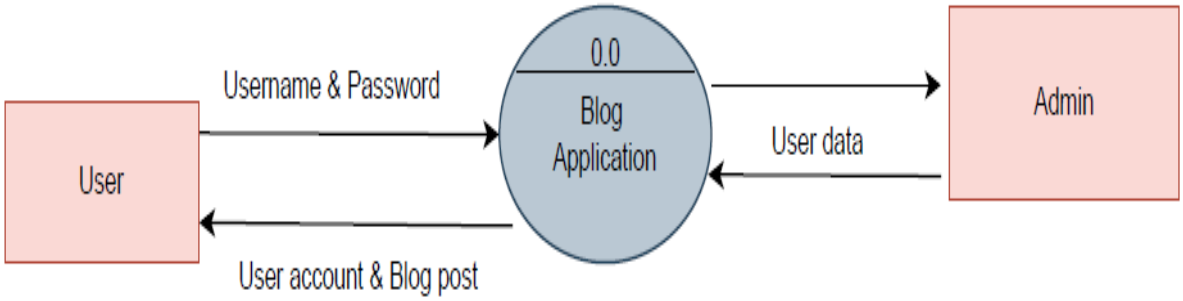

Fig. 4: Level 0 DFD

2. **Level 1 DFD**

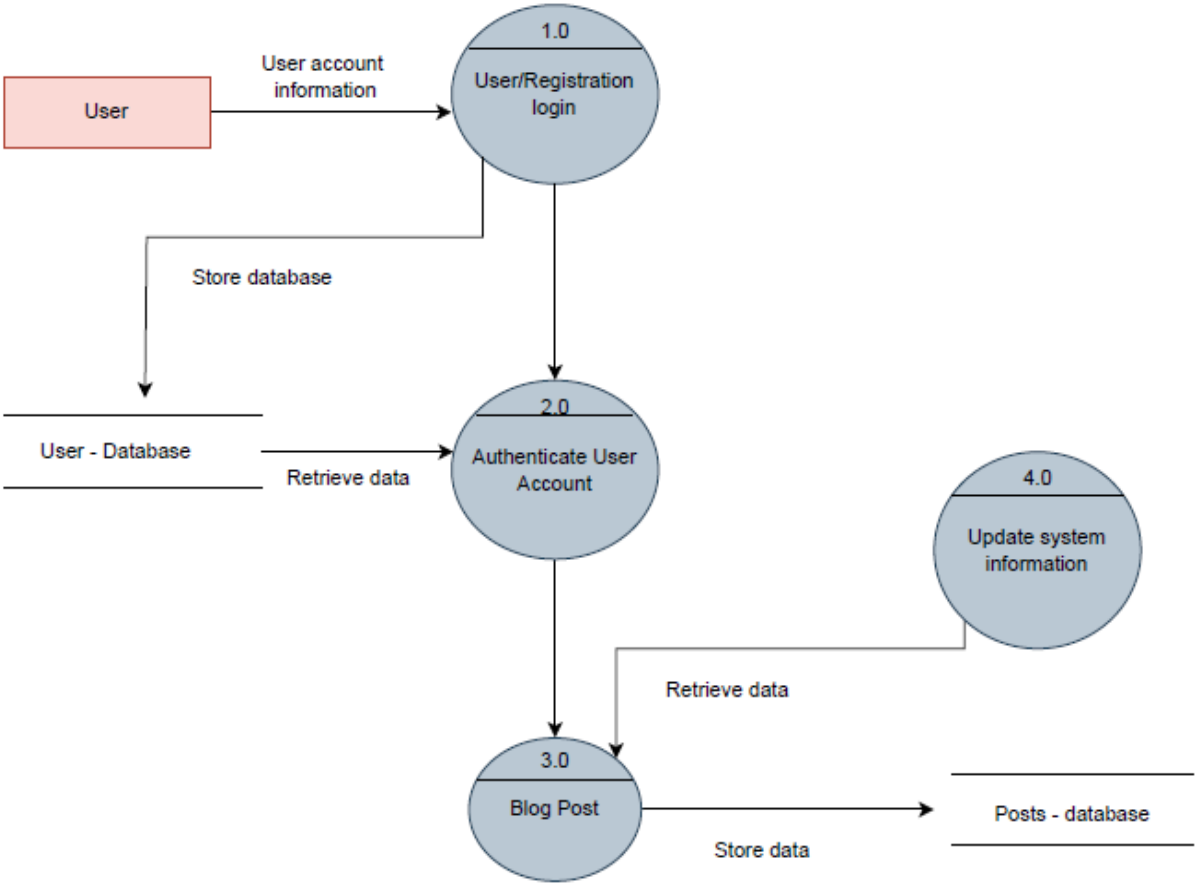

Fig. 5: Level 1 DFD

3. **Level 2 DFD**

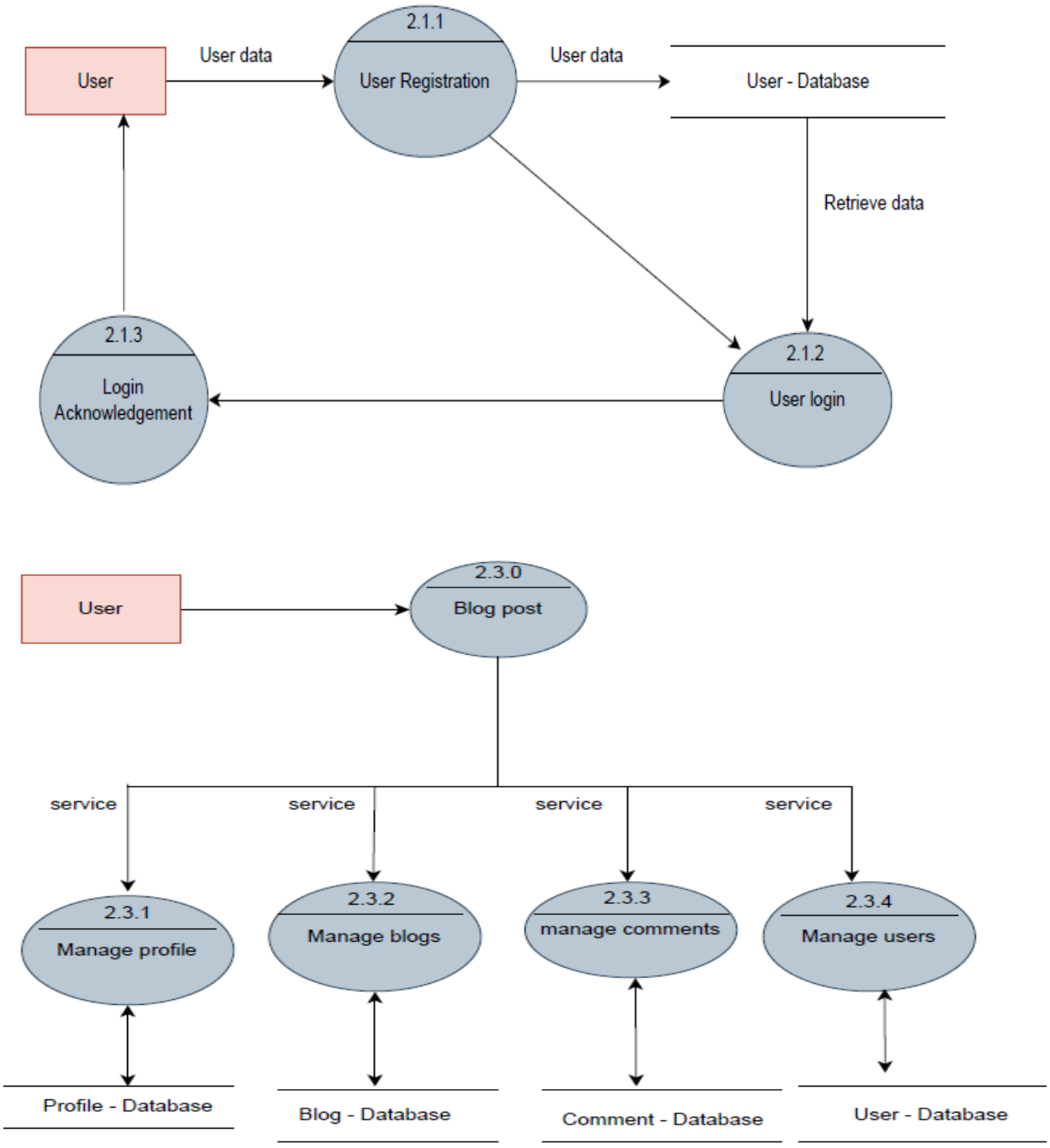

Fig. 6: Level 2 DFD

3.2.2. UML Diagrams

1. **Use Case Diagram**

