Tribhuvan University Institute of Science and Technology

Texas International College, Kathmandu

Supervisor's Recommendation

I hereby recommend that this project report be prepared under my supervision by Ms. Anusha Maharjan (23734), Ms. Anusha Nepal (23735) and Mr. Manish Gautam (24704) Entitled "**Quiz App Using SVM classification Algorithm**" is satisfactory in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc.CSIT) is recommended for the final evaluation.

…….

Mr. Ram Prasad Subedi

Project Supervisor

Department of B.Sc. CSIT

Tribhuvan University Institute of Science and Technology

Texas International College, Kathmandu

Letter of Approval

This is to certify that the project was prepared by Mr. Manish Gautam , Mrs. Anusha Maharjan and Mrs. Anusha Nepal Entitled "**Quiz App Using SVM Classification Algorithm**" in partial fulfillment of requirements for the degree of B.Sc. in Computer Science and Information technology (B.Sc.CSIT), Institute of Science and Technology, Tribhuvan University has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

……..

Ram Prasad Subedi

Project supervisor

……

Omkar Basnet

Head Of Department

Texas International College

……..

Internal Examiner

……..

Sarbin Sayami

External Examiner

Acknowledgment

The successful completion of our project stands as a testament to the invaluable support and assistance we received from numerous individuals and organizations. We extend our sincere gratitude to each of them for their unwavering commitment throughout the project's duration.

Our project supervisor, Mr. Ram Prasad Subedi, deserves special mention for his keen interest, guidance, and provision of essential ideas, information, and knowledge. His mentorship was crucial in the development of a functional web application and in ensuring our project report adhered to the required standards and norms.

Texas International College, Kathmandu, deserves our deepest appreciation for their constant guidance, supervision, and provision of essential ICT infrastructure. A special acknowledgment goes to Mr. Omkar Basnet, the HOD of B.Sc. CSIT, Mr. Sulav Nepal, the B.Sc. CSIT incharge, whose tireless efforts played a pivotal role; without his support, our project would have faced significant challenges.

We express our heartfelt thanks to the seniors and teaching staff of the B.Sc. CSIT department for their unwavering support, as well as the non-teaching staff for their timely assistance. Gratitude extends to the students of Texas International College and others who contributed vital statistical data, playing an indispensable role in the project's success.

Lastly, we extend our thanks and appreciation to each colleague for their encouragement and steadfast support throughout the challenging yet rewarding journey of developing the project.

Abstract

Introducing "Quiz App," a web application designed to revolutionize education learning methods. This platform employs a smart system where questions are randomly chosen, providing multiple users with personalized tests.

Using the Agile methodology, the high-level design incorporates cutting-edge technologies such as Python, HTML , CSS, Js, Vue.js, Tailwind CSS, Django. The system ensures fairness in testing through a randomization implemented in the database presenting questions in a random order to every user. Additionally, a Support Vector Machine (SVM) algorithm categorizes questions based on user performance, facilitating a tailored learning experience.

The feasibility study demonstrates technical, operational, and economic viability, with the entire project outlined in a Gantt chart spanning three months. The application boasts user-friendly operation, demonstrated through a flowchart illustrating the system's processes and a use case diagram showcasing user interactions.

For users of the Quiz App, the experience is designed to be straightforward and enjoyable. A personal dashboard displays past test history and available quizzes, making it easy to navigate. Taking a test is a simple process with clear instructions and feedback upon completion. The app provides insights into performance, highlighting strengths and areas for improvement. Results are sent automatically, streamlining the process. The aim is to create a positive and user-friendly environment for individuals using the app, making learning and testing an accessible and enjoyable experience.

Keywords: *Quiz App, Django, SVM*

Table of Contents

Supervisor's Recommendation i
Letter of Approval ii
Acknowledgment iii
Abstract iv
List of Abbreviations vii
List of Figures viii
List of Tables ix
Chapter 1 : INTRODUCTION 1
1.1 Introduction 1
1.2 Problem Statement 1
1.3 Objectives 1
1.4 Scope and Limitation 2
1.5 Development Methodology 2
1.6 Report Organization 3
Chapter 2 : BACKGROUND STUDY AND LITERATURE REVIEW 4
2.1 Background Study 4
2.2 Literature Review 4
Chapter 3 : SYSTEM ANALYSIS 6
3.1 System Analysis 6
3.1.1 Requirements Analysis 6
i. Functional Requirements 6
ii. Non- Functional Requirements 8
3.1.2 Feasibility Analysis 9
i. Technical Feasibility 9
ii. Operational Feasibility 9
iii. Economic Feasibility 9
iv. Schedule Feasibility 9
3.1.3 Analysis 10
Chapter 4 : SYSTEM DESIGN 15
4.1 System Architecture and Overview 15
4.2 System Diagrams 15
4.2.1 System flow chart of Quiz App 15
4.3 Algorithm Details 16
Chapter 5 : IMPLEMENTATION AND TESTING 18

5.1 Implementation 18
5.1.1 Tools Used 18
5.1.2 Implementation Details of Modules 19
5.2 Testing 25
5.2.1 Unit Testing 25
5.2.2 Integration Testing 26
5.2.3 System Testing: 27
5.3 Result Analysis: 27
Chapter 6 : CONCLUSION AND FUTURE RECOMMENDATION 29
6.1 Conclusion 29
6.2 Future Recommendation 29
References 31
Bibliography 32
APPENDICES 33

List of Abbreviations

AI	Artificial Intelligence
CSS	Cascading Style Sheets
CSIT	Computer Science and Information Technology
DBMS	Database Management System
ERD	Entity Relationship Diagram
HTML	Hypertext Markup Language
ICT	Information and Communication Technology
JS	JavaScript
MCQ	Multiple Choice Questions
SDLC	System Development Life Cycle
SQL	Structured Query Language
SVM	Support Vector Machine
TU	Tribhuvan University
URL	Uniform Resource Locator

List of Figures

Figure 1.1Agile Methodology of Quiz App *3*
Figure 3.1 Use Case Diagram of Quiz App *7*
Figure 3.2 Sequence Diagram of Quiz App *10*
Figure 3.3 ER Diagram of Quiz App *11*
Figure 3.4 Class Diagram of Quiz App *12*
Figure 3.5 Object Diagram of Quiz App *13*
Figure 3.6 Activity Diagram of Quiz App *14*
Figure 4.1 Flow Chart of Quiz App *16*
Figure 5.1 Working Mechanism of Django framework *19*

List of Tables

Table 5. 1 Test Case for Login 25
Table 5. 2 Test case for Test taking and Availability 26

CHAPTER 1 : INTRODUCTION

1.1 Introduction

Introducing "Quiz App," an innovative web application designed to empower individuals in their journey to learning. Quiz App is a web-based tool that simplifies quiz creation, management, and participation. It offers administrators features like a dashboard, question pool, and test management, while users can access personalized dashboards, take tests, and receive results.

Utilizing an intelligent system, the Quiz App randomly selects questions from a pool, enabling multiple users to take tests and receive results based on the performance which can also be accessed by the admin to monitor user performance. Its mission is to simplify, enhance enjoyment, and foster adaptability in learning, ensuring effortless expertise development.

With Quiz App, admins set questions to see how well you're doing, making your learning experience unique. So, let's explore Quiz App together – where learning is not just easy but also fun.

1.2 Problem Statement

The issue arises in the current methods of learning, where paper tests create delays and lack adaptability. The Quiz App proposes a swift online alternative for personalized quizzes. Admins set questions, users take tests, and shared data enables a more adaptable learning experience. This solution addresses the limitations of rigid educational methods, fostering flexibility and personalization. The Quiz App acts as a bridge between admins and users, revolutionizing the learning process by introducing quick online assessments that cater to individual needs, overcoming the constraints of traditional approaches in education.

1.3 Objectives

The objectives of "Quiz App" are multi-faceted, aiming to create a dynamic and effective learning platform. Here are the key objectives:

- To simplify quiz creation, organization, and analysis through an intuitive dashboard.
- To utilize svm algorithm for categorization of the question.

1.4 Scope and Limitation

The scope for this project is to educate learners as being supervised by the administrator so that one can improve their knowledge in specific field.

Some of scopes are:

- It can be used by Colleges and Tutors to take examination, test and provide feedback.
- The platform adapts to individual user needs, allowing personalized learning.
- With a user-friendly interface, "Quiz App" caters to users of different skill levels.
- Admins can set questions, ensuring relevance and alignment with learning objectives.

Some limitations of system are:

- While covering a wide range, "Quiz App" may not include every niche area.
- The availability of learning resources depends on administrator contributions and the platform's database.
- Users are restricted to participating in tests only in the multiple-choice question (MCQ) format.

1.5 Development Methodology

The Quiz App is developed using the Agile Methodology, selected for its effectiveness in managing dynamic projects with uncertain requirements. This ensures the app's adaptability to evolving user needs and the agility to make swift adjustments, essential for a dynamic educational platform [1]. Guided by Agile principles, the development process emphasizes early deliveries, continuous collaboration, and responsiveness, allowing the Quiz App to remain adaptable and effective in the evolving educational landscape.

The adoption of Agile involves decomposing the Quiz App project into collaborative, iterative phases. This contemporary approach facilitates continuous adaptation to evolving requirements, placing significant importance on customer(team) collaboration. Through frequent increments and a focus on adaptability, the Quiz App's development under Agile aims to deliver a responsive, user-centric educational platform capable of quickly responding to emerging educational trends.

The key principles include customer collaboration, frequent deliverables, and an emphasis on adaptability, promoting a responsive and customer-centric development process [1].

The basic principles are:

- Changing Requirements: Embrace changing requirements, even late in development, for the customer's competitive advantage.
- Motivated Individuals: Build projects around motivated individuals, providing the needed environment, support, and trust for task completion.
- Effective Communication: Face-to-face conversation is the most efficient method for conveying information to and within a development team.

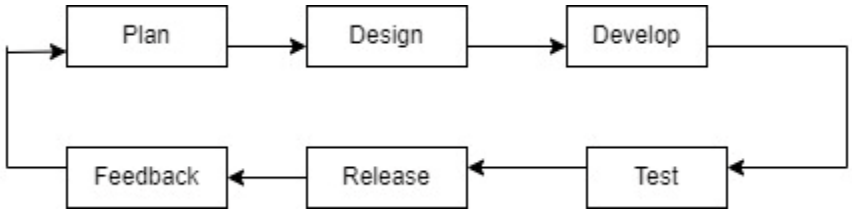


Figure 1.1Agile Methodology of Quiz App

1.6 Report Organization

The report is separated into different chapters. Each chapter contains its sub chapters with its contents. The preliminary section of the report consists of title page, letters, abstracts, acknowledgements, table of contents, list of figures and list of tables.

The first chapter contains the introduction and its sub chapters, second chapters consist of background study and literature review, third chapter contains system analysis and its sub chapters. The fourth chapter contains system design and its sub chapters, fifth contains implementation and testing and sixth contains its conclusion and future recommendation.

CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

"Quiz App" is a web application designed for use in colleges and businesses. It uses Django, Bootstrap, JavaScript, and smart algorithms to create custom tests. Admins get a dashboard and manage tests, while users have personalized dashboards to take tests and see results. Django is a Python web framework for building robust and scalable web applications [2]. Python is a versatile programming language. Bootstrap is a front-end framework for responsive web design. Pandas is a data manipulation library. SQLite3 is a lightweight, embedded relational database. Pickle is a Python module for object serialization, allowing data to be saved and loaded efficiently [2]. These technologies are often used together in web development, where Django handles the backend, Python supports application logic, Bootstrap ensures a responsive design, Pandas manages data, SQLite3 stores data, and Pickle aids in data serialization.

