Preprint · August 2022

E - PARIKSHA (Online Exam Portal)- for skill Development of Students

A Project Report Submitted in partial fulfillment of the degree of

Master of Computer Application

By

Vivek Kumar(19MCMC13)
Badal Kumar (19MCMC43)

School of Computer and Information Sciences University of Hyderabad, Gachibowli
Hyderabad - 500046, India

JUNE, 2022

CERTIFICATE

This is to certify that the Project Report entitled "**E - PARIKSHA (Online Exam Portal)**" submitted by **Vivek Kumar** bearing **Reg.No. 19MCMC13** and **Badal Kumar** bearing **Reg.No. 19MCMC43** in partial fulfillment of the requirements for the award of Master of Computer Application is a bonafide work carried out by them under my supervi-

sion and guidance.

The Project Report has not been submitted previously in part or in full to this or any other

University or Institution for the award of any degree or diploma.

Dr. Nagamani M
Internal Supervisor
Assistant Professor
University of Hyderabad

Ramashankar Darivemula
External Supervisor
Software Engineer
Jocata, Hyderabad

Prof. Chakarvarthy Bhagvati, **Dean**,

School of Computer and Information Sciences,
University of Hyderabad

DECLARATION

We, **Vivek Kumar** and **Badal Kumar** hereby declare that this dissertation entitled "**E -PARIKSHA (Online Exam Portal)**" submitted by us under the guidance and supervision of **Dr. Nagamani M** is a bonafide work. We also declare that it has not been submitted previously in part or in full to this or any other University or Institution for the award of

any degree or diploma.

Date:

VIVEK KUMAR
Reg. No.: 19MCMC13

Signature of the Student

BADAL KUMAR
Reg. No.: 19MCMC43

Signature of the Student

Acknowledgments

The projects consumed a huge amount of work, research and dedication. These projects would not have been done successfully if we did not had the support of many individuals and organizations. Therefore, We would like to extend our sincere gratitude to all of them. We feel great pleasure to express our deep sense of gratitude to **Dr Nagamani M**, Our in-ternal supervisor who showed his continuous support and guidance, throughout my work.

We are grateful to **Prof. Chakravarthy Bhagvati**, Dean of the School (SCIS), University of Hyderabad for providing the facilities and extending their cooperation during the course.

We would like to thank **Jocata** for providing us with this opportunity to intern with them and get hands-on experience on how corporate companies function and work. We would like to thank **Mr Ramashankar Darivemula** , our manager and thier entire team.

We express our heart full indebtedness to Mr Ramashankar Darivemula , our external su-pervisor for providing us valuable insights, guiding us, supporting us in completing and these projects.

VIVEK KUMAR
BADAL KUMAR

Abstract

Now a days every Educational institute is focusing on online education where they start-ing to taking classes in online mode. Here we

introducing an online quiz application where student can attempt a quiz and test knowledge.

The E-PARIKSHA (online Exam Portal) is a web application for taking an online quiz in an efficient manner and no time wasting checking the paper. The main objective of E-PARIKSHA (online Exam Portal) is to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast re-sults. students can attend Quiz according to their convenience and time and there is no need of using extra thing like paper, pen etc. This can be used in educational institutions as well as in corporate world. It's also use-full for student they can give those quiz for practice or we can use anywhere any time as it is a web-based Application (user location doesn't matter). No restriction that examiner has to be present when the candidate takes the test. This Web Application provides facility to conduct online quiz from worldwide. It saves time as it allows a 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. The administrator has the privilege to create, modify and delete the Quiz, questions of the quiz, Categories of the Quiz, and also evaluate different contexts. Students can see their results and other activity where a student not only judges their knowledge/skill but o they can improve their knowledge/skill at the same time.

Contents

v

List of Figures

Chapter 1

Introduction

In this chapter we are going to explain in detail about E-Pariksha, its problem statement, motivation, objective and so on. We will explore different dimensions of it.

1.1 E-PARIKSHA

1.1.1 Problem Statment

E-PARIKSHA is a software developed to conduct an Online Quiz based on time con-straints. Portal is accessed by entering the user-

name and password which is added to the database.Before the start of the Quiz, the rules and regulations are displayed which includes description of time limit, number of questions to be answered and scoring methods.Quiz starts by displaying questions with four options each based on categories chosen ,like Gen-eral Knowledge, Verbal Reasoning and Computer Science.If the time exceeds the given time or the user enters the submit button then the quiz will be ended. Final score will be displayed and updated in the database with username.

1.1.2 Objective

1. It will simplify the task of users to test their knowledge and reduce the paperwork.

2. It is also helpful for educational institutes to conduct exams for students in asyn- chronous mode.

3. The result can be displayed after completion of the exam in the form of dashboard.

1.1.3 Motivation

The following are the points which motivate us to do the project on Code Readability:

1. **Time Saving:** For building it, we don't need a huge investment as it is going to an online service that will predict and suggest related information. There is no time-wasting in traveling or making the paper as a copy or giving them to students and doing all the basic regions manually.

2. **Convenience:** It is quite user-friendly which takes basic information of yours and you are all set to go and test and use this application.

3. **Importance:** An online exam provides flexibility and security to the examination process. Once all the questions are uploaded in the system, the system can shuffle and give questions in different orders to different students. This minimizes the chance of cheating.

1.1.4 Scope

The Following points are most valuable.

- **Conduct Exam effortless. It reduces exam anxiety among test takers.**

- **Promote social interaction between the test taker and experts.**

- **Prevents cheating.**

- **Safe and secure data.**
- **Can get result instantly**

1.2 About Company

JOCATA Financial Advisory and Technology Services Private Limited is a Private in-corporated on 16 November 2010. It is classified as a non-govt company and is registered at the Registrar of Companies, era bad.

Jocata Financial Advisory and Technology Services Private Limited's Annual General Meeting was last held on 30 November 2020 and as per records from Ministry of Corporate Affairs, its balance sheet was last filed on 31 March 2020.

2

1.3 Internship Overview

1.3.1 Company Orientation and Security Procedures

We were briefed about the company and its history. Then assigned a company email ID and a Security Password. We have been provided with AWS workspace credentials and Other related passwords and VPN access. Furthermore, they helped us to set up our project. The company mainly intended to work on financial projects.

1.3.2 Our Works

In this project we work as an intern for learning purposes:

- **Creating Rest API:** In this particular application, we have to create a rest API that will have CRUD property i.e., create, read, update and delete. Where we will be using REST API HTTPS methods i.e., GET, POST, PUT, AND DELETE to achieve our task. We will be testing our application using POSTMAN and application builds for testing APIs.

- **Designing Frontend:** we also design a webpage using an angular frontend frame-work.To get started with Angular development, you will need a modern code edi-tor (like VS Code), NodeJS and NPM installed globally, and a browser to test on. Through Node's package manager (NPM), Angular and all its dependencies can be brought into your workspace. After the dependencies have been installed, you can use Angular's built-in Command Line Interface (CLI) to create your first application. CLI can be used for many functions like creating new components, serving the app locally, and building the app for production.

3

Chapter 2

Literature Survey

In this chapter, we are going to explain about application architecture i.e., three-tier architecture further in detail about E-PARIKSHA (online exam Portal) and related tools and technologies.

2.1 Application architecture patterns

This chapter includes the details about my application architecture patterns. Here I am using three-tier architecture for developing full-stack applications.

2.1.1 Three-Tier Architecture

Three-tier architecture, which separates applications into three logical and physical com-puting tiers, is the predominant software architecture for traditional client-server applica-tions. Three-tier architecture is a well-established software application architecture that

organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The three tier is -

1. **Presentation tier** – it is user interface and communication layer of the application

2. **Application tier** – it is known as the logic tier or middle tier and is the heart of the application.

3. **Data tier -** sometimes called database tier, data access tier or back-end of application.