Fig. 7: Use Case Diagram

2. **Sequence Diagram**

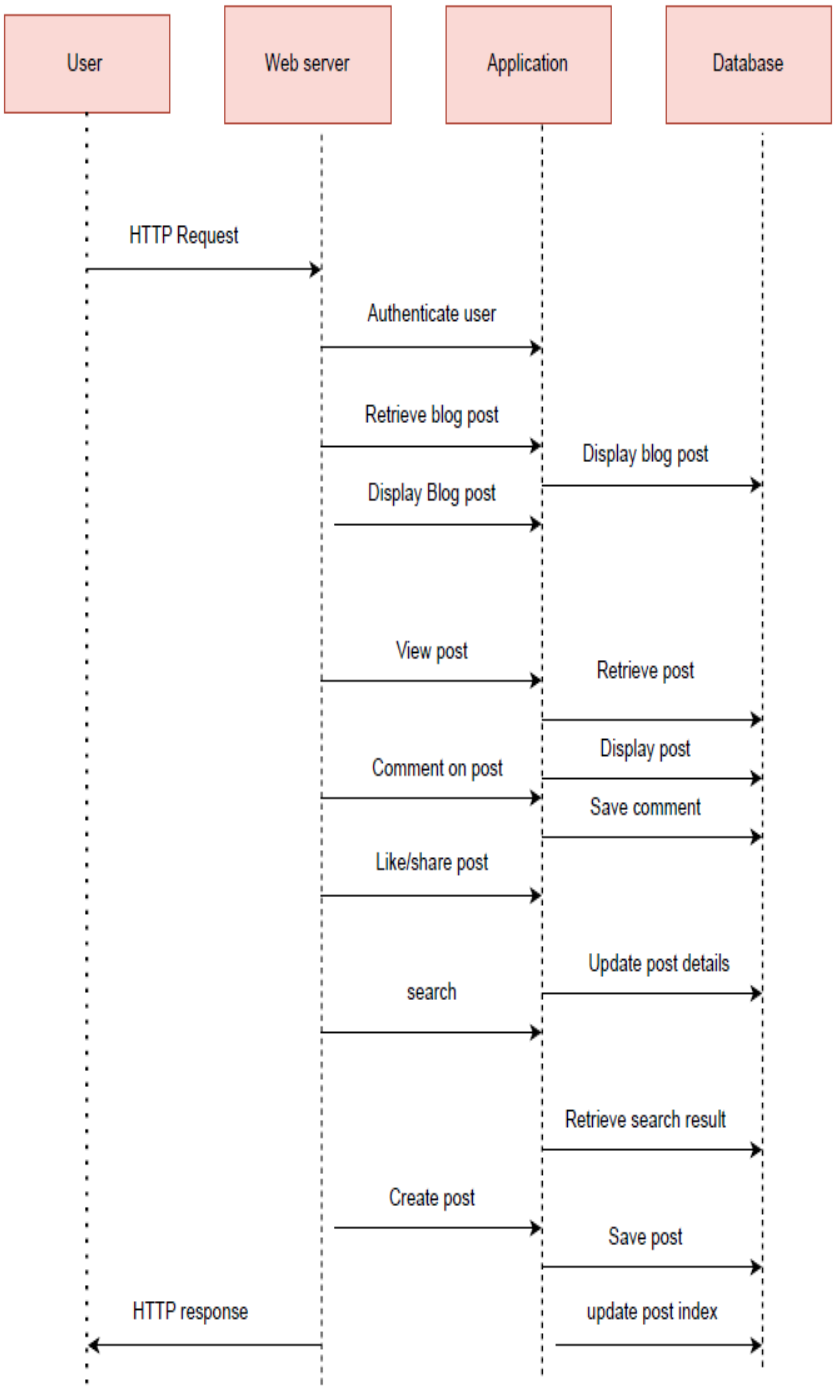

Fig. 8: Sequence Diagram

3. **Activity Diagram**

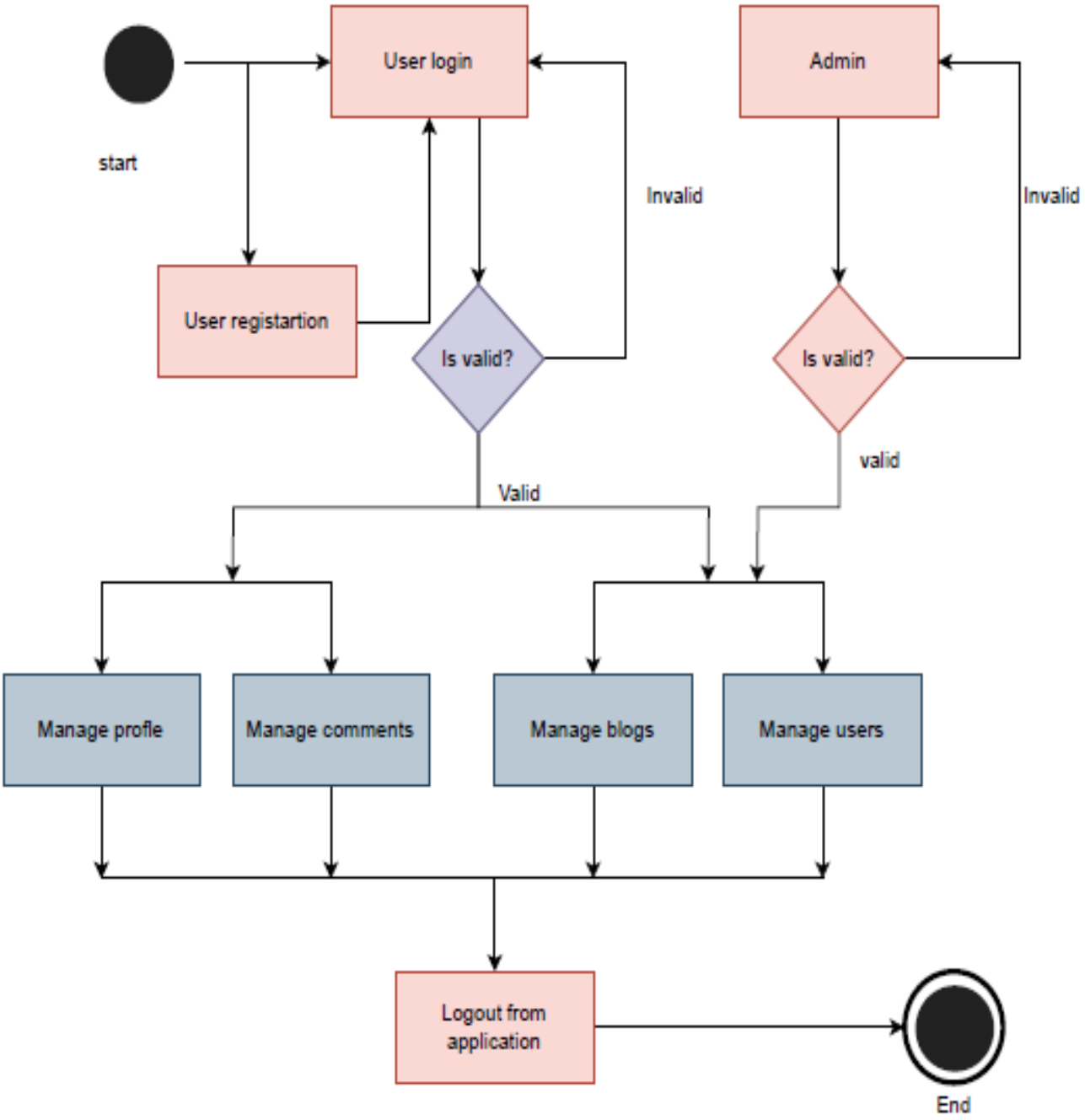

Fig. 9: Activity Diagram

4. **Class Diagram**

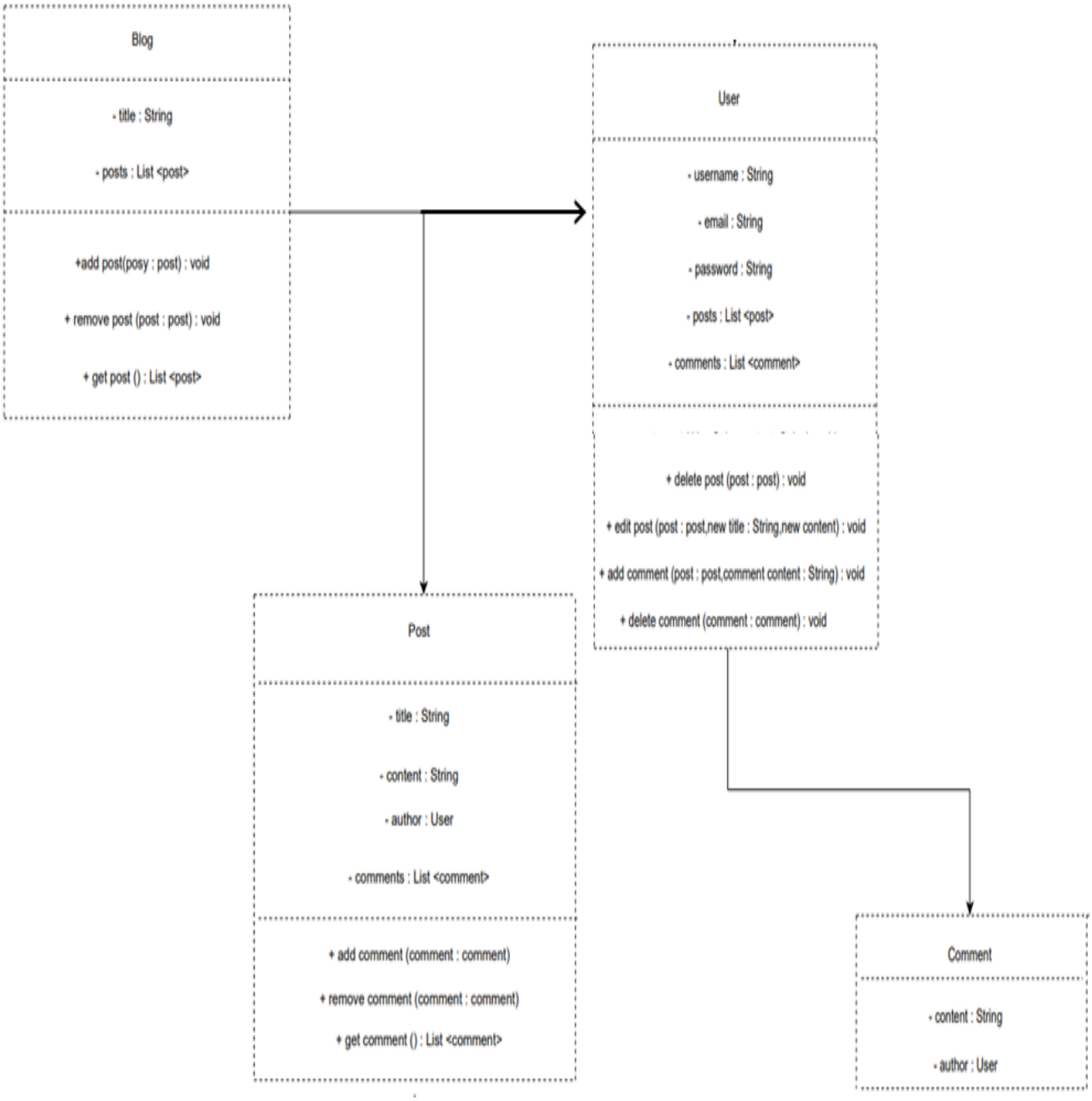