That's why we created "Quiz App," a new webapp designed to simplify learning. Think of it as a helpful friend who gives you random questions. Our goal is to make learning straightforward and fun. We observed that other coding websites can be intimidating for beginners, featuring difficult problems and intense competition. "Quiz App" stands out – it's free, user-friendly, and adapts to your skill level.

2.2 Literature Review

Platforms such as MCQ Hall, TestPortal, LeetCode, and CodeQuizzes may be daunting for newcomers. They lack personalized monitoring, with no admin overseeing individual users like a teacher does with students. The platforms host challenging problems, fostering intense competition that might discourage beginners. Many questions are advanced, mirroring those used by top companies like Amazon and Facebook. Unfortunately, some prioritize business over a conducive learning environment, even offering premium versions with paid features. This can hinder the learning journey for those seeking a supportive and educational coding experience. Some similar systems to our project for taking test especially regarding to coding and feedback on it are:

Test Portal

Test portal introduces AI-generated quizzes, tests and exams. We always look for new ways to make your online assessments easier, better and much more productive. That's why Test portal can now automatically generate quizzes, tests and exams based on our content.

LeetCode

LeetCode is an online platform for coding interview preparation. The service provides coding and algorithmic problems intended for users to practice coding. LeetCode has gained popularity among job seekers and coding enthusiasts as a resource for technical interviews and coding competitions. The primary goal of LeetCode is to help users enhance their coding skills through a vast collection of programming problems. These problems cover a wide range of topics, including algorithms, data structures, databases, and more. LeetCode provides a platform for users to practice solving these problems using various programming languages such as Python, Java, C++, and others.

Quizlet

Quizlet is a web tool and a mobile app that boosts students learning through several study tools that include flashcards and game-based quizzes. You can either design your own study sets from scratch or search for pre-made sets to customize and use in your teaching. Quizlet takes information and converts it into flashcards, quizzes, and games, so that users can study the same information in a variety of forms.

MCQ Hall

MCQ Hall is a platform for conducting MCQ Exams connecting students with institutes, book writers, book publishers and teachers [3]. Any creator can join MCQ Hall and make their online presence strong. Also, MCQ Hall provides a way to get connected with students remotely, conduct exams on a regular basis as Live or mock, prepare hassle free results with efficiency and accuracy, give live feedback to students analyzing their performance and many more.

SVM

Support Vector Machines (SVM) are a group of supervised learning algorithms, which can be used for classification or regression purposes. SVM is a machine learning algorithm commonly employed for classification tasks, and in this case, it aids in dynamically categorizing questions into different difficulty levels or other specified categories.

CHAPTER 3 : SYSTEM ANALYSIS

3.1 System Analysis

3.1.1 Requirements Analysis

Requirement analysis is the process of precisely identifying, defining, and documenting the various requirements that are related to a particular business objective. Requirements gathering help in clearly understanding the needs of the customer, defining the scope of the project, and assessing the timescales and resources required to complete it. There are two types of requirements which are as follows:

i. Functional Requirements

This requirement determines what the system should do. In order to make the application functional, we require the following:

User Authentication and Authorization:

The Quiz App ensures secure user registration and login, allowing users to create accounts with personal details. Different user roles, including administrators and regular users, are implemented to ensure appropriate access levels.

Admin Dashboard:

Admins have centralized control over question management, test creation, and result analysis. They can add, edit, or delete questions, create customized tests, and access detailed statistics on total questions and exam performance.

User Dashboard:

The user dashboard simplifies test-taking, offering multiple tests and immediate results. It provides a comprehensive view of test history, available quizzes, and performance insights over time.

Question Pool:

Admins manage the question pool, categorizing questions based on difficulty levels. The Support Vector Machine (SVM) algorithm dynamically categorizes questions, ensuring a personalized learning experience.

System Optimization:

Randomization ensures fairness by presenting questions in a classification order. Admins can refresh question categories using SVM-based processes, and the user interface is designed for intuitive operation.

Use-Case-Diagram of Quiz App:

The use case diagram for the Quiz App outlines the interactions between two main actors: the Administrator (Admin) and the User. Admins can manage the question pool by adding, editing, or deleting questions, create customized tests, and access detailed result reports. On the other hand, users can take tests assigned by admins, receive result, and view their performance history through a personalized dashboard.

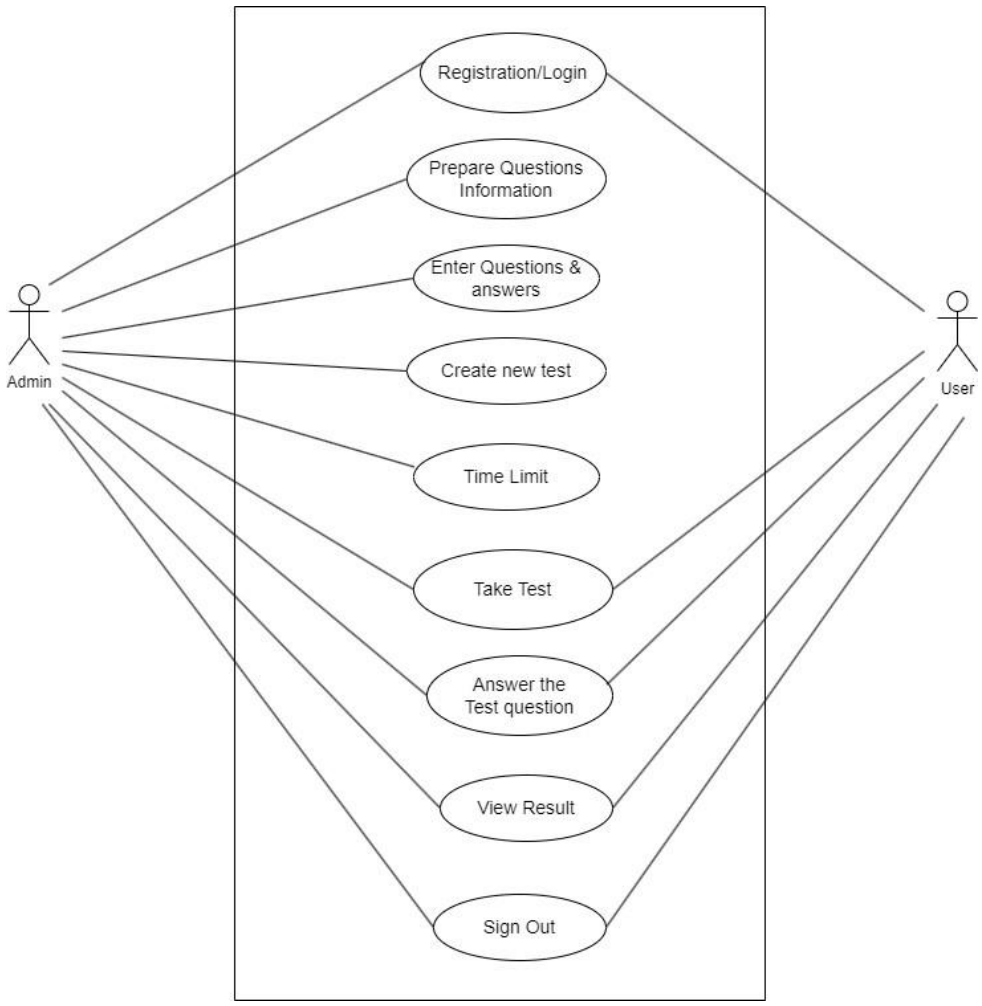


Figure 3.1 Use Case Diagram of Quiz App

ii. Non- Functional Requirements

Non-functional requirements define the aspects of a system that are not directly related to its specific behaviors or features but are crucial for its overall performance, usability, and reliability. Like:

Performance:

The app should respond to user interactions within a specified time frame (e.g., load a quiz in 2 seconds) and should be able to support a large number of simultaneous users without significant performance degradation.

Scalability:

The system should be able to handle an increasing number of users and quizzes without a significant increase in response time.

Reliability:

The app should have a high level of availability, with minimal downtime for maintenance, and must ensure data integrity and reliability in storing and retrieving user and quiz information.

Availability:

The app must specify the percentage of time the system should be available (e.g., 99.9% uptime).

Security:

The quiz app implements secure user authentication and authorization mechanisms and protects against common security threats, such as SQL injection and cross-site scripting.

Compatibility:

The quiz app ensures compatibility with a range of devices, browsers, and operating systems and confirms compatibility with various screen sizes and resolutions, especially for mobile devices.

3.1.2 Feasibility Analysis

The feasibility study helps to determine the benefits of the proposed system in society and organization. It also determines if the system can be built successfully with cost, time, and effort. The study was conducted by analyzing the collected requirements.

i. Technical Feasibility

The technical feasibility of a quiz app relies on a comprehensive evaluation covering technology choices, infrastructure needs, security, scalability, and performance. Important considerations for a quiz app include connecting with other systems, following industry rules, and ensuring it's easy to use and reliable through testing and backup plans. These factors are crucial for ensuring the app works well and meets standards.

ii. Operational Feasibility

Operational feasibility for a Quiz App revolves around practicality and user acceptance. It requires evaluating the ease of integration into existing processes, assessing user-friendliness, and considering the impact on daily operations.

iii. Economic Feasibility

The technological feasibility of a quiz app hinges on a thorough examination of the technology stack's appropriateness, scalability, integration capabilities, security measures, and performance.

iv. Schedule Feasibility

The schedule feasibility shows the time taken to develop the software This software has completed within three months. The project is divided into the tasks and milestones.

The comprehensive feasibility analysis encompasses several critical dimensions, including Technical, Operational, Economic, Schedule, Legal, Cultural, Political, Social, and Ethical feasibility. But for a quiz app project, feasibility is analyzed in four key areas: Technical, Operational, Economic, and Schedule.

3.1.3 Analysis

During the development of the Quiz App, we employed an object-oriented approach, which is a well-established methodology for creating web applications that are organized, adaptable, and easy to maintain.

In this approach, we emphasize the use of "objects," which are instances of classes. These objects not only store data but also have built-in behaviors. In the context of the Quiz App, we structured the application around key entities like users, quizzes, questions, and answers. Each of these entities is represented with its own set of attributes and methods.

Data Modelling:

Data modeling is the process of diagramming data flows. When creating a new or alternate database structure, the designer starts with a diagram of how data will flow into and out of the database. The data model will normally consist of entity types, attributes, relationships, integrity rules, and the definitions of those objects. This is then used as the start point for interface or database design.

