TABLE OF CONTENT

List of Figures

ABSTRACT

The Online Examination Portal web application is being featured in this document, that will include all characteristics and procedures which are required. This document includes acute information about objectives, scope, technical details, feasibility of the system. Online Examination Portal is very beneficial for Educational Institute's to formulate an entire exam, saving time to check the papers and concoct mark sheets. Online exam portal helps students to offer a quick and easy way to appear for the test. It also provides the results immediately after the examination with 100% accuracy and security. Student can enter to perform exam only with their valid username and password. This examination contains multiple choice questions and appropriate number of options.

In the recent times, due to Covid, when schools & colleges were shut down & entire mode of education went online, there was not option for teachers to evaluate their students & promote them to next standards. The availability of Exam Portal application will resolve such issues and will enable students to give exam from any part of the world. It also offers facility to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast results.

CHAPTER – 1

INTRODUCTION

The Exam Portal web application is an online test simulator to hold online examination test in a decisive and skilled manner and thus avoiding wastage of time for manually examining the test paper. The major objective of this web based online examination system is to competently evaluate the student/candidate thoroughly through an automated system that not only saves a lot of time but also gives fast, improved, and accurate results. It allows the students/candidate appearing for the exam to give the test/papers according to their convenience from any location by simply using internet and time over the manual/traditional pen and paper format. The online examination system provides the student/candidate with a quick and easy way to show up for the test/examination. It also imparts and produces results of the examination immediately after the examination with 100% accuracy and security. Student can appear for the exam only with their valid credentials such as valid username and valid password. This examination consists of multiple-choice questions and appropriate number of options. This system provides time limit to each user. The candidate can verify their results immediately after the completion of the exam.

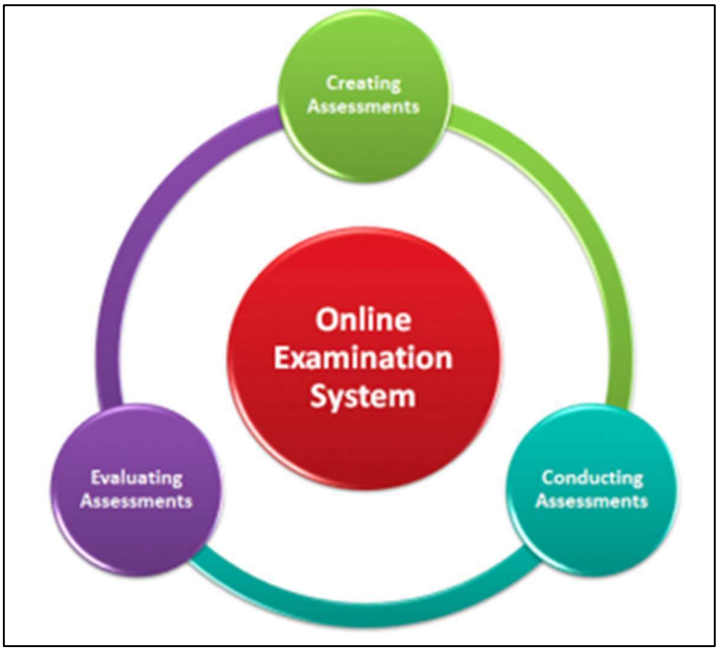


Fig 1.1: Overview of Online Examination Portal

CHAPTER – 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM AND ITS DISADVANTAGES

Presently, most of the institutions are following pen & paper-based approach of conducting exams. Although it has several benefits but in total it's not very efficient system. The offline examination system has numerous downsides, for example, trouble in investigating the exam physically, more invigilators are needed to take a test of numerous understudies. Exam results are not precise since figuring's are executed physically, the possibility of losing test's results is excessive in present frameworks, result checking is time-consuming as it was physically done, at one time counted number of students can appear for the exam. The advancement in data technology and using it systematically and appropriately will overcome the current mistake in the manual framework. This system is required to prepare registration\application form, question paper for the students and required to print a lot of number manually. To calculate how many students registered, and verification of details of these students in a month by hand is very difficult. This requires quite a lot of time and wastage of money as it requires quite lot of manpower to do that. Another factor that takes into account that is the possibility of errors. Some of the few mentioned key points which are downsides of existing pen & paper manual examination system:

- Time Consuming for creating question paper.
- Time to check right and wrong answers.
- Calculation of Marks.
- Human error.
- Limitation of no of student can give examination at a time.
- Student needs to come exam centre for giving test.

2.2 PROPOSED SYSTEM

This Web Application provides facility to conduct online examination worldwide. We have studied the manual examination system of a college and identified the possible automation for the examination system in the college. It saves time as it allows number of students to give the

exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by the server. Administrator has a privilege to create, modify and delete the test. Papers and its questions. User can register, login and give the test with his specific id, and can see the results as well. The system can be easily operated, as it is unambiguous, handy where on spot entries can be considered.

The proposed system which is accessed by the user is categorized as:

Admin: They are the ones who operate the whole system. They can create exams for students for specific Subjects. They can add, delete, and update the question from the system. They can also manage the results and rank the students according to their marks.

Student: They have given privileges only for registering themselves into the system. Students can also check their results and see their ranks once the exam is over.

2.3 PROJECT OBJECTIVES

The general objective of the online examination system is to minimize the traditional, manual p en and paper format of examination and introduce the examinees and examiners to tranquil and effortless mode of examination. In view of a long-term usage the online examination is here to stay worldwide which enables the Educational Institutes to evaluate the exams in contented and secure form.

Below mentioned are the crucial intents of the Online Examination System:

1. To provide an interface through which student can appear for examination online for objective as well as subjective type questions.
2. To provide registration for students done by themselves.
3. The Online Examination System inspects the candidate appearing for the exam through the online objective test
4. The answer/response chosen by the candidate will be verified by an automated mechanism.
5. It is an integrated Online Examination System that minimizes the paperwork for both the examiners and examinees.
6. It allows the examiners to create/prepare the question paper.

2.4 PROJECT SCOPE

Online Examination System is widely used as compared to other examination method these days. Online examination system can be used by corporate organizations and educational institution as well. As it is convenient and adaptable, web-based application can be used anywhere and anytime. As every coin has two faces, the same goes with the software every software may have some instances of bugs, errors, security related problems or system faults. There may be occurrence of different problems or system faults for example, computer collapse or crashes due to power supply problem will invalidate efforts of number of students. Online Examination system is designed for Educational Institutions (like schools, colleges, universities, training Institutions).

- It can be used anywhere any time as it is a web-based application.
- This system will provide better security and transparency in the examination.
- The system handles all the operations and generates reports as soon as the test is finish.
- The type of questions is only multiple choice or true and false.

2.5 PROJECT METHODOLOGY

This work plan to build up a framework to identify a wide assortment of unfair practices during the web assessment and lead a fair test. Our proposed online assessment framework has two stages, the initial one is the preparation stage, and the second one is the test stage. In the preparation stage, the competitor needs to verify himself before beginning the test by utilizing login username and password. In the second section that's the test stage, the candidate gives an exam. The advantages of the proposed framework are that the ease of giving exam for the student and reduction of manual labour work done by evaluator. The new proposed framework is easy to understand, and quick entries can be made in this framework. In a complete cycle, no manual integration is required. Understudy can test from any spot of the world 24x7; there are no topographical boundaries—100% rightness in marks computation and result affirmation. The diverse inquiry set for various applicants.

Structured Systems Analysis and Design Methodology (SSADM). It was used to conduct this study. This work also has compelling reasons for using this approach. The SSADM approach is widely used in the analysis and design phases of system growth. It receives a prescriptive way to deal with data framework improvement.

CHAPTER – 3

LITERATURE SURVEY

Various investigates been done regarding the matter of an online assessment framework which can be addressed as given point.

Fagbol et al. [1] proposed a system called a Computer Based System (CBS). CBS is a web-based Online Exam System (OES) designed to help an examination process and resolve challenges such as absence of scheduling adaptability for automation, an applicant log-off upon a permission period, outcome integrity, an assurance, an independent execution, need for adaptability, robustness, built for supporting examination process and address challenges such as exam behaviour, auto checking, auto accommodation and a report generation of the exam result.

Ayo et al. [2] proposed a model called E-examination implementation. The software was created at a Nigerian private university. Developing such software is to conduct the Joint Admission Matriculation Board (JAMB) entrance exam for all Nigerian universities. Convent University, a private university in Nigeria, was responsible for developing and testing this program. They considered the program to be instrumental in programming and ordinary investigation.

Wei et al. [3] built a framework called Online Assessment Framework (OEF). OEF upholds some exceptional fundamental features like auto-generation of rank and results, auto-generation of questions, working inquiries like programming, altering MS Word, PowerPoint, MS Windows, Excel, and so on.

Rashad et al. [4] proposed a web-based framework named Exam Management Assessment (EMA).EMA has all essential highlights like overseeing assessment, assessing understudy's answers, conducting the assessment, and incorporating auto imprint for the accommodation, secure login.

Arvind Singh, Niraj Shrike, Kiran Shetty [5] proposed a system called OES. OES is a customizable system. Students' answers are checked automatically and fastly.

Guzan and Conejo et al. [6] proposed OES called SITTIE Automatic Assessment Environment. SITTIE is a web-based tool for creating and modifying adaptive experiments. It can be used to achieve instructional goals by combining self-evaluation test questions with feedback and hints. Other features include resumption ability, multi- invigilators, random question collection, irregular inquiries circulation, random distribution of choice.

CHAPTER – 4

FEASIBILITY STUDY

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high-level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

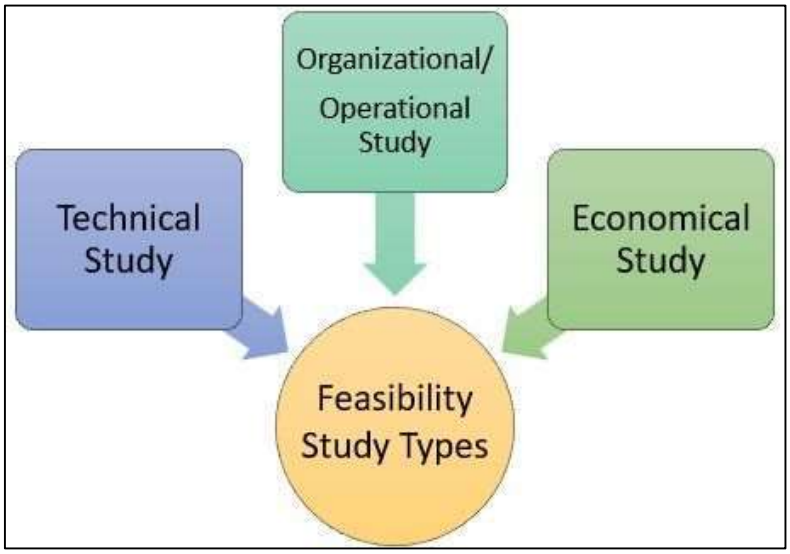


Fig 4.1: Types of Feasibility Study

The system has been tested for feasibility in the following points.

1. Technical Feasibility

2. Economical Feasibility

3. Operational Feasibility

4.1 TECHNICAL FEASIBILITY

This website is very much technically feasible. This website is very much concerned with specifying equipment and the website will successfully satisfy almost all the user's requirements. The technical need for this system may vary considerably but might include:

a. The facility to produce report instantly after finishing exam.

b. Response time under certain conditions.

c. Ability to process a file at a particular speed.

Therefore, the basic input/output of all files is identified. So, the project can easily be built up and it will also be technically feasible.

4.2 ECONOMICAL FEASIBILITY

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

4.3 OPERATIONAL FEASIBILITY

In this project, the management will know the details of each project where he may be presented, and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessaries.

CHAPTER – 5

SOFTWARE AND HARDWARE REQUIREMENTS

5.1 SOFTWARE REQUIREMENTS:

Backend Frameworks: Spring Boot & Java

Front End Frameworks: HTML5, CSS3, JavaScript, Bootstrap, & ReactJS

Runtime Environment: Windows System

Database Tool: MySQL

IDE: VS Code & IntelliJ Idea

5.2. HARDWARE REQUIREMENTS:

PROCESSOR: Intel core i3 or Above

RAM: 8 GB or above

HDD: 2 GB or more (Free Space)

O.S: Windows 8 or higher

Screen: VGA Monitor or Laptop Screen

Mouse: Standard Mouse

Keyboard: Standard Keyboard (QWERTY)

Backup Device: CD or Pen Drive (PD) minimum 4 GB

Power Backup: UPS/ 4 Cell Batteries

CHAPTER – 6

SYSTEM IMPLEMENTATION

6.1 TECHNOLOGY USED

6.1.1 Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today's digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well.

While most modern Java applications combine the Java runtime and application together, there are still many applications and even some websites that will not function unless you have a desktop Java installed. Java.com, this website, is intended for consumers who may still require Java for their desktop applications – specifically applications targeting Java 8. Developers as well as users that would like to learn Java programming should visit the dev.java website instead and business users should visit oracle.com/java for more information.

How Java Works?

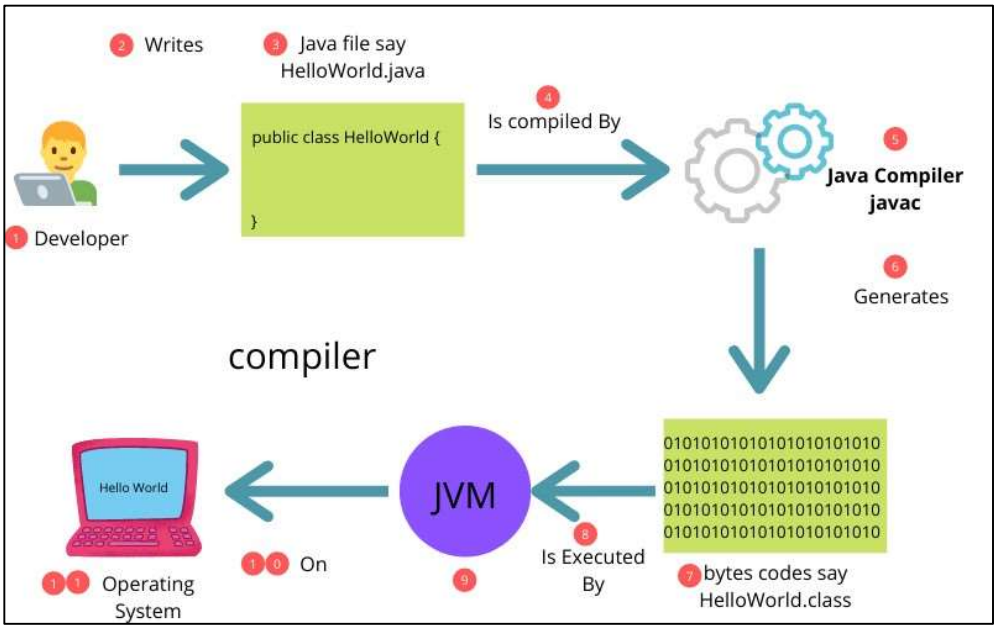


Fig 6.1: Working of Java

The Java software platform consists of the JVM, the Java API, and a complete development environment. The JVM parses and runs (interprets) the Java bytecode. The Java API consists of an extensive set of libraries including basic objects, networking and security functions; Extensible Markup Language (XML) generation; and web services. Taken together, the Java language and the Java software platform create a powerful, proven technology for enterprise software development.