Fig. 10: Class Diagram

3.2.3. Decision Tree

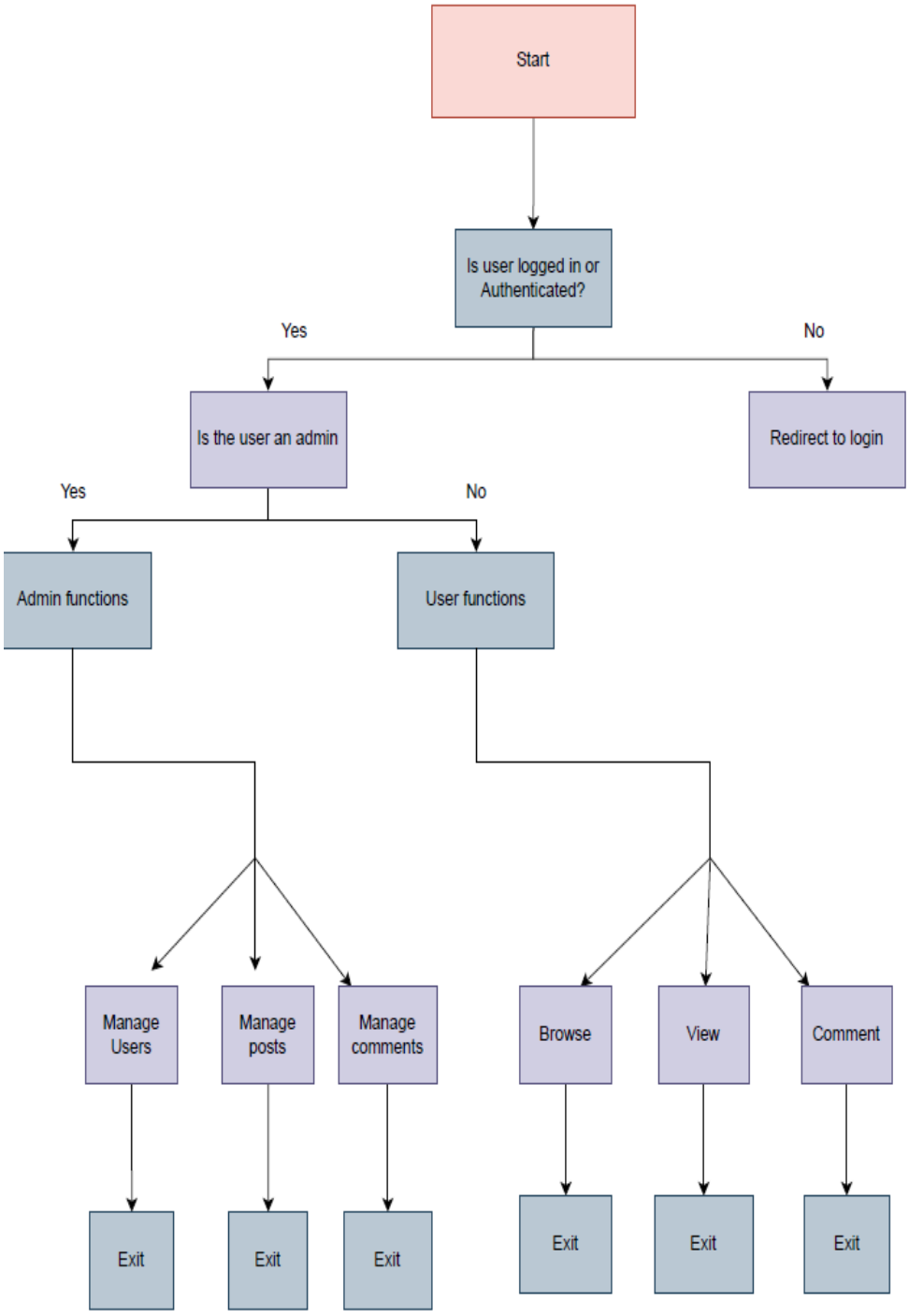

Fig. 11: Decision Tree

3.2.4. Data Dictionary

Field Name	Attribute	Data Type	Field Length	Description
User	1. Username	String	50	The Username is chosen by user.
	2. Email	String	100	The email Address is Associated with user.
	3. Password	String	50	The password used to Authenticate the user.
Blog Post	1. ID	String/Integer	-	A unique Identifier for blog Post.
	2. Title	String	200	The title of the blog post.
	3. Content	String	-	The Content Body of the blog Post.
	4. Date Created	Date & Time	-	The Date & Time when the blog post was created.
	5. Comments	List of Comments	-	Comments Made on Post
Comment	1. ID	String/Integer	-	A unique Identifier for blog Comment.
	2. Content	String	500	The Content Body of the blog Post.
	3. Date Created	Date & Time	-	The Date & Time when the comments was made.
Blog Application	1. User	List of users	-	Collection of registered users.
	2. Blog Post	List of Blog Post	-	Collection of blog posts created by user.