Sequence Diagram:

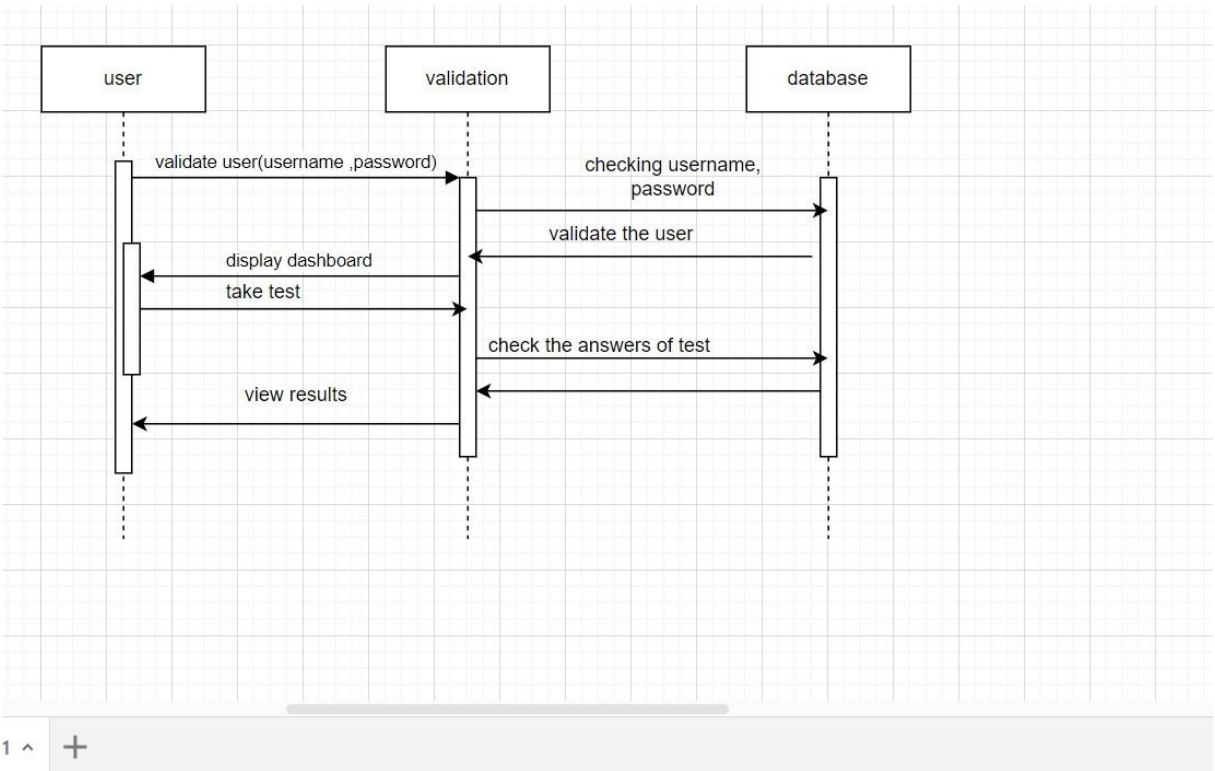


Figure 3.2 Sequence Diagram of Quiz App

ER Diagram:

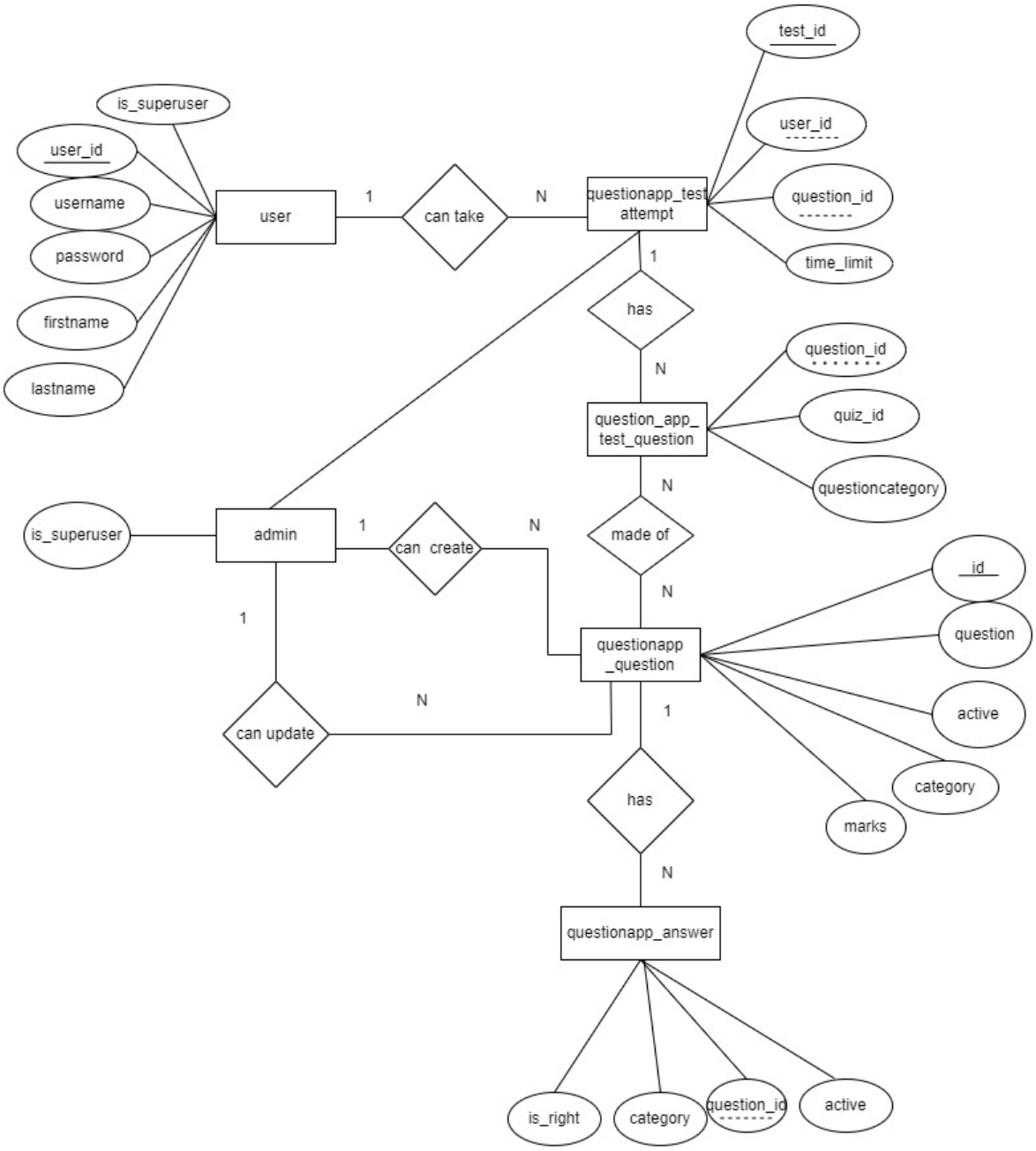


Figure 3.3 ER Diagram of Quiz App

Class Diagram:

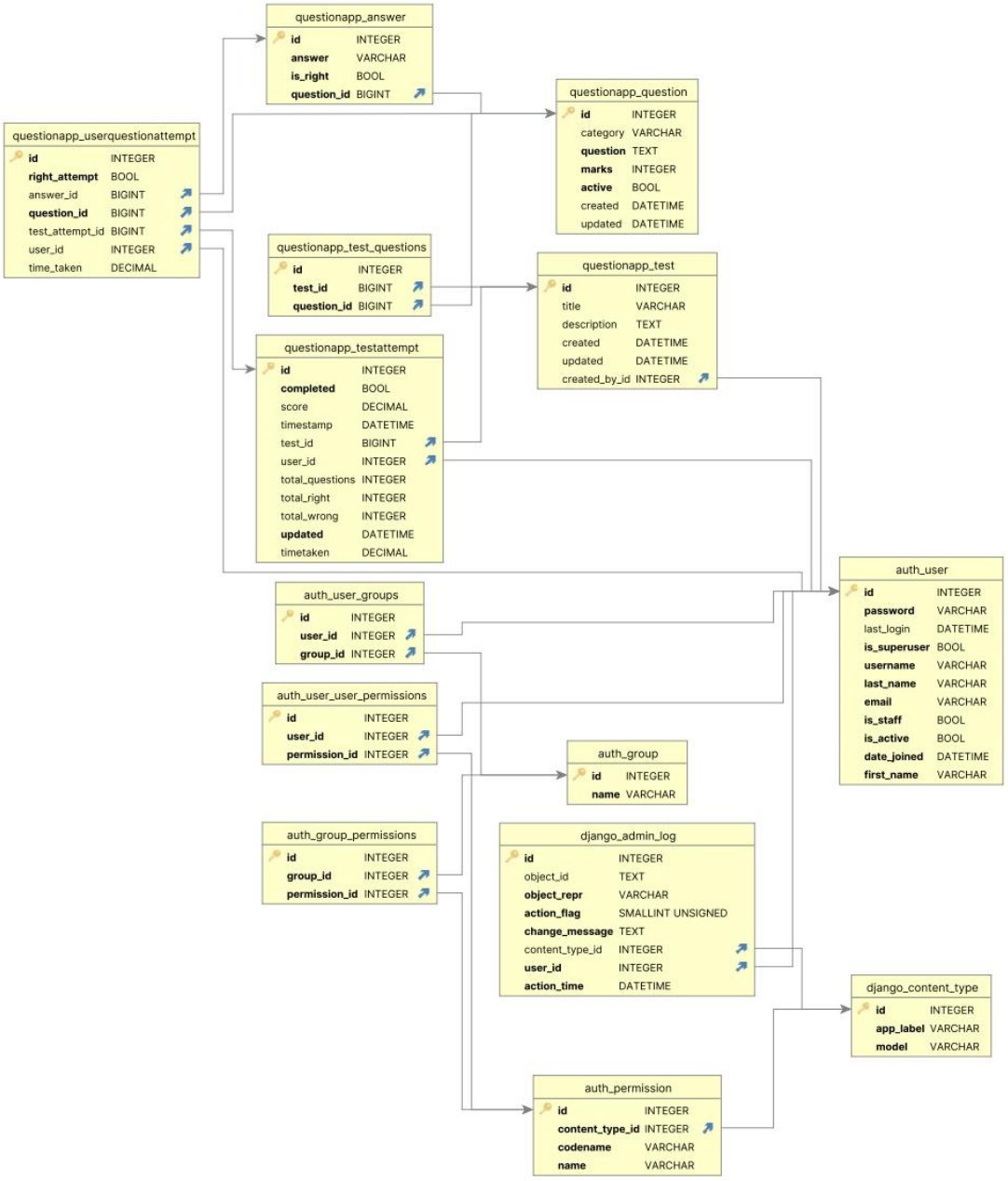


Figure 3.4 Class Diagram of Quiz App

Object Diagram of Quiz App:

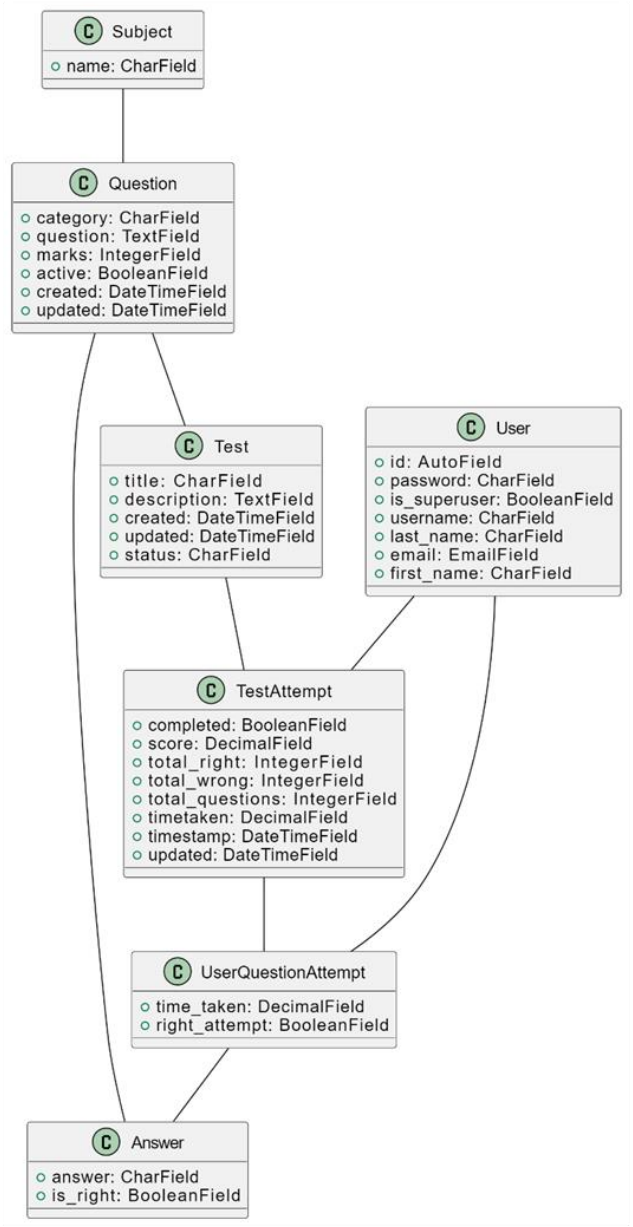


Figure 3.5 Object Diagram of Quiz App

Activity Diagram of Quiz App:

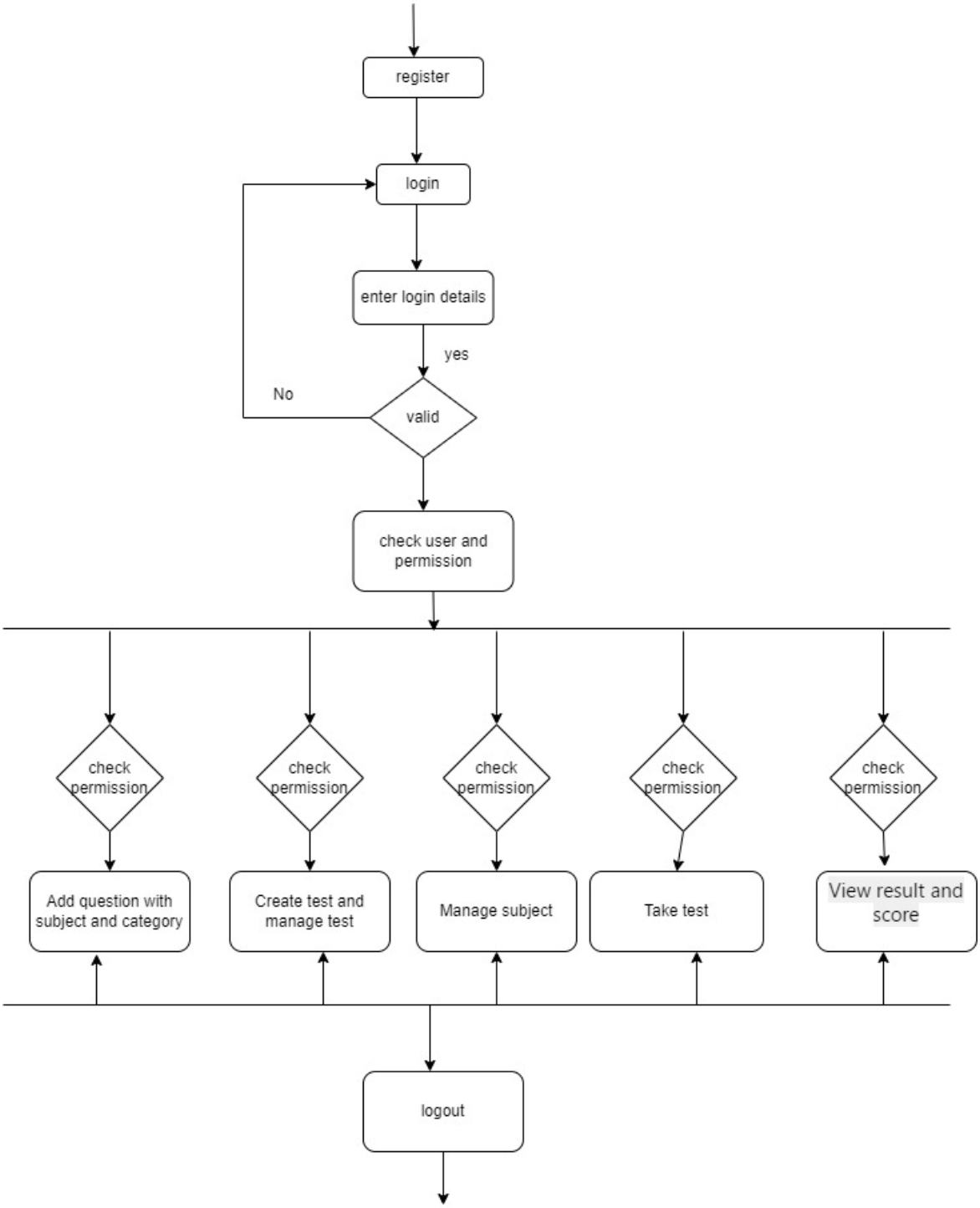


Figure 3.6 Activity Diagram of Quiz App

CHAPTER 4 : SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. Generally, this topic deals with the module, database design, user interface design and the program design.

4.1 System Architecture and Overview

The required system that makes use of the internet and computer peripherals is developed. This system is supported by any different devices that can access the internet.

4.2 System Diagrams

A system diagram is a visual representation of a system, its components, and the relationships between those components. It is a graphical model that shows how different parts of a system work together to achieve a specific function or goal. Typically, a system diagram consists of various symbols and shapes that represent the different components of the system. These components are connected by lines or arrows that show the flow of data, energy, or other resources between them. System diagrams can be used for a variety of purposes, such as analysis, design, and communication. They are particularly useful for understanding complex systems and identifying areas for improvement or optimization.

4.2.1 System flow chart of Quiz App

A system flowchart is a type of diagram that shows the flow of inputs, processes, and outputs within a system. It is a graphical representation of a system's logic that illustrates how data and information move through the various stages of a process or operation. System flowcharts are useful for a variety of purposes, such as documenting and analyzing existing processes, designing new processes, and communicating system processes to stakeholders.

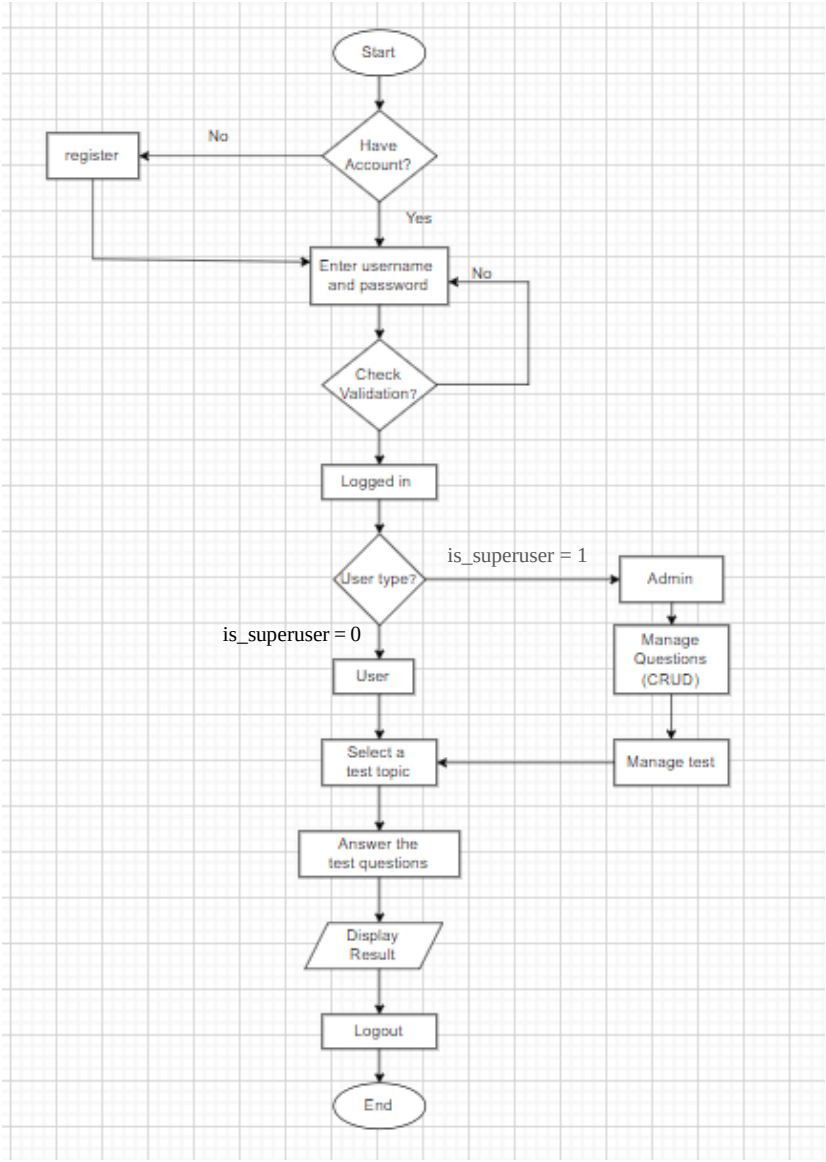


Figure 4.1 Flow Chart of Quiz App

4.3 Algorithm Details

Support Vector machine

In the context of the Quiz App, the Support Vector Machine (SVM) algorithm is used for categorizing questions in the question pool based on user performance. SVM is a machine learning algorithm commonly employed for classification tasks, and in this case, it aids in dynamically categorizing questions into different difficulty levels or other specified categories.

SVM algorithm is applied in our project “Quiz App” in following ways:

Dynamic Categorization:

The SVM algorithm analyzes user performance data, such as the correctness of responses, completion time, or other relevant metrics.

Category Assignment:

Based on the analysis, questions are dynamically assigned to different categories, such as easy, hard, or any specified category. This dynamic categorization ensures a personalized and adaptive learning experience for each user.

Refresh Mechanism:

The system may have a feature, possibly triggered by an admin, to refresh the question categories using the SVM-based categorization process. This ensures that the question pool remains relevant and adapts to evolving user performance patterns.

Quiz App is utilizing SVM with three dimensions—number of attempts, number of correct attempts, and time taken—it suggests a more advanced approach to dynamically categorizing and adapting the question pool based on user performance.

By incorporating SVM into the Quiz App, the aim is to enhance the system's adaptability and provide users with quizzes that are tailored to their individual learning levels and preferences.

For a three-dimensional feature space, the decision boundary (hyperplane) is a plane. The formula for a linear SVM decision function in a three-dimensional space is:

$f(x)=w1 \cdot x1+w2 \cdot x2+w3 \cdot x3+b$

Here:

- $f(x)$ is the decision function.
- $x1$,$x2$,$x3$ are the input features.
- $w1$,$w2$,$w3$ are the weights assigned to the features.
- b is the bias term.