Components of Java

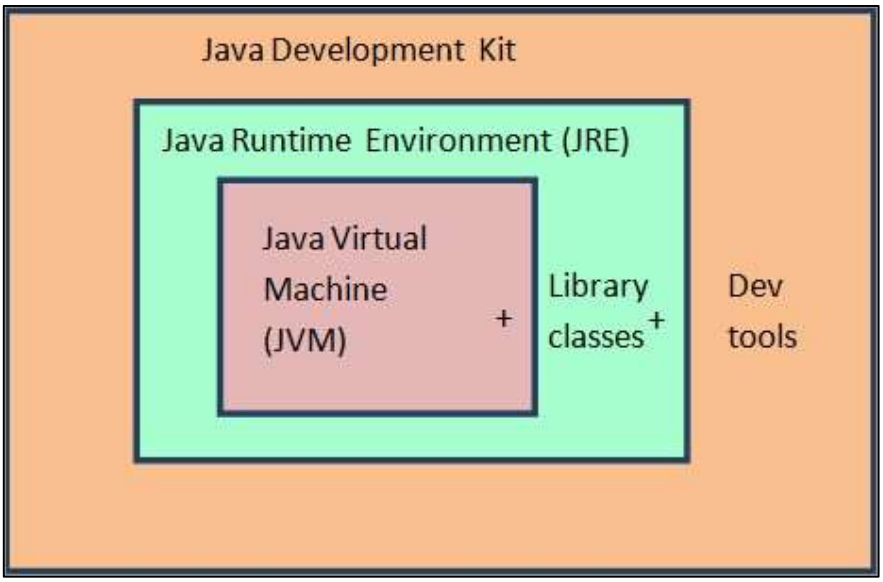


Fig 6.2: Components of Java

6.1.2 Spring Boot

The Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application. Spring enables you to build applications from "plain old Java objects" (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Examples of how you, as an application developer, can benefit from the Spring platform:

1. Make a Java method execute in a database transaction without having to deal with transaction APIs.
2. Make a local Java method an HTTP endpoint without having to deal with the Servlet API.

3. Make a local Java method a message handler without having to deal with the JMS API.
4. Make a local Java method a management operation without having to deal with the JMX API.

Java Spring Boot (Spring Boot) is a tool that makes developing web application and micro services with Spring Framework faster and easier through three core capabilities:

1. Autoconfiguration
2. An opinionated approach to configuration
3. The ability to create standalone applications

These features work together to provide you with a tool that allows you to set up a Spring-based application with minimal configuration and setup.

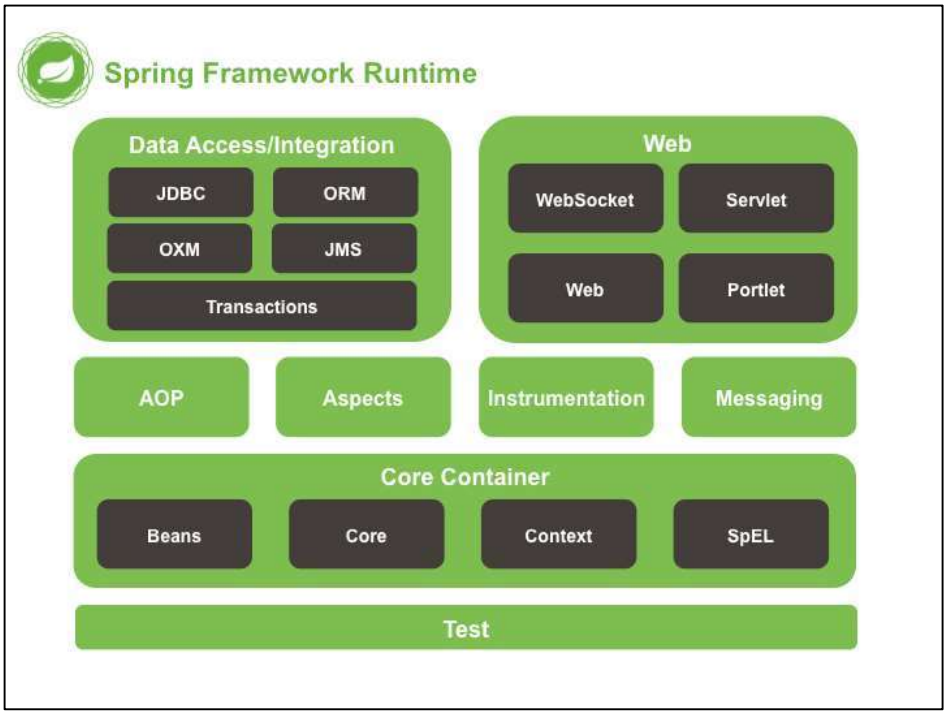


Fig 6.3: Spring Framework Runtime

6.1.3 HTML5

Hypertext Markup Language revision 5 (HTML5) is markup language for the structure and presentation of World Wide Web contents. It is client-side markup language and is platform independent. HTML5 consists of various tags which allows us to structure data in a definite way.

Some of the most used tags are heading tags, paragraph tags, division tags, anchor tags etc. One of the helpful aspects of HTML is, it can embed programs written in a scripting language like JavaScript and CSS. HTML is not considered a programming language as it can't create dynamic functionality. Instead, with HTML, web users can create and structure sections, paragraphs, and links using elements, tags, and attributes.

6.1.4 CSS3

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language**.**

There are 3 ways to write CSS in our HTML file.

- Inline CSS
- Internal CSS
- External CSS

6.1.5 JavaScript

JavaScript is a cross-platform, object-oriented language used to interact with web pages (e.g., complex animation, clickable buttons, pop-up menus, etc.). There are also more advanced JavaScript versions such as Node.js, which allow you to add more functionality to a website rather than downloading files (such as real-time interaction between multiple computers). Within the host environment (for example, a web browser), JavaScript can be linked to its native objects to provide system control over them.

6.1.6 ReactJS

React is a JavaScript-based UI development library. Facebook and an open-source developer community run it. Although React is a library rather than a language, it is widely used in web development. The library first appeared in May 2013 and is now one of the most commonly used frontend libraries for web development.

React offers various extensions for entire application architectural support, such as Flux and React Native, beyond mere UI.

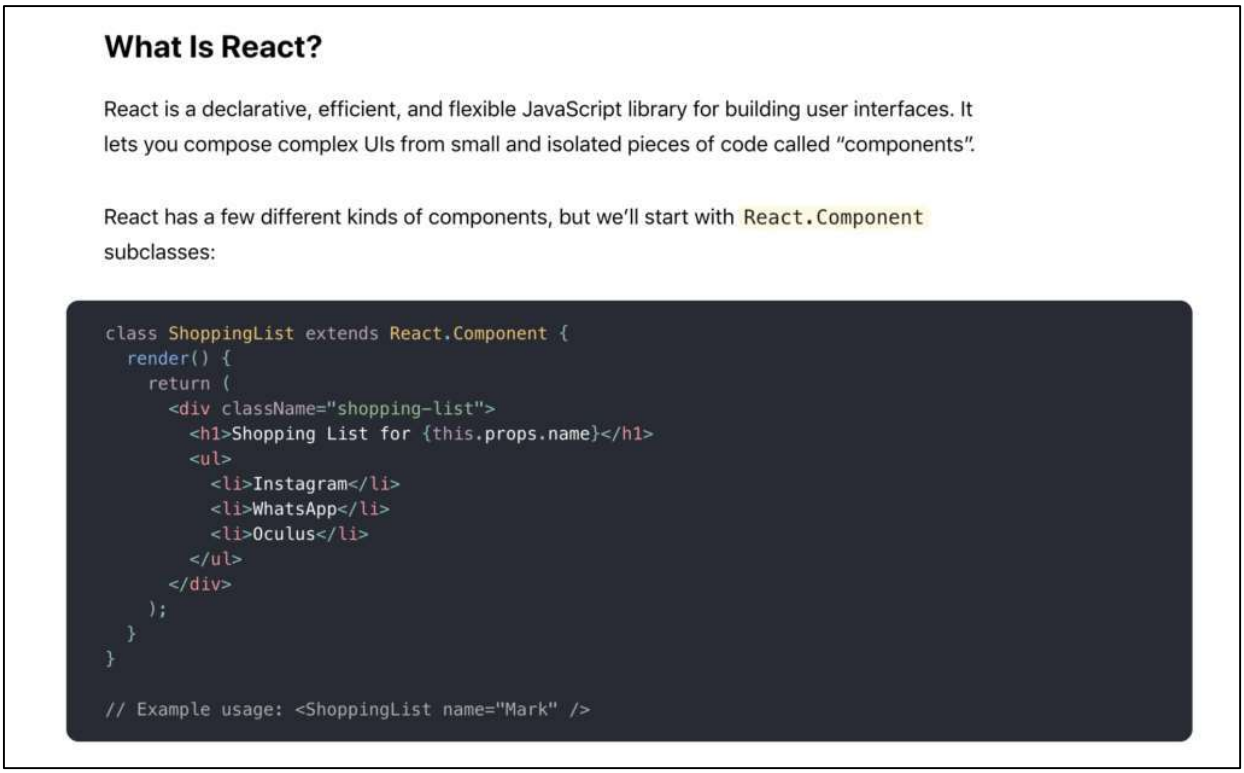


Fig 6.4: ReactJS Sample Code

6.1.7 MySQL

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching, and writing would not be so fast and easy with those type of systems.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

- MySQL is released under an open-source license. So, you have nothing to pay to use it.
- MySQL is a very powerful program. It handles a large subset of the functionality of the most expensive and powerful database packages.

- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.

6.2 System Architecture

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client server architecture in which presentation, application processing, and data management functions are logically separated. For example, an application that uses middleware to service data requests between a user and database employees multi-tier architecture. The most widespread use of multi-tier architecture is the three-tier architecture.

Three-tier architecture

Three-tier architecture is a client–server architecture in which the user interface, functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. The three-tier model is a software architecture pattern. Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology.

Presentation tier

The tier is the user interface of the application, where users interact with the application. Its main purpose is to display information to and collect data from users. This is the top-most level in the architecture and can be run on the web browser or a desktop/mobile application.

Application tier

This is the heart of the application, better known as the logic tier or middle tier. It coordinates all business logic of the application, prescribes how business objects interact with each other. It handles collected information from the Presentation tier. During the process, this tier may need to access the Data-tier to retrieve or modify the data. In a three-tier application, all

communication goes smoothly through the Application tier. The Presentation tier and the Data-tier cannot communicate directly with one another.

Data-tier

The Data-tier is sometimes referred to as the database tier, where it stores and manages the data processed by the Application tier.

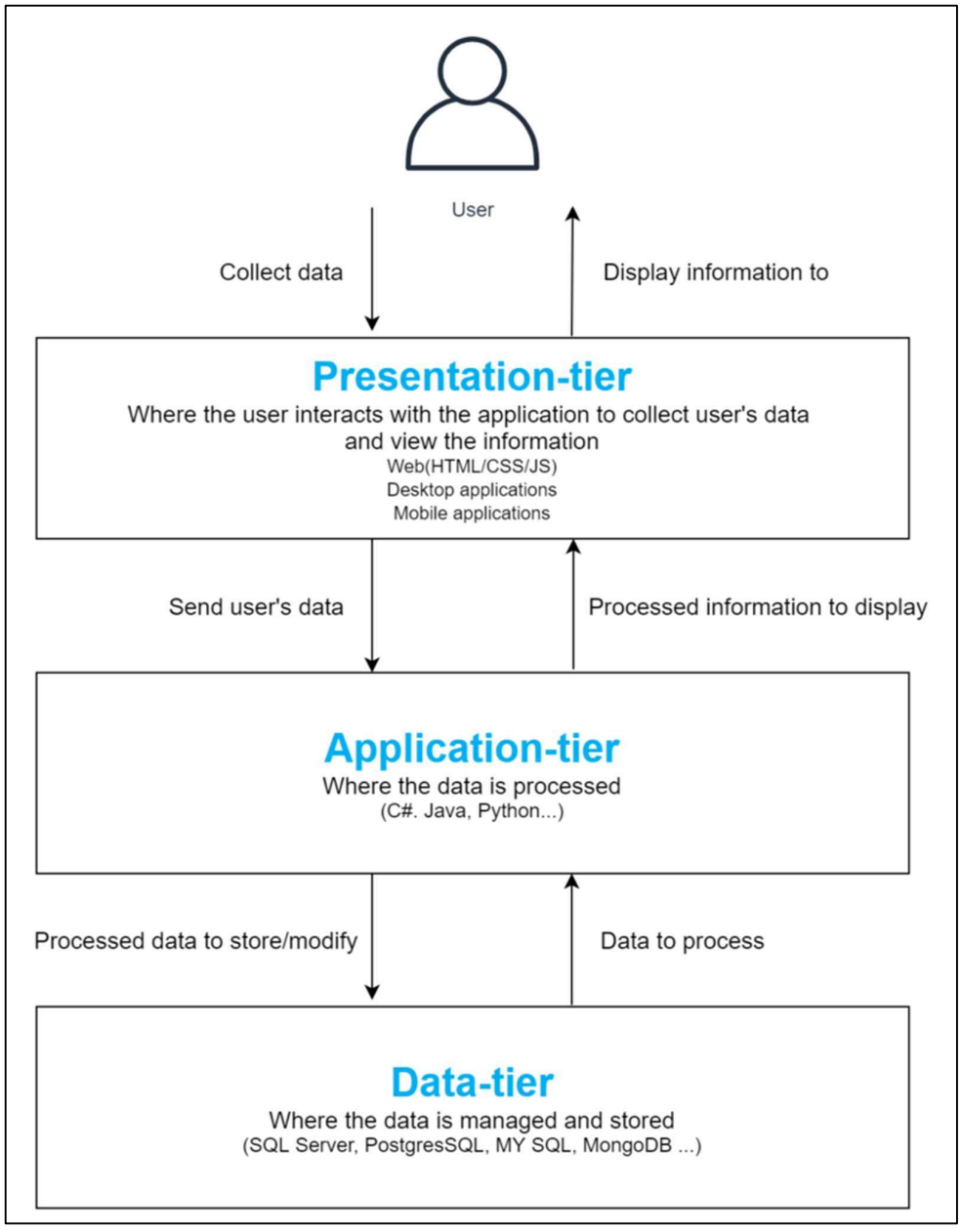


Fig 6.5: 3-tier Architecture

CHAPTER - 7

REQUIREMENT ANALYSIS AND SPECIFICATIONS

The main goal of the online examination system is to realize the system to send the question and examination automatically. After the examination is over, the system can also read the paper under a certain plan according to the reference answer. To achieve this goal, we need to manage examination questions systematically and effectively, automatically generate examination papers, and retrieve papers on time.

7.1 Functional Requirements

This system will be used in 2 User Modules which are Administrator or Faculty and Student. As all of these have different requirements, the modules are designed to meet their needs and avoid any type of confusion. The features of all 2 User Modules have been described below.

- **ADMIN**
  i. Register themselves.
  ii. Login themselves.
  iii. Add/Delete/Update Categories.
  iv. Add/Delete/Update Quizzes.
  v. Add/Delete/Update Questions.
  vi. Able to view results.

- **USER**
  i. Register themselves.
  ii. Login themselves.
  iii. See the available Categories.
  iv. See and attempt the available Quizzes.
  v. See and attempt to solve the Questions.
  vi. Able to view results.

7.2 Non-Functional Requirements

Usability

Usability is a quality attribute used to access how easy the interface is to use. Usability is ease of use. It tells how user friendly the interface is. It includes memorability, learnability, and satisfaction. Our software interface has all the above quality. Any kind of user can easily understand the interface.

Reliability

Reliability is how much the system is consistent in different platforms. The ability of an apparatus, system to consistently perform its required function, on demand and without degradation or failure.

Integrity

Integrity means doing the right thing in a reliable way. Data integrity is a fundamental component of security. In its broadcast use, “Data Integrity” refers to the accuracy and consistency of data stored in a database, data mart or another construct. Data integrity is the overall completeness, accuracy, and consistency of data.

PERFORMANCE

Performance is also a major non-functional requirement. Performance Requirements about resources required, response time, transaction rate or anything else having to do with performance.

CHAPTER - 8

MODULE DESCRIPTION

This section includes the modules, architecture and various elements that are combined to form the whole system's framework**.**

8.1 Registration Module: It will allow new user to get register on the website. User needs to provide his/her full name, email id, contact number, & password while registering.

8.2 Login Module: In this module user can login to the website by entering login ID and password. After Login the system links to Profile page.

