A FIELD PROJECT REPORT ON

# QUIZ APPLICATION FOR GENERAL KNOWLEDGE ON CODING

**Submitted**

*In partial fulfillment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE ENGINEERING**

by

231FA04B83 : V.YASASVI LAKSHMI SAI

231FA04C13 : KRISHNA

231FA04C70 : HEMA SRI

231FA04F83 : TEJA SRI

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

Under the Guidance of

Dr. Nerella Sameera

Assistant Professor, CSE

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF COMPUTING AND INFORMATICS**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH**

**(Deemed to be University)**

**Vadlamudi, Guntur -522213, INDIA**

**APRIL-2025.**

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

## CERTIFICATE

This is to certify that the field project entitled "**QUIZ APPLICATION FOR GENERAL KNOWLEDGE ON CODING**" being submitted by **V.Yasasvi Lakshmi sai (231FA04B83), T.Krishna (231FA04C13), M.Hema sri (231FA04C70) ,R.Teja sri (231FA04F83)** in partial fulfilment of **Bachelor of Technology(B.tech)** in the **Department of Computer science engineering, Vignans Foundation For Science Technology & Research (Deemed to be University),** Vadlamudi, Guntur District, Andhra Pradesh, India,

This is a bonafide work carried out by them under my guidance and supervision.

[illegible]
**Guide**

[illegible]
**Project Review Committee**

[illegible]
**HoD, CSE**

HoD
Dept. of Computer Science & Engineering
VFSTR Deemed to be University
VADLAMUDI - 522 213
Guntur Dist., A.P., India.

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

## DECLARATION

**Date: 26-04-2025**

We hereby declare that the work presented in the field project titled **"QUIZ APPLICATION FOR GENERAL KNOWLEDGE ON CODING"** is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Dr. Nerella Sameera, Assistant Professor** in partial fulfilment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

| | | |
|---|---|---|
| V.Yasasvi Lakshmi sai | 231FA04B83 | [illegible] |
| Krishna | 231FA04C13 | [illegible] |
| M.Hema sri | 231FA04C70 | [illegible] |
| R.Teja sri | 231FA04F83 | [illegible] |

# CHAPTER-01
# INTRODUCTION

# INTRODUCTION

Programming languages like C, Java, and Python form the backbone of modern software development. Each of these languages has its own syntax, logic, and application areas, making them essential for students and professionals alike. This Quiz Application is designed to help users test and improve their knowledge of these three core programming languages. With a user-friendly interface and a range of carefully crafted multiple-choice questions, the application serves as an interactive learning tool for coding enthusiasts.

## 1.1 Problem Definition

The goal is to develop a Quiz Application that enables users to test their fundamental knowledge of three popular programming languages: C, Java, and Python. The application should present a series of multiple-choice questions related to these languages and allow users to select answers, calculate scores, and view results.

## 1.2 Existing System

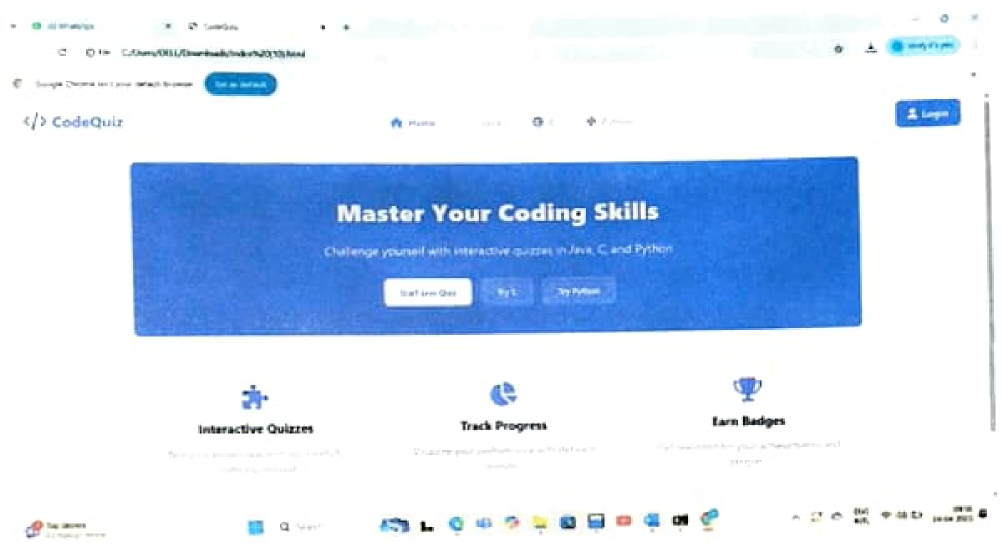
As a result, users who are beginners or looking to test their basic knowledge might find it difficult to measure their understanding effectively.
Currently, the existing systems for quiz-based applications in coding mostly rely on simple question-and-answer interfaces, where users can attempt a series of multiple-choice questions. These systems, however, often lack the following key features