For classification, if $f(x)$ is greater than zero, the data point is classified into one class, and if it's less than zero, the point is classified into the other class. The vector $= [\ 1\ ,\ 2\ ,\ 3\ ]$ $w=[w1\ ,w2\ ,w3\ ]$ is perpendicular to the decision boundary. The training process involves finding the optimal values for w and b based on the training data and the chosen SVM algorithm.

CHAPTER 5 : IMPLEMENTATION AND TESTING

5.1 Implementation

In this phase, Quiz App was being implemented by using various implementing tools as per the design of a system. Here real system coding was started and executed to make system ready for testing phase.

5.1.1 Tools Used

a. Front End

A "front-end" application is one that application users interact with directly. In simple words. While creating a front end, different components relating to the software development were used. They are listed below:

- Bootstrap v5.1

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. We are using bootstrap for client-side design like: login page, dashboard etc.

- HTML 5/CSS 3:

HTML and CSS were used for editing Bootstrap to make more attractive web page.

- Vue.js

b. Back End:

A "back-end" application or program serves indirectly in support of the frontend services, usually by being closer to the required resource or having the capability to communicate with the required resource. Back end possess database and administrator manages it. The following parts are contained in the back end of our system:

- Django version 4.2.8 (Python3 Web Development Framework)

Django is a free and open-source, Python-based web framework that runs on a web server. It follows the model–template–views architectural pattern.

Integrating Pandas, NumPy, Matplotlib, and Scikit-learn in a Django application enhances its capabilities for data manipulation, numerical operations, visualization, and machine learning tasks. Pandas facilitates efficient data handling, NumPy supports numerical computations, Matplotlib enables data visualization, and Scikit-learn empowers the implementation of machine learning models within the Django environment.

c. Database Platforms:

The system is developed using DB sqlite3. Using db.sqlite3 offers simplicity, portability, and low overhead, making it suitable for smaller projects like Quiz App. It's included in Python, requiring no additional installations.

5.1.2 Implementation Details of Modules

In Django, the term "controllers" from the traditional Model-View-Controller (MVC) pattern is equivalent to what Django refers to as "views." In this framework, the models represent the "M," which are the classes that define the structure and behavior of the data. The "V," or views in the traditional MVC sense, are realized through Django's templates or HTML files, which handle the presentation layer.

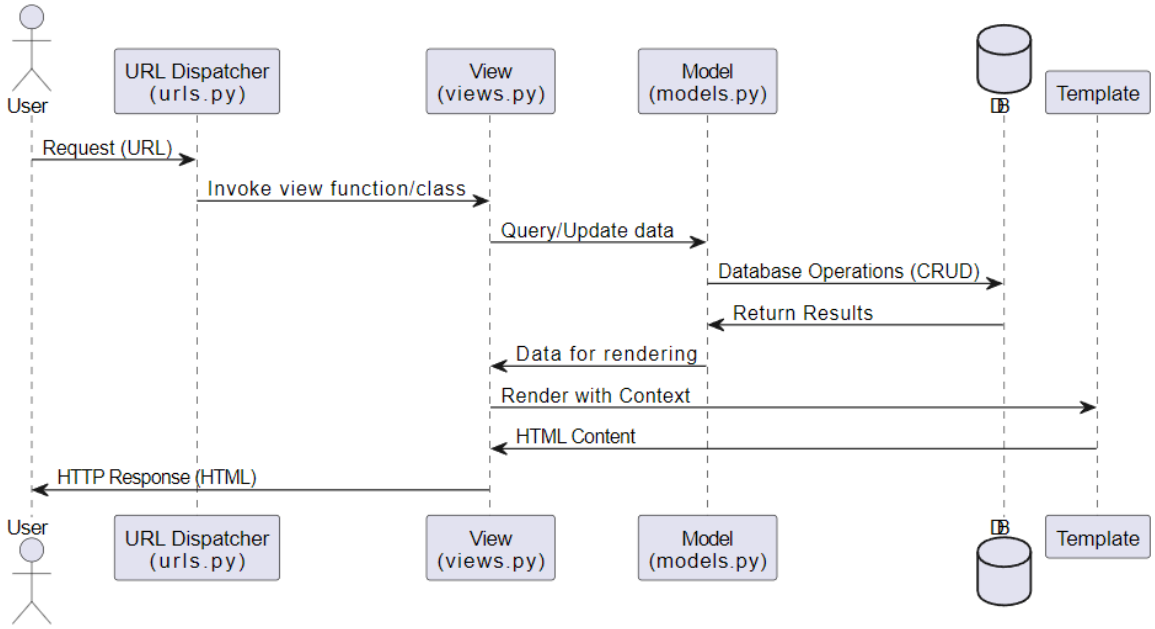


Figure 5.1 Working Mechanism of Django framework

Some Controllers of Quiz App:

User Authentication Controllers:

The user authentication is managed by three main controllers. The loginPage controller oversees the user login process, authenticating credentials and redirecting successful logins to the dashboard, which is handled by the home controller. The signupPage controller is responsible for user registration, handling data validation and account creation, and subsequently redirects new users to the loginPage for initial login. The logoutPage controller takes care of logging out users, redirecting them back to the loginPage post-

logout. Apps related to user authentication and registration, such as 'django.contrib.auth', 'django.contrib.sessions', and 'dashboard', are essential for these processes. Controllers within these apps handle user authentication and validation. The AUTH_PASSWORD_VALIDATORS setting defines password validation rules.

```python
def loginPage(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username,password=password)
        if user is not None:
            login(request,user)
            return redirect('dashboard:home')
        else:
            messages.info(request,"Username or Password Incorrect")

    context ={

    }
    return render(request,'login.html',context)
```

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dashboard',
    'questionapp',
    'django_extensions',
    'crispy_forms',
    'crispy_tailwind',
    'rest_framework',

]
```

Dashboard and Test Management Controllers:

The home controller acts as the central dashboard, offering a comprehensive view of various statistics and management tools for tests and questions. The testPage controller is designed to display the details of a specific test, providing insights into individual test components. The list_test controller is responsible for listing all available tests, along with functionalities to add new tests or modify existing ones. The add_test controller facilitates the addition of new tests, ensuring a seamless redirection back to the test list once the submission is complete. Lastly, the edit_test controller provides the capability to make alterations to a specific test, followed by a redirect back to the overall test list for continued management and oversight.

Question Management Controllers:

In Quiz App application, question management is streamlined through specific controllers. The list_questions controller displays all existing questions and offers the functionality to add new ones, with a direct link to question_edit for making edits to individual questions. The question_add controller is dedicated to incorporating new questions into a specific test, enhancing the test's content. Meanwhile, question_edit is exclusively focused on providing a user-friendly interface for editing the details of a particular question, ensuring that the questions remain relevant and up-to-date.

Subject Management and Option (Answer) Management Controllers:

the management of answer options and subjects is efficiently handled by dedicated controllers. The manage_option controller serves as the central hub for managing a question's options, offering functionalities to edit, delete, and add new options. The edit_option controller allows for the modification of individual options, subsequently

redirecting back to the main option management page. In contrast, the delete_option controller focuses on removing specific options, guiding users back to the option management interface upon completion. The add_option_by_question controller is specialized in adding new options to a given question. Complementing these, the subject management aspect of your application is streamlined through the create_subject controller, which facilitates the creation of new subjects and leads to a comprehensive subject list display handled by the list_subject controller.

```
def create_subject(request):
    form = SubjectForm()
    if request.method == "POST":
        form = SubjectForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('dashboard:list_subject')

    context = {
        'form':form
    }
    return render(request,'dashboard/subject/addsubject.html',context)

def list_subject(request):
    subjects = Subject.objects.all()

    context  = {
        'subjects':subjects
    }

    return render(request,'dashboard/subject/list_subject.html',context)
```

Test Taking, Submission and Test Attempts Viewing Controllers:

the process of taking tests and managing test attempts is streamlined through a set of specialized controllers. The test_index controller lists all available tests for users, guiding them to initiate or continue tests. The GetQuestionView API controller is pivotal in providing users with the next question in an ongoing test, and it efficiently handles test completion, scoring, summarize results. Complementarily, the QuestionSubmissionView API controller is responsible for managing the submission of answers during a test, ensuring that user responses are correctly recorded. To offer users insights into their testing history, the view_test_attempts and view_test_attempts_by_test controllers display a comprehensive overview of the user's test attempts, including detailed views for specific tests.

```python
def view_test_questions(request,test_id):
    test = Test.objects.get(id=test_id)
    questions = test.questions.all().select_related('subject')
    context = {
        'test':test,
        'questions':questions
    }
    return render(request,'dashboard/view_test_questions.html',context)
```

Statistics, CSV Upload, and other Controllers:

Controllers for statistics and CSV uploads play a crucial role in enhancing the functionality of your platform. The questionstatics controller is adept at updating question categories, utilizing data derived from user attempts and performance to refine the categorization process. For efficient data management, the UploadCsv API controller provides the capability to upload questions in bulk via CSV files, streamlining the question-adding process. Additionally, the AddQuestionsToTestAPIView API controller serves an essential function by allowing the addition of questions to specific tests, thereby customizing and enriching test content. The view_test_questions controller offers a detailed view of the questions associated with a particular test, ensuring transparency and accessibility of test content. Finally, the add_option controller facilitates the addition of new options to existing questions, enabling dynamic question modification and enhancement.

```python
@login_required()
def view_test_attempts(request):
    test_attempts = TestAttempt.objects.filter(user=request.user).order_by('-id')
    context = {
        'test_attempts':test_attempts
    }
    return render(request,'dashboard/test_attempts.html',context)
```

```python
@login_required()
def view_test_attempts_by_test(request,test_id):
    test_attempts = TestAttempt.objects.filter(user= request.user , test__id = test_id).order_by('-id')
    test = Test.objects.get(id=test_id)
    print(test_attempts)
    context = {
        'test':test,
        'test_attempts':test_attempts
    }
    return render(request,'dashboard/test_attempts_index.html',context)
```

Implementation of an Algorithm:

The MCQ.ipynb notebook is a Jupyter Notebook file that demonstrates the entire process of data generation, analysis, machine learning model training, evaluation, and model persistence using SVM for classifying MCQs into two categories: "Easy" and "Hard." It performs following actions:

Data Generation:

generate_mcq_data(num_samples): Generates synthetic data for multiple-choice questions (MCQs) with two classes: "Easy" and "Hard." It creates random values for attributes like "AverageTimeTaken," "NumAttempts," and "NumCorrectAttempts" for both classes.

Data Analysis:

df = pd.read_csv(...): Reads the generated data from a CSV file into a Pandas DataFrame and displays the first 10 rows.

print_dataframe_info(df): Prints information about the DataFrame, including its shape, size, and count of non-null values.

get_label_counts(df): Calculates and returns the count of each unique value in the 'Label' column, showing the distribution of classes.

Data Filtering:

filter_data(df): Filters the data to create two subsets, 'easy_df' and 'hard_df,' each containing 100 samples from the respective class.

Data Preprocessing:

process_data(df): Prepares the data for training by dropping the 'Label' column from the DataFrame, resulting in feature data (X).

Data Label Extraction:

get_label(df): Extracts the 'Label' column as the target variable (y).

Train-Test Split:

split_data(X, y): Splits the data into training and testing sets (X_train, X_test, y_train, y_test) using a 80-20 ratio.

```
from sklearn.model_selection import train_test_split

def split_data(X, y):
    """
    Split the data into training and testing sets.