8.3 Category Module: It allows the admins to add, update, or delete category. Normal user (students) can see the quizzes based on their categories.

8.4 Quiz Module: It allows the admins to add, update, or delete quiz. Each quiz belongs to one category. This allows to filter the quiz by category. Normal user (students) can see the quizzes and if they desire, they can attempt the quiz.

8.5 Exam Module: This is the main module which handles the logic used when a student starts the quiz. Timer will run for each quiz, & once it finishes, quiz will auto-submit. The quiz can also be submitted by the student beforehand, by clicking on the Submit button. The exam module will pass the data entered by student to result module.

8.6 Result Module: This module produces the result for each attempted quiz. The attempted question by students is matched against the correct answer saved in database. And based on that the result is generated.

CHAPTER - 9

SDLC METHODOLOGY

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands. The SDLC defines and outlines a detailed plan with stages, or phases, that each encompass their own process and deliverables. Adherence to the SDLC enhances development speed and minimizes project risks and costs associated with alternative methods of production.

9.1 Why SDLC is Required?

Carrying a project without any action plan can be a disaster and can eventually lead to its downfall if not delivered on time. From allocating resources to deployment, it all must go through a pipeline to align the whole piece of the development cycle. That's the primary reason, SDLC came into the limelight, and after witnessing glorious success and became an enormous hit. Providing a high-class phase that also matches the customer's requirement in terms of cost, time and efficiency are among the major aim of this cycle.

9.2 How does the SDLC work?

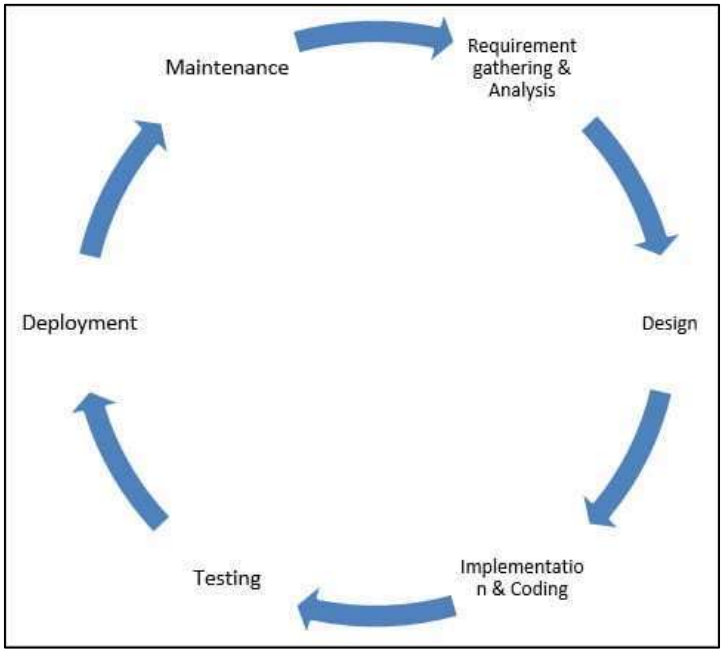


Fig 8.1: Steps of SDLC

Requirement Gathering and Analysis Phase

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only. Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.The planning phase encompasses all aspects of project and product management. This typically includes resource allocation, capacity planning, project scheduling, cost estimation, and provisioning. During the planning phase, the development team collects input from stakeholders involved in the project, customers, sales, internal and external experts, and developers. This input is synthesized into a detailed definition of the requirements for creating the desired software. The team also determines what resources are required to satisfy the project requirements, and then infers the associated cost.

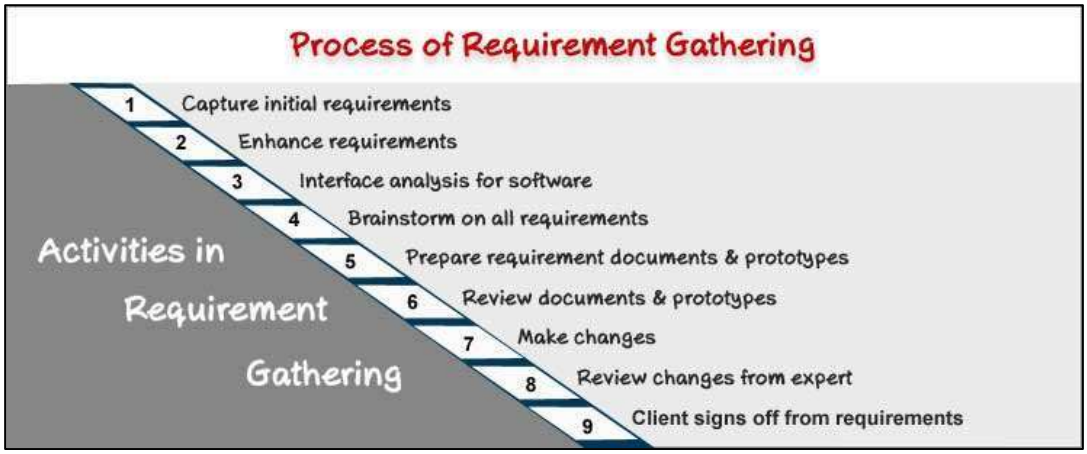


Fig 8.2: Process of Requirement Gathering

Expectations are clearly defined during this stage as well; the team determines not only what is desired in the software, but also what is NOT. The tangible deliverables produced from this phase include project plans, estimated costs, projected schedules, and procurement needs.

Design Phase

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

Testing Phase

The phase entails the evaluation of the created software. The testing team evaluates the developed product(s) to assess whether they meet the requirements specified in the 'planning' phase. Assessments entail the performance of functional testing: unit testing, code quality testing, integration testing, system testing, security testing, performance testing and acceptance testing, as well as non-functional testing. If a defect is identified, developers are notified. Validated (actual) defects are resolved, and a new version of the software is produced.

Deployment/Release Phase

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation. In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

Maintenance Phase

After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

9.3 Spiral Model

The Spiral Method is described by Barry Boehm in his 1986 paper "A Spiral Model of Software Development and Enhancement." The Spiral Model boils down to a meta-model, which evaluates the specific risk profile of the project before recommending an approach that blends aspects of the other popular methodologies of the day, including Iterative and Waterfall. As such, it rejects a one size fits all approach to process model adoption.