4

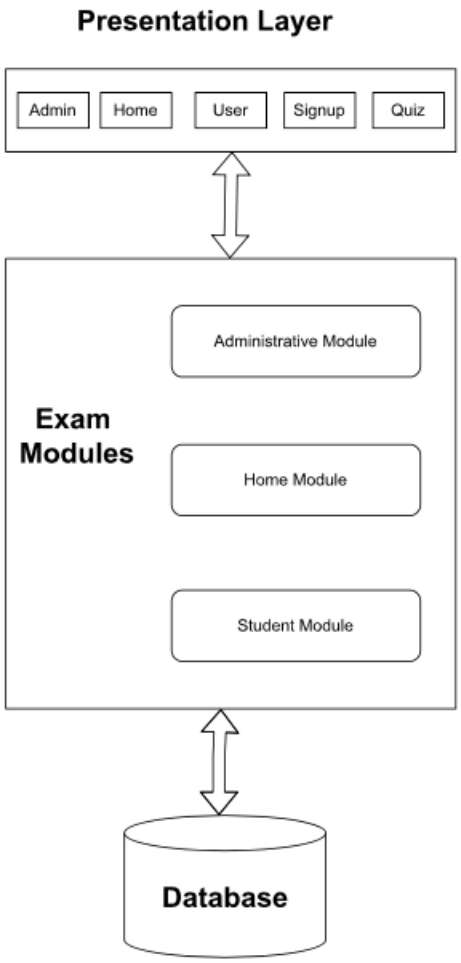

Figure 2.1: Three Tier Architecture

Advantages
The chief benefit of the three-tier architecture is its logical and physical separation of

functionality and more in the following.

1. **Improved security:** Because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

2. **Improved scalability:** Any tier can be scaled independently of

the others as needed.

3. **Faster development:** Because each tier can be developed simultaneously by differ-ent teams, an organization can bring the application to the market faster, and pro-grammers can use the latest and best languages and tools for each tier.

5

2.2 Related Articles and their key Points

Many different researches have focused on the subject of an online examination system. This work can be represented as follows:

SIETTE: Guzman and Conejo (2005) proposed an online examination system called System of Intelligent Evaluation using Tests for Tele-education (SIETTE). SIETTE is a web-based environment to generate and construct adaptive tests. It can be used for instructional objectives via combining adaptive student self-assessment test questions with hints and feedback. SIETTE supports secure login and portability features. On the other hand, the other features: resumption capability, multi-instructor, random question selection, ran-dom questions distribution and random choices distribution is missing.

EMS: Rashad Et. Al. (2010) proposed a web-based online examination system called Exam Management System (EMS). EMS manages the examination and auto-grading for student's exams and supports conducting exams, collects the answers, auto mark the sub-missions, and produce the reports for the test. EMS supports secure login, multi-instructor, and portability features. However, the other features: resumption capability, random ques-tion selection, random questions distribution, and random choices distribution is missing [4]. Arvind Singh, Niraj Shirke, Kiran Shette 2011: The project evaluates the examiners by using the online examination system concept. The exams will be totally customizable. This system will check results automatically based on student's answers.

CBTS: Fagbola et. al. (2013) developed a Computer Based Test System (CBTS). CBTS is a web-based online examination system developed to address issues such as lack of tim-ing flexibility for automation candidates log-off upon expiration of allowed time, result integrity, guaranty, stand-alone deployment, need for flexibility, robustness, designed to support the examination processes and overcome challenges framing the conduct of exam-ination, auto- marking, auto- submission, and generation report of examination result

2.3 E-PARIKSHA (online quiz portal) background -

E-Pariksha is a concept that originated with software engineering

and computer science that arrived with the addition of computing technologies and helped people to develop their skills. It is conduct web application that improve accessibility of remote students.

6

Today many organizations conduct online exams worldwide successfully and issue re-sults online. There are some advantages and disadvantages to the online examination. The advantage is that, "It can be conducted remote candidates' evaluations for answers can be conducted can be fully automated for multiple choice". The question can be evaluated manually or through an automated system depending on the nature of the question and re-quirements. The disadvantage is there is no method to identify whether the exact student takes the exam.

2.4 Tools And Technologies Used

2.4.1 Angular

Angular is a platform and framework for building single-page client applications using HTML CSS JS and TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

2.4.2 Virtual-DOM

Virtual-DOM is a JavaScript object, each object contains all the information needed to create a DOM, when the data changes it will calculate the change between the object and the real tree, which will help optimize render DOM tree. It can be assumed that a virtual model can handle client data.

2.4.3 Component

Angular components are a subset of directives, always associated with a template. Unlike other directives, only one component can be instantiated for a given element in a template. A component must belong to an Ng Module in order for it to be available to another com-ponent or application. It's a function that handles HTML tag creation as well as an example of Virtual-DOM processing capability. Any changes of data at any time will be processed and updated immediately by Virtual-DOM.

7

2.4.4 Java

Java is an object-oriented programming language with a high level of abstraction and as few implementation dependencies as possible. Most firms believe it to be one of the fastest, most secure, and most trustworthy programming languages for developing their projects.

Jocata core programming language is Java as it uses java framework and module to develop its software.

2.4.5 Spring Boot

Spring Boot makes it simple to construct self-contained, production-ready Spring-based applications that can be" simply run." We offer an unbiased perspective on the Spring plat-form and third-party libraries so you can get started quickly. The majority of Spring Boot apps only require a minimal amount of Spring configuration.

MVC (Model View Controller)
The Model-View-Controller (MVC) is an architectural pattern that separates an appli-cation into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.

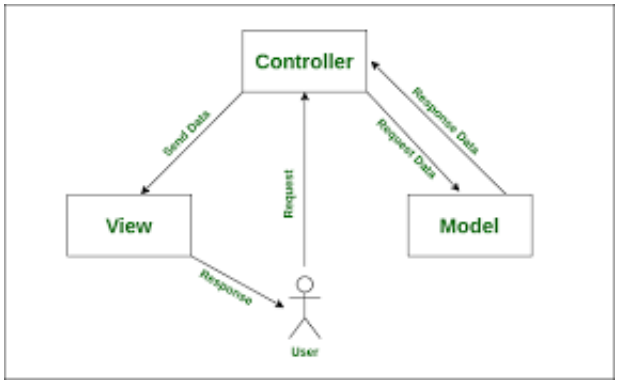

Figure 2.2: Model-View-Controller

Model The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

8

View The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controllers Controllers act as an interface between Model and View components to pro-cess all the business logic and incoming requests,

manipulate data using the Model com-ponent and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

2.4.6 Database

MySql One of the reasons MySQL is the world's most popular open source database is that it provides comprehensive support for every application development need. Within the database, support can be found for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more.

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results.

9

Chapter 3

Project: Design And Implementation

This Chapter explains the design aspect of the **E-Pariksha** Online portal with a UML di-agram also DFD for the implementation of the project along with the complex architecture of the system.

3.1 Design

Here it deals with the designing phase of the Software development life cycle (SDLC) under we will discuss about the different diagrams like DFD, Use Case diagram, Entity Relationship Diagram, and architecture of the system.

3.1.1 Decomposition diagram

A decomposition diagram depicts the breakdown of a complicated entity, such as a pro-cess, organization, data topic area, or other sorts of object, into lower-level, more specific components. Division diagrams, for example, can depict organizational structure or the decomposition of functions into processes. As of now, the quiz maker response has also been given to Admin.

Function of Decomposition diagram Functional decomposition breaks down a large, complex process into an array of smaller, simpler units or tasks, fostering a better under-standing of the overall process.

A functional decomposition diagram contains the whole function or project along with all of the necessary sub-tasks needed to complete it.

10

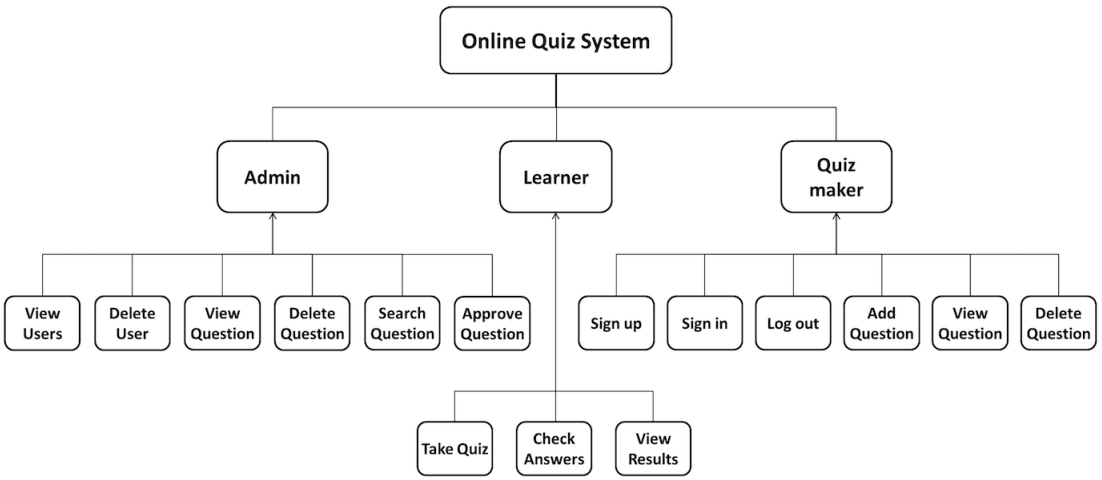

Figure 3.1: Decomposition diagram of E-Pariksha

3.1.2 Flow Diagram

A flow diagram is a visual representation of data flowing through a system or a process. The FD also includes information on each entity's outputs and inputs, as well as the process itself. There is no control flow, no decision rules, and no loops in a data-flow diagram.