    Parameters:
    X (array-like): The input features.
    y (array-like): The target variable.

    Returns:
    X_train (array-like): The training set of input features.
    X_test (array-like): The testing set of input features.
    y_train (array-like): The training set of target variable.
    y_test (array-like): The testing set of target variable.
    """
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
    return X_train, X_test, y_train, y_test
```

Model Training:

classfier = svm.SVC(...): Initializes an SVM classifier with a linear kernel, gamma set to 'auto,' and regularization parameter (C) set to 2.

classfier.fit(X_train, y_train): Trains the SVM classifier on the training data.

Model Evaluation:

y_predict = classfier.predict(X_test): Predicts class labels for the test data. print_classification_report(y_test, y_predict): Prints a classification report showing precision, recall, F1-score, and support for each class, as well as overall accuracy.

Model Prediction:

ModelPredict(your_data): Demonstrates how to use the trained model for prediction by providing new data ('your_data') and printing the predicted labels.

```
def ModelPredict(your_data):
    """
    Predicts the class labels for the given data using the classifier.

    Args:
        your_data (dict): The input data for prediction. It should be a dictionary with the following keys:
            - 'AverageTimeTaken': A list of integers representing the average time taken for each data point.
            - 'NumAttempts': A list of integers representing the number of attempts for each data point.
            - 'NumCorrectAttempts': A list of integers representing the number of correct attempts for each data point.

    Returns:
        None
    """
    df_test = pd.DataFrame(your_data)
    predictions = classfier.predict(df_test)
    print("Predictions:", predictions)
```

Model Saving:

model_filename = 'SVMmodel.pkl': Defines a filename for saving the trained SVM model. with open(model_filename, 'wb') as model_file: Saves the trained classifier using the pickle library.

```
import pickle

model_filename = 'SVMmodel.pkl'
with open(model_filename, 'wb') as model_file:
    pickle.dump(classfier, model_file)

"""
This code saves the trained classifier object to a file using pickle.

Parameters:
- model_filename (str): The name of the file to save the model to.

Returns:
None
"""
```

Model Loading (Optional):

load_model(model_filename): Defines a function to load a pre-trained SVM model from a file.

```
def load_model(model_filename):
    """
    Load a pre-trained model from a file.

    Parameters:
    model_filename (str): The filename of the model file to load.

    Returns:
    object: The loaded model object.
    """
    with open(model_filename, 'rb') as model_file:
        loaded_model = pickle.load(model_file)

    return loaded_model

# to load the existing modle.
with open (model_filename,'rb') as model_file:
    LoadModel = pickle.load(model_file)
```

5.2 Testing

Software testing is the process of evaluating a software item to detect differences between given input and expected output. It also assesses the feature of a software item. Testing assesses the quality of the product.

Various types of testing were done for verification and validation of Quiz App.

Testing was divided into various parts:

5.2.1 Unit Testing

The Unit testing part of a testing methodology is the testing of individual software modules or components that make up an application or system. As work was divided and after coding it was parallelly tested and after getting bug it was made bug free.

Table 5. 1 Test Case for Login

S.N	Test case Id	Test description	Expected Result	Actual Result	Pass/fail
1	TC-01	Open browser and enter URL	Dashboard page should be displayed with user/admin login at navbar.	Dashboard displayed with user/admin login.	pass
2	TC-02	Enter valid data in login form like username and password.	It should redirect to home page.	Home page displayed	pass

3	TC-03	Enter valid data in username and invalid in password.	Error message as Error: Username or Password Incorrect	Error message as Error: Username or Password Incorrect	pass
4	TC-04	Enter valid data for username and password for admin	Home page is displayed and admin statistics is shown.	Displays dashboard with admin statistics.	pass
5	TC-05	Enter invalid username and password	Error: Username or Password Incorrect	Error: Username or Password Incorrect	pass

5.2.2 Integration Testing

The Integration testing part of a testing methodology is the testing of the different modules/components that have been successfully unit tested when integrated together to perform specific tasks and activities. The test is often done on both the interfaces between the components and the larger structure being constructed, if its quality property cannot be assessed from its components. After integrating the requirements, we tested it, it was fine and satisfactory.

Table 5. 2 Test case for Test taking and Availability

S.N	Test case Id	Test description	Expected Result	Actual Result	Pass/fail
1	TC-01	After login, dynamic dashboard should be displayed with various statistics.	Dynamic and up to date dashboard will be displayed.	Dynamic and up to date dashboard is displayed with all the statistics.	pass
2	TC-02	Upon taking a test, it should display the result and record it and display result.	Completing test will show results and update result page.	Completing test , it shows results and update result page.	pass
3	TC-03	Creating a test, the test will be ready and available to all.(live)	Creating a test by admin, it will be available to test for all the user.	Test is available to all the user including admin.	pass

5.2.3 System Testing:

The system testing part of a testing methodology involves testing the entire system for errors and bugs. This test was carried out by interfacing the hardware and software components of the entire system, and then testing it as a whole. This testing was listed under the black-box testing method, where the software was checked for user expected working conditions as well as potential exception and edge conditions.

5.3 Result Analysis:

After all these testing methods and plans Quiz App was working successfully and as per our requirements and functionality. Here system was completely verified and validated as per our functional and nonfunctional requirements. Admin was successfully able to create test on various subject fields with multiple question and set time limit for each test. Users were able to login in by creating an account and give the test. Upon taking the test, the questions were randomly generated from the question pool of a test to maintain fairness in the examination. The test was successfully performed under the given time. After providing the test, the percentage and others details were immediately shown to the user and recorded and is displayed in result and score tab. Admins were able to see those results and can determine knowledge level of the user. The dynamic dashboard was also updated and statistics was displayed. Also, users were able to successfully logout from the application.
The MCQ.ipynb notebook uses a dataset that is read from a CSV file. The result from the notebook after training the SVM classifier and evaluating its performance:
This is a classification report that provides various metrics for both the "Easy" and "Hard" classes, as well as overall accuracy:

- Precision: Indicates the fraction of true positives out of all predicted positives.
- Recall: Indicates the fraction of true positives out of all actual positives.
- F1-score: Represents the harmonic mean of precision and recall, providing a balance between the two metrics.
- Support: The number of samples in each class.
- Accuracy: Overall classification accuracy, which is the fraction of correctly classified samples.

```
                precision    recall  f1-score   support

          Easy       1.00      1.00      1.00        42
          Hard       1.00      1.00      1.00        38

      accuracy                           1.00        80
     macro avg       1.00      1.00      1.00        80
  weighted avg       1.00      1.00      1.00        80
```

Above given code is an example of the result from the notebook after training the SVM classifier and evaluating its performance.

```python
data = {'AverageTimeTaken': [25, 30, 40],
        'NumAttempts': [5, 8, 12],
        'NumCorrectAttempts': [4, 7, 10]}