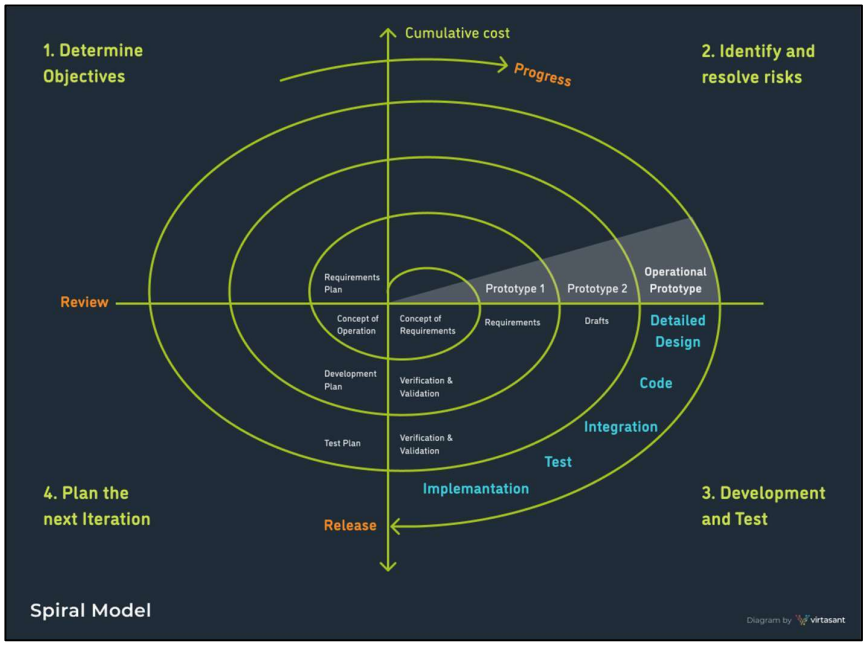


Fig 8.3: Spiral Model

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

Develop next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

CHAPTER - 10

SYSTEM DIAGRAMS

10.1 Database Schema

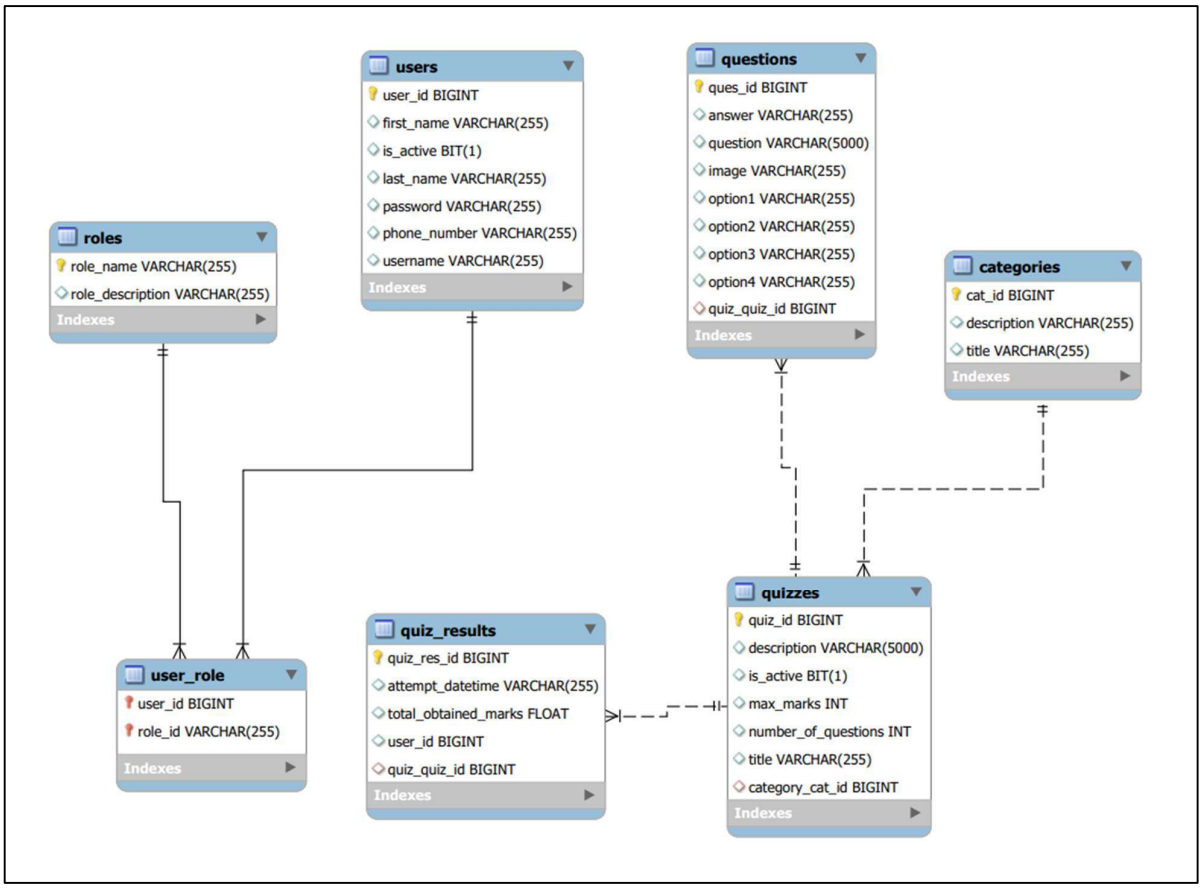


Fig 10.1: Database Schema Design

10.2 Use Case Diagram

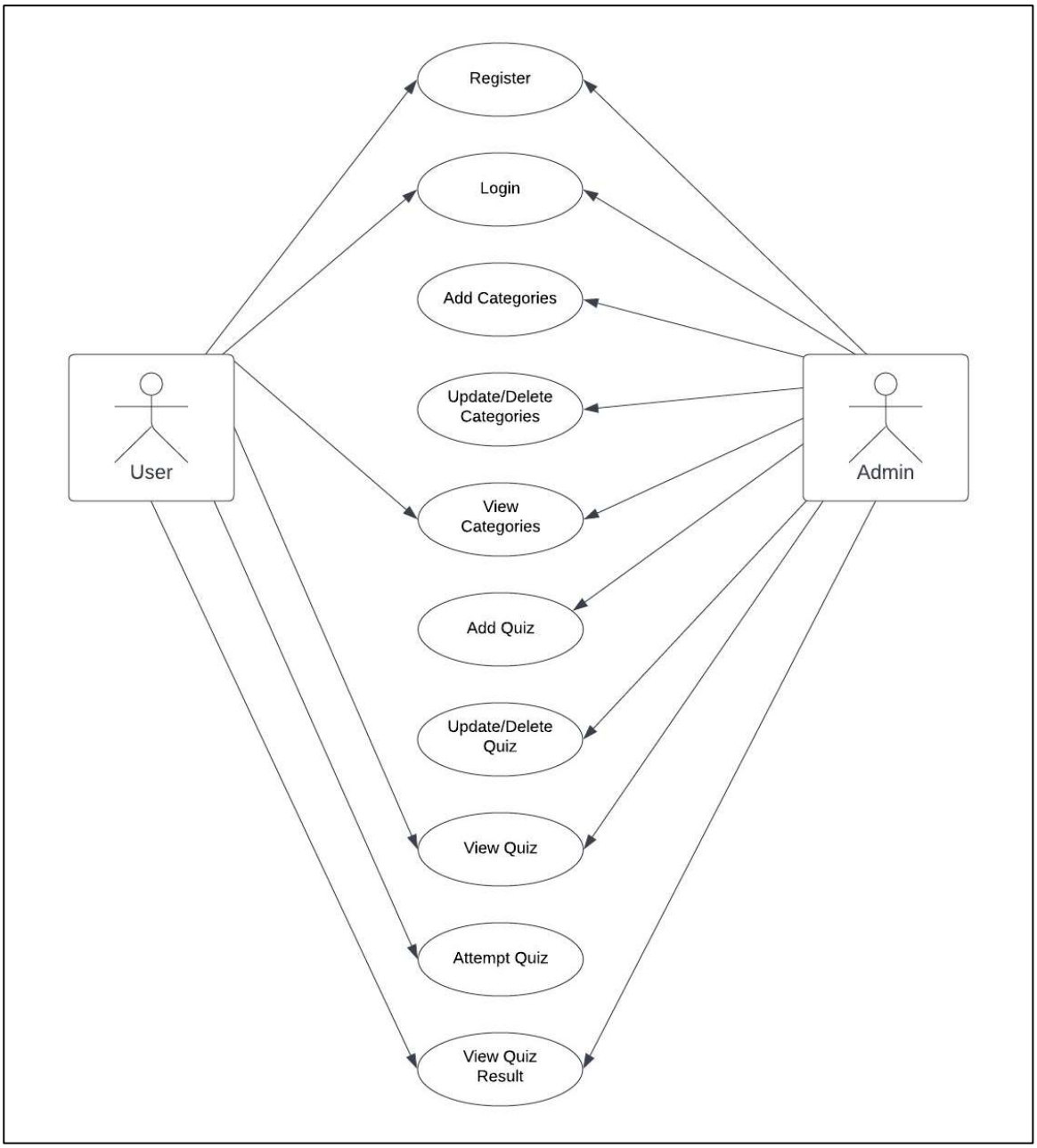


Fig 10.1: Use Case Diagram

10.3 Flow Chart

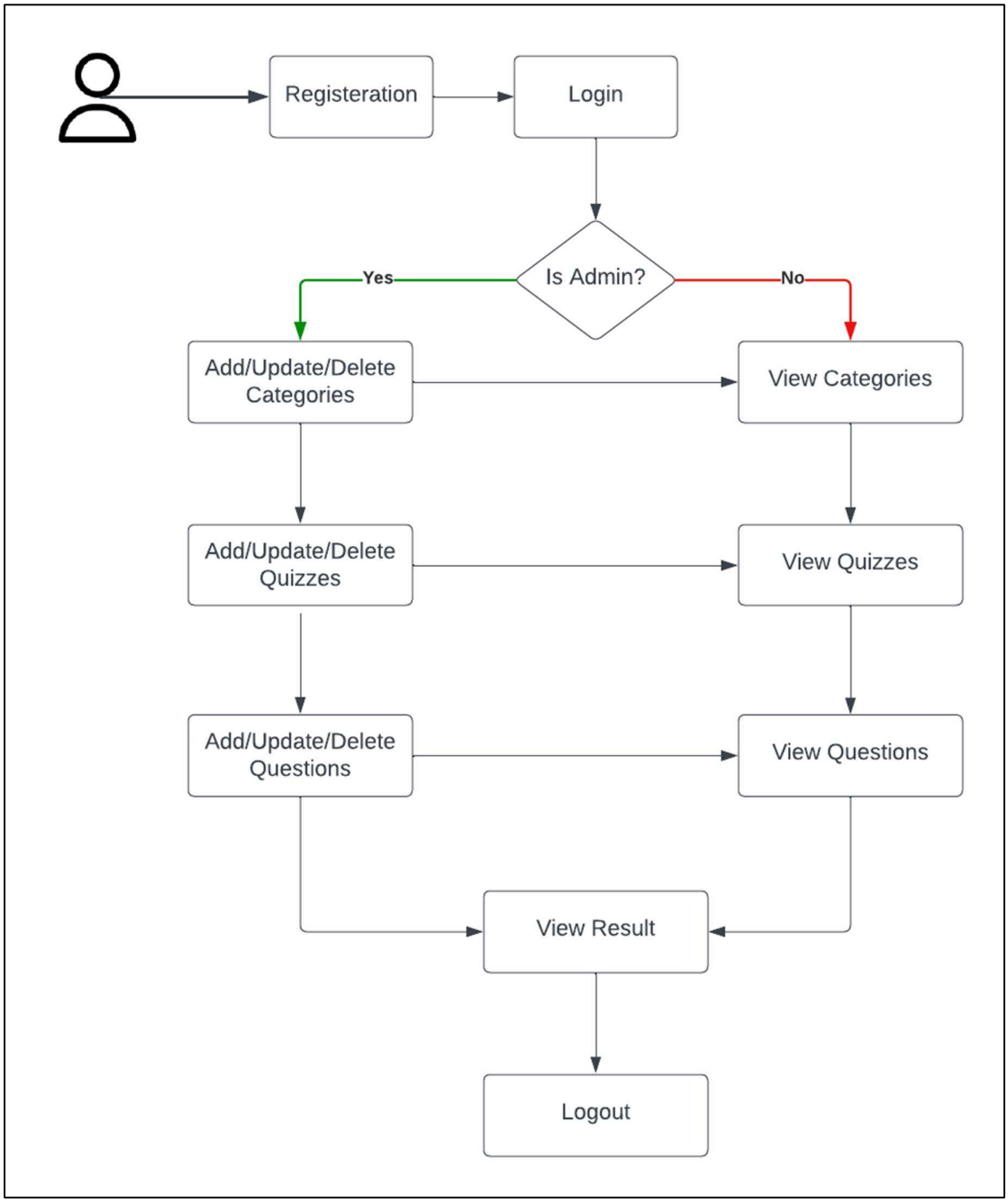


Fig 10.1: Flow Chart of the process from Registration till Logout.

CHAPTER - 11

CODING

11.1 Backend

ExamPortalBackendApplication.java

```
package com.project.examportalbackend;

import com.project.examportalbackend.models.Role;
import com.project.examportalbackend.repository.RoleRepository;
import org.springframework.boot.ApplicationRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import java.util.Arrays;

@SpringBootApplication
public class ExamPortalBackendApplication {

	public static void main(String[] args) {
		SpringApplication.run(ExamPortalBackendApplication.class, args);
	}

	@Bean
	public ApplicationRunner initializer(RoleRepository roleRepository) {
		return args -> roleRepository.saveAll(Arrays.asList(
				Role.builder().roleName("USER").roleDescription("Default
Role provided to each user").build(),

	Role.builder().roleName("ADMIN").roleDescription("Superuser, who has access for
all functionality").build()));
	}
}
```

User.java

```
package com.project.examportalbackend.models;

import lombok.*;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

@NoArgsConstructor
```

```
@AllArgsConstructor
@Getter
@Setter
@ToString
@Entity
@Table(name = "users") //annotations
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "user_id")
    private long userId;  // field

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "username", unique = true)
    private String username;

    @Column(name = "password")
    private String password;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "is_active")
    private boolean isActive = true;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(name = "user_role",
            joinColumns = {
                    @JoinColumn(name = "user_id")
            },
            inverseJoinColumns = {
                    @JoinColumn(name = "role_id")
            }
    )
    private Set<Role> roles;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Set<SimpleGrantedAuthority> authorities = new HashSet<>();
        this.roles.forEach(role -> authorities.add(new
SimpleGrantedAuthority(role.getRoleName())));
        return authorities;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
```

```
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return isActive;
    }
}
```

Role.java

```
package com.project.examportalbackend.models;

import lombok.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@NoArgsConstructor
@AllArgsConstructor
@ToString
@Setter
@Getter
@Builder
@Entity
@Table(name = "roles")
public class Role {

    @Id
    @Column(name="role_name")
    private String roleName;

    @Column(name="role_description")
    private String roleDescription;
}
```

QuizResult.java

```
package com.project.examportalbackend.models;

import lombok.*;

import javax.persistence.*;

@Entity
@Getter
@Setter
@ToString
@NoArgsConstructor
```

```
@AllArgsConstructor
@Table(name = "quiz_results")
public class QuizResult {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long quizResId;

    @Column(name = "user_id")
    private Long userId;

    @Column(name = "total_obtained_marks")
    private float totalObtainedMarks;

    @Column(name = "attempt_datetime")
    private String attemptDatetime;

    @ManyToOne(fetch = FetchType.EAGER)
    private Quiz quiz;


}
```

Quiz.java

```
package com.project.examportalbackend.models;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Getter
@Setter
@ToString
@Table(name = "quizzes")
public class Quiz {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long quizId;

    @Column(name = "title")
    private String title;

    @Column(name = "description", length = 5000)
    private String description;

    @Column(name = "max_marks")
    private int maxMarks;
```

```
    @Column(name = "number_of_questions")
    private int numberOfQuestions;

    @Column(name = "is_active")
    private boolean iActive = false;

    @ManyToOne(fetch = FetchType.EAGER)
    private Category category;

    @OneToMany(mappedBy = "quiz", fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JsonIgnore
    private List<Question> questions = new ArrayList<>();

    @OneToMany(mappedBy = "quiz", fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JsonIgnore
    private List<QuizResult> quizResults = new ArrayList<>();
}
```

Question.java

```
package com.project.examportalbackend.models;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.*;

@Entity
@Getter
@Setter
@ToString
@Table(name = "questions")
public class Question {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long quesId;

    @Column(name = "question", length = 5000)
    private String content;

    @Column(name = "image")
    private String image;

    @Column(name = "option1")
    private String option1;

    @Column(name = "option2")
    private String option2;

    @Column(name = "option3")
    private String option3;

    @Column(name = "option4")
    private String option4;

    @Column(name = "answer")
```

```
    private String answer;

    @ManyToOne(fetch = FetchType.EAGER)
    private Quiz quiz;
}
```

LoginResponse.java

```
package com.project.examportalbackend.models;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
public class LoginResponse {
    private User user;
    private String jwtToken;
}
```

LoginRequest.java

```
package com.project.examportalbackend.models;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
public class LoginRequest {
    private String username;
    private String password;
}
```

Category.java

```
package com.project.examportalbackend.models;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Getter
@Setter
@ToString
```

```
@Table(name = "categories")
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long catId;

    @Column(name = "title")
    private String title;

    @Column(name = "description")
    private String description;

    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
    @JsonIgnore
    private List<Quiz> quizzes = new ArrayList<>();
}
```

UserRepository.java

```
package com.project.examportalbackend.repository;

import com.project.examportalbackend.models.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

RoleRepository.java

```
package com.project.examportalbackend.repository;

import com.project.examportalbackend.models.Role;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepository extends JpaRepository<Role, String> {
}
```

QuizResult.java

```
package com.project.examportalbackend.repository;

import com.project.examportalbackend.models.QuizResult;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface QuizResultRepository extends JpaRepository<QuizResult, Long> {
    List<QuizResult> findByUserId(Long userId);
}
```

QuizRepository.java

```
package com.project.examportalbackend.repository;

import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.models.Quiz;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface QuizRepository extends JpaRepository<Quiz, Long> {
    List<Quiz> findByCategory(Category category);
}
```

QuestionRepository.java

```
package com.project.examportalbackend.repository;

import com.project.examportalbackend.models.Question;
import com.project.examportalbackend.models.Quiz;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;


public interface QuestionRepository extends JpaRepository<Question, Long> {
    List<Question> findByQuiz(Quiz quiz);
}
```

CategoryRepository.java

```
package com.project.examportalbackend.repository;


import com.project.examportalbackend.models.Category;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CategoryRepository extends JpaRepository<Category, Long> {
}
```

QuizResultController.java

```
package com.project.examportalbackend.controllers;

import com.project.examportalbackend.models.Question;
import com.project.examportalbackend.models.Quiz;
import com.project.examportalbackend.models.QuizResult;
import com.project.examportalbackend.services.QuestionService;
import com.project.examportalbackend.services.QuizResultService;
import com.project.examportalbackend.services.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.time.ZoneId;
```

```
import java.time.ZonedDateTime;
import java.util.*;

@CrossOrigin
@RestController
@RequestMapping("/api/quizResult")
public class QuizResultController {

    @Autowired
    private QuestionService questionService;
    @Autowired
    private QuizService quizService;

    @Autowired
    private QuizResultService quizResultService;

    @PostMapping(value = "/submit", params = {"userId", "quizId"})
    public ResponseEntity<?> submitQuiz(@RequestParam Long userId, @RequestParam Long
quizId, @RequestBody HashMap<String,String> answers) {
        Quiz quiz = quizService.getQuiz(quizId);
        int totalMarks = quiz.getMaxMarks();
        int totalQuestions = quiz.getNumberOfQuestions();
        float marksPerQuestion = (float)totalMarks/totalQuestions;

        Question question = null;
        int numCorrectAnswers = 0;
        for(Map.Entry<String, String> m : answers.entrySet()){
            Long quesId = Long.valueOf(m.getKey());
            question = questionService.getQuestion(Long.valueOf(m.getKey()));
            if(question != null) {
                if(question.getAnswer().equals(m.getValue())){
                    numCorrectAnswers++;
                }
            }
        }
        float totalObtainedMarks = numCorrectAnswers*marksPerQuestion;

        QuizResult quizResult = new QuizResult();
        quizResult.setUserId(userId);
        quizResult.setQuiz(quizService.getQuiz(quizId));
        quizResult.setTotalObtainedMarks(totalObtainedMarks);
        final ZonedDateTime now = ZonedDateTime.now(ZoneId.of("Asia/Kolkata"));
        quizResult.setAttemptDatetime(now.toLocalDate().toString() + " " +
now.toLocalTime().toString().substring(0,8));

        quizResultService.addQuizResult(quizResult);
        return ResponseEntity.ok(quizResult);
    }

    @GetMapping(value = "/", params = "userId")
    public ResponseEntity<?> getQuizResults(@RequestParam Long userId){
        List<QuizResult> quizResultsList =
quizResultService.getQuizResultsByUser(userId);
        Collections.reverse(quizResultsList);
        return ResponseEntity.ok(quizResultsList);
    }
}
```

QuizController.java

```
package com.project.examportalbackend.controllers;

import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.models.Quiz;
import com.project.examportalbackend.services.CategoryService;
import com.project.examportalbackend.services.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
@RequestMapping("/api/quiz")
public class QuizController {

    @Autowired
    private QuizService quizService;

    @Autowired
    private CategoryService categoryService;

    @PostMapping("/")
    public ResponseEntity<?> addQuiz(@RequestBody Quiz quiz) {
        return ResponseEntity.ok(quizService.addQuiz(quiz));
    }

    @GetMapping("/")
    public ResponseEntity<?> getQuizzes() {
        return ResponseEntity.ok(quizService.getQuizzes());
    }

    @GetMapping("/{quizId}")
    public ResponseEntity<?> getQuiz(@PathVariable Long quizId) {
        return ResponseEntity.ok(quizService.getQuiz(quizId));
    }

    @GetMapping(value = "/", params = "catId")
    public ResponseEntity<?> getQuizByCategory(@RequestParam Long catId) {
        Category category = categoryService.getCategory(catId);
        return ResponseEntity.ok(quizService.getQuizByCategory(category));
    }

    @PutMapping("/{quizId}")
    public ResponseEntity<?> updateQuiz(@PathVariable Long quizId, @RequestBody Quiz
quiz) {
        if (quizService.getQuiz(quizId) != null) {
            return ResponseEntity.ok(quizService.updateQuiz(quiz));
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Quiz with id : " +
String.valueOf(quizId) + ", doesn't exists");
    }

    @DeleteMapping("/{quizId}")
    public ResponseEntity<?> deleteQuiz(@PathVariable Long quizId) {
```

```
        quizService.deleteQuiz(quizId);
        return ResponseEntity.ok(true);
    }
}
```

QuestionController.java

```
package com.project.examportalbackend.controllers;

import com.project.examportalbackend.models.Question;
import com.project.examportalbackend.models.Quiz;
import com.project.examportalbackend.services.QuestionService;
import com.project.examportalbackend.services.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin
@RestController
@RequestMapping("/api/question")
public class QuestionController {

    @Autowired
    private QuestionService questionService;

    @Autowired
    private QuizService quizService;

    @PostMapping("/")
    public ResponseEntity<?> addQuestion(@RequestBody Question question) {
        return ResponseEntity.ok(questionService.addQuestion(question));
    }

    @GetMapping("/")
    public ResponseEntity<?> getQuestions() {
        return ResponseEntity.ok(questionService.getQuestions());
    }

    @GetMapping("/{questionId}")
    public ResponseEntity<?> getQuestion(@PathVariable Long questionId) {
        return ResponseEntity.ok(questionService.getQuestion(questionId));
    }

    @GetMapping(value = "/", params = "quizId")
    public ResponseEntity<?> getQuestionsByQuiz(@RequestParam Long quizId) {
        Quiz quiz = quizService.getQuiz(quizId);
        List<Question> questions = quiz.getQuestions();
        return ResponseEntity.ok(questions);
    }

    @PutMapping("/{questionId}")
    public ResponseEntity<?> updateQuestion(@PathVariable Long questionId, @RequestBody
Question question) {
```

```
        if (questionService.getQuestion(questionId) != null) {
            return ResponseEntity.ok(questionService.updateQuestion(question));
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Question with id : "
+ String.valueOf(questionId) + ", doesn't exists");
    }

    @DeleteMapping("/{questionId}")
    public ResponseEntity<?> deleteQuestion(@PathVariable Long questionId) {
        questionService.deleteQuestion(questionId);
        return ResponseEntity.ok(true);
    }
}
```

CategoryController.java

```
package com.project.examportalbackend.controllers;

import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.services.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
@RequestMapping("/api/category")

public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    @PostMapping("/")
    public ResponseEntity<?> addCategory(@RequestBody Category category) {
        return ResponseEntity.ok(categoryService.addCategory(category));
    }

    @GetMapping("/")
    public ResponseEntity<?> getCategories() {
        return ResponseEntity.ok(categoryService.getCategories());
    }

    @GetMapping("/{categoryId}")
    public ResponseEntity<?> getCategory(@PathVariable Long categoryId) {
        return ResponseEntity.ok(categoryService.getCategory(categoryId));
    }