3.2.5. Decision Table

Scenario	Condition 1	Condition 2	Condition 3	Actions
User Registration	Username not already taken.	Email format is valid and not already registered.	Password meets Requirements.	Create new User Account.
User Login	Username/Email & Password match.	-	-	Grant Access.
Create Blog Post	User is logged in	Title and content are not empty.	-	Create new blog Post.
Edit Blog Post	User is Logged in	User is the author of the blog Post.	New Title content is not Empty.	Edit Existing Blog Post.
Delete Blog Post	User Is logged in	User is the Author of the blog Post.	-	Delete Existing Blog Post.
Comment On Blog Post	User is Logged in	Content of the comment is not empty.	-	Add the Comment to Blog Post.
Search Blog Posts.	Keyword are Printed.	-	-	Retrieve and display the matching Blog Posts.

4. RESULTS AND EXPLANATION

4.1. Implementation Approaches

Building an AWS-based web application involves several key steps and decisions to effectively utilize Amazon Web Services (AWS) for deploying and managing the application. Initially, it is crucial to plan the architecture by gathering requirements and designing a suitable structure, whether it be microservices or monolithic. Selecting appropriate AWS services is the next step, involving choices like EC2 instances or AWS Lambda for compute resources, Amazon S3 for storage, and Amazon RDS or DynamoDB for databases. Networking setup with a Virtual Private Cloud (VPC) and security configurations using AWS IAM and other security services is essential.

Following setup, the development phase includes coding the application and integrating AWS services such as S3 for storage and RDS for databases. Deployment is facilitated through CI/CD pipelines using AWS CodePipeline, CodeBuild, and CodeDeploy, ensuring seamless and automated deployment processes. Scaling and load balancing are managed using Auto Scaling groups and Elastic Load Balancers (ELB) to handle traffic fluctuations and ensure high availability.

Monitoring and logging are crucial for maintaining the application's health and performance, achieved through Amazon CloudWatch and CloudWatch Logs. Security measures, including data encryption with AWS KMS and compliance adherence, are paramount to protect sensitive data and meet industry standards. Cost management involves using AWS Budgets to monitor expenses and optimizing resource usage to minimize costs. Regular maintenance and updates, including security patches and application updates, ensure the application remains secure and up-to-date. This comprehensive approach leverages AWS's robust ecosystem to build, deploy, and manage scalable and secure web applications effectively.

4.2. Pseudo Code

User Authentication Functions

```
function login(userCredentials):


    if authenticationSuccessful:
        currentUser = getUserInfo(userCredentials)
        showHomePage()
    else:
        showErrorMessage()

function logout():
    user loggedout

function getUserInfo(userCredentials):
```

Home Page Functions

```
function showHomePage():
    displayBlogs(loadBlogs())

function loadBlogs():


    return blogs

function displayBlogs(blogs):
    # Display a list of blogs on the home page
    for each blog in blogs:
        displayBlogSummary(blog)

function displayBlogSummary(blog):
    # Display a summary of a blog to the full post

function shareUserBlog(userBlog):
    # Share user blog to external apps
```

Blog Creation Functions

```
function createBlog(user, title, content):


    if blogCreatedSuccessfully:
        showHomePage()
    else:
        showErrorMessage()
```

Blog Details Functions

```
function viewBlogDetails(blog):
    displayBlog(blog)
    displayFeedbacks(loadFeedbacks(blog))
    shareBlog(blog)

function displayBlog(blog):
    # Display the full content of a blog post

function loadFeedbacks(blog):


    return feedbacks

function displayFeedback(feedbacks):
    # Display a list of feedbacks for a blog
    for each feedback in feedbacks:
        displayFeedback(feedback)
```