ModelPredict(data)
```

Python

```
Predictions: ['Easy' 'Easy' 'Easy']
```

```python
import pickle

model_filename = 'SVMmodel.pkl'
with open(model_filename, 'wb') as model_file:
    pickle.dump(classfier, model_file)
```

Above given code, saves the trained classifier object to a file using pickle after the prediction.

CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion

The development and testing phases of the Quiz App have been successfully completed, resulting in a robust and functional educational platform. Through the utilization of Django, Bootstrap, JavaScript, and smart algorithms, the Quiz App offers a user-friendly interface for both administrators and users, streamlining quiz creation, management, and participation processes.

In summary, the Quiz App is a groundbreaking tool that changes how we learn. With modern technology and a smart approach, it makes learning fair, easy, and enjoyable. The system adapts to each user, categorizes questions smartly, and ensures a positive experience. The study shows it's technically and economically sound. The user-friendly design and adaptive learning create a fun environment. The Quiz App is a fresh, complete, and user-friendly way to learn – making education not just simple but also fun.

6.2 Future Recommendation

Looking forward, the Quiz App has significant potential for improvement, especially in enhancing its usability and effectiveness for online education. One key enhancement could be the implementation of an algorithm for subject categorization. This would simplify the process of adding questions by allowing users to contribute without worrying about assigning them to specific categories. By organizing questions automatically, the app becomes more versatile and accessible to a wider range of students.

Expanding the variety of subjects available on the app would also be beneficial. By including a broader range of topics such as mathematics, literature, science, and history, the app can cater to the needs of a more diverse user base, ensuring its relevance across various educational fields.

Streamlining the creation of superuser accounts is another area for improvement. Simplifying the process through a user-friendly interface or smoother authentication procedures can make it easier for administrators to manage access and delegate responsibilities within the app.

Integrating gamification elements and optimizing the app for mobile devices can significantly enhance the learning experience. By adding interactive games and ensuring

compatibility with smartphones and tablets, the app becomes more engaging and accessible, allowing users to learn on the go.

Additionally, implementing personalized feedback and study material suggestions can greatly benefit learners. By analyzing test results and user preferences, the app can offer tailored feedback and recommend additional study materials, enhancing comprehension and retention for individual users.

Overall, by incorporating these simpler yet impactful enhancements, the Quiz App can become a valuable tool for online education, providing a user-friendly platform that meets the diverse needs of students and educators alike.

REFERENCES

[1] I. Sommerville, Software Engineering, Eighth Edition, Essex, England: Pearson Education Limited, 2007.

[2] W. S. Vincent, "Django for Beginners," in *Django for Beginners: Build websites with Python & Django*, USA, Independently Published, 2018-2020, p. 308.

BIBLIOGRAPHY

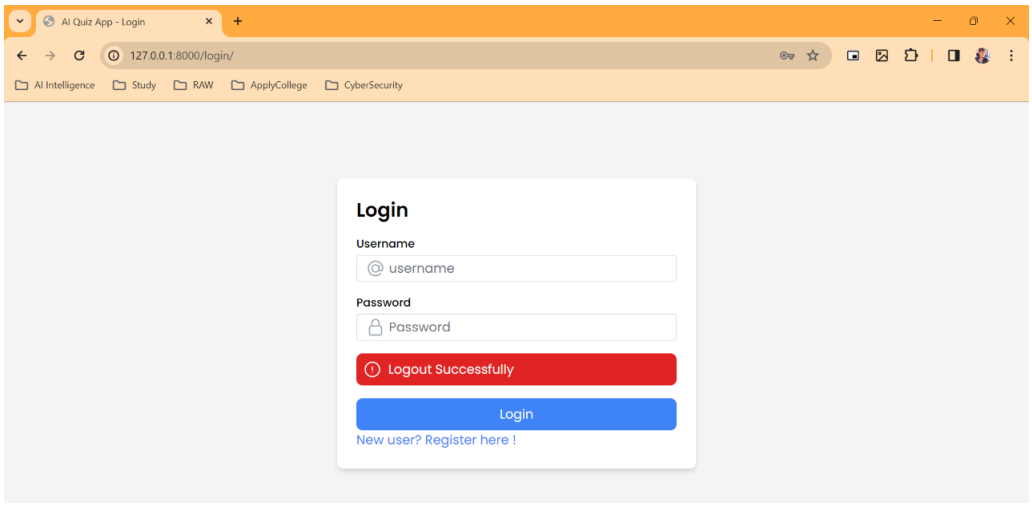
[1] McqHall, "McqHall," DalloTech Pvt. Ltd, [Online]. Available: https://mcqhall.com/. [Accessed 22 10 2023].

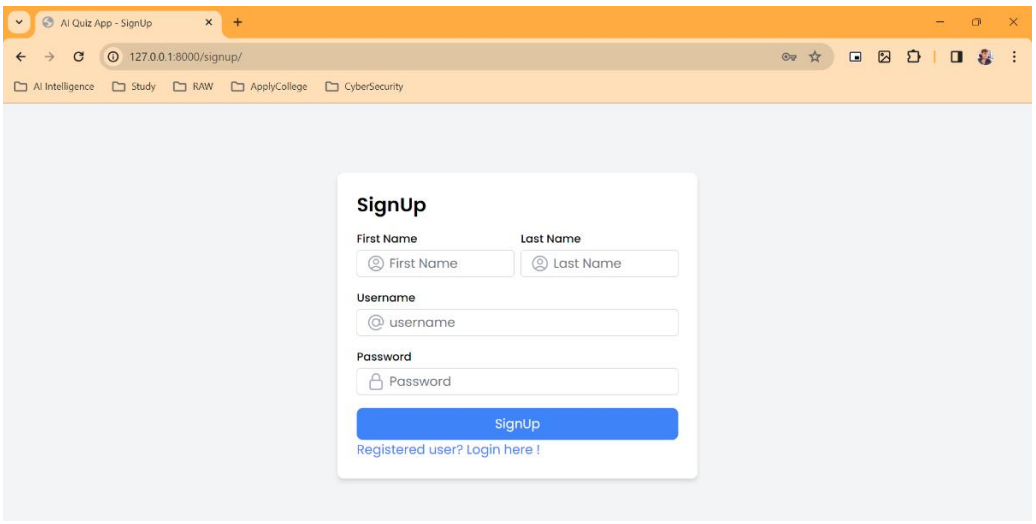
[2] W3Schools, "W3Schools," 1993. [Online]. Available: https://www.w3schools.com/. [Accessed 11 11 2023].

APPENDICES

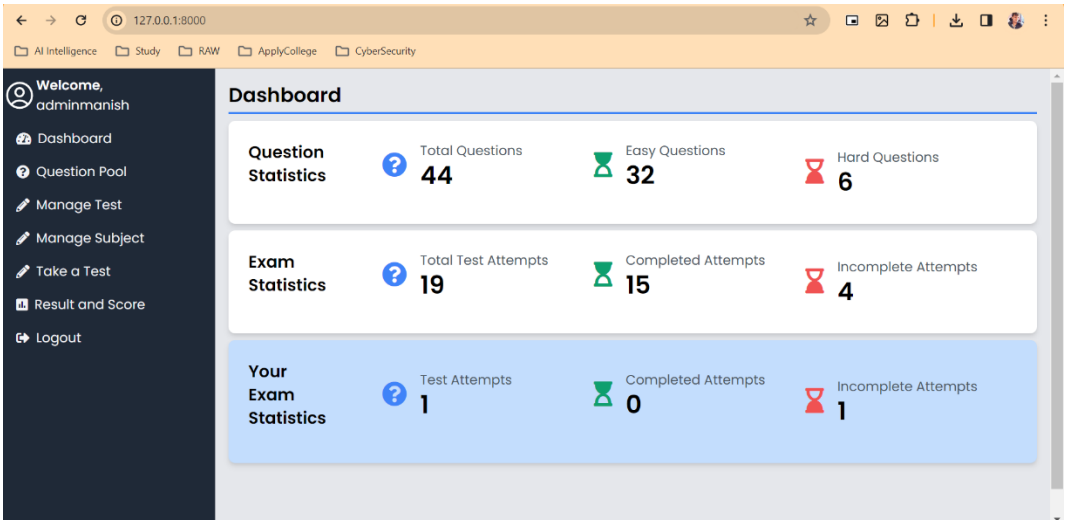
Screenshots:

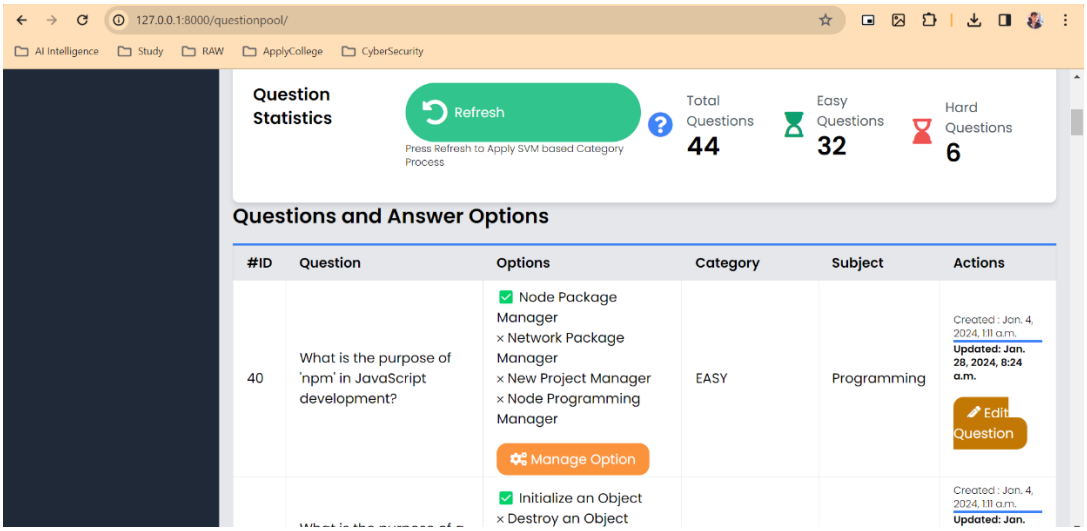
Login/Signup Form

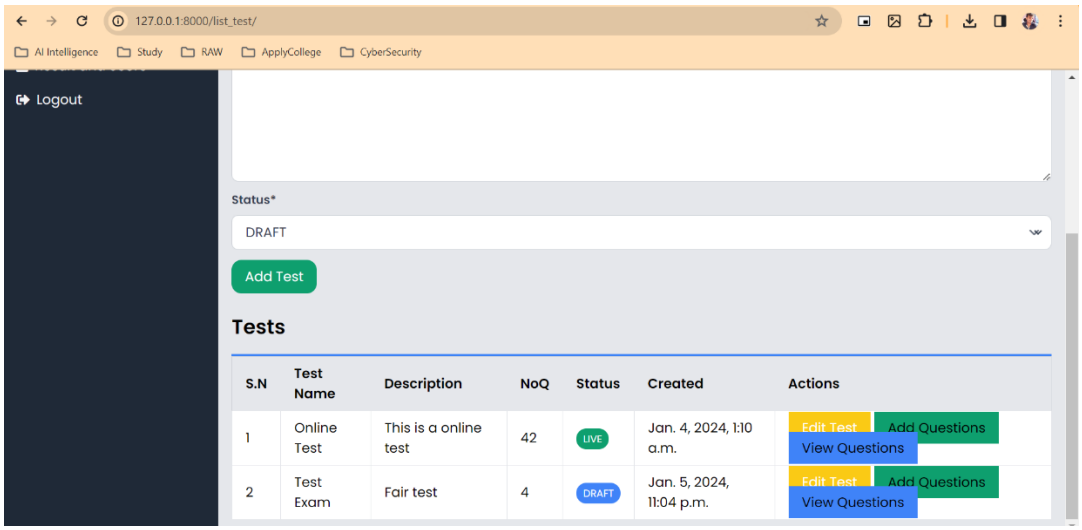




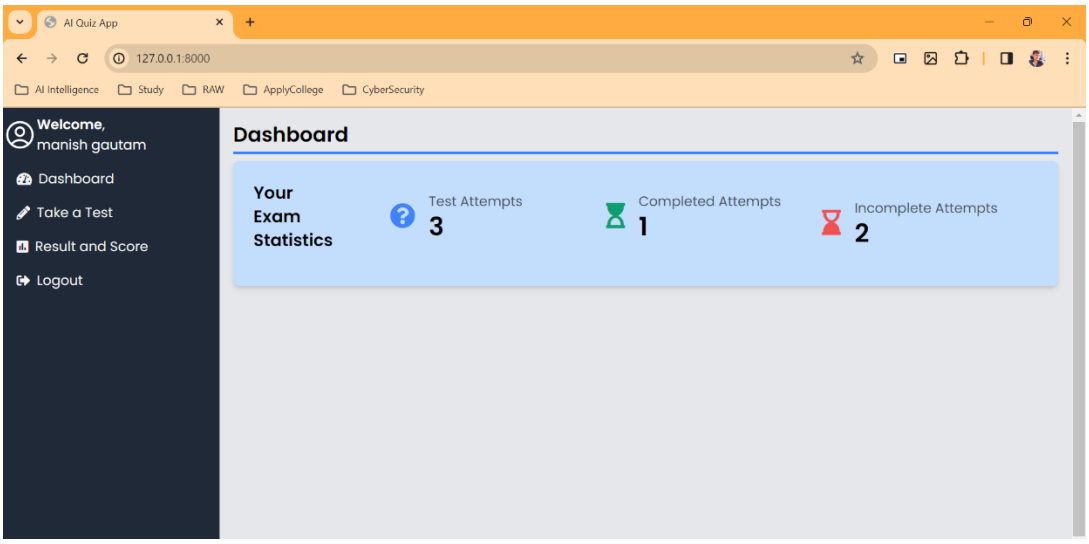
Admin Panel:

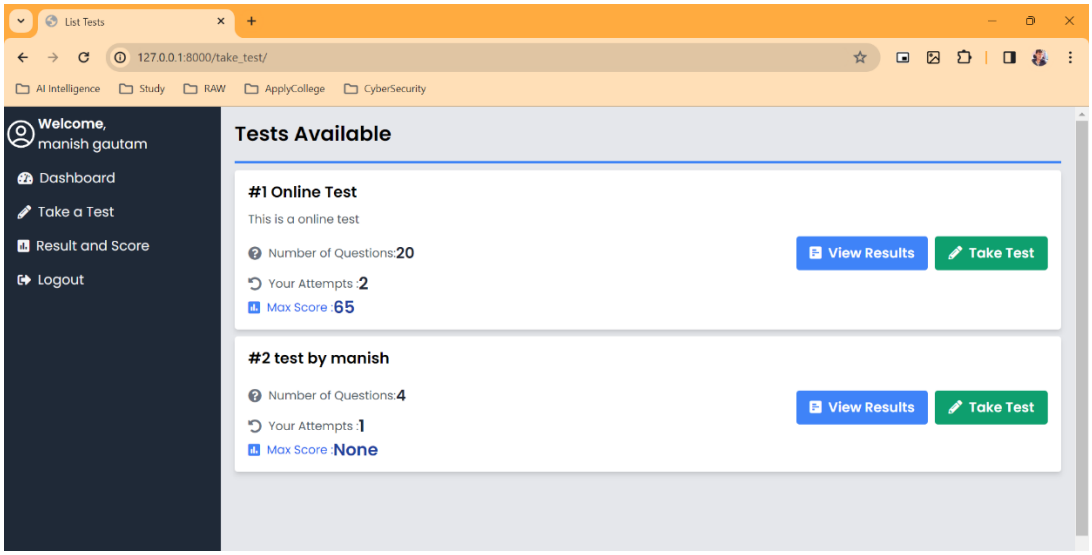


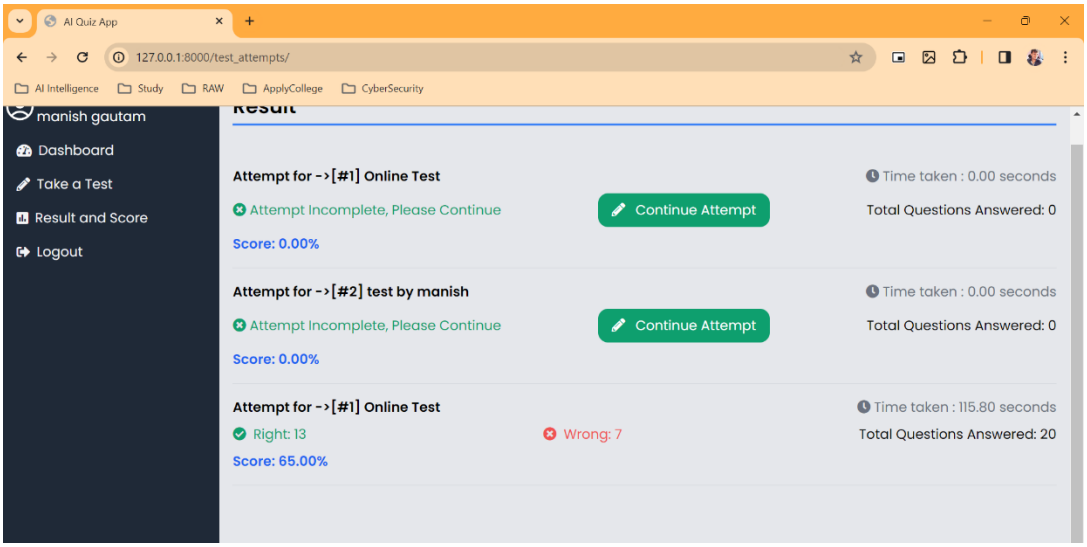

127.0.0.1:8000/questionpool/
AI Intelligence
Study
RAW
ApplyCollege
CyberSecurity
Question Statistics
Refresh
Press Refresh to Apply SVM based Category Process
Total Questions
44
Easy Questions
32
Hard Questions
6
Questions and Answer Options
#ID
Question
Options
Category
Subject
Actions
40
What is the purpose of 'npm' in JavaScript development?
Node Package Manager
× Network Package Manager
× New Project Manager
× Node Programming Manager
Manage Option
EASY
Programming
Created : Jan. 4, 2024, 1:11 a.m.
Updated: Jan. 28, 2024, 8:24 a.m.
Edit Question
Initialize an Object
× Destroy an Object
Created : Jan. 4, 2024, 1:11 a.m.
Updated: Jan.


127.0.0.1:8000/list_test/
AI Intelligence
Study
RAW
ApplyCollege
CyberSecurity
Logout
Status*
DRAFT
Add Test
Tests
S.N
Test Name
Description
NoQ
Status
Created
Actions
1
Online Test
This is a online test
42
LIVE
Jan. 4, 2024, 1:10 a.m.
Edit Test
Add Questions
View Questions
2
Test Exam
Fair test
4
DRAFT
Jan. 5, 2024, 11:04 p.m.
Edit Test
Add Questions
View Questions

User Panel:


AI Quiz App
127.0.0.1:8000
AI Intelligence
Study
RAW
ApplyCollege
CyberSecurity
Welcome, manish gautam
Dashboard
Take a Test
Result and Score
Logout
Dashboard
Your Exam Statistics
Test Attempts
3
Completed Attempts
1
Incomplete Attempts
2


List Tests
127.0.0.1:8000/take_test/
AI Intelligence
Study
RAW
ApplyCollege
CyberSecurity
Welcome,
manish gautam
Dashboard
Take a Test
Result and Score
Logout
Tests Available
#1 Online Test
This is a online test
Number of Questions:20
Your Attempts :2
Max Score :65
View Results
Take Test
#2 test by manish
Number of Questions:4
Your Attempts :1
Max Score :None
View Results
Take Test


AI Quiz App
127.0.0.1:8000/test_attempts/
AI Intelligence
Study
RAW
ApplyCollege
CyberSecurity
manish gautam
Dashboard
Take a Test
Result and Score
Logout
Attempt for ->[#1] Online Test
Time taken : 0.00 seconds
Attempt Incomplete, Please Continue
Continue Attempt
Total Questions Answered: 0
Score: 0.00%
Attempt for ->[#2] test by manish
Time taken : 0.00 seconds
Attempt Incomplete, Please Continue
Continue Attempt
Total Questions Answered: 0
Score: 0.00%
Attempt for ->[#1] Online Test
Time taken : 115.80 seconds
Right: 13
Wrong: 7
Total Questions Answered: 20
Score: 65.00%

Snippets of major source code components:

mcq data generator.py

Data works > mcq data generator.py > ...