    @PutMapping("/{categoryId}")
    public ResponseEntity<?> updateCategory(@PathVariable Long categoryId, @RequestBody
Category category) {
        if (categoryService.getCategory(categoryId) != null) {
            return ResponseEntity.ok(categoryService.updateCategory(category));
        }
```

```
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Category with id : "
+ String.valueOf(categoryId) + ", doesn't exists");
    }

    @DeleteMapping("/{categoryId}")
    public ResponseEntity<?> deleteCategory(@PathVariable Long categoryId) {
        categoryService.deleteCategory(categoryId);
        return ResponseEntity.ok(true);
    }
}
```

AuthController.java

```
package com.project.examportalbackend.controllers;

import com.project.examportalbackend.models.LoginRequest;
import com.project.examportalbackend.models.LoginResponse;
import com.project.examportalbackend.models.User;
import com.project.examportalbackend.services.AuthService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
@RequestMapping("/api")
public class AuthController {

    @Autowired
    AuthService authService;


    @PostMapping("/register")
    public User registerUser(@RequestBody User user) throws Exception {
        return authService.registerUserService(user);
    }

    @PostMapping("/login")
    public LoginResponse loginUser(@RequestBody LoginRequest loginRequest) throws
Exception {
        return authService.loginUserService(loginRequest);
    }

}
```

JwtUtil.java

```
package com.project.examportalbackend.configurations;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;
```

```
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtUtil {
    private final String SECRET_KEY = "exam-portal";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 *
10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
    }
}
```

JwtRequestFilter.java

```
package com.project.examportalbackend.configurations;

import com.project.examportalbackend.services.implementation.UserDetailsServiceImpl;
import io.jsonwebtoken.ExpiredJwtException;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class JwtRequestFilter extends OncePerRequestFilter {
    @Autowired
    private UserDetailsServiceImpl userDetailsServiceImpl;

    @Autowired
    private JwtUtil jwtUtil;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {

        final String requestTokenHeader = request.getHeader("Authorization");

        String username = null;
        String jwtToken = null;

        if (requestTokenHeader != null && requestTokenHeader.startsWith("Bearer ")) {
            jwtToken = requestTokenHeader.substring(7);
            try {
                username = jwtUtil.extractUsername(jwtToken);
            } catch (IllegalArgumentException e) {
                System.out.println("Unable to get JWT Token");
            } catch (ExpiredJwtException e) {
                System.out.println("JWT Token has expired");
            }
        } else {
            System.out.println("JWT token does not start with Bearer");
        }

        if (username != null && SecurityContextHolder.getContext().getAuthentication()
== null) {

            UserDetails userDetails =
userDetailsServiceImpl.loadUserByUsername(username);
            if (jwtUtil.validateToken(jwtToken, userDetails)) {
                UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken
= new UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
                usernamePasswordAuthenticationToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToke
```

```
n);
            } else {
                System.out.println("Token is not valid");
            }
        }
        filterChain.doFilter(request, response);
    }
}
```

AuthService.java

```
package com.project.examportalbackend.services;

import com.project.examportalbackend.models.LoginRequest;
import com.project.examportalbackend.models.LoginResponse;
import com.project.examportalbackend.models.User;

public interface AuthService {
    User registerUserService(User user) throws Exception;

    LoginResponse loginUserService(LoginRequest loginRequest) throws Exception;
}
```

CategoryService.java

```
package com.project.examportalbackend.services;

import com.project.examportalbackend.models.Category;

import java.util.List;

public interface CategoryService {

    Category addCategory(Category category);

    List<Category> getCategories();

    Category getCategory(Long catId);

    Category updateCategory(Category category);

    void deleteCategory(Long categoryId);
}
```

QuestionService.java

```
package com.project.examportalbackend.services;

import com.project.examportalbackend.models.Question;
import com.project.examportalbackend.models.Quiz;
```

```
import java.util.List;

public interface QuestionService {

    Question addQuestion(Question question);

    List<Question> getQuestions();

    Question getQuestion(Long quesId);

    Question updateQuestion(Question question);

    void deleteQuestion(Long questionId);

    //Extra Methods
    List<Question> getQuestionsByQuiz(Quiz quiz);

}
```

QuizResultService.java

```
package com.project.examportalbackend.services;

import com.project.examportalbackend.models.QuizResult;

import java.util.List;

public interface QuizResultService {
    QuizResult addQuizResult(QuizResult quizResult);
    List<QuizResult> getQuizResults();

    List<QuizResult> getQuizResultsByUser(Long userId);
}
```

QuizService.java

```
package com.project.examportalbackend.services;

import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.models.Quiz;

import java.util.List;


public interface QuizService {

    Quiz addQuiz(Quiz quiz);

    List<Quiz> getQuizzes();

    Quiz getQuiz(Long quizId);

    Quiz updateQuiz(Quiz quiz);
```

```
    void deleteQuiz(Long quizId);

    // Extra methods
    List<Quiz> getQuizByCategory(Category category);
}
```

AuthServiceImpl.java

```
package com.project.examportalbackend.services.implementation;

import com.project.examportalbackend.configurations.JwtUtil;
import com.project.examportalbackend.models.LoginRequest;
import com.project.examportalbackend.models.LoginResponse;
import com.project.examportalbackend.models.Role;
import com.project.examportalbackend.models.User;
import com.project.examportalbackend.repository.RoleRepository;
import com.project.examportalbackend.repository.UserRepository;
import com.project.examportalbackend.services.AuthService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.HashSet;
import java.util.Set;

@Service
public class AuthServiceImpl implements AuthService {

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private UserDetailsServiceImpl userDetailsServiceImpl;
    @Autowired
    private JwtUtil jwtUtil;
    @Autowired
    private AuthenticationManager authenticationManager;

    @Override
    public User registerUserService(User user) throws Exception {
        User temp = userRepository.findByUsername(user.getUsername());
        if (temp != null) {
```

```
            throw new Exception("User Already Exists");
        } else {
            Role role = roleRepository.findById("USER").isPresent() ?
roleRepository.findById("USER").get() : null;
            Set<Role> userRoles = new HashSet<>();
            userRoles.add(role);
            user.setRoles(userRoles);
            user.setPassword(passwordEncoder.encode(user.getPassword()));
            return userRepository.save(user);
        }
    }

    public LoginResponse loginUserService(LoginRequest loginRequest) throws
Exception {

        authenticate(loginRequest.getUsername(), loginRequest.getPassword());
        UserDetails userDetails =
userDetailsServiceImpl.loadUserByUsername(loginRequest.getUsername());
        String token = jwtUtil.generateToken(userDetails);
        return new
LoginResponse(userRepository.findByUsername(loginRequest.getUsername()), token);
    }

    private void authenticate(String username, String password) throws Exception {
        try {
            authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(username, password));
        } catch (DisabledException e) {
            throw new Exception("USER_DISABLED", e);
        } catch (BadCredentialsException e) {
            throw new Exception("INVALID_CREDENTIALS", e);
        }
    }
}
```

CategoryServiceImpl.java

```
package com.project.examportalbackend.services.implementation;
import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.repository.CategoryRepository;
import com.project.examportalbackend.services.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CategoryServiceImpl implements CategoryService {

    @Autowired
    private CategoryRepository categoryRepository;

    @Override
    public Category addCategory(Category category) {
        return categoryRepository.save(category);
```

```
    }

    @Override
    public List<Category> getCategories() {
        return categoryRepository.findAll();
    }

    @Override
    public Category getCategory(Long catId) {
        return categoryRepository.findById(catId).isPresent() ?
categoryRepository.findById(catId).get() : null;
    }

    @Override
    public Category updateCategory(Category category) {
        return categoryRepository.save(category);
    }

    @Override
    public void deleteCategory(Long categoryId) {
        categoryRepository.delete(getCategory(categoryId));
    }
}
```

QuestionServiceImpl.java

```
package com.project.examportalbackend.services.implementation;

import com.project.examportalbackend.models.Question;
import com.project.examportalbackend.models.Quiz;
import com.project.examportalbackend.repository.QuestionRepository;
import com.project.examportalbackend.services.QuestionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class QuestionServiceImpl implements QuestionService {

    @Autowired
    QuestionRepository questionRepository;

    @Override
    public Question addQuestion(Question question) {
        return questionRepository.save(question);
    }

    @Override
    public List<Question> getQuestions() {
        return questionRepository.findAll();
    }
```

```
    @Override
    public Question getQuestion(Long quesId) {
        return questionRepository.findById(quesId).isPresent() ?
questionRepository.findById(quesId).get() : null;
    }

    @Override
    public Question updateQuestion(Question question) {
        return questionRepository.save(question);
    }

    @Override
    public void deleteQuestion(Long questionId) {
        questionRepository.delete(getQuestion(questionId));
    }

    @Override
    public List<Question> getQuestionsByQuiz(Quiz quiz) {
        return questionRepository.findByQuiz(quiz);
    }
}
```

QuizResultServiceImpl.java

```
package com.project.examportalbackend.services.implementation;

import com.project.examportalbackend.models.QuizResult;
import com.project.examportalbackend.repository.QuizResultRepository;
import com.project.examportalbackend.services.QuizResultService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class QuizResultServiceImpl implements QuizResultService {

    @Autowired
    private QuizResultRepository quizResultRepository;

    @Override
    public QuizResult addQuizResult(QuizResult quizResult) {
        return quizResultRepository.save(quizResult);
    }

    @Override
    public List<QuizResult> getQuizResults() {
        return quizResultRepository.findAll();
    }

    @Override
    public List<QuizResult> getQuizResultsByUser(Long userId) {
        return quizResultRepository.findByUserId(userId);
```

```
    }
}
```

QuizServiceImpl.java

```
package com.project.examportalbackend.services.implementation;

import com.project.examportalbackend.models.Category;
import com.project.examportalbackend.models.Quiz;
import com.project.examportalbackend.repository.QuizRepository;
import com.project.examportalbackend.services.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class QuizServiceImpl implements QuizService {

    @Autowired
    public QuizRepository quizRepository;

    @Override
    public Quiz addQuiz(Quiz quiz) {
        return quizRepository.save(quiz);
    }

    @Override
    public List<Quiz> getQuizzes() {
        return quizRepository.findAll();
    }

    @Override
    public Quiz getQuiz(Long quizId) {
        return quizRepository.findById(quizId).isPresent() ?
quizRepository.findById(quizId).get() : null;
    }

    @Override
    public Quiz updateQuiz(Quiz quiz) {
        return quizRepository.save(quiz);
    }

    @Override
    public void deleteQuiz(Long quizId) {
        quizRepository.deleteById(quizId);
    }

    @Override
    public List<Quiz> getQuizByCategory(Category category) {
        return quizRepository.findByCategory(category);
    }
}
```

UserDetailsServiceImpl.java

```
package com.project.examportalbackend.services.implementation;

import com.project.examportalbackend.models.User;
import com.project.examportalbackend.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
        if (user != null)
            return user;
        throw new UsernameNotFoundException("User Not Found!");
    }
}
```

application.properties

```
spring.application.name=exam-portal-app
server.port=8081

spring.datasource.url=jdbc:mysql://localhost:3306/exam-
portal?createDatabaseIfNotExist=true&allowPublicKeyRetrieval=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=root@

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto = update

spring.jpa.properties.hibernate.show_sql=true
```

11.2 Frontend

App.js

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import "./App.css";
import Header from "./components/Header";
import AdminAddCategoryPage from "./pages/admin/categories/AdminAddCategoryPage";
import AdminCategoriesPage from "./pages/admin/categories/AdminCategoriesPage";
```

```
import AdminUpdateCategoryPage from
"./pages/admin/categories/AdminUpdateCategoryPage";
import AdminProfilePage from "./pages/admin/AdminProfilePage";
import LoginPage from "./pages/LoginPage";
import RegisterPage from "./pages/RegisterPage";
import AdminQuizzesPage from "./pages/admin/quizzes/AdminQuizzesPage";
import AdminAddQuiz from "./pages/admin/quizzes/AdminAddQuiz";
import AdminUpdateQuiz from "./pages/admin/quizzes/AdminUpdateQuiz";
import AdminQuestionsPage from "./pages/admin/questions/AdminQuestionsPage";
import AdminAddQuestionsPage from "./pages/admin/questions/AdminAddQuestionsPage";
import AdminUpdateQuestionPage from
"./pages/admin/questions/AdminUpdateQuestionPage";
import UserProfilePage from "./pages/users/UserProfilePage";
import UserQuizzesPage from "./pages/users/UserQuizzesPage";
import UserQuizManualPage from "./pages/users/UserQuizManualPage";
import UserQuestionsPage from "./pages/users/UserQuestionsPage";
import UserQuizResultPage from "./pages/users/UserQuizResultPage";

const App = () => {
  return (
    <Router>
      <Header />
      <Routes>
        <Route path="/" element={<LoginPage />} />
        <Route path="/login" element={<LoginPage />} />
        <Route path="/register" element={<RegisterPage />} />
        <Route path="/adminProfile" element={<AdminProfilePage />} />
        <Route path="/adminCategories" element={<AdminCategoriesPage />} />
        <Route path="/adminAddCategory" element={<AdminAddCategoryPage />} />
        <Route
          path="/adminUpdateCategory/:catId"
          element={<AdminUpdateCategoryPage />}
        />
        <Route path="/adminQuizzes" element={<AdminQuizzesPage />} />
        <Route path="/adminAddQuiz" element={<AdminAddQuiz />} />
        <Route path="/adminUpdateQuiz/:quizId" element={<AdminUpdateQuiz />} />
        <Route path="/adminQuestions" element={<AdminQuestionsPage />} />
        <Route path="/adminAddQuestion" element={<AdminAddQuestionsPage />} />
        <Route
          path="/adminUpdateQuestion/:quesId"
          element={<AdminUpdateQuestionPage />}
        />
        <Route path="/profile" element={<UserProfilePage />} />
        <Route path="/quizzes" element={<UserQuizzesPage />} />
        <Route path="/quiz/*" element={<UserQuizzesPage />} />
        <Route path="/quizManual/" element={<UserQuizManualPage />} />
        <Route path="/questions/" element={<UserQuestionsPage />} />
        <Route path="/quizResults/" element={<UserQuizResultPage />} />
      </Routes>
    </Router>
  );
};

export default App;
```

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { Provider } from "react-redux";
import store from "./store";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);

reportWebVitals();
```

AdminAddCategoryPage.js

```
import React, { useState } from "react";
import "./AdminAddCategoryPage.css";
import { Button, Form } from "react-bootstrap";
import { useDispatch } from "react-redux";
import * as categoriesConstants from "../../../constants/categoriesConstants";
import FormContainer from "../../../components/FormContainer";
import Sidebar from "../../../components/Sidebar";
import {
  addCategory,
  fetchCategories,
} from "../../../actions/categoriesActions";
import swal from "sweetalert";
import { useNavigate } from "react-router-dom";

const AdminAddCategoryPage = () => {
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState("");
  const token = JSON.parse(localStorage.getItem("jwtToken"));

  const dispatch = useDispatch();
  const navigate = useNavigate();

  const submitHandler = (e) => {
    e.preventDefault();
    const category = { title: title, description: description };
    addCategory(dispatch, category, token).then((data) => {
      if (data.type === categoriesConstants.ADD_CATEGORY_SUCCESS) {
        swal("Category Added!", `${title} succesfully added`, "success");
      } else {
        swal("Category Not Added!", `${title} not added`, "error");
      }
```

```
      // navigate("/adminCategories");
    });
  };

  return (
    <div className="adminAddCategoryPage__container">
      <div className="adminAddCategoryPage__sidebar">
        <Sidebar />
      </div>
      <div className="adminAddCategoryPage__content">
        <FormContainer>
          <h2>Add Category</h2>
          <Form onSubmit={submitHandler}>
            <Form.Group className="my-3" controlId="title">
              <Form.Label>Title</Form.Label>
              <Form.Control
                type="text"
                placeholder="Enter Category Title"
                value={title}
                onChange={(e) => {
                  setTitle(e.target.value);
                }}
              ></Form.Control>
            </Form.Group>