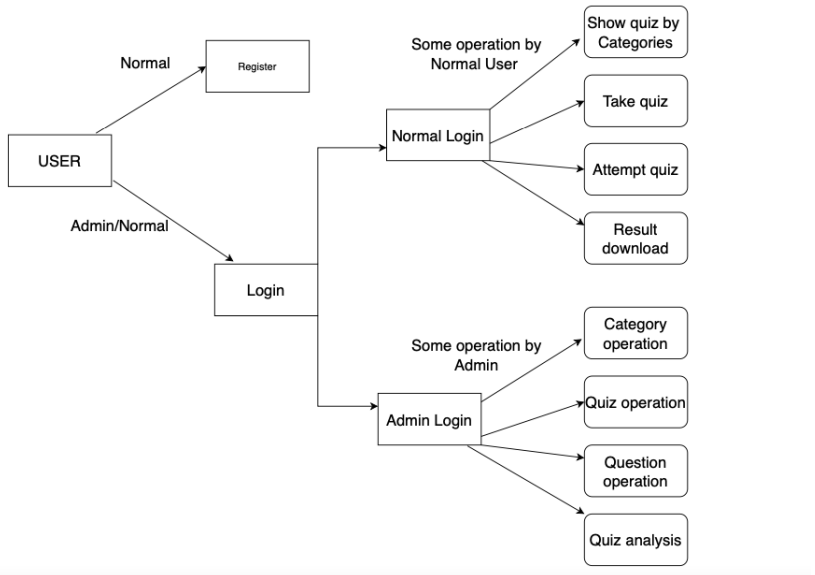

Figure 3.2: Flow Diagram of E-Pariksha

11

3.1.3 Use Case Diagram

The unified modeling language used is use case diagram. A use case is a set of scenarios that describes an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

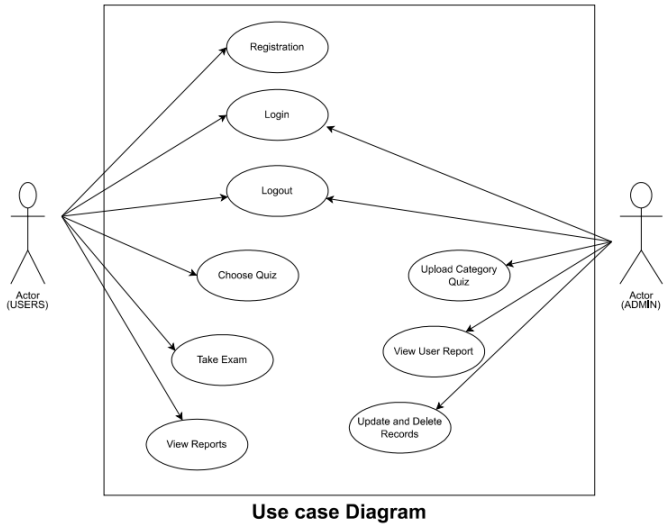

Figure 3.3: Use Case Diagram of E-Pariksha

3.1.4 Entity Relation diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system. An ERD uses data modeling tech-niques that can help define business processes and serve as the foundation for a relational database.

Importance of ERDs Entity relationship diagrams provide a Visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After arelational databaseis rolled out, an ERD can still serve as a reference point, should any debugging or business process re-engineering be needed later.

12

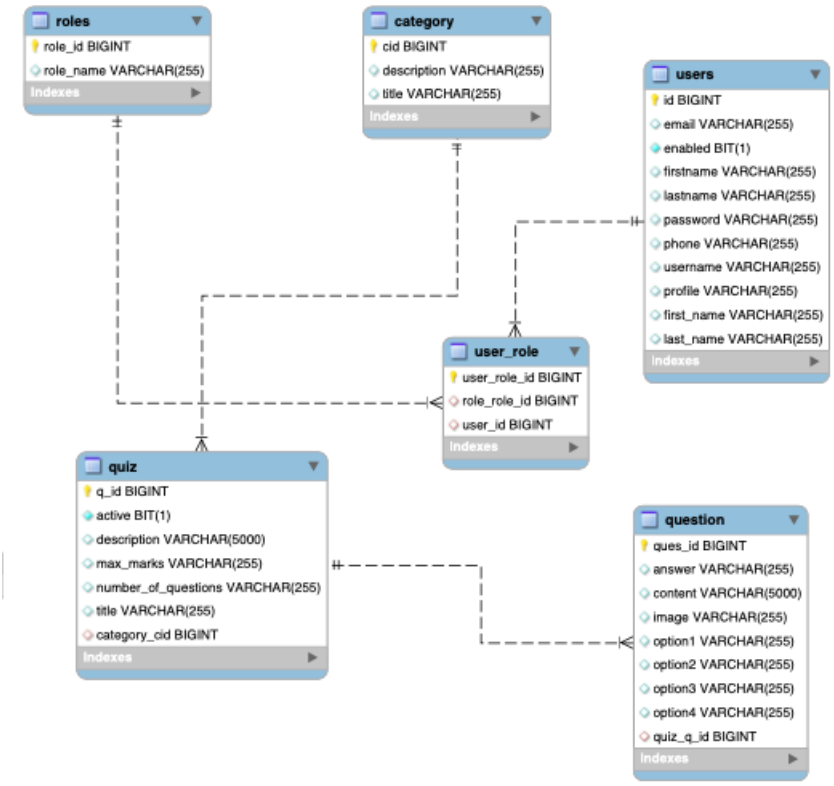

Figure 3.4: Entity Relation diagram of E-Pariksha

13

3.2 Implementation

In this section you can see some Implementation part of application and it's modules.

3.2.1 Home

It's a first entry page represents as home page of application in working face.

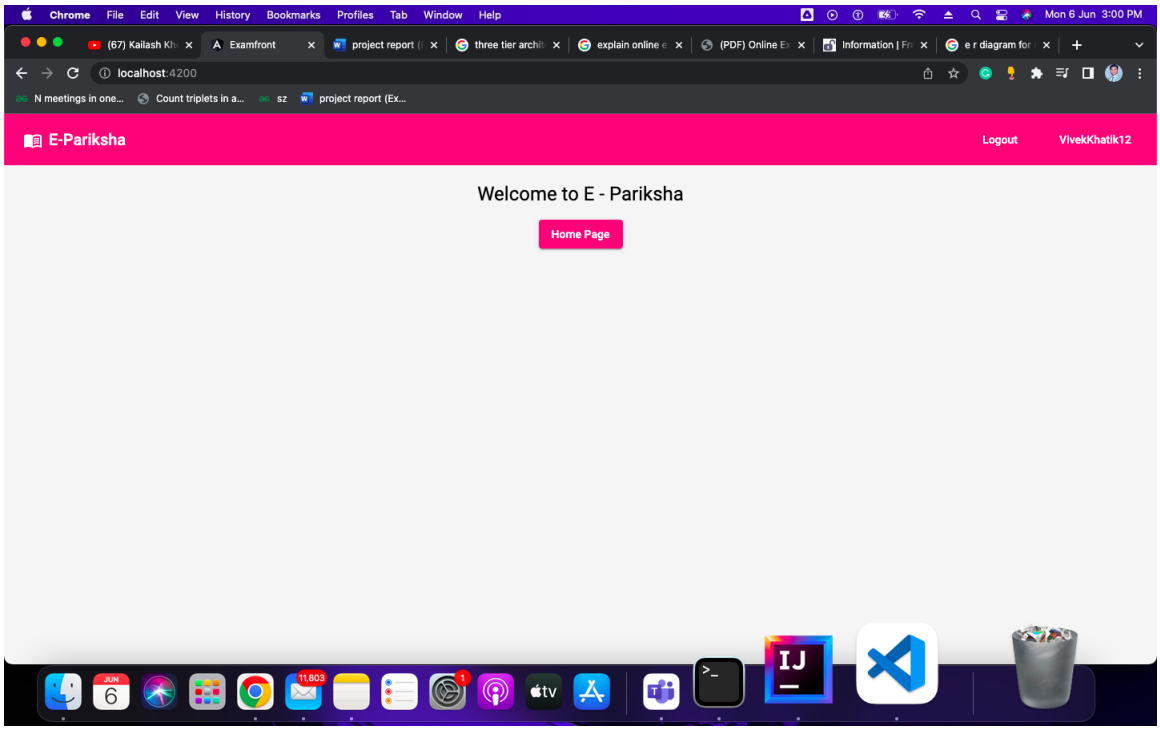

Figure 3.5: screenshot of the Home page of E-PARIKSHA

Next Steps:

- **Login:** User can login and the system will open the next window ac- cording to the role.
- **Registration:** if the user is not registered then needs to register on the registration page.

14

3.2.2 Registration

In this registration form, the User And Admin can register themselves.The

system will provide proper authentication while entering the data into the

registration form.

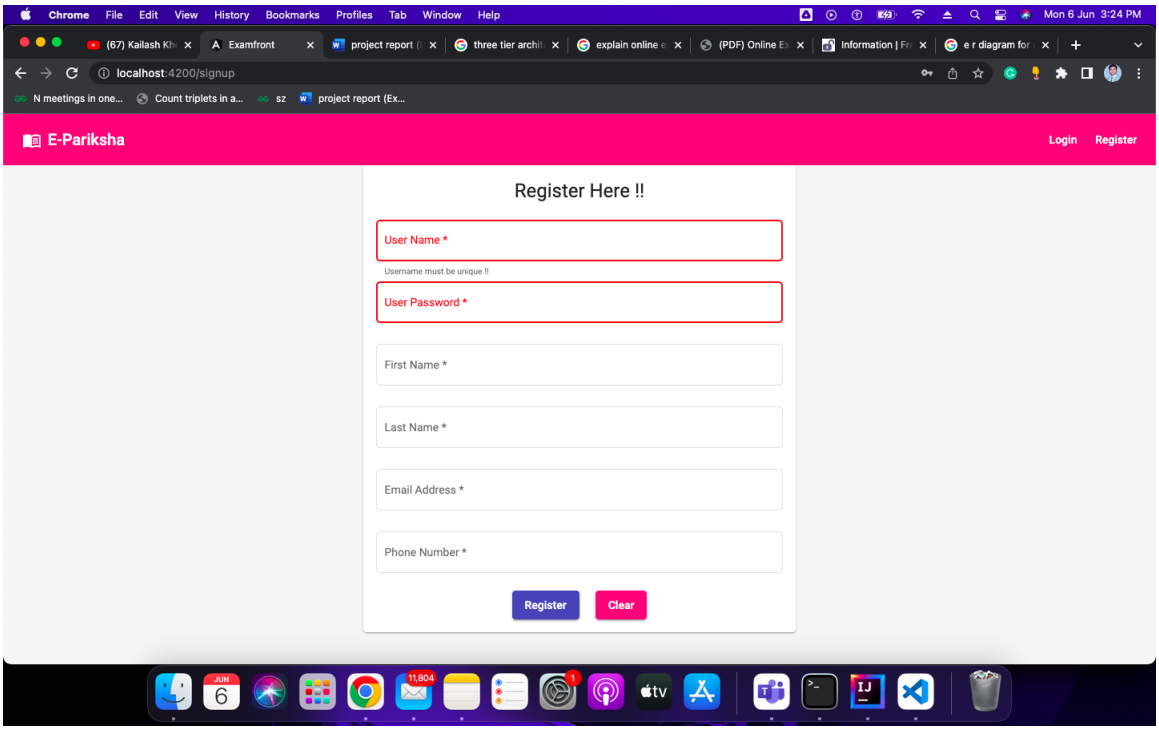

Figure 3.6: Registration

Next Steps:

- **Login:** After registration, the user can login with the registered username and password the system will open the next page according to the user's role.

15

Listing 3.1: Sample code for User Controller class : Controller

```
1 //=================================================
2 //code sinnepet which will work as a RestController for User.
3 //=================================================
4
```

```
                                    @RestController
5                                   @RequestMapping("/user")
6                                   @CrossOrigin("*")
7                                   public class UserController
8                                   {
```

```
9
10 @Autowired
11 private UserService userService;
12
13 @Autowired
14 private BCryptPasswordEncoder bCryptPasswordEncoder;
15
16 //creating user
17 @PostMapping("/")
18 public User createUser(@RequestBody User user) throws
Exception {
19
20 user.setProfile("default.png");
21 //encoding password with bcryptpasswordencoder
22
23 user.setPassword(this.bCryptPasswordEncoder.encode(user.
getPassword()));
24
25 Set<UserRole > roles = new HashSet <>();
26
27 Role role = new Role();
28 role.setRoleId(45L);
29 role.setRoleName("NORMAL");
30
31 UserRole userRole = new UserRole();
```

```
32 userRole.setUser(user);
33 userRole.setRole(role);
34
35 roles.add(userRole);
16
36
37                  }                   return
38                                      this.userService.createUser(user,
                                        roles);
39
                                        @GetMapping("/{username}")
                                        public User ge-
40                                      tUser(@PathVariable("username")
41                                      String username
                                        ) {
42                  }                   return
43                                      this.userService.getUser(username);
44
                                        //delete the
45                                      user by id
46                                      @DeleteMapping("/{userId}")
                                        public void
                    }                   dele-
                                        teUser(@PathVariable("userId")
                                        Long userId)
47                                      {
48                                      this.userService.deleteUser(userId);
49                                      }
50
17
```

3.2.3 LogIn

The login system of the Application system will ask the user to provide the username and password according to that user will get the access as User or admin.

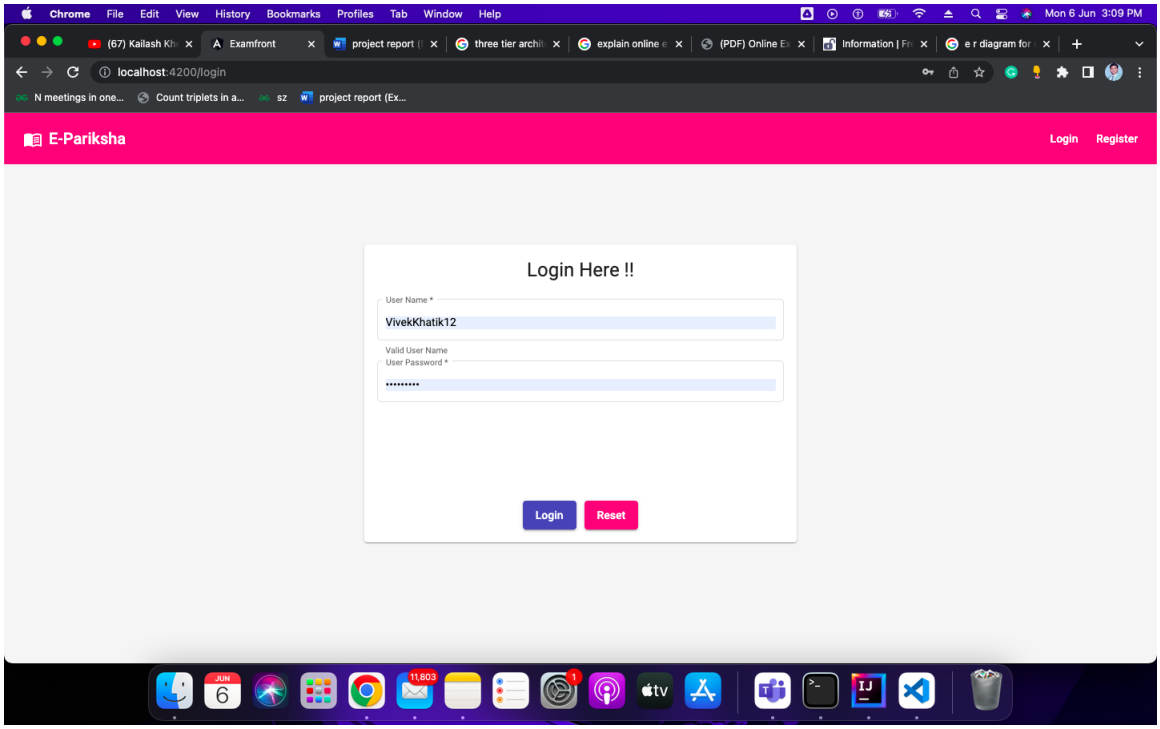

Figure 3.7: screenshot of the LogIn page of E-PARIKSHA

Next Steps:

- **As an Admin:** if the user is recognized as an admin by the system then the admin page will open.
- **As an User:** if the user is recognized as an Normal User by the system then the admin page will open.