## 1.3 Proposed System

1. Multiple Choice Questions (MCQs):
    - Questions are related specifically to C, Java, and Python.
    - Each question has four options with only one correct answer.
2. Language Selection:
    - Users can select a quiz category based on the programming language they want to be tested on.
3. Time Limit Per Question:
    - A timer is set for each question (e.g., 30 or 60 seconds).
    - If the user fails to answer within the time limit, the question is marked as unanswered.
    - Helps simulate real-time test conditions and improves time management skills.
4. Score Calculation:
    - The application calculates and displays the total score at the end based on correct answers.
5. Answer Summary Graph:
    - After the quiz, users are shown a graphical summary (e.g., bar chart or pie chart) of:

        - Total Questions
        - Questions Attempted
        - Correct Answers
        - Incorrect Answers
    - This feature helps users visually analyse their performance.

6. Result Display:
    - The final result includes the total score and optionally the time taken and accuracy percentage.



## 1.4 Literature Review

Programming quizzes are becoming increasingly popular as a method for evaluating coding knowledge. A study by **Liu et al. (2019)** explored the impact of coding quizzes on learning programming languages, specifically **C** and **Java**. The research found that quizzes focusing on specific programming language concepts significantly boosted students' understanding and retention. This highlights the importance of **language-specific quizzes**, which provide a focused, concise, and effective learning environment for beginners and advanced programmers alike.

Additionally, **Kapoor and Sharma (2020)** discussed the effectiveness of **timed coding challenges** in coding competitions, showing that time constraints during quizzes simulate real-world coding scenarios and prepare students for **interviews and exams**. This aligns with our proposed feature of time-limited questions in the application, which can enhance decision-making under pressure.

Studies like those by **Hattie and Timperley (2007) on** the power of feedback suggest that immediate and constructive feedback enhances learning significantly. In the case of coding quizzes, providing feedback not only on whether the solution is correct but also on the efficiency and optimization of the code could encourage students to refine their problem-solving approaches. Moreover, using automated feedback systems powered by AI can offer personalized suggestions for improvement based on the student's performance in real-time.

A study by **Liu et al. (2019**) explored the impact of coding quizzes on learning programming languages, specifically C and Java. The research found that quizzes focusing on specific programming language concepts significantly boosted students' understanding and retention. This highlights the importance of language-specific quizzes, which provide a focused, concise, and effective learning environment for beginners and advanced programmers alike. Language-specific quizzes allow students to master foundational concepts and syntax before applying them in more complex scenarios, reinforcing core skills and improving long-term retention.

# CHAPTER-02
# SYSTEM REQUIREMENTS

# 2 : SYSTEM REQUIREMENTS

## Front-End Requirements

The front-end of the Event Management System is responsible for delivering a user-friendly and responsive interface. It allows users to interact with the system for event creation, registration, browsing events, and more. The following technologies are used:

### 1. HTML (HyperText Markup Language)

- Used to structure web pages and content.
- Forms the skeleton of the application (e.g., forms for registration, login, event details).
- Ensures semantic organization of elements (headers, buttons, tables, etc.).

### 2. CSS (Cascading Style Sheets)

- Used to style the HTML content.
- Responsible for the visual appearance: layout, fonts, colors, and responsiveness.
- Enables responsive design using media queries for compatibility across devices (desktop, tablet, mobile).

### 3. JavaScript

- Provides dynamic behavior and interactivity to the web pages.
- Used for client-side validation of forms (e.g., checking if required fields are filled).
- Handles real-time updates (e.g., countdown timers, live search).
- Can interact with APIs or back-end for real-time data fetching without reloading pages (using AJAX or Fetch API).

## Back-End Requirements

The back-end of the Event Management System handles the core functionality such as processing data, managing users, storing event details, and ensuring secure transactions. It communicates with the front-end and the database to deliver a seamless user experience.

### 1: PHP (Hypertext Preprocessor)

PHP is a widely-used open-source scripting language suited for web development.

- ✓ Use Cases:
  - Handling form submissions (event registration, login).

- Communicating with MySQL databases for CRUD operations.

## 2: Python (with Flask or Django)

Python is a powerful and flexible language, ideal for rapid development.

- Use Cases:
  - Flask (lightweight) or Django (full-featured) can be used for building the server-side logic.
  - REST API development to serve front-end requests.
  - Integrating third-party services (e.g., payment gateways, email APIs).

## 3: Node.js (JavaScript Runtime)

Node.js allows JavaScript to be used on the server-side, enabling full-stack JS development.

- Use Cases:
  - Building RESTful APIs to handle data operations.
  - Real-time features like chat support or live event updates using WebSockets.
  - Efficient handling of concurrent user requests.