            <Form.Group className="my-3" controlId="description">
              <Form.Label>Description</Form.Label>
              <Form.Control
                style={{ textAlign: "top" }}
                as="textarea"
                rows="5"
                type="text"
                placeholder="Enter Category Description"
                value={description}
                onChange={(e) => {
                  setDescription(e.target.value);
                }}
              ></Form.Control>
            </Form.Group>

            <Button
              className="my-3 adminAddCategoryPage__content--button"
              type="submit"
              variant=""
            >
              Add
            </Button>
          </Form>
        </FormContainer>
      </div>
    </div>
  );
};

export default AdminAddCategoryPage;
```

AdminUpdateCategoryPage.js

```
import React, { useState } from "react";
import "./AdminUpdateCategoryPage.css";
import { Button, Form } from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
import swal from "sweetalert";
import { useParams } from "react-router-dom";
import * as categoriesConstants from "../../../constants/categoriesConstants";
import FormContainer from "../../../components/FormContainer";
import Sidebar from "../../../components/Sidebar";
import { updateCategory } from "../../../actions/categoriesActions";
import { useNavigate } from "react-router-dom";

const AdminUpdateCategoryPage = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const params = useParams();
  const catId = params.catId;

  const oldCategory = useSelector((state) =>
    state.categoriesReducer.categories.filter((cat) => cat.catId == catId)
  )[0];
  const [title, setTitle] = useState(oldCategory ? oldCategory.title : "");
  const [description, setDescription] = useState(
    oldCategory ? oldCategory.description : ""
  );
  const token = JSON.parse(localStorage.getItem("jwtToken"));

  const submitHandler = (e) => {
    e.preventDefault();
    const category = { catId: catId, title: title, description: description };
    updateCategory(dispatch, category, token).then((data) => {
      if (data.type === categoriesConstants.UPDATE_CATEGORY_SUCCESS) {
        swal("Category Updated!", `${title} succesfully updated`, "success");
      } else {
        swal("Category Not Updated!", `${title} not updated`, "error");
      }
    });
    navigate("/adminCategories");
  };

  return (
    <div className="adminUpdateCategoryPage__container">
      <div className="adminUpdateCategoryPage__sidebar">
        <Sidebar />
      </div>
      <div className="adminUpdateCategoryPage__content">
        <FormContainer>
          <h2>Update Category</h2>
          <Form onSubmit={submitHandler}>
            <Form.Group className="my-3" controlId="title">
              <Form.Label>Title</Form.Label>
```

```
              <Form.Control
                type="text"
                placeholder="Enter Category Title"
                value={title}
                onChange={(e) => {
                  setTitle(e.target.value);
                }}
              ></Form.Control>
            </Form.Group>

            <Form.Group className="my-3" controlId="description">
              <Form.Label>Description</Form.Label>
              <Form.Control
                style={{ textAlign: "top" }}
                as="textarea"
                rows="5"
                type="text"
                placeholder="Enter Category Description"
                value={description}
                onChange={(e) => {
                  setDescription(e.target.value);
                }}
              ></Form.Control>
            </Form.Group>

            <Button
              className="my-3 adminUpdateCategoryPage__content--button"
              type="submit"
              variant=""
            >
              Update
            </Button>
          </Form>
        </FormContainer>
      </div>
    </div>
  );
};

export default AdminUpdateCategoryPage;
```

AdminQuestionPage.js

```
import React, { useEffect, useState } from "react";
import "./AdminQuestionsPage.css";

import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import { Button } from "react-bootstrap";
import { fetchQuestionsByQuiz } from "../../../actions/questionsActions";
import Sidebar from "../../../components/Sidebar";
import Question from "../../../components/Question";
import Loader from "../../../components/Loader";
```

```
const AdminQuestionsPage = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const urlParams = new URLSearchParams(window.location.search);
  const quizId = urlParams.get("quizId");
  const quizTitle = urlParams.get("quizTitle");

  const questionsReducer = useSelector((state) => state.questionsReducer);
  const [questions, setQuestions] = useState(questionsReducer.questions);
  const token = JSON.parse(localStorage.getItem("jwtToken"));
  let answers = {};

  const addNewQuestionHandler = () => {
    navigate(`/adminAddQuestion/?quizId=${quizId}`);
  };

  useEffect(() => {
    if (!localStorage.getItem("jwtToken")) navigate("/");
  }, []);

  useEffect(() => {
    fetchQuestionsByQuiz(dispatch, quizId, token).then((data) =>
      setQuestions(data.payload)
    );
  }, []);

  return (
    <div className="adminQuestionsPage__container">
      <div className="adminQuestionsPage__sidebar">
        <Sidebar />
      </div>
      <div className="adminQuestionsPage__content">
        <h2>{`Questions : ${quizTitle}`}</h2>
        <Button
          className="adminQuestionsPage__content--button"
          onClick={addNewQuestionHandler}
        >
          Add Question
        </Button>
        {questions ? (
          questions.map((q, index) => {
            return (
              <Question
                key={index}
                number={index + 1}
                answers={answers}
                question={q}
                isAdmin={true}
              />
            );
          })
        ) : (
          <Loader />
        )}
```

```
      </div>
    </div>
  );
};

export default AdminQuestionsPage;
```

AdminProfilePage.js

```
import React, { useEffect } from "react";
import { Table } from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import Sidebar from "../../components/Sidebar";
import "./AdminProfilePage.css";
import Image from "react-bootstrap/Image";
import { fetchCategories } from "../../actions/categoriesActions";
import { fetchQuizzes } from "../../actions/quizzesActions";

const AdminProfilePage = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const loginReducer = useSelector((state) => state.loginReducer);
  const user = loginReducer.user;
  const token = JSON.parse(localStorage.getItem("jwtToken"));

  useEffect(() => {
    if (!localStorage.getItem("jwtToken")) navigate("/");
  }, []);

  useEffect(() => {
    fetchCategories(dispatch, token);
  }, [dispatch]);

  useEffect(() => {
    fetchQuizzes(dispatch, token);
  }, [dispatch]);

  return (
    <div className="adminProfilePage__container">
      <div className="adminProfilePage__sidebar">
        <Sidebar />
      </div>
      <div className="adminProfilePage__content">
        <Image
          className="adminProfilePage__content--profilePic"
          width="20%"
          height="20%"
          roundedCircle
          src="images/user.png"
        />

        <Table bordered className="adminProfilePage__content--table">
```

```
          <tbody>
            <tr>
              <td>Name</td>
              <td>{`${user.firstName} ${user.lastName}`}</td>
            </tr>
            <tr>
              <td>Username</td>
              <td>{user.username}</td>
            </tr>
            <tr>
              <td>Phone</td>
              <td>{user.phoneNumber}</td>
            </tr>
            <tr>
              <td>Role</td>
              <td>{user.roles[0].roleName}</td>
            </tr>
            <tr>
              <td>Status</td>
              <td>{`${user.active}`}</td>
            </tr>
          </tbody>
        </Table>
      </div>
    </div>
  );
};

export default AdminProfilePage;
```

UserProfilePage.js

```
import React, { useEffect } from "react";
import { Table } from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import Image from "react-bootstrap/Image";
import { fetchCategories } from "../../actions/categoriesActions";
import { fetchQuizzes } from "../../actions/quizzesActions";
import SidebarUser from "../../components/SidebarUser";
import "./UserProfilePage.css";
const UserProfilePage = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const loginReducer = useSelector((state) => state.loginReducer);
  const user = loginReducer.user;
  const token = JSON.parse(localStorage.getItem("jwtToken"));

  useEffect(() => {
    fetchCategories(dispatch, token);
  }, [dispatch]);

  useEffect(() => {
```

```
    fetchQuizzes(dispatch, token);
  }, [dispatch]);

  useEffect(() => {
    if (!localStorage.getItem("jwtToken")) navigate("/");
  }, []);

  return (
    <div className="userProfilePage__container">
      <div className="userProfilePage__sidebar">
        <SidebarUser />
      </div>
      {user && (
        <div className="userProfilePage__content">
          <Image
            className="userProfilePage__content--profilePic"
            width="20%"
            height="20%"
            roundedCircle
            src="images/user.png"
          />