18

3.2.4 LogIn As an Admin

If the system finds the user an admin then the admin access page is going to open where admin can perform their task efficiently

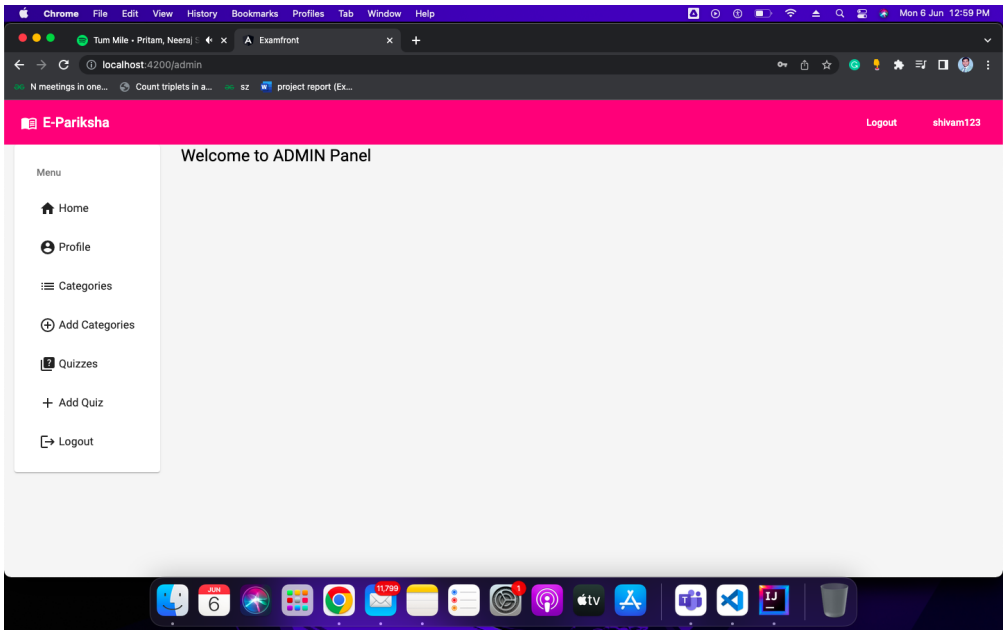

Figure 3.8: screenshot of the Admin Page of E-PARIKSHA

Next Steps:

- **Category Operations:** In this page the admin can perform category opretions like add, delete, update.
- **Quizess Operations:** In this page the admin can perform Quizzes operations of different categories.
- **Questions Operations:** In this page the admin can per- form Questions operations of different Quizzes.

19

3.2.5 LogIn As an Admin - Category Opretions

Next step: Admin can add , Delete , Update Category of Quizess.

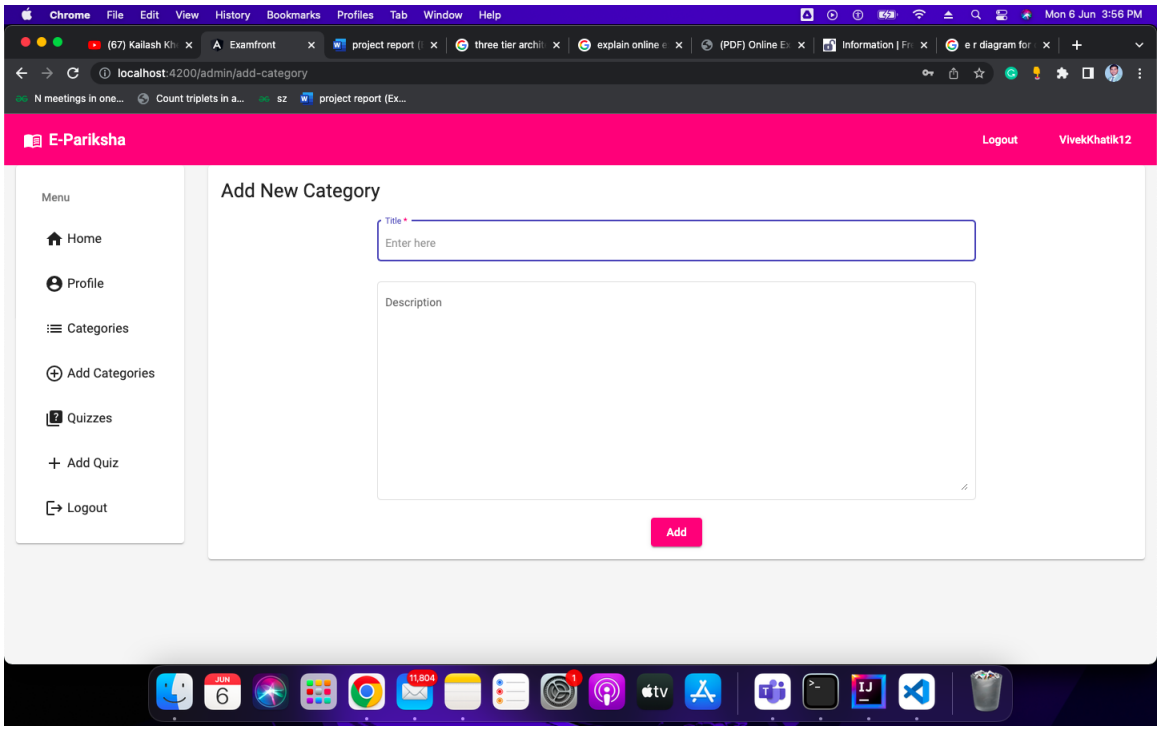

Figure 3.9: LogIn As an Admin - Category Operations of E-PARIKSHA

Next Steps:

- **Quizess Opretions:** Admin can add, delete,update operations with Quizess.
- **Questions Opretions...**Also can add, delete,update operations with Questions of the Quizess.

20

Listing 3.2: Sample code for Category Controller class : Controller

```
1 //=========================================================
2 //code sinnepet which will work as a RestController for Category.
3 //=========================================================
4
```

```
@RestController
@RequestMapping("/category")
@CrossOrigin("*")
public class CategoryController {

@Autowired
private CategoryService categoryService;

//add category
@PostMapping("/")
public ResponseEntity <Category > addCategory(@RequestBody Category category) {
Category category1 = this.categoryService.addCategory(category);
return ResponseEntity.ok(category1);
}

//get category
@GetMapping("/{categoryId}")
public Category getCategory(@PathVariable("categoryId") Long categoryId) {
return this.categoryService.getCategory(categoryId);
}

```

```
                                    //get all categories
                                    @GetMapping("/")
26                                  public ResponseEntity
27                                  <?> getCategories() {
28                                  return ResponseEn-
29                                  tity.ok(this.categoryService.
```

```
30                  }               getCategories());
```

```
31
32 //update category
33 @PutMapping("/")
21
34 public Category updateCategory(@RequestBody Category cate-
   gory
```

```
35                  }               ) {
                                    return
36                                  this.categoryService.updateCategory(category);
```

```
37
```

```
                                    //delete category
                                    @DeleteMapping("/{categoryId}")
38                                  public void deleteCate-
39                                  gory(@PathVariable("categoryId")
40                                  Long
```

```
                                    categoryId) {
41                  }
                                    this.categoryService.deleteCategory(categoryId);
42
```

```
43
44 }
```

22

3.2.6 LogIn As an Admin - Quiz Opretions

Next step: Admin can Add , Delete , Update Quizess of different category.