## 2.1 Hardware & Software Requirements

This section outlines the minimum and recommended hardware and software configurations required to develop, deploy, and run the Event Management System efficiently.

### Hardware Requirements

| Component | Minimum Requirement | Recommended Requirement |
|---|---|---|
| Processor | Intel Core i3 or equivalent | Intel Core i5/i7 or equivalent |
| RAM | 4 GB | 8 GB or higher |
| Storage | 250 GB HDD | 512 GB SSD or higher |
| Display | 1024×768 resolution | Full HD (1920×1080) |
| Internet | Basic broadband connection | High-speed internet |

### Software Requirements

| Category | Software |
|---|---|
| Operating System | Windows 10/11, macOS, or any modern Linux distribution |
| Web Browser | Google Chrome, Mozilla Firefox, Microsoft Edge |
| Front-End Tools | HTML, CSS, JavaScript, Bootstrap (optional) |
| Back-End Options | PHP (with Apache), Python (Flask/Django), or Node.js |
| Database | MySQL / PostgreSQL / MongoDB |

# CHAPTER-03
# SYSTEM DESIGN

## 3: SYSTEM DESIGN

System Design outlines how the quiz application is structured — how its components interact, the data flow, and how users interact with the system.

### 1. User Interface (UI) - What the user sees and interacts with:

- **Quiz Start Screen:**
  - Displays instructions (e.g., time limit, number of questions).
  - Allows the user to select a programming language (Java, Python, C).
  - A "Start Quiz" button.
- **Quiz Screen:**
  - Displays the current question clearly.
  - Presents multiple-choice options for each question.
  - A timer displaying the remaining time (updates dynamically).
  - Navigation buttons (e.g., "Next," "Previous" - optional, depending on your desired flow).
  - A progress indicator (e.g., "Question 3 of 10").
- **Result Screen:**
  - Displays the total time taken.
  - Shows the number of questions attempted.
  - Indicates the number of correct answers.
  - Indicates the number of incorrect answers.
  - **Graphical Representation:** A circular chart (pie chart or donut chart) visually representing the percentage of correct and incorrect answers.
  - Option to review attempted questions (with correct answers highlighted).

### 2. Application Logic (Backend) - What makes the application work:

- **Question Management:**
  - **Storage:** A way to store the quiz questions, their options, and the correct answers for each language. This could be:
    - Simple data structures within the application
    - External files (e.g., JSON, CSV).
    - A database (more scalable for a larger number of questions).
  - **Retrieval:** Logic to fetch a set of 10 random questions for the selected language.
- **Quiz Session Management:**
  - **Initialization:** When a user starts a quiz, a session is created. This involves selecting the questions and starting the timer.
  - **State Tracking:** Keeping track of the current question, the user's answers, and the time elapsed.
  - **Answer Submission:** Handling the user's selection for each question.
  - **Timer Logic:** Implementing the 10-minute timer and triggering the end of the quiz when time runs out.
  - **Scoring:** Evaluating the user's answers against the correct answers.
- **Result Generation:**
  - Calculating the number of attempted, correct, and incorrect answers.
  - Generating the data needed for the graphical representation (e.g., percentages of correct and incorrect answers).

### 3. Data Storage (Optional, but recommended for scalability):

- **Database (e.g., SQLite, PostgreSQL, MySQL):**
  - Storing questions, options, and correct answers, potentially organized by language.
  - Could also store user scores and quiz history if you want to add those features later.
- **File System (e.g., JSON, CSV files):**
  - A simpler way to store quiz data, suitable for a smaller number of questions.

### System Flow:

1. **User Interaction:** The user opens the application and sees the "Quiz Start Screen."
2. **Language Selection:** The user selects a programming language (Java, Python, or C).

3. **Quiz Initiation:** The user clicks "Start Quiz."
4. **Backend Processing:**
    - The application logic retrieves 10 random questions for the selected language from the data storage.
    - A quiz session is created, and the 10-minute timer starts.
5. **Quiz Display:** The first question is displayed on the "Quiz Screen" along with the timer and options.
6. **Answering Questions:** The user selects an answer and proceeds to the next question. The application logic records their choices.
7. **Time Runs Out or All Questions Answered:**
    - If the timer reaches zero, the quiz ends automatically.
    - If the user answers all 10 questions, the quiz ends.
8. **Result Calculation:** The application logic calculates the results (attempted, correct, incorrect).
9. **Result Display:** The "Result Screen" is displayed, showing the statistics and the circular graph representing the performance.