          <Table bordered className="userProfilePage__content--table">
            <tbody>
              <tr>
                <td>Name</td>
                <td>{`${user.firstName} ${user.lastName}`}</td>
              </tr>
              <tr>
                <td>Username</td>
                <td>{user.username}</td>
              </tr>
              <tr>
                <td>Phone</td>
                <td>{user.phoneNumber}</td>
              </tr>
              <tr>
                <td>Role</td>
                <td>{user.roles[0].roleName}</td>
              </tr>
              <tr>
                <td>Account Status</td>
                <td>{`${user.enabled}`}</td>
              </tr>
            </tbody>
          </Table>
        </div>
      )}
    </div>
  );
};
export default UserProfilePage;
```

CHAPTER – 12

TESTING

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

12.1 Levels of software testing

Software level testing can be majorly classified into 4 levels:

i. **Unit Testing:** A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

ii. **Integration Testing:** A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

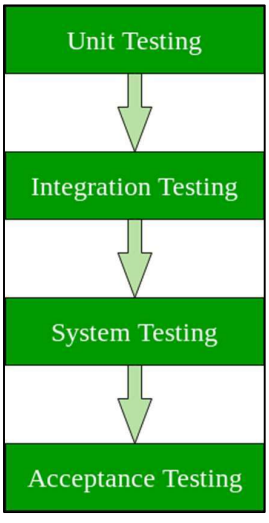


Fig 11.1: Levels of Testing

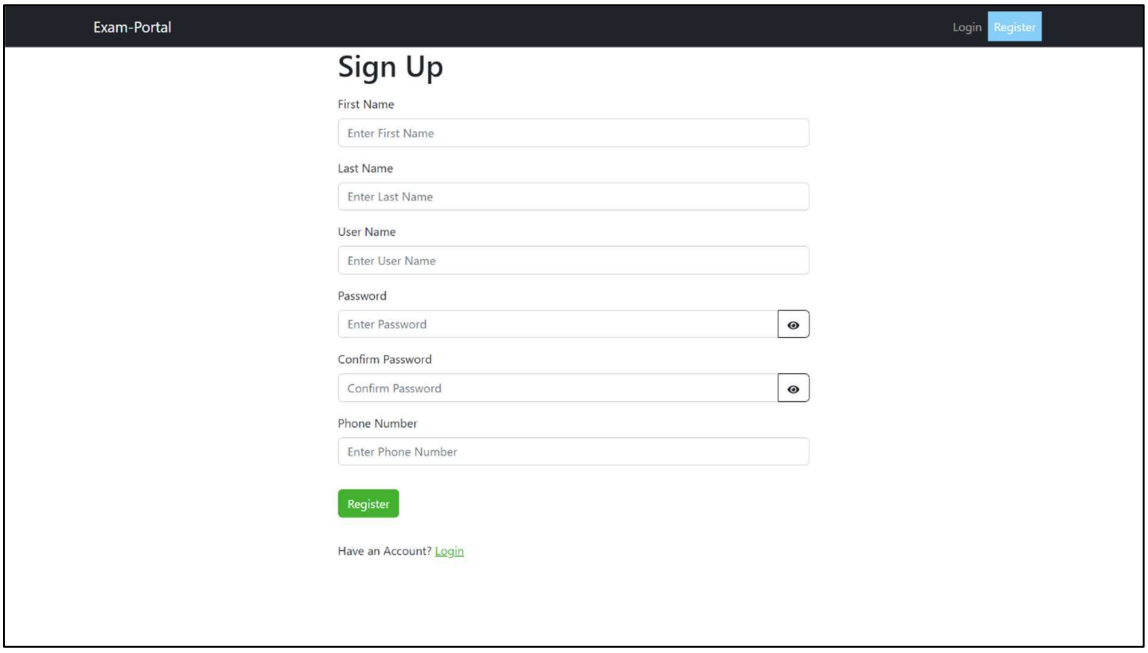
iii. **System Testing:** A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

iv. **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

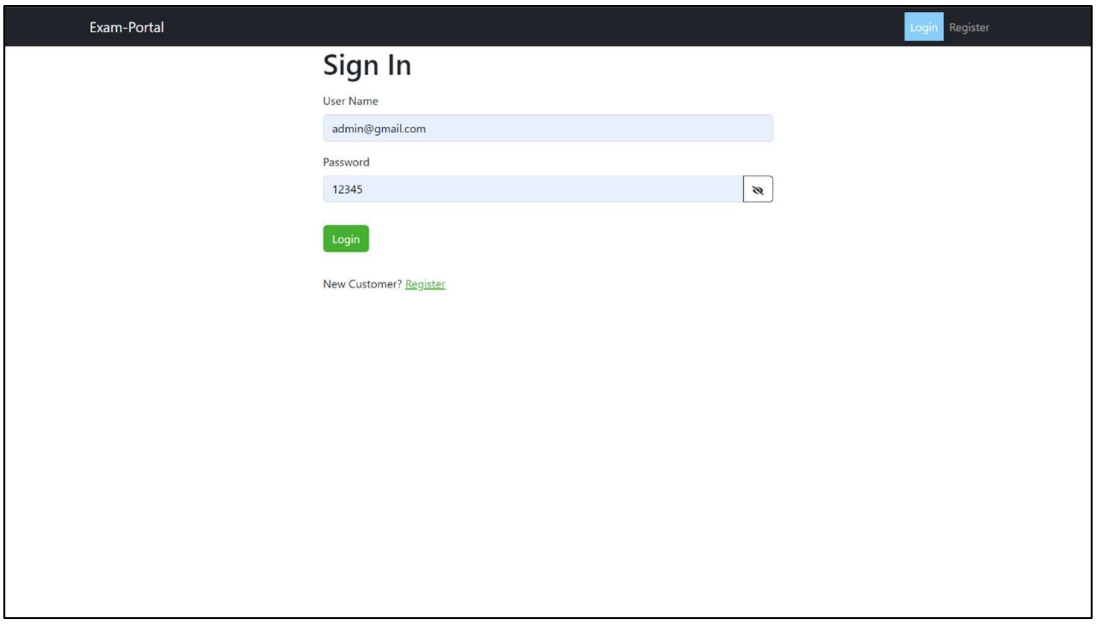
CHAPTER – 13

SCREENSHOTS

i. Registration Page



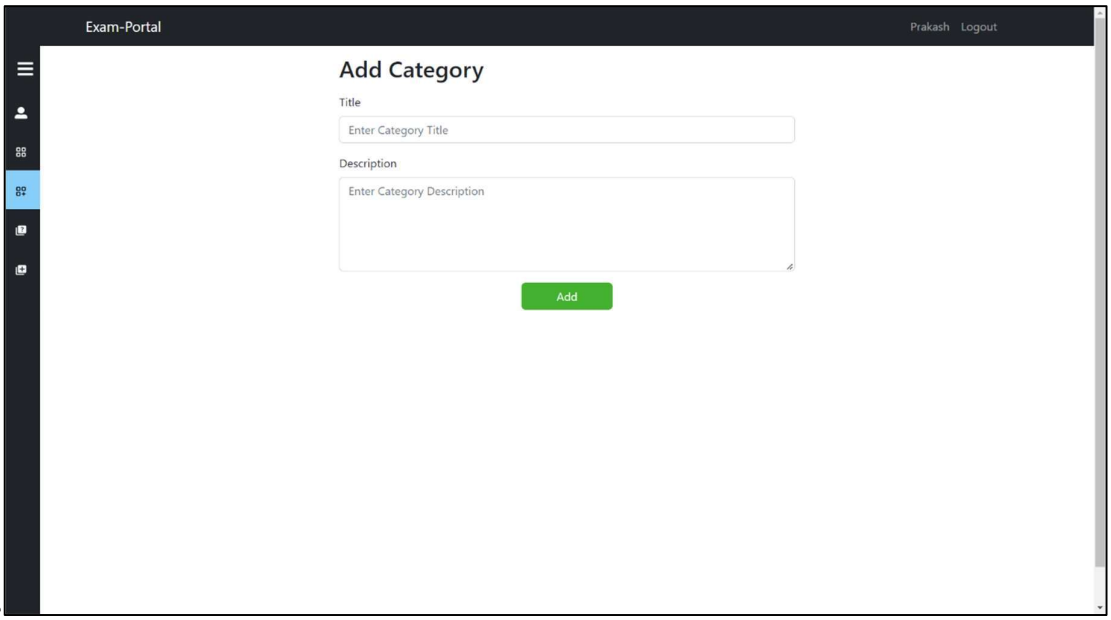
ii. Login Page



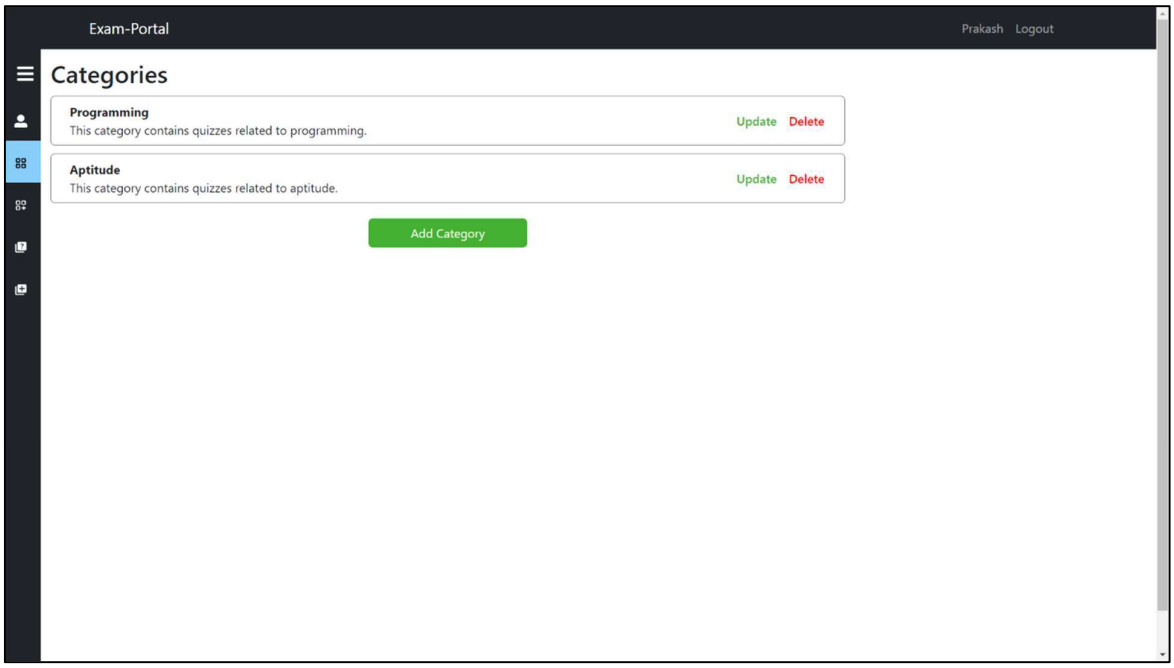
iii. Profile Page

iv. Add Category Page (Admin)

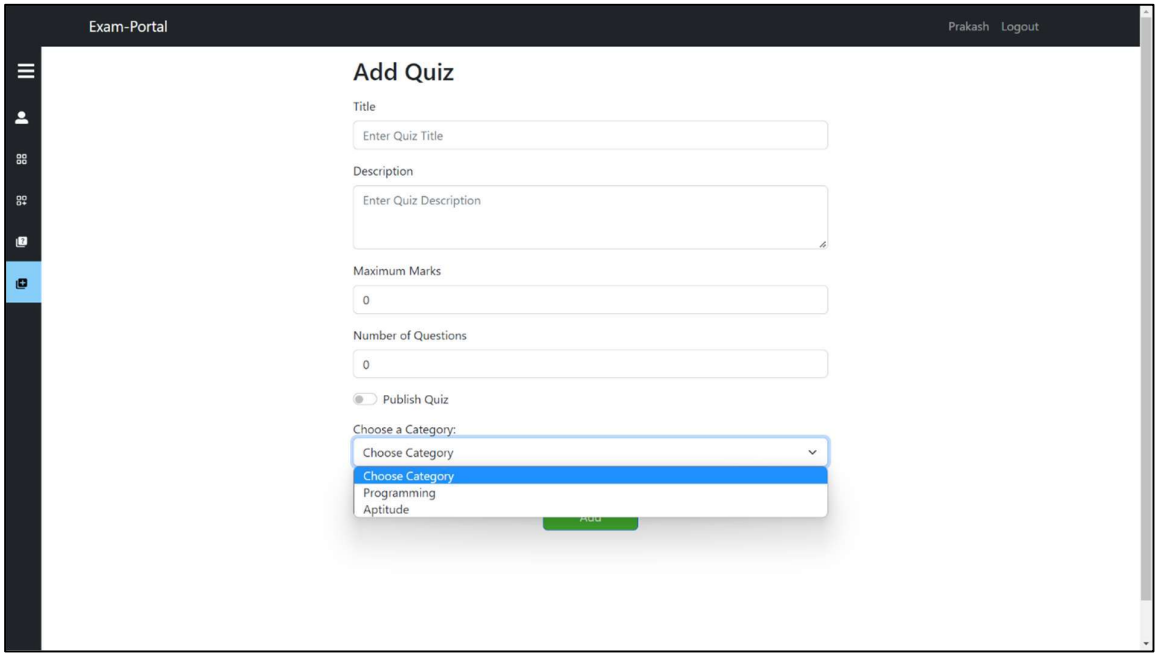
e



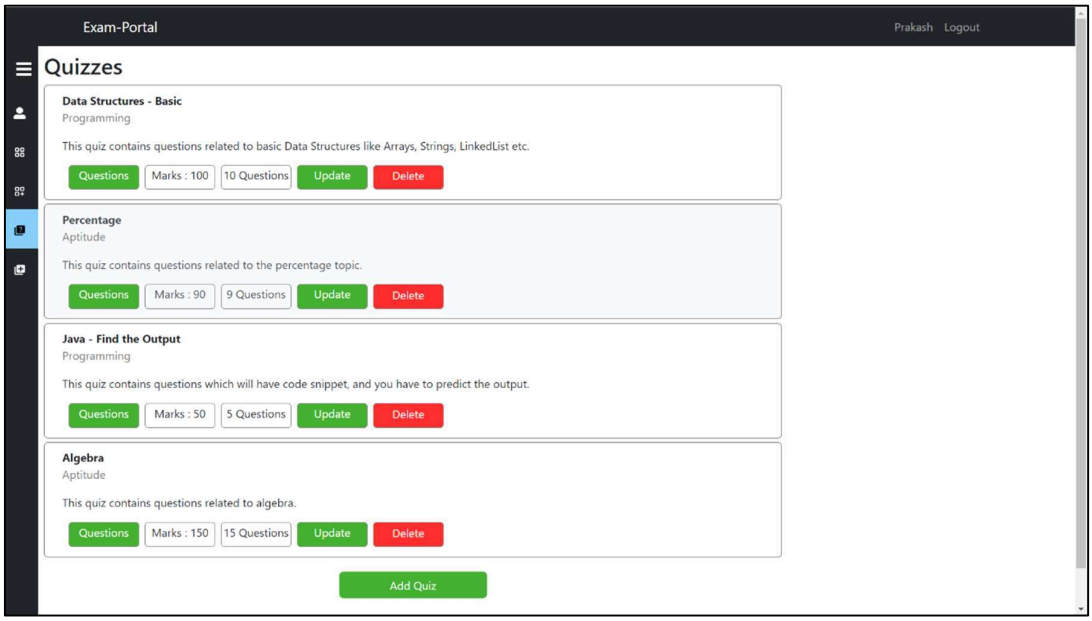
v. Categories Page (Admin)



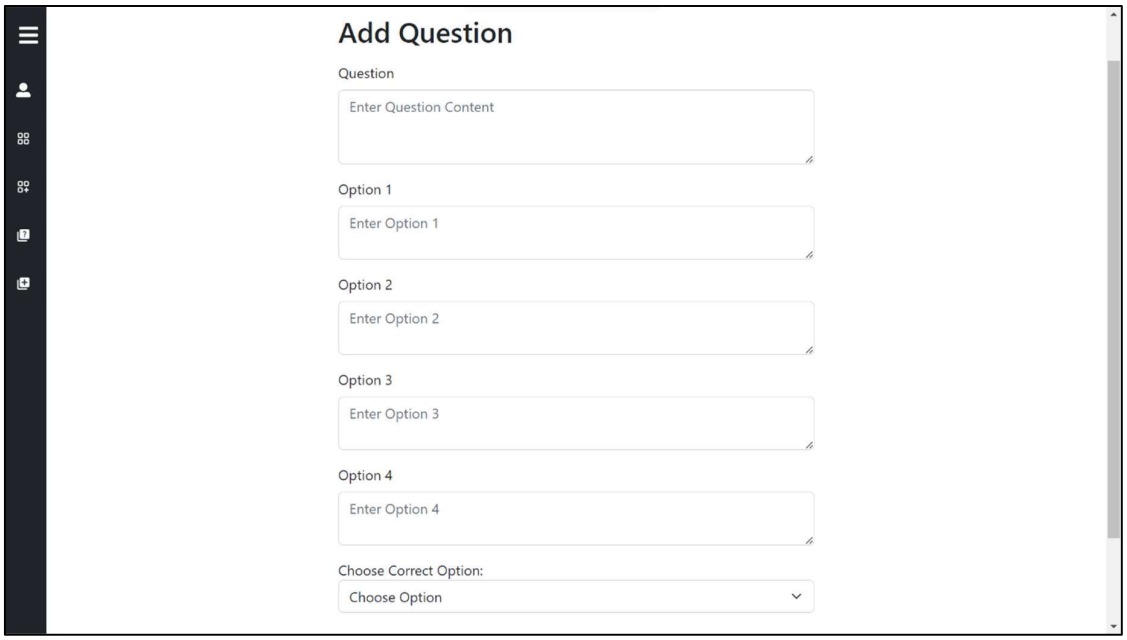
vi. Add Quiz Page (Admin)



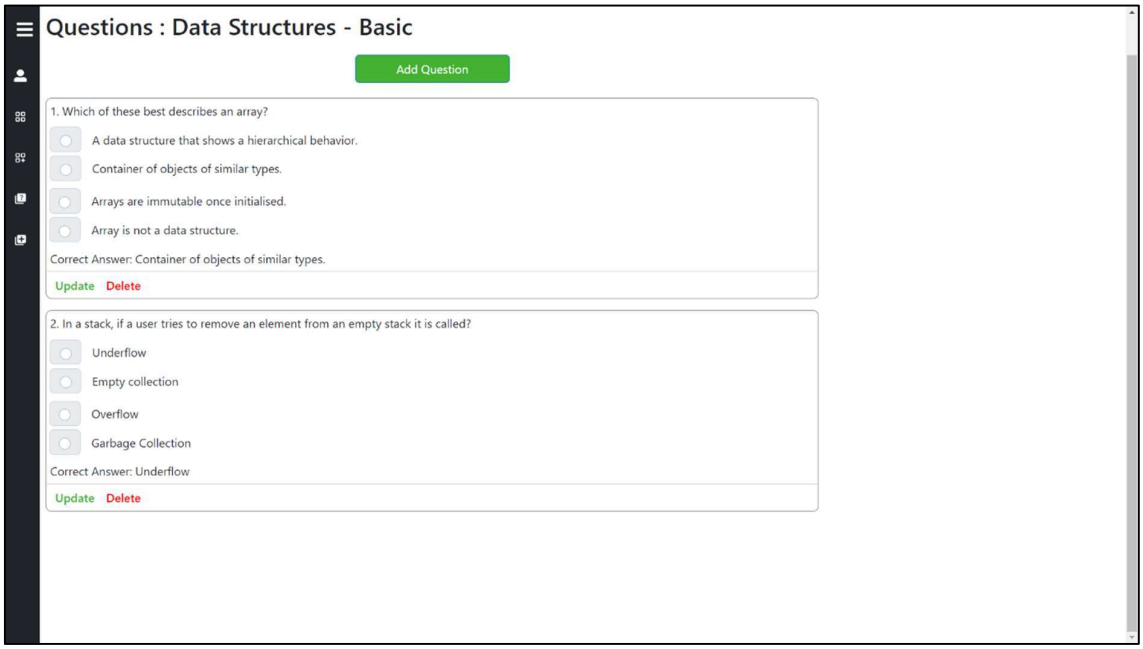
vii. Quiz Page (Admin)



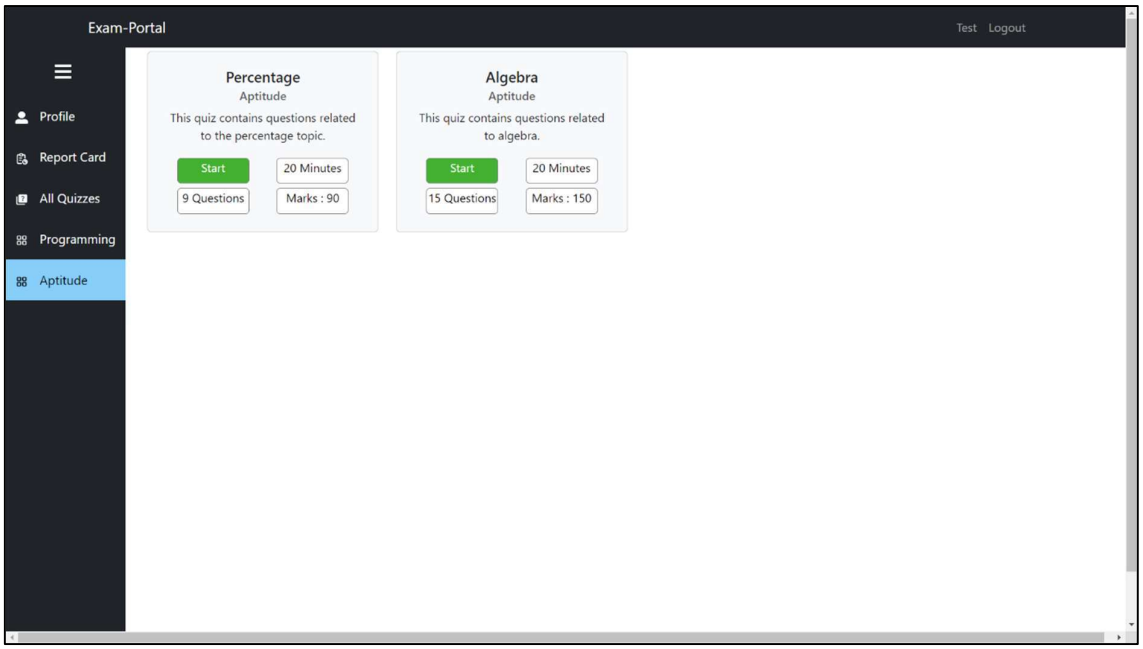
viii. Add Question Page (Admin)



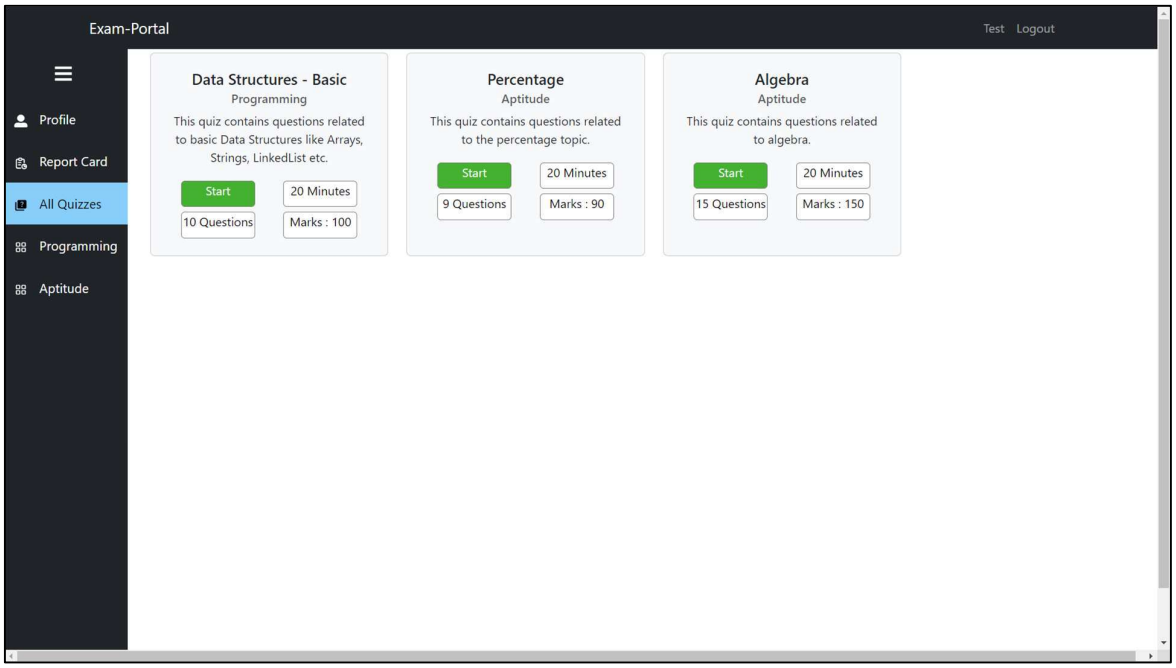
ix. Question Page (Admin)



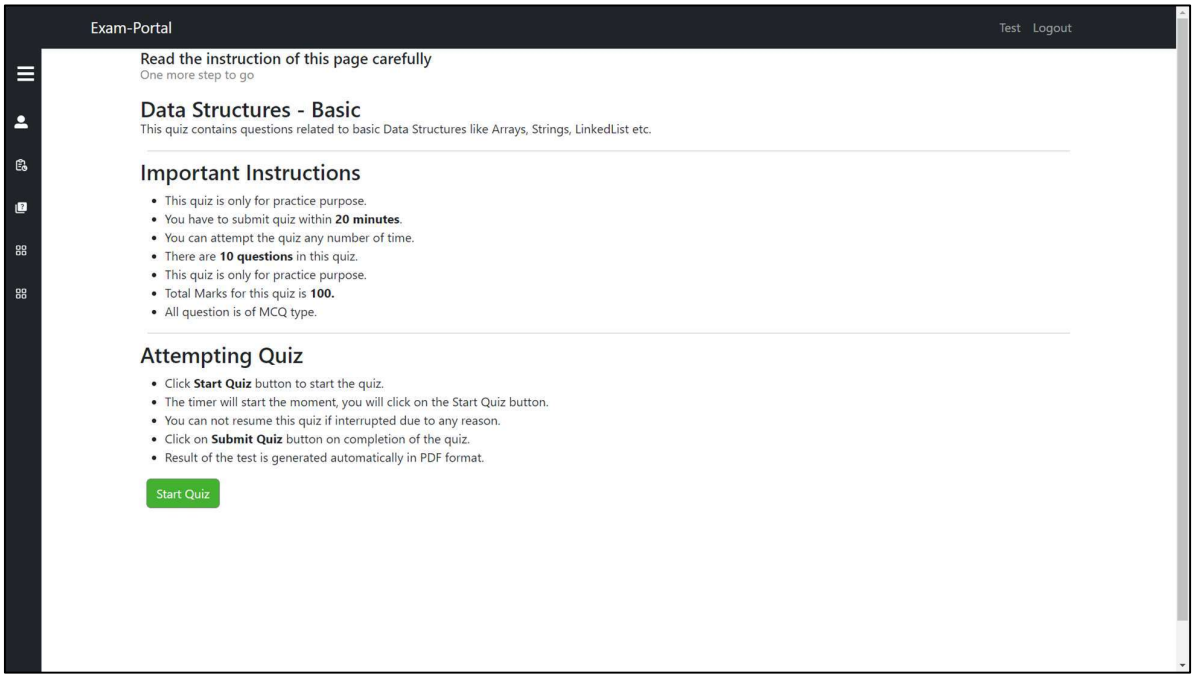
x. Categories Page (User)



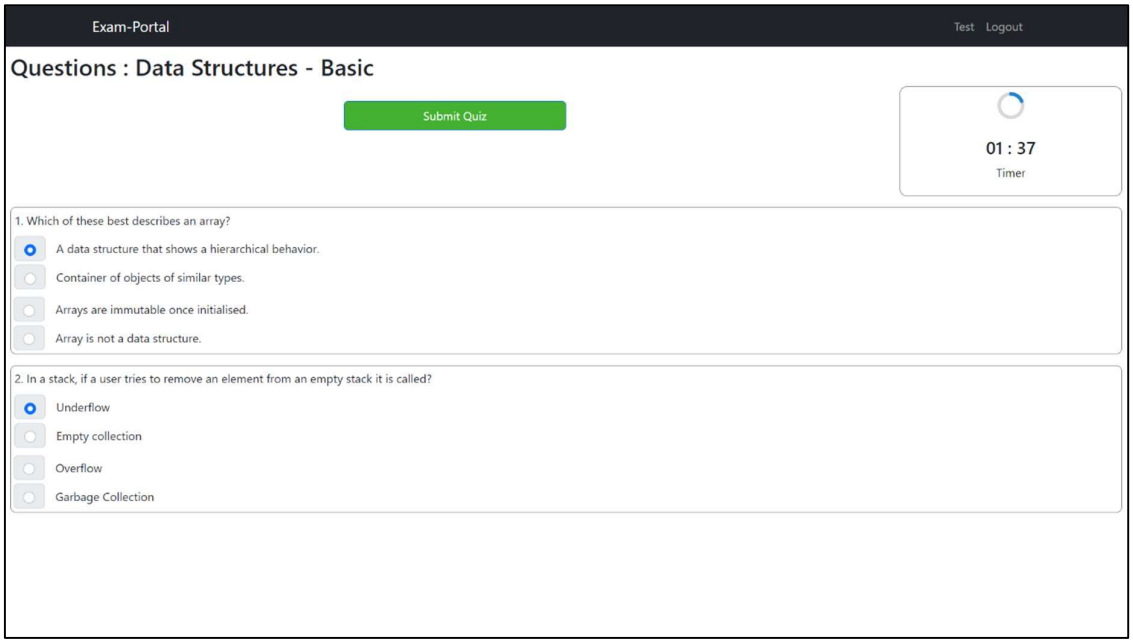
xi. All Quizzes Page (User)



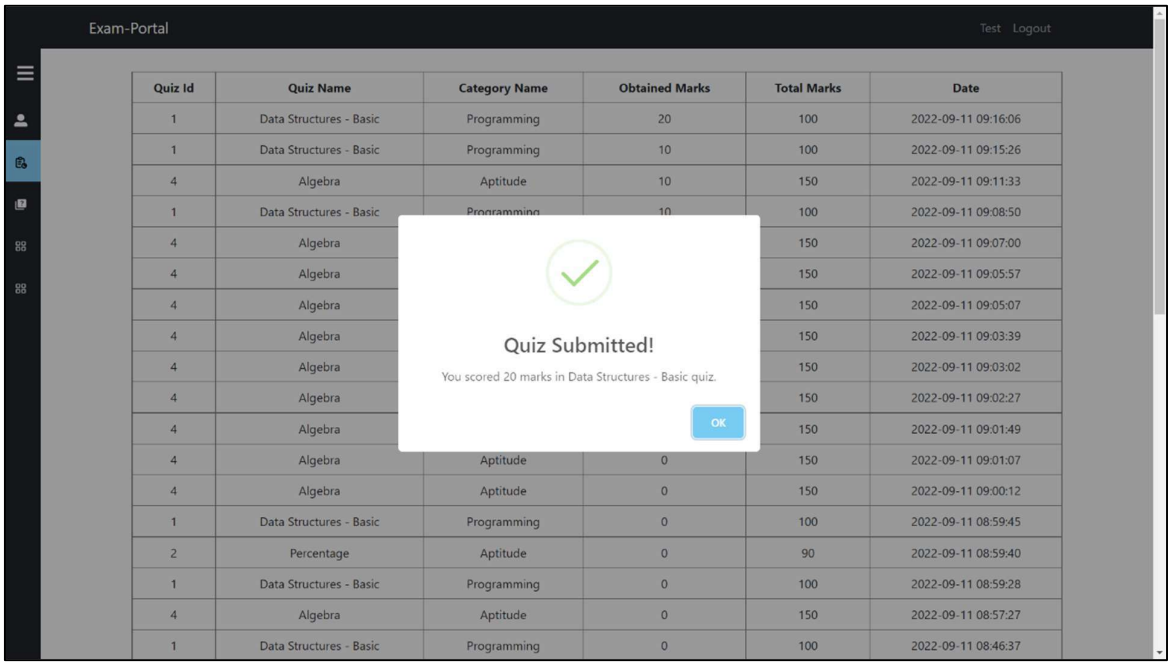
xii. Quiz Manual Page (User)



xiii. Question Page (User)

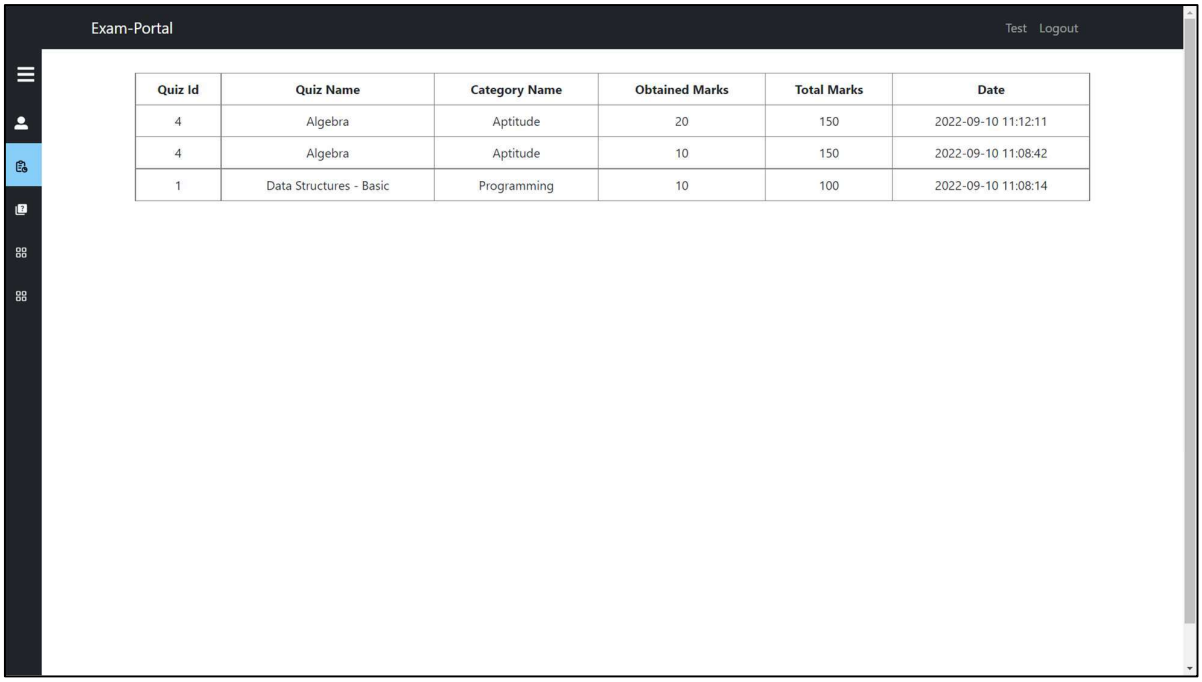


xiv. Quiz Submitted Alert (User)



Quiz Id	Quiz Name	Category Name	Obtained Marks	Total Marks	Date
1	Data Structures - Basic	Programming	20	100	2022-09-11 09:16:06
1	Data Structures - Basic	Programming	10	100	2022-09-11 09:15:26
4	Algebra	Aptitude	10	150	2022-09-11 09:11:33
1	Data Structures - Basic	Programming	10	100	2022-09-11 09:08:50
4	Algebra			150	2022-09-11 09:07:00
4	Algebra			150	2022-09-11 09:05:57
4	Algebra			150	2022-09-11 09:05:07
4	Algebra			150	2022-09-11 09:03:39
4	Algebra			150	2022-09-11 09:03:02
4	Algebra			150	2022-09-11 09:02:27
4	Algebra			150	2022-09-11 09:01:49
4	Algebra	Aptitude	0	150	2022-09-11 09:01:07
4	Algebra	Aptitude	0	150	2022-09-11 09:00:12
1	Data Structures - Basic	Programming	0	100	2022-09-11 08:59:45