Blog Feedback Functions

```
function feedbackOnBlog(blog, content):


    if feedbackPostedSuccessfully:
        showSuccessMessage()
    else:
        showErrorMessage()

function sendPrompt(message)
    if messageSendSuccesfully:
        receiveResponce(message)
    else:
        showErrorMessage()
```

}

function receiveResponce(message):
    return response;

Profile Page Functions

function viewProfile(user):
    displayUserProfile(user)
    displayUserBlogs(loadUserBlogs(user))

function displayUserProfile(user):
    # Display user's profile information

function loadUserBlogs(user):

    return userBlogs

function displayUserBlogs(userBlogs):
    # Display a list of blogs posted by the user
    for each blog in userBlogs:
        displayBlogSummary(blog)

function editUserBlogs(userBlogs):
    # Update user blog

Feedback function

function feedback():

Question Page Functions

function showCommuntyPage():
    displayQuestion(loadQuestion())

function loadQuestions():

    return Questions

```
function displayQuestion(questions):
    # Display a list of question on the home page
    for each question in questions:
        # display each questions

function displayQuestionSummary(question):
    # Display a summary of a question to the full post
```

Question Creation Functions

```
function createQuestion(user, title, content):


    if questionCreatedSuccessfully:
        showCommunityPage()
    else:
        showErrorMessage()
```

Question Details Functions

```
function viewQuestionDetails(question):
    displayQuestion(question)
    displayAnswer(loadAnswers(Question))

function displayQuestion(question):
    # Display the full content of a question post

function loadAnswers(question):


    return answers

function displayAnswers(answers):
    # Display a list of answers for a question
    for each answer in answers:
        # display all answer for each question
```

Answer Functions

```
function answerOnQuestion(Question, content):


    if answerPostedSuccessfully:
        showSuccessMessage()
```

```
    else:
        showErrorMessage()

function upvoteAnswerOnQuestion(Question):


    if ipvotePostedSuccessfully:
        showSuccessMessage()
    else:
        showErrorMessage()
```

Event Page Functions

```
function showEventPage():
    displayEvent(loadEvents())
function loadEvents():


    return Event

function displayEvent(Events):
    # Display a list of Events on the home page
    for each event in events:
        # display each events

function displayEventSummary(event):
    # Display a summary of a event to the full post
```

Event Creation Functions

```
function createEvent(user, title, content):


    if eventCreatedSuccessfully:
        showEventsPage()
    else:
        showErrorMessage()
```

Event Details Functions

```
function viewEventsDetails(event):
    displayEvent(event)
```

```
function displayEvent(event):
    # Display the full content of a event post

function loadEvent(event):
```

Main Application

```
function main():
    if userIsLoggedIn():
        showHomePage()
    else:
        promptUserToLogIn()

main()
```

4.3. Testing

Our testing strategy encompassed both automated unit testing and manual user testing to ensure the functionality, reliability, and usability of our agriculture community application. Leveraging Django test framework, we wrote automated tests for each component and module, validating their behavior and conformity to specified requirements. This rigorous unit testing approach allowed us to identify and rectify bugs and issues early in the development cycle, contributing to the overall stability and robustness of the application.

In parallel, we conducted manual user testing by engaging users to assess the application's usability and user experience. These users were tasked with performing specific scenarios and actions within the application, such as browsing agricultural blogs, community questions and answers, and accessing events. Through observation and feedback collection, we gained valuable insights into user interactions, navigation patterns, and overall satisfaction levels. This qualitative feedback played a crucial role in refining the application's user interface and functionality to better align with the needs and preferences of our target audience, ensuring a positive user experience upon deployment.

5. CONCLUSION

In conclusion, the AWS BlogWiz: Unleashing the Power of Django in the Cloud; We have developed a scalable and secure blog application that not only satisfies the requirements of contemporary web development but also offers a smooth experience for both content writers and readers by utilising the strength of AWS services in conjunction with Django. This project provides evidence of how well cloud computing works for creating scalable and reliable online applications.

6. REFERENCES

- https://docs.aws.amazon.com/lambda/latest/dg/foundation-networking.html

- https://docs.aws.amazon.com/autoscaling/ec2/userguide/ts-as-healthchecks.html