## 3.1. Modules Description

### A. User Interface (UI) Module

- Language selection screen
- Quiz screen (displays questions & options)
- Result screen (score + performance chart)
- Simple, beginner-friendly layout

### B. Quiz Controller

- Controls quiz flow: next question, validate answers
- Manages the timer for each question or the full quiz
- Handles scoring logic and correctness tracking

### C. Question Bank Module

- Stores questions categorized by language
- Each entry: question, 4 options, correct answer

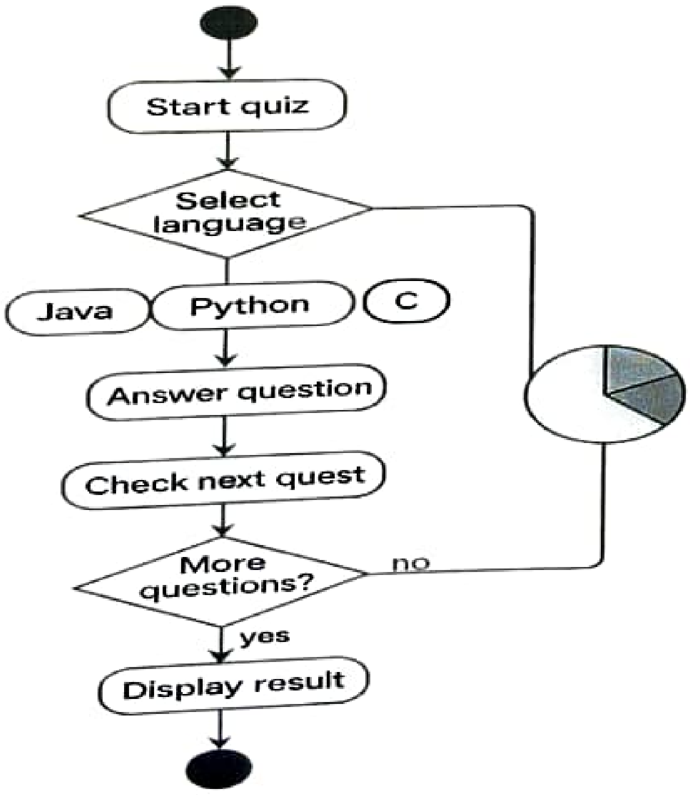
- Can be stored in a file (CSV, JSON) or database (like SQLite)

**D. Result & Graph Module**

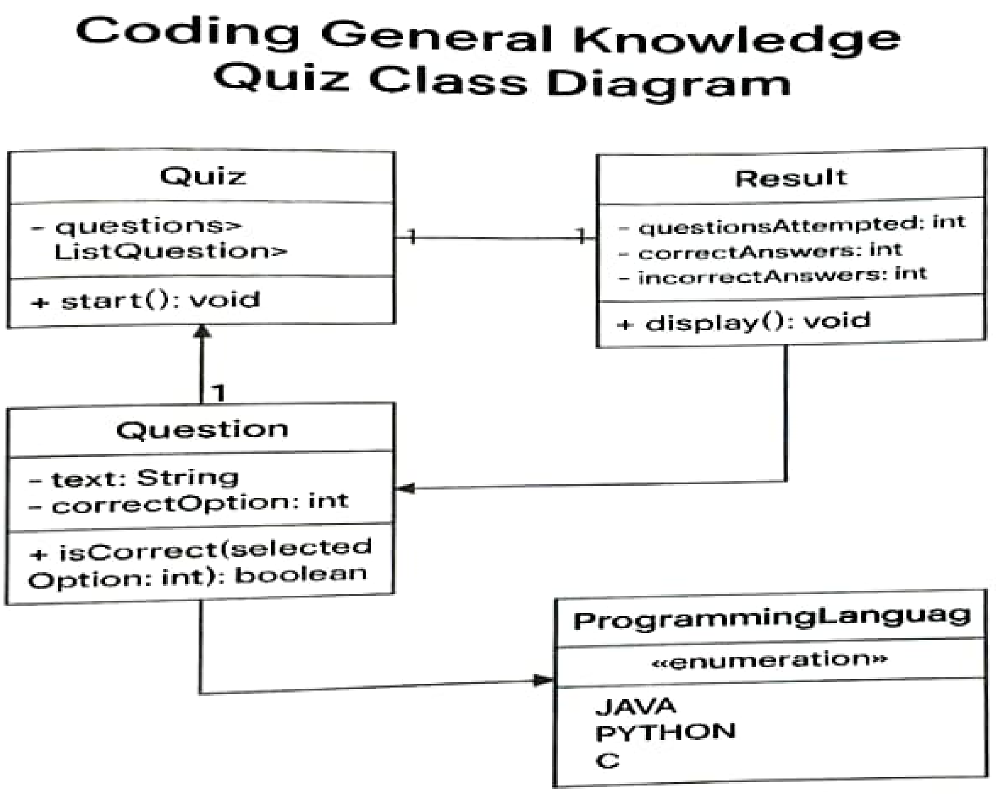
- Displays