



PROJECT REPORT ON MOCK TEST PORTAL

Submitted To:

Mr. Dinesh Bhardwaj

Asst. Prof. of Computer Science Department.

Submitted By:

Umesh Kumar (10572024602)

Rohit Kumar (10722024610)

CONTENTS

Table of Contents

1. INTRODUCTION	4
1.1 Online Mock Test - An Overview	4
1.2 Scope of the Project	4
1.3 Study of Existing System.....	4
1.4 Objectives:	4
1.5 Profile of The Problem	5
2. REQUIREMENT ANALYSIS STEPS	6
2.1 FUNCTIONAL REQUIREMENTS	6
2.2 NON FUNCTIONAL REQUIREMENT.....	7
3. SYSTEM ANALYSIS	9
3.1 Proposed System.....	9
3.1.1 Defining the Problem.....	9
3.1.2 Developing Solution Strategies	9
3.1.3 Flow Diagrams.....	10
3.2 System Specification	13
3.2.1 Hardware Specification	13
3.2.2 Software Specification.....	13
4. SOFTWARE DESIGN	14
4.1 Interface Design:.....	14
4.1.1 Home Page.....	14
4.1.2 Login Page	15
4.1.3 Signup Page	15
4.1.4 Dashboard Page	16
4.1.5 Test Page.....	16
4.1.6 Add Question Page (Admin Only).....	18
4.2 Database Design:	18
4.3 Coding:	20
4.3.1 Add Questions Module (for admin only):.....	20
4.3.2 Get Question Module:.....	21
5. IMPLEMENTATION.....	24
6. TESTING.....	25
6.1 Techniques Used in Testing.....	25
6.2 Test Case.....	25

7. Maintenance.....	28
8. BIBLIOGRAPHY AND REFERENCES	29
8.1 Bibliography:	29
8.2 References:	29

1. INTRODUCTION

1.1 Online Mock Test - An Overview

The Online Mock Test project aims to create a platform for conducting and managing online mock tests. The system will provide a user-friendly interface for creating, scheduling, and administering tests, as well as generating and analyzing test results. The project will be implemented using HTML, CSS, JavaScript, and Node.js (Express.js).

1.2 Scope of the Project

The scope of the project includes the development of a web-based application that allows users to create and take mock tests in various fields, such as education, recruitment, and certification. The system will include features such as user authentication, test creation, scheduling, test-taking, result analysis, and feedback. The project aims to provide a comprehensive solution for conducting online mock tests, which can be customized to meet the specific needs of different users.

1.3 Study of Existing System

Currently, there are several online mock test platforms available, but they either have limited features or are not user-friendly. Many of them also require users to pay a subscription fee or restrict the number of tests that can be created and taken. Therefore, there is a need for a comprehensive and user-friendly online mock test platform that is accessible to everyone. The proposed system aims to address these issues and provide a seamless experience for test creators and takers alike. SYSTEM ANALYSIS.

1.4 Objectives:

The objectives of this project are as follows:

- To provide a web-based platform for users to take mock tests online.
- To help students and job seekers prepare for competitive exams by giving them a simulated experience of the actual test.
- To allow users to register and login to the application in a secure and user-friendly manner.
- To provide a system for generating randomized tests from a database of questions and answers.
- To provide a dashboard for users to view their scores and performance.
- To ensure the application is responsive, accessible, and scalable.

- To provide a project that demonstrates proficiency in HTML, CSS, JavaScript, and Node.js (specifically, Express.js).

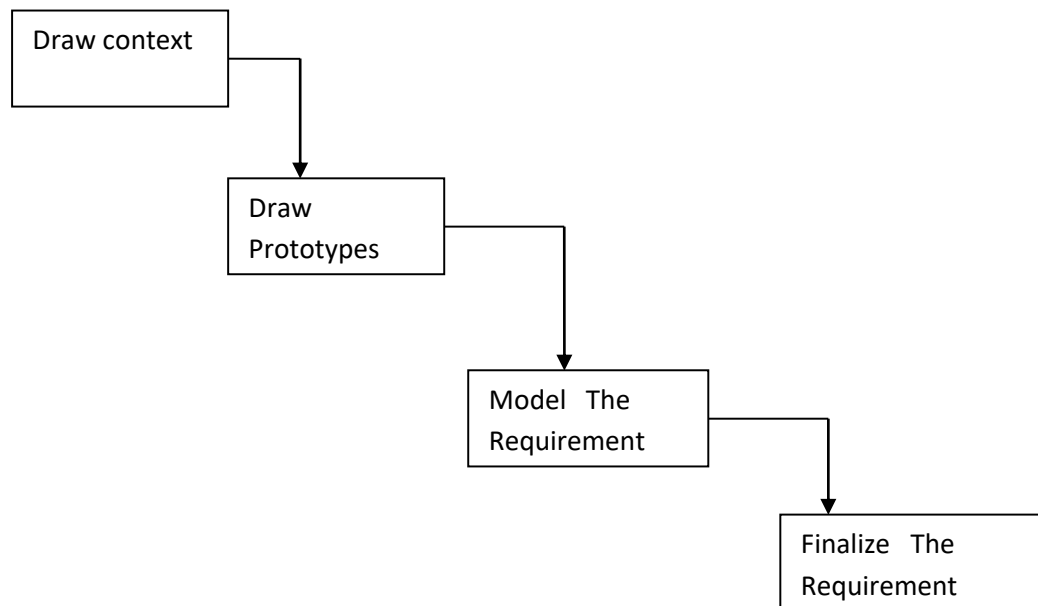
1.5 Profile of The Problem

The profile of the problem is a detailed description of the problem that the project aims to solve. It includes information about the context, scope, and impact of the problem. In the case of an online mock test project, the profile of the problem could include the following:

- **Context:** The context of the problem could be the lack of accessible and affordable exam preparation resources for students and job seekers.
- **Scope:** The scope of the problem could be the need for a platform that provides users with a simulated experience of the actual exam, including randomized tests and instant feedback.
- **Impact:** The impact of the problem could be that students and job seekers are not adequately prepared for competitive exams, which can have negative consequences such as reduced employment opportunities and limited career advancement. By providing an online mock test platform, the project aims to address this problem and help users achieve their goals.

The profile of the problem is an important aspect of any project, as it helps to clearly define the problem and its impact on the target audience. This information can be used to guide the design and development of the project, ensuring that it meets the needs of the users and addresses the root causes of the problem.

2. REQUIREMENT ANALYSIS STEPS



2.1 FUNCTIONAL REQUIREMENTS

Functional requirements define the fundamental actions that must take place in the software in accepting the inputs and in processing and generating the outputs. These are listed as “shall” statements starting with “The system shall....

Register Module– This module is provided users to register themselves to assess the site.

- **Input** – Name, username, password and re-password.
- **Process** – After entering Name, username, password and re-password by user process of validation occur to identify whether information is valid or not and if it is valid then user will be added to database.
 - There is other process that check the input enter by user in username field (every time the filed is updated) to verify that username is available or not.
- **Output** – after successful registration of user the success message will be send to the client and current page will be redirected to login page.

Login Module– This module is provided for administrator and users such as basic user or admin who have registered themselves in the system. These logins are provided according to the need of the systems.

- **Input** – Username and password
- **Process** – After entering username and password by user process of validation occur to identify whether username and password is available in database or not.
- **Output** – Registered user can access website and can use the services.

Test Module– As users are the main visitor of site, the following facilities are available through this module.

- User Can select Topic of Test In dashboard
- And give the test on test page
- After submitting the test the result page will be shown.

Input – Topic , subtopic and test length

- **Process**– request will sent to the server through fetch request containing the above inputs.
- **Output**– array of Question containing 4 options and correct Ans will be return and show on test page

2.2 NON FUNCTIONAL REQUIREMENT

- **Performance:** Define the maximum acceptable response times, throughput, and other relevant metrics that your application should meet.
- **Availability:** Specify the required uptime for your application, as well as any backup or recovery procedures that should be implemented in case of downtime.
- **Security:** Outline the required security measures, such as encryption, access control, and authentication mechanisms that your application should have.
- **Usability:** Define the usability standards that your application should meet, including ease of use, accessibility, and user interface design.

- **Scalability:** Describe the capacity requirements that your application should be able to handle, such as the number of concurrent users, transactions, and data volumes.
- **Maintainability:** Define the level of maintainability required, such as the ease of adding new features, fixing bugs, and updating the application.
- **Interoperability:** Specify the interfaces and protocols that your application should be compatible with, such as web services, APIs, and messaging systems.
- **Compliance:** Identify any regulatory or industry standards that your application must comply with, such as HIPAA, PCI, or GDPR.

3. SYSTEM ANALYSIS

3.1 *Proposed System*

Online Mock tests have become increasingly popular as a means of helping students and job seekers prepare for competitive exams. There are several reasons why online mock tests are useful:

- **Simulates the actual exam environment:** Online mock tests simulate the actual exam environment, which can help users become familiar with the types of questions, time limits, and overall structure of the exam. This can help users reduce exam anxiety and perform better on the actual exam.
- **Provides instant feedback:** Online mock tests provide instant feedback to users, allowing them to see which questions they answered correctly and incorrectly. This can help users identify their strengths and weaknesses and focus their study efforts accordingly.
- **Saves time and effort:** Online mock tests can save users time and effort by providing a convenient and efficient way to prepare for exams. Users can take mock tests at any time and from anywhere, without having to travel to a physical testing center.
- **Cost-effective:** Online mock tests can be more cost-effective than traditional exam preparation methods, such as attending coaching classes or purchasing study materials. This can make exam preparation more accessible to a wider range of users.
- **Increases confidence:** Online mock tests can help users increase their confidence by providing them with a sense of familiarity and preparedness for the actual exam. This can help users perform better on the actual exam and achieve their goals.

3.1.1 **Defining the Problem**

The proposed system is an online mock test platform that will enable students to take practice tests for various competitive exams. With the increasing popularity of online exams, there is a growing demand for online mock tests that can simulate the actual exam environment. However, existing online mock test platforms have limitations such as limited question banks, poor user experience, and lack of flexibility in test customization. Therefore, there is a need for a comprehensive online mock test platform that addresses these limitations.

3.1.2 **Developing Solution Strategies**

- To address the limitations of existing online mock test platforms, the proposed system will have the following features:
- **Comprehensive question bank:** The platform will have a large question bank with questions for various competitive exams such as JEE, NEET, UPSC, SSC,

and more. This will ensure that students have access to a diverse set of questions to practice.