2	Percentage	Aptitude	0	90	2022-09-11 08:59:40
1	Data Structures - Basic	Programming	0	100	2022-09-11 08:59:28
4	Algebra	Aptitude	0	150	2022-09-11 08:57:27
1	Data Structures - Basic	Programming	0	100	2022-09-11 08:46:37

xv. Result Page (User)



Quiz Id	Quiz Name	Category Name	Obtained Marks	Total Marks	Date
4	Algebra	Aptitude	20	150	2022-09-10 11:12:11
4	Algebra	Aptitude	10	150	2022-09-10 11:08:42
1	Data Structures - Basic	Programming	10	100	2022-09-10 11:08:14

CHAPTER – 14

FUTURE SCOPE AND CONCLUSION

14.1 Future Scope

Steady improvement in innovation has prompted a fast development of the evaluation business. Large numbers of the training organizations and colleges are building up their interest in web tests through online assessment programming for their understudies rather than pen and paper-based test. It demonstrates that online assessment software is the future of evaluation techniques. We have made an online assessment framework, and we will be dealing with it in the future and will make a great deal of improvement like-

i. Adding the screen-sharing & live camera feature to enhance the security and prevent any kind of suspicious activities.
ii. Voice acknowledgment.
iii. Fingerprint validation.
iv. Facial recognition acknowledgment.

14.2 Conclusion

This paper introduces a secure system for internet-based exam invigilation, whose work maintains academic integrity in e-learning. This web-based system is convenient and user-friendly for utilizing, from the candidate's perspective, as it only needs one laptop or smartphone. The key concept is to minimize the amount of traditional, pen-paper form of examination and convert all forms of documentation to digital form. It was observed that the data/information required can be obtained with much simplicity, accuracy and in a secure manner, through the computerized system. The user with limited knowledge about the computer may also access the system with ease. The system also yields detailed result as required.

REFERENCES

[1] M. Fagbol; Baale Adebisi & Oke A. (2013). Computer Based Test (CBT) System for University Academic Enterprise Examination

[2] Ayo, Charles & Akinyemi, I.O. & Adebiyi, Ayodele & Ekong, Uyinomen. (2007). The prospects of Eexamination implementation in Nigeria. The Turkish Online Journal of Distance Education.

[3] Wei L., Cong Z., & Zhiwei Y. (2010). Fingerprint based identity authentication for online examination system. 2010 Second International Workshop on Education Technology and Computer Science. Doi:10.1109/etcs.2010.409

[4] M. Z. Rashad; M. S. Kandil; A. E. Hassan, and M. A. Zaher, "An Arabic Web-Based Exam Management System", International Journal of Electrical and Computer Sciences (IJECS-IJENS), Vol. 10, No. 1, pp. 35-41, February 2010. Doi:10.1109/etcs.2010.413

[5] Arvind Singh; Niraj Shirke; Kiran Shette, "Online Examination Portal," 2011. Doi:10.1109/etcs.2010.415

[6] Conejo R., Guzmán E., & Trella M. (2015). The siette automatic assessment environment. International Journal of Artificial Intelligence in Education, 26(1), 270-292. Doi:10.1007/s40593-015-0078-4