```python
import numpy as np
import pandas as pd

def generate_mcq_data(num_samples):
    """
    Generate multiple-choice question (MCQ) data for two classes: "Easy" and "Hard".

    Parameters:
    num_samples (int): Number of samples to generate for each class.

    Returns:
    pd.DataFrame: A DataFrame containing the generated MCQ data.

    Example:
    >>> generate_mcq_data(200)
       AverageTimeTaken  NumAttempts  NumCorrectAttempts Label
    0         29.671568            2                   3  Easy
    1         33.812526            3                   4  Easy
    2         47.654152            1                   2  Easy
    3        157.931957            9                   1  Hard
    4        322.654789            7                   0  Hard
    """
    # Set seed for reproducibility
```

mcq data generator.py

Data works > mcq data generator.py > ...

```python
    # Set seed for reproducibility
    np.random.seed(42)

    # Generating data for the "Easy" class
    easy_data = {
        'AverageTimeTaken': np.random.uniform(10, 60, num_samples),  # 10 to 60 seconds
        'NumAttempts': np.random.randint(1, 5, num_samples),  # 1 to 4 attempts
        'NumCorrectAttempts': np.random.randint(2, 5, num_samples)  # 2 to 4 correct attempts
    }

    easy_df = pd.DataFrame(easy_data)
    easy_df['Label'] = 'Easy'

    # Generating data for the "Hard" class
    hard_data = {
        'AverageTimeTaken': np.random.uniform(120, 600, num_samples),  # 2 to 10 minutes
        'NumAttempts': np.random.randint(5, 11, num_samples),  # 5 to 10 attempts
        'NumCorrectAttempts': np.random.randint(0, 2, num_samples)  # 0 to 1 correct attempts
    }

    hard_df = pd.DataFrame(hard_data)
    hard_df['Label'] = 'Hard'
```

mcq data generator.py

Data works > mcq data generator.py > ...

```python
    # Combine both datasets
    dataset = pd.concat([easy_df, hard_df], ignore_index=True)

    # Shuffle the dataset
    dataset = dataset.sample(frac=1).reset_index(drop=True)

    return dataset

# Generate MCQ data with 200 samples for each class
mcq_data = generate_mcq_data(200)

# Display the first few rows of the dataset
print(mcq_data.head())

# Save the dataset to a CSV file if needed
mcq_data.to_csv('question_classification_dataset.csv', index=False)
```

models.py

questionapp > models.py > ...

```python
from django.db import models
from django.utils import timezone
from django.contrib.auth import get_user_model
User  = get_user_model()
# Create your models here.
class Subject(models.Model):
    name = models.CharField(max_length=50,)

    def __str__(self):
        return self.name
class Question(models.Model):
    CATEGORY_CHOICES = [('HARD', 'HARD'),('EASY','EASY'),('UNCATEGORIZED','UNCATEGORIZED')]
    category = models.CharField(max_length=20,choices=CATEGORY_CHO (function) null: Any rue,default="UNCATEGORIZED")
    subject = models.ForeignKey(Subject,on_delete=models.SET_NULL,null=True,blank=True,related_name='questions')
    question = models.TextField()
    marks = models.IntegerField(default=1)
    active = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True,null=True,)
    updated = models.DateTimeField(auto_now=True,null=True,)
    def __str__(self):
        return self.question

```

models.py

questionapp > models.py > ...

```python

class Answer(models.Model):
    question = models.ForeignKey(Question,on_delete=models.CASCADE,related_name='answers')
    answer = models.CharField(max_length=300)
    is_right = models.BooleanField(default=False)
    def __str__(self):
        return self.answer



class Test(models.Model):
    TEST_STATUS = [
        ('DRAFT','DRAFT'),
        ('LIVE','LIVE')
    ]
    title = models.CharField(max_length=100,null=True,)
    questions = models.ManyToManyField(Question,related_name="tests",blank=True)
    description = models.TextField(blank=True,null=True)
    created = models.DateTimeField(auto_now_add=True,null=True,)
    updated = models.DateTimeField(default=timezone.now,null=True,)
    status = models.CharField(max_length=20,choices=TEST_STATUS,default="DRAFT")
    created_by = models.ForeignKey(User,on_delete=models.CASCADE,null=True,blank=True)
```

models.py

questionapp > models.py > ...

```python
    def __str__(self):
        return self.title

class TestAttempt(models.Model):
    user = models.ForeignKey(User,on_delete=models.SET_NULL,null=True)
    test = models.ForeignKey(Test,on_delete=models.SET_NULL,null=True,related_name='test_attempts')
    completed = models.BooleanField(default=False)
    score = models.DecimalField(default=0,max_digits=8,decimal_places=2,null=True)
    total_right = models.IntegerField(default=0,null=True)
    total_wrong = models.IntegerField(default=0,null=True)
    total_questions = models.IntegerField(default=0,null=True)
    timetaken = models.DecimalField(default=0,max_digits=10,decimal_places=2,null=True)
    timestamp = models.DateTimeField(auto_now_add=True,null=True,)
    updated= models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"#{self.id} {self.user} - {self.test}"


class UserQuestionAttempt(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE,null=True)
    test_attempt = models.ForeignKey(TestAttempt,on_delete=models.CASCADE,null=True,related_name='questions_attempt')
    question = models.ForeignKey(Question,on_delete=models.CASCADE,related_name='userquestionattempts')
    answer = models.ForeignKey(Answer,on_delete=models.CASCADE,null=True,blank=True)
    time_taken = models.DecimalField(default=0,max_digits=10,decimal_places=2,null=True) # in seconds
    right_attempt = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.user} -> {self.right_attempt} -> {self.answer}"

```

Log of visit to Supervisor:

Texas International College

Department of Computer Science and Information Technology
Mitrapark, Chabahil, Kathmandu

Project Log Report (CSIT – 2076 Batch)

Group Member Details			Project Title
S.N.	Name	T.U. Roll No.	
1.	Anusha Maharjan	23734/076	Quiz App
2.	Anusha Nepal	23735/076	
3.	Manish Gautam	29709/076	

S.N.	Date	Topics Discussed	Signature
1	21st Jan 2024 (07/10/2080)	Project overview and algorithm discussion	
2	22st Jan 2024 (08/10/2080)	Project overview and functionalities	
3	26st Jan 2024 (12/10/2080)	Documentation review	
4	28th Jan 2024 (14/10/2080)	project and Documentation Review	
5	15th march 2024 (2/12/2080)	Documentation Review	
6			
7			

Ram Prasad Subedi
Project Supervisor Name

Subu Nepal
Academic Coordinator
IT Department (B.Sc. CSIT | BCA)
Texas International College