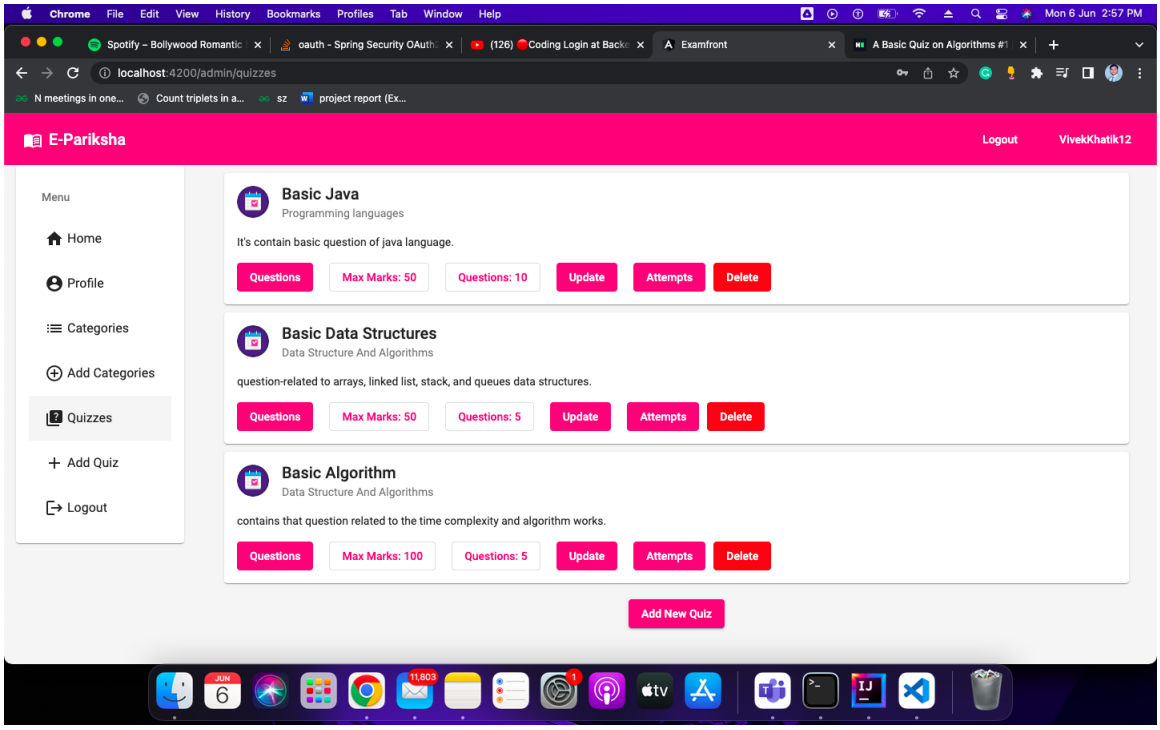

Figure 3.10: LogIn As an Admin - Quiz Opretions

Next Steps:

- **Questions Opretions:** Can add, delete,update operations

with Questions of the Quizess.

23

Listing 3.3: Sample code for Quiz Controller class : Controller

```
1 //=========================================================
2 //code sinnepet which will work as a RestController for Quiz.
3 //=========================================================
4
```

```
                                    @RestController
                                    @CrossOrigin("*")
5                                   @RequestMapping("/quiz")
6                                   public class QuizController
7                                   {
8                                   @Autowired
9                                   private QuizService
10                                  quizService;
```

```
11
```

```
                                    //add quiz service
                                    @PostMapping("/")
                                    public ResponseEntity
12                                  <Quiz>
13                                  add(@RequestBody Quiz
14                                  quiz) { return ResponseEn-
15                                  tity.ok(this.quizService.addQuiz(quiz));
16                                  }
```

```
17

18 //update quiz

19

20 @PutMapping("/")

21 public ResponseEntity <Quiz> update(@RequestBody Quiz quiz)
{

22 return ResponseEntity.ok(this.quizService.updateQuiz(quiz
```

```
23                  }                               ));
```

```
24
```

```
                                    //get quiz
                                    @GetMapping("/")
25                                  public ResponseEntity
26                                  <?> quizzes() {
27                                  return ResponseEn-
28                                  tity.ok(this.quizService.getQuizzes());
29                                  }
```

```
30
```

```
                                    //get single quiz
                                    @GetMapping("/{qid}")
31                                  public Quiz
32                                  quiz(@PathVariable("qid")
33                                  Long qid) { return
34                                  this.quizService.getQuiz(qid);
35                                  }
```

```
36
```

24

```
                                    //delete the quiz
                                    @DeleteMapping("/{qid}")
37                                  public void
38                                  delete(@PathVariable("qid")
39                                  Long qid) {
40                                  this.quizService.deleteQuiz(qid);
41                                  }
```

```
42

43 @GetMapping("/category/{cid}")

44 public List<Quiz> getQuizzesOfCategory(@PathVariable("cid")
```

```
                                                Long cid) {
45                                              Category
                                                category =
                        }                       new
                                                Category();
                                                category.setCid(cid);
46
47                                              return
48                                              this.quizService.getQuizzesOfCategory(category);
49

                                                //get active quizzes
                                                @GetMapping("/active")
50                                              public List<Quiz>
51                                              getActiveQuizzes() {
52                                              return
53                                              this.quizService.getActiveQuizzes();
54                                              }
55

                                                //get active quizzes of
                                                category
                                                @GetMapping("/category/active/{cid}")
                                                public List<Quiz>
56                                              getActive-
57                                              Quizzes(@PathVariable("cid")
58                                              Long


                                                cid) {
                                                Category category = new
                                                Category();
59                                              category.setCid(cid);
60                                              return
61                                              this.quizService.getActiveQuizzesOfCategory(
```

```
62
63                  }                  }                  category);
```

25

3.2.7 LogIn As an Admin - Question Opretions

Next step: Admin can add , Delete , Update questions of different Quizess of different category.