- Customizable tests: The platform will allow users to create customized tests by selecting questions from the question bank. This will enable students to focus on specific topics or subjects.
- Exam-like environment: The platform will simulate the actual exam environment by providing a timer, disabling navigation between questions, and randomizing the order of questions.
- Performance analytics: The platform will provide detailed performance analytics such as score, time taken, and accuracy for each test. This will enable students to identify their strengths and weaknesses and improve their performance.
- User-friendly interface: The platform will have a user-friendly interface that is easy to navigate and use.

STRUCTURE OF PROJECT:

❖ Before Login

- Home Page
- Login
 - User login
 - Admin login
- Register

❖ After User Login

- Home Page
- Dashboard
- Test Page
- Logout

❖ After Admin Login

- Home Page
- Dashboard
- Test Page
- Add Question Page
- Logout

3.1.3 Flow Diagrams

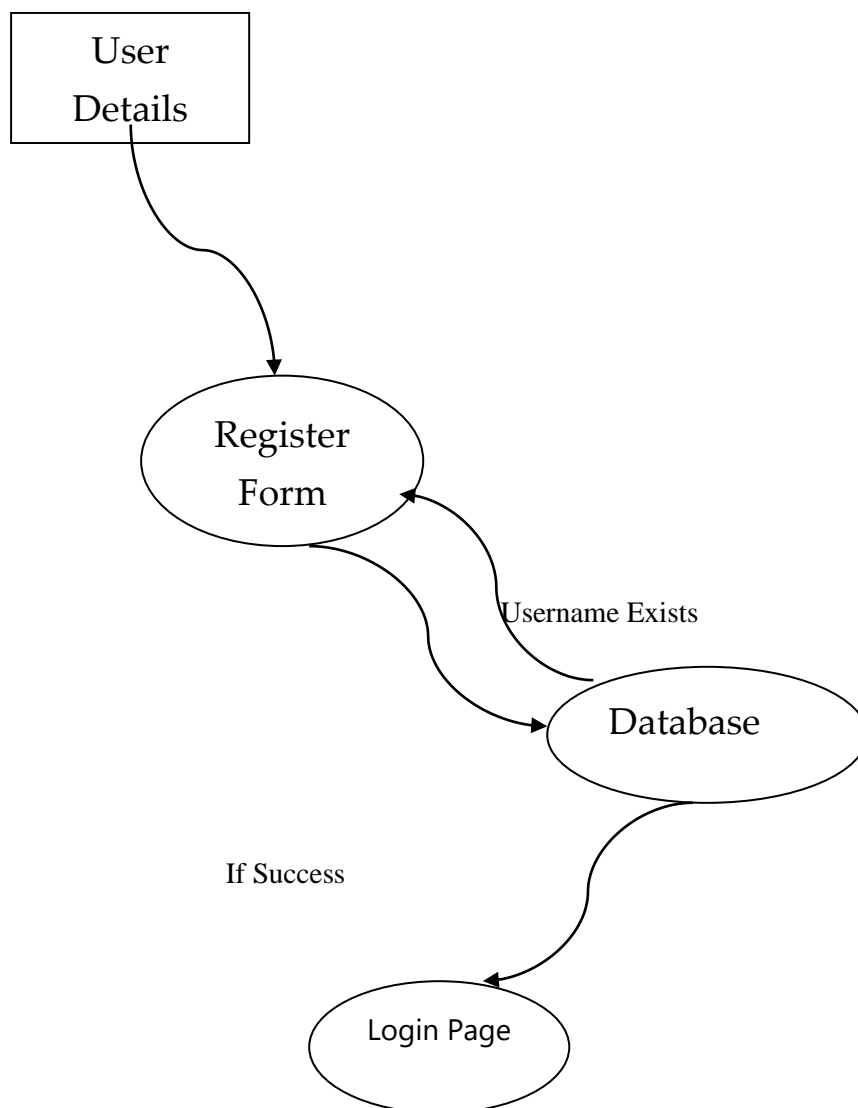
The flow diagram for the proposed system is as follows:

- User registration: Users can register on the platform by providing their basic details such as name, email, and password.