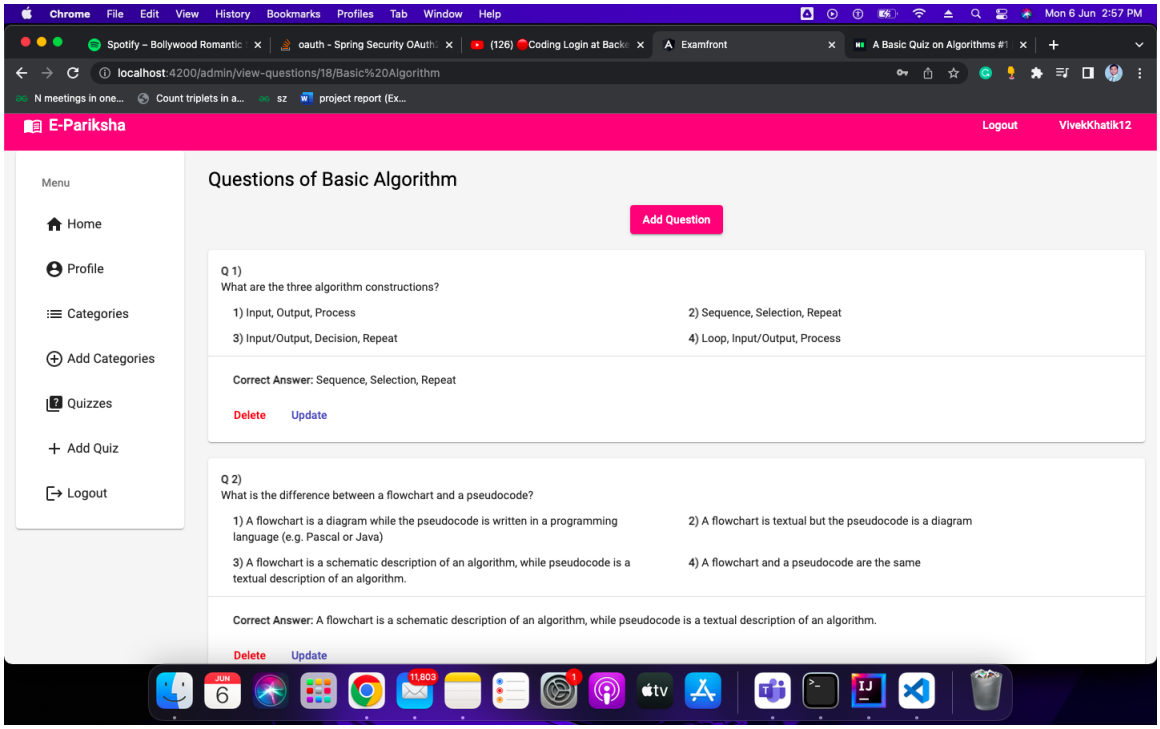

Figure 3.11: LogIn As an Admin - Question Opretions

26

Listing 3.4: Sample code for Quiz Controller class : Controller

```
                                    //=====================================
                                    //code sinnepet which will
                                    work as a RestController
                                    for Quiz.
1                                   //=====================================
2                                   @RestController
3                                   @CrossOrigin("*")
4                                   @RequestMapping("/question")
5                                   public class
6                                   QuestionController {
7                                   @Autowired
8                                   private QuestionService
9                                   service;
```

```
10

11 @Autowired

12 private QuizService quizService;

13

14 //add question

15 @PostMapping("/")

16 public ResponseEntity <Question > add(@RequestBody Question

question) {

17 return ResponseEntity.ok(this.service.addQuestion(

18                  }                                   question));
```

```
19

20 //update the question

21 @PutMapping("/")

22 public ResponseEntity <Question > update(@RequestBody Ques-
tion

question) {

23 return ResponseEntity.ok(this.service.updateQuestion(
```

```
24                  }                                   question));

25
26 //get all question of any quid
27 @GetMapping("/quiz/{qid}")
28 public ResponseEntity <?> getQuestionsOfQuiz(@PathVariable("
qid") Long qid) {
29
30 Quiz quiz = this.quizService.getQuiz(qid);
31 Set<Question > questions = quiz.getQuestions();
32 List list = new ArrayList(questions);
27
33 if (list.size() > Integer.parseInt(quiz.
getNumberOfQuestions())) {
34 list = list.subList(0, Integer.parseInt(quiz.
                                    getNumberOfQuestions()
                                    + 1)); }
35                                  Collections.shuffle(list);
36                                  return
37                                  ResponseEntity.ok(list);

38
39 }
40
41 @GetMapping("/quiz/all/{qid}")
42 public ResponseEntity <?> getQuestionsOfQuizAdmin(
```

```
                                    @PathVariable("qid") Long
                                    qid) {
                                    Quiz quiz = new Quiz();
                                    quiz.setqId(qid);
43                                  Set<Question >
44                                  questionsOfQuiz =
45                                  this.service.
```

```
46                  }               getQuestionsOfQuiz(quiz);
                                    return
47                                  ResponseEn-
                                    tity.ok(questionsOfQuiz);
48
```

```
                                    //get single question
                                    @GetMapping("/{quesId}")
49                                  public Question
50                                  get(@PathVariable("quesId")
51                                  Long quesId) { return
52                                  this.service.getQuestion(quesId);
53                                  }
```

```
54
```

```
                                    //delete question
                                    @DeleteMapping("/{quesId}")
55                                  public void
56                                  delete(@PathVariable("quesId")
57                                  Long quesId) {
58                                  this.service.deleteQuestion(quesId);
59                                  }
```

```
60

61 //eval quiz

62 @PostMapping("/eval-quiz")

63 public ResponseEntity <?> evalQuiz(@RequestBody
List<Question >
```

questions) {

64 System.out.println(questions);

65 double marksGot = 0;

28

		int correctAnswers = 0;
		int attempted = 0;
		for (Question q :

66		questions) {
67		//single questions
68		Question question =
69		this.service.get(q.getQuesId());
70		if (ques-
71		tion.getAnswer().equals(q.getGivenAnswer()))

72	{	//correct
73		correctAnswers++;

74

75 double marksSingle = Double.parseDouble(questions

.get(0).getQuiz().getMaxMarks()) / questions.

76	//	size(); this.questions[0].quiz.maxMarks / this.

questions.length;

77 marksGot += marksSingle;

78

79 }

80

```
                                    if (q.getGivenAnswer() !=
                                    null) {
                                    attempted++;
                                    }
                                    };
                                    Map<String , Object>
81                                  map= new HashMap
82                                  <>();
83                                  map.put("marksGot",
84                                  marksGot);
85                                  map.put("correctAnswers",
86                                  correctAnswers);
87                                  map.put("attempted",
88                                  attempted);
89
90
                                                        Map<String
                                                        , Object>
                                                        map =
                                                        Map.of("marksGot",
91                          //                          marksGot , "
                                                        correctAnswers",
                                                        correctAn-
                                                        swers ,
                                                        "attempted",
                                                        attempted);
92                          }                           return
                                                        ResponseEn-
                                                        tity.ok(map);
93                                                      }
94
```

29

3.2.8 LogIn As an User

Next step: As we can see the this page we have multiple quiz of multiple category so now User can attempt the quiz according of their interest and test knowledge.

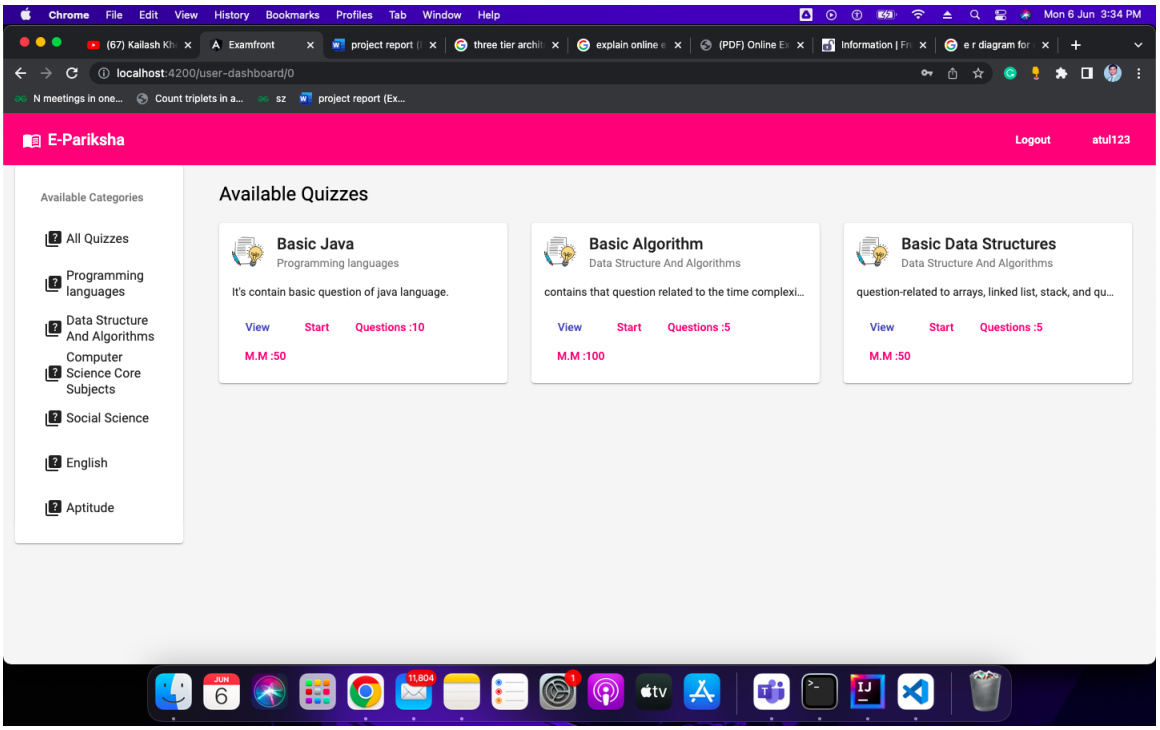

Figure 3.12: LogIn As an User in E-PARIKSHA

Next Steps:

- **Select Anyone Quiz..**

30

3.2.9 Attempt Quiz

Next step: Now User can able to see a number of questions there and every question have 4 options 1 ,2,3 4, and the page has one automatic timer also after finishing that timer the quiz will submit autometically.

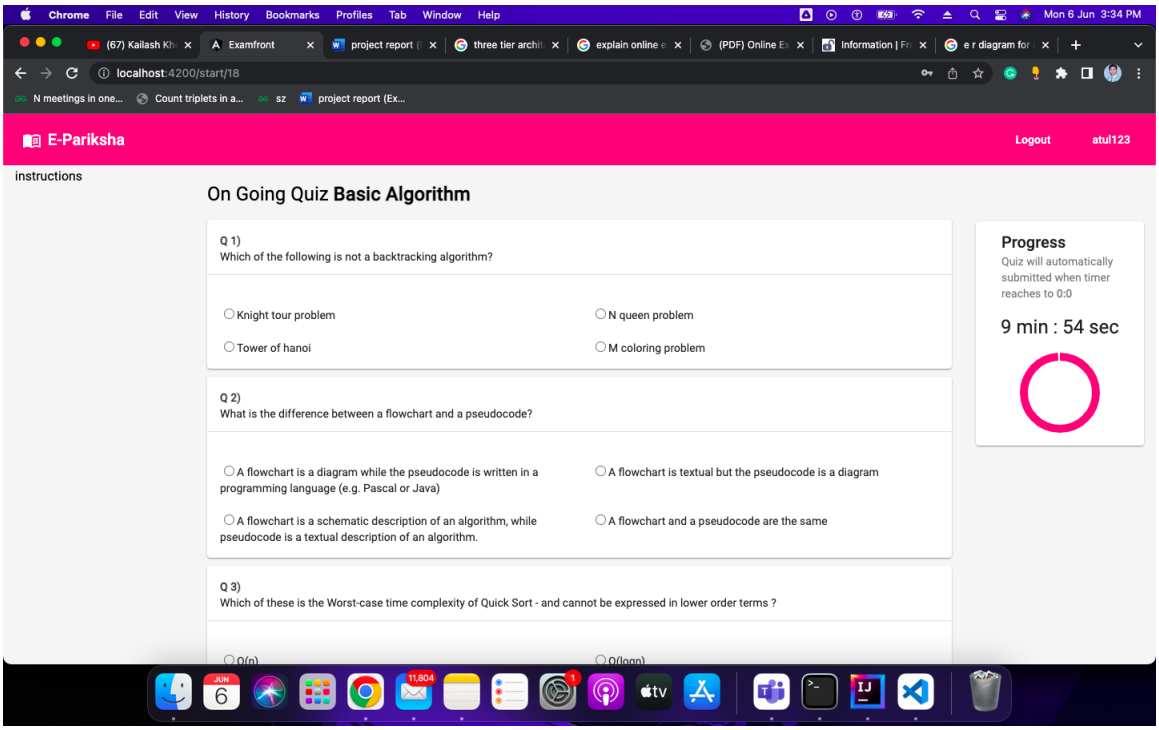

Figure 3.13: Attempt Quiz

Next Steps:

- **Submit....**

31

3.2.10 Result

Next step: User can see that result of given quiz.

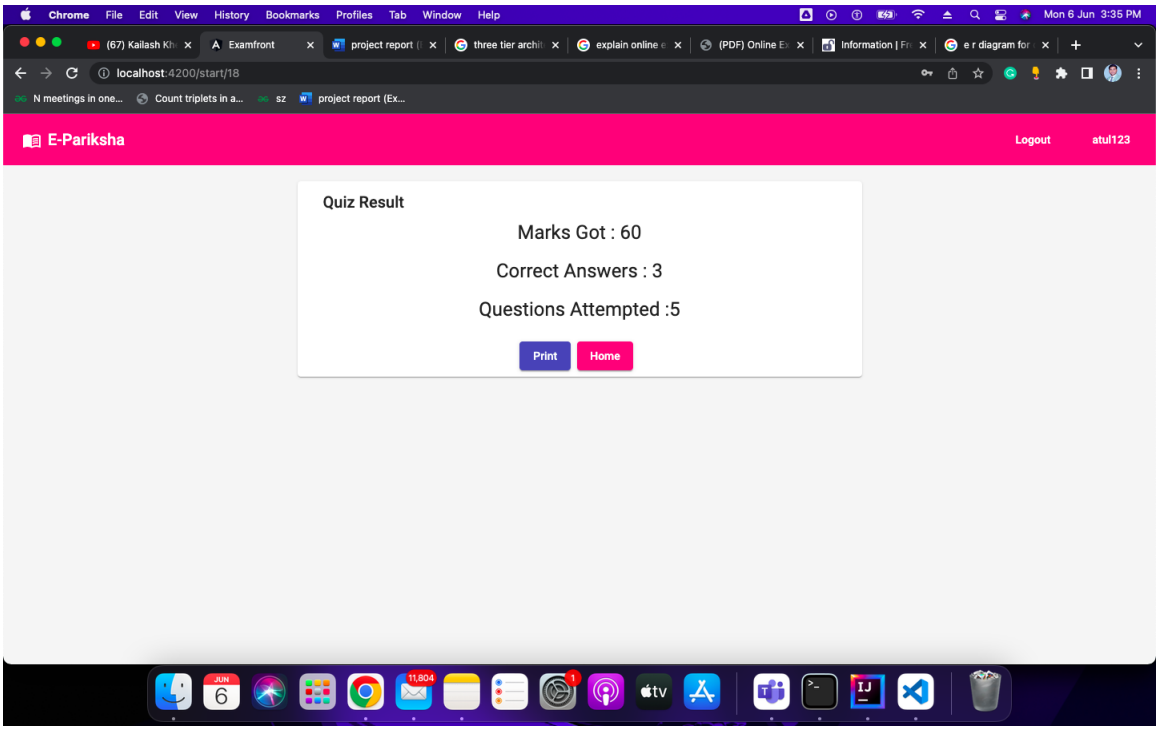

Figure 3.14: Result

Next Steps:

- **End.**

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The Project E-Pariksha as the intern project is to develop an online portal for skill development accessible by the internship company to monitor the interns' skill set. The present project work involves Java, Spring Boot, and Angular JS. Part of the work created the web portal in which users can ac-cess the examination page after properly registering through the registration form page. and get authenticated from the login page based on their role.

If the role is user, then the user can attempt the exam, and if the role is admin, then the admin has many facilities like adding the quiz questions, categories of the quiz, deleting the quiz, seeing marks of students, etc.

GOAL Achieved.

- Reduced entry work.
- Easy retrieval of information.
- Reduced errors due to human intervention.
- Portable and flexible for further enhancement.
- Web enabled.

33

4.2 Limitations of the work

Let us look at the disadvantages of an online examination system:

- Challenges in Technology Adoption..
- Infrastructural Barriers.
- Maintenance also Require..
- Susceptible to Cheating .
- Transitioning to Open-Book Exams.

4.3 Future Work:

As shown in the screen short that project still in working face So the next component we going to work in home page, and add more security features like live camera and Id Access. Also, domain for E-Pariksha website.

Home Page: In future we are going to build a home page in such a way that when anyone comes to the application, he first going to see home page with multiple category quizzes if user want to take quiz that that going to authenticate.

Some other future works -

1. In future we want to add some coding exams like Gfg ,Hacker Rank coding Platform.

2. Facilitating communication between users, by allowing users to launch mobile apps to communicate with customer care, or send messages to service center and avail free information related to quizzes and its related technologies and issues as well.

34

3. Providing multiple categories of quizzes to relate to every student as user experience.

4. Admin also can make some quizzes as paid so we are going to add a payment gateway also in the existing system.

4.4 Recommendation:

The project has been accomplished and an application was developed to solve the aforementioned problems. For further development, here are some recommendations on this project: The application should support an auto-mated time setting to let the student know how many hours and minutes are left for them to complete the examination.

35

Bibliography

[1] Introduction: Insight Details, Jocata GRID

[2] Literature Survey : Application architecture patterns, Data manage-ment, External API and Service Discovery

[3] Description Help: postman, Camunda Modeler, Dbeaver

[4] Consuming REST API, url and KEY

[5] K. S. Prasad Reddy, Beginning Spring Boot, "RESTful Web Services :Building REST APIs Using Spring Boot,

[6] Decomposition of Web Based Online Quiz System
e

[7] A Project on Online MCQ Quiz Application
p

36

[8] Spring MVC
https://javatpoint.com/

[9] K. S. Prasad Reddy, Beginning Spring Boot, "Working with JPA :BSpring Data JPA, pp. 84-97, 2015.

37