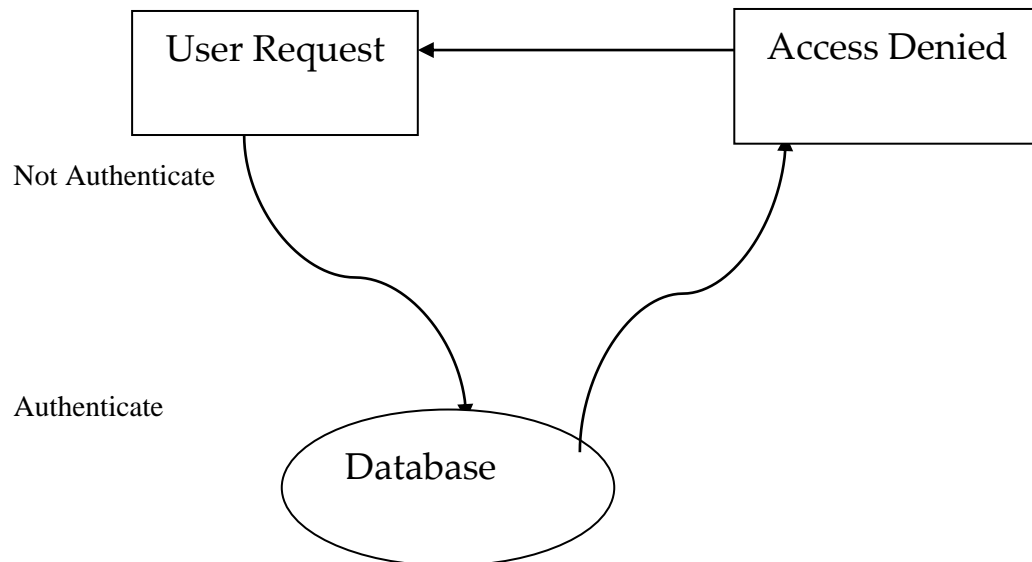
- Test selection: Users can select the test they want to take from a list of available tests.
- Test customization: Users can customize the test by selecting the number of questions, difficulty level, and specific topics or subjects they want to focus on.
- Test-taking: Users can take the test in a timed, exam-like environment.
- Performance analytics: Users can view their performance analytics after completing the test.
- Test history: Users can view their test history and performance analytics for previous tests taken on the platform.
- Payment: Users can make payments to access premium features such as access to more tests and advanced performance analytics.
- The flow diagram will ensure that the user experience is streamlined and intuitive, enabling users to take tests easily and efficiently.

(0 level DFD)

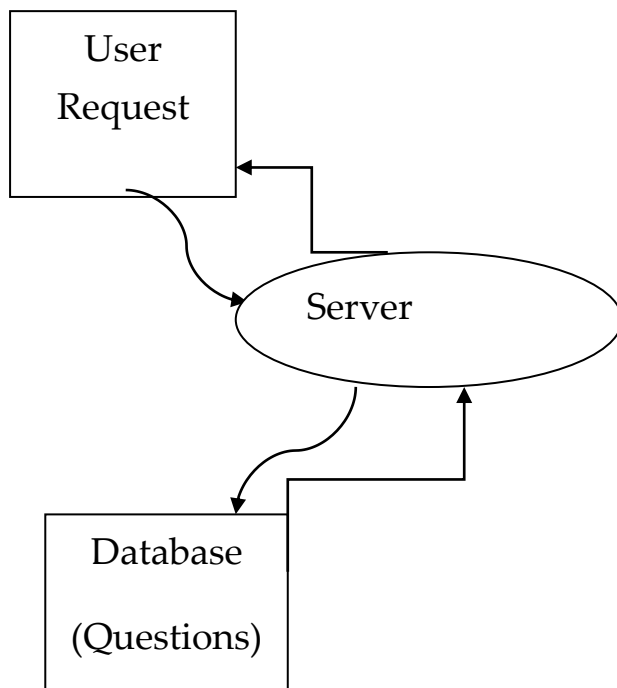
For Registration:



For Login:



For Starting Test:



3.2 System Specification

3.2.1 Hardware Specification

The hardware requirements for the proposed online mock test system are as follows:

- Processor: Intel Core i3 or higher
- RAM: 4 GB or higher
- Storage: 128 GB or higher
- Display: 1366 x 768 or higher resolution

3.2.2 Software Specification

The software requirements for the proposed online mock test system are as follows:

- Operating System: Windows 10 or higher, macOS, or Linux
- Web Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge
- Text Editor: Visual Studio Code, Sublime Text, or Atom
- Server-Side Framework: Node.js with Express.js
- Database: MongoDB or MySQL

These software specifications are necessary for developing and deploying the online mock test system. The server-side framework and database are crucial for creating a dynamic and responsive web application. The choice of text editor is up to the developer's preference, but it should be capable of handling HTML, CSS, and JavaScript files. Additionally, a modern web browser is necessary for testing and using the system.

4. SOFTWARE DESIGN

4.1 Interface Design:

The interface design of the Online Mock Test system is developed to provide a user-friendly experience for the users. The design is developed to be simple and efficient, keeping in mind the user's perspective. The interface design is developed using HTML and CSS, which is compatible with various devices like desktops, laptops, and mobiles.

The interface design consists of multiple pages, which are designed in such a way that the user can navigate between them easily. The pages are designed using a responsive design approach, which adjusts the page layout according to the device used to access the system. The system also provides interactive features like buttons, forms, and drop-down lists to enhance the user experience.

4.1.1 Home Page

The home page consists of a header, body, and footer section. The header section contains the logo of the system and a navigation menu. The navigation menu includes links to various pages like home, about us, contact us, etc.

The body section of the home page contains a large banner with a title and a subtitle. Below the banner, there are two sections. The left section has a 'Get Started' button, which redirects the user to the signup page. The right section has a 'Login' button, which redirects the user to the login page.

TestPrep

Home

About

Contact

Login

Welcome To TestPrep

A Place To Test Your Exam Preparation

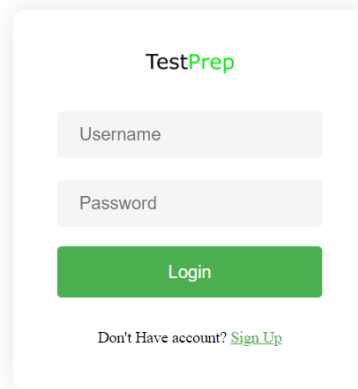
Get Started >>

1000+ user already using This



4.1.2 Login Page

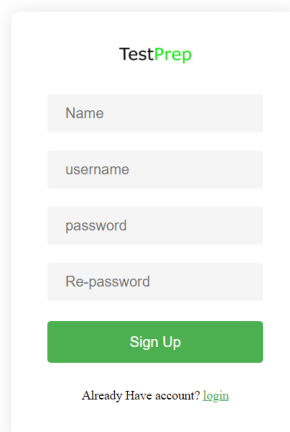
The login page has a simple design with a login form in the center of the page. The form has two input fields for email and password and a submit button. The page also includes a link to the signup page for new users and a forgot password link.



The login page mockup features the TestPrep logo at the top. Below it are two input fields labeled 'Username' and 'Password'. A green 'Login' button is positioned below the password field. At the bottom, there is a link that reads 'Don't Have account? [Sign Up](#)'.

4.1.3 Signup Page

The signup page has a similar design to the login page. It also has a form in the center of the page with input fields for name, email, password, and confirm password. The form also includes a submit button and a link to the login page for existing users.



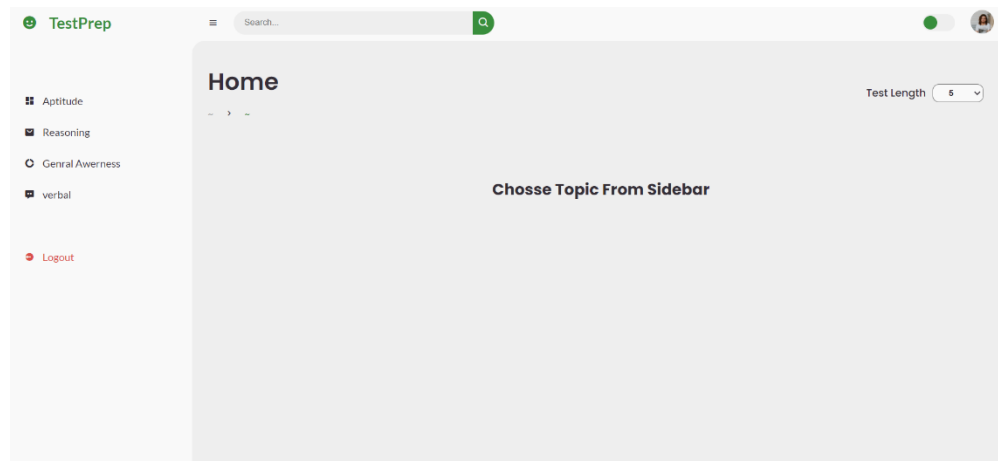
The signup page mockup features the TestPrep logo at the top. Below it are four input fields labeled 'Name', 'username', 'password', and 'Re-password'. A green 'Sign Up' button is positioned below the 'Re-password' field. At the bottom, there is a link that reads 'Already Have account? [login](#)'.

4.1.4 Dashboard Page

The dashboard page is an HTML document that serves as the main interface of a web application called "TestPrep". It consists of a sidebar and a content section. The sidebar contains the brand logo, navigation menu with links to different topics such as Aptitude, Reasoning, General Awareness, and Verbal. The bottom menu in the sidebar contains a logout link.

The content section includes a navbar with a search bar, a switch mode button, and a user profile icon. The main section includes a heading with a dropdown list of test length and a box-info section that displays "Choose Topic From Sidebar" text.

The HTML document also includes links to external CSS and Boxicons libraries, as well as a JavaScript file that is used to manipulate the DOM elements in the HTML document.



4.1.5 Test Page

The above HTML code defines the structure of a test page. It consists of a set of rules and a "Start Test" button. When the user clicks on the "Start Test" button, the timer will start and questions will be displayed on the page.

After the test is completed, the results will be displayed in a table with information about the total number of questions, the number of correct, incorrect and unattempted questions, and the score. There is also a button to print the questions and results.

This HTML code uses an external CSS file and JavaScript file to define the styling and behavior of the test page respectively.

Rules:

- Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem laborum eos rem sequi veniam necessitatibus aut, tempore est amet assumenda provident velit nostrum officia enim omnis vero aperiam ad incididunt.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem laborum eos rem sequi veniam necessitatibus aut, tempore est amet assumenda provident velit nostrum officia enim omnis vero aperiam ad incididunt.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem laborum eos rem sequi veniam necessitatibus aut, tempore est amet assumenda provident velit nostrum officia enim omnis vero aperiam ad incididunt.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem laborum eos rem sequi veniam necessitatibus aut, tempore est amet assumenda provident velit nostrum officia enim omnis vero aperiam ad incididunt.

Test Length: 5

Start Test

02:55

Question 1

Find the average of 1.11, 0.01, 0.101, 0.001, 0.11

- ☐ 0.2664
- ☐ 0.2554
- ☐ 0.1264
- ☐ 0.1164

Next

1

Submit

Test Result:

Topic	APTITUDE (Average)
TotalQuestions	5
Correct	0
Wrong	0
Unattempted	5
Score	0 / 5

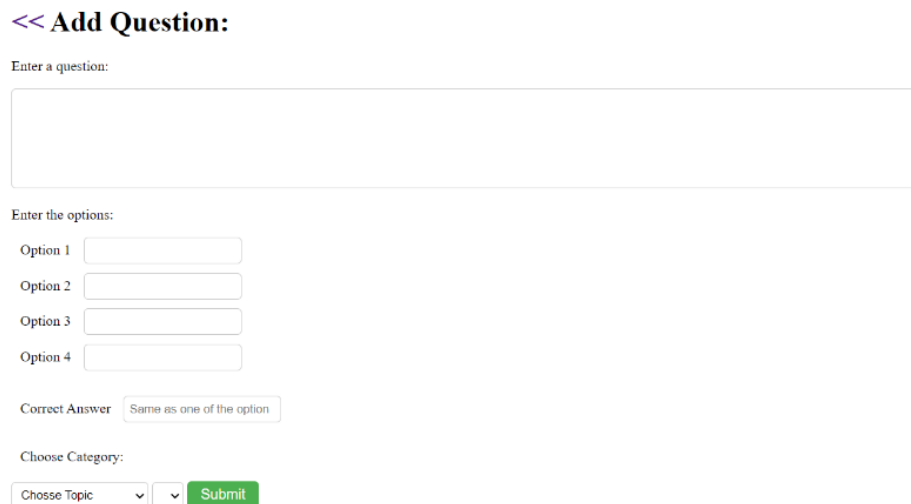
Print

4.1.6 Add Question Page (Admin Only)

The interface design of the "addQuestion" page is simple and intuitive, with a clear layout that makes it easy for users to input the necessary information. The design approach used is responsive design, which means that the interface is optimized for use on various devices, including desktops, laptops, tablets, and smartphones. This ensures that users can access the system from any device without encountering any issues with the interface.

The page uses HTML, CSS, and JavaScript programming languages to create an interactive and responsive interface. The HTML provides the structure of the page, while CSS defines the style and layout of the elements. JavaScript is used for validation and interactivity, such as checking if the required fields are filled and displaying error messages if necessary.

Overall, the interface design of the "addQuestion" page is user-friendly, with a simple and responsive layout that ensures a seamless user experience.



The screenshot shows a web form titled "<< Add Question:". It includes a large text input field for the question, followed by four smaller input fields for options labeled "Option 1" through "Option 4". Below these is a "Correct Answer" section with a dropdown menu currently set to "Same as one of the option". At the bottom, there is a "Choose Category:" section with a "Chosse Topic" dropdown and a green "Submit" button.

4.2 Database Design:

The database design of the Online Mock Test system is developed using MySQL. The database consists of multiple tables to store user data and question data. The database is designed to efficiently store and retrieve data, ensuring data consistency and integrity.

The database design consists of the following tables:

User table:

The user table stores user data like Name, Usernam, and password. It has the following columns:

- name: The name of the user
- username(primary Key): The username of user

- password: The hashed password of the user

Server: 127.0.0.1 » Database: project » Table: users

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	username	text	utf8mb4_general_ci		No	None			Change Drop More
2	password	text	utf8mb4_general_ci		No	None			Change Drop More
3	name	text	utf8mb4_general_ci		No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

Question tables:

The questions table stores question data like question, options, and answer. It has the following columns:

- question_id (primary key): A unique identifier for each question
- question: The actual question
- option1: The first option for the question
- option2: The second option for the question
- option3: The third option for the question
- option4: The fourth option for the question
- correct_ans: The correct answer for the question.
- topic: subtopic of main topic.

Server: 127.0.0.1 » Database: project » Table: aptitude

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	question	text	utf8mb4_general_ci		No	None			Change Drop More
3	option1	text	utf8mb4_general_ci		No	None			Change Drop More
4	option2	text	utf8mb4_general_ci		No	None			Change Drop More
5	option3	text	utf8mb4_general_ci		No	None			Change Drop More
6	option4	text	utf8mb4_general_ci		No	None			Change Drop More
7	correct_ans	text	utf8mb4_general_ci		No	None			Change Drop More
8	topic	text	utf8mb4_general_ci		No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

...Other three table are also similar to this one.

4.3 Coding:

The coding for the Online Mock Test system is developed using HTML, CSS, JavaScript, and Node.js (Express.js) for server-side scripting. The code is modularly designed, ensuring easy maintenance and scalability of the system.

The code for the system consists of multiple modules that perform specific tasks, including:

4.3.1 Add Questions Module (for admin only):

This module is a function named `addDataToSQL` that takes in parameters for a question, options, correct answer, main topic, and subtopic. The function's purpose is to insert this data into a MySQL database table, with the main topic being the name of the table.

The function uses a SQL query to insert the data into the table, using placeholders for the question, options, correct answer, and subtopic parameters. The placeholders are filled in by passing an array of values to the `connection.query()` method, which is part of the MySQL module for Node.js.

If there are any errors during the insertion process, an error will be thrown. Otherwise, the function will log a message to the console indicating that the data was added successfully. This module is likely used to add new questions to the system's question bank, allowing administrators to expand the available pool of questions for users to be tested on.

```
const addDataToSQL = (question, options, correct_ans, maintopic, subtopic) => {
  const sql = `INSERT INTO ${maintopic} (question, option1, option2, option3, option4, correct_ans, topic) VALUES (?, ?, ?, ?, ?, ?, ?)`;
  connection.query(sql, [question, options[0], options[1], options[2], options[3], correct_ans, subtopic], function (error, results, fields) {
    if (error) throw error;
    console.log("Data added successfully");
  });
};
```

End Point this call this function:

```
router.post('/add', function (req, res) {
  console.log(req.body)
  const { question, options, correct_ans, mainTopic, subTopic } = req.body
  addDataToSQL(question, options, correct_ans, mainTopic, subTopic)
  res.status(200).json({ status: 'success' });
})
```

Fetch request from frontend for this end point:

```
const Question = document.querySelector("#question-area")
const option1 = document.getElementById('option1')
const option2 = document.getElementById('option2')
const option3 = document.getElementById('option3')
const option4 = document.getElementById('option4')
const correct_ans = document.getElementById('correct_ans')
const options = [formatString(option1.value), formatString(option2.value), formatString(option3.value), formatString(option4.value)]

if (!Question.value || !option1.value || !option2.value || !option3.value || !option4.value || !correct_ans.value || !mainTopic.value || !subTopic.value) {
    alert("Please fill in all fields");
    return;
}
if (!options.includes(correct_ans.value)) {
    alert("correct ans must be from 4 options")
    return
}

fetch('/api/add', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        question: formatString(Question.value),
        options: options,
        correct_ans: formatString(correct_ans.value),
        mainTopic: mainTopic.value,
        subTopic: subTopic.value
    })
}).then(response => response.json()).then(data => { //process data send from sever})
```

4.3.2 Get Question Module:

This module is responsible for retrieving questions from the database for a given topic and subtopic.

The GetQuestions function takes three parameters: maintopic which is the main topic of the question (e.g. aptitude, reasoning), subtopic which is the subtopic (e.g. average, number series), and length which is the number of questions to retrieve.

First, an empty array called Questions is created to store the retrieved questions. Then, a SQL query is constructed using the main topic, subtopic, and length parameters to retrieve random questions from the corresponding table in the database.

The connection.query function is used to execute the query asynchronously and return the results. The results are then iterated over using a forEach loop to create an object for each question with its id, question text, options, correct answer, and topic. These objects are pushed to the Questions array.

Finally, the Questions array is returned with the retrieved questions.

```

const GetQuestions = async (maintopic, subtopic, length) => {
  const Questions = []
  const query = `SELECT * FROM ${maintopic} WHERE topic = '${subtopic}' ORDER BY RAND() LIMIT ${length}`

  const results = await new Promise((resolve, reject) => {
    connection.query(query, (error, results, fields) => {
      if (error) {
        reject(error)
      } else {
        resolve(results)
      }
    })
  })

  results.forEach(result => {
    const item = {
      "id": result.id,
      "question": result.question,
      "options": [
        result.option1,
        result.option2,
        result.option3,
        result.option4,
      ],
      'ans': result.correct_ans,
      'topic': maintopic
    }
    Questions.push(item)
  })

  return Questions
}

```

End Point That call this function:

```

router.get('/questions', async function (req, res) {
  const testTopic = (req.query.Topic)
  const testSubTopic = (req.query.subTopic)
  const testlength = Number(req.query.testlength)
  if (!testlength) {
    return
  }
  let questions = await GetQuestions(testTopic, testSubTopic, testlength)

  res.send({
    'code': 200,
    'success': 'Data fetched successfully',
    'data': questions
  })
})

```

Fetch request from frontend for this end point:

```
const params = new Proxy(new URLSearchParams(window.location.search), {
  get: (searchParams, prop) => searchParams.get(prop),
});

let __Topic = params.Topic;
let __SubTopic = params.subTopic;
let __testlength = params.testlength;
const QuestionData = fetch(`/api/questions?testlength=${__testlength}&Topic=${__Topic}&subTopic=${__SubTopic}`)
  .then(response => response.json())
```

Each module is designed using best coding practices, ensuring code readability and maintainability. The system also includes input validation to prevent data inconsistencies or security vulnerabilities. Additionally, the code is optimized for fast and efficient data retrieval and processing, providing a smooth user experience.

5. IMPLEMENTATION

Implementation is a crucial phase of the software development life cycle (SDLC) where the designed system is developed and tested. In this phase, the project team converts the design into an operational system by writing the code, installing software components, and integrating the system modules.

In our project, we implemented the designed system using the agile methodology, which involves continuous feedback and iterations between the project team and stakeholders. We started by creating a detailed project plan, including the task breakdown, timeline, and resource allocation. The project plan was regularly updated throughout the implementation phase to ensure that we stay on track and meet the project objectives.

The implementation phase involved writing the code for the system, integrating the different modules, and testing the system for bugs and errors. We used a combination of programming languages and technologies to implement the system, including MySQL, HTML, CSS, and JavaScript. We also used popular open-source frameworks and libraries such as Express , Nodejs to speed up the development process and ensure code quality.

During the implementation phase, we faced several challenges, such as database schema changes, integration issues, and compatibility issues with third-party libraries. However, we were able to overcome these challenges by using best practices in software development, including version control, unit testing, and continuous integration.

Overall, the implementation phase was a success, and we were able to deliver a fully functional system that met the project requirements and objectives. The system was thoroughly tested for functionality, performance, security, and user experience, and we were able to resolve all issues before the system was deployed to the production environment.

Post Implementation Review

After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against predefined requirement. It determines how well the system continues to meet the performance specifications.

6. TESTING

Testing is a critical part of software development, and our project was no exception. To ensure that the system functions as intended and meets the requirements, we employed various testing techniques.

6.1 Techniques Used in Testing

We used both manual and automated testing techniques to ensure the quality of our system. The testing techniques used are as follows:

Unit Testing: We performed unit testing to test each unit or component of the system in isolation. This helped us to identify any defects or issues early on in the development cycle.

Integration Testing: We also performed integration testing to test the interaction between different modules of the system. This helped us to identify any defects or issues that may arise due to the integration of different modules.

System Testing: We conducted system testing to test the entire system as a whole. This helped us to ensure that all the modules of the system are integrated correctly and function as intended.

Acceptance Testing: We performed acceptance testing to ensure that the system meets the requirements and is ready for deployment.

6.2 Test Case

We created test cases for each module of the system to ensure that the system functions as intended. Here are some examples of the test cases we performed:

Unit Test Case for Login Functionality:

- Verify that the user is able to enter the correct username and password to log in successfully.
- Verify that the user is not able to log in with an incorrect username or password.
- Verify that the user is redirected to the dashboard page after successful login.
- Integration Test Case for Adding Questions:
- Verify that the user is able to add a question with all the required fields filled in.
- Verify that the system displays an error message if the user tries to add a question without filling in all the required fields.
- Verify that the added question is displayed in the list of questions.

System Test Case for Generating Results:

- Verify that the system is able to generate Result for each Test.
- Verify that the Result are generated accurately and display the required information.
- Verify that the Result are generated in the correct format.

- Acceptance Test Case for Test Taking:
- Verify that the user is able to start a Test and answer all the questions.
- Verify that the system displays the correct answers after the user has completed the Test.
- Verify that the system calculates the user's score correctly and displays it to the user.

We performed many more test cases to ensure the quality of our system. We used various tools such as Selenium, JUnit, and TestNG for automated testing. We also performed manual testing to ensure that the system functions correctly from a user's perspective. Overall, our testing efforts helped us to identify and fix defects early on in the development cycle, ensuring a high-quality final product.

LOGIN FOR USER:

Serial No	Description	Expected Result	Actual Result	Result
1.	This page contains 2 fields user name and password and a login button to submit the information. User is entering correct information.	Dashboard Page should open after successful login.	Dashboard Page is opening after successful login by user.	Passed
2.	If either user name or password is filled incorrect or left blank.	An alert msg should be displayed and user should be asked fill the information again.	When wrong information is entered by user then an error alert is displayed.	Passed

USER REGISTRATION PAGE:

Serial No	Description	Expected Result	Actual Result	Result
1.	User registration page 1 consist of detail information about User and a submit button to submit the information .Here user is entering correct information.	After submitting information User registration page 2 should be displayed.	After submitting information User registration page 2 is displayed.	Passed
2.	If the information entered by user in incorrect or left somewhere blank.	An error message should be displayed and ask the user to fill the information again.	An error message is occurred if the information is incorrect or left blank.	Passed

7. MAINTENANCE

Maintenance is a critical part of the software development life cycle, and it ensures that the software continues to function as intended and meet the user's changing needs. For the mock test project, maintenance may involve several activities, including:

1. **Bug Fixing:** Bugs may be identified after the software is deployed, and maintenance involves fixing these bugs to ensure the software is functioning as intended.
2. **Updates and Enhancements:** As user needs change over time, new features and updates may need to be added to the mock test software. Maintenance involves identifying the necessary updates and enhancements and implementing them.
3. **Security Updates:** Security is a significant concern for any software application, and maintenance may involve implementing security updates to ensure the mock test software is secure and free from vulnerabilities.
4. **Performance Optimization:** The mock test software may need to be optimized for better performance. Maintenance involves identifying areas where the software can be optimized and implementing changes to improve performance.
5. **Compatibility Issues:** Over time, the mock test software may encounter compatibility issues with other software or operating systems. Maintenance involves identifying these issues and implementing changes to ensure compatibility.

Overall, maintenance is a crucial part of ensuring the mock test software is functioning as intended and meeting the user's changing needs. It is essential to allocate resources and budget for maintenance activities to ensure the software's long-term success.

8. BIBLIOGRAPHY AND REFERENCES

8.1 Bibliography:

- GeeksforGeeks. (n.d.). GeeksforGeeks: A computer science portal for geeks. Retrieved March 17, 2023, from <https://www.geeksforgeeks.org/>
- Google LLC. (n.d.). Google. Retrieved March 17, 2023, from <https://www.google.com/>
- YouTube LLC. (n.d.). YouTube. Retrieved March 17, 2023, from <https://www.youtube.com/>

8.2 References:

- GeeksforGeeks. (2021, July 30). Data Structures and Algorithms. YouTube. <https://www.youtube.com/playlist?list=PLqM7alHxFySGqCvcwfqqMrteqWukz9ZoE>
- Google LLC. (2022). About Google. Google. <https://about.google/intl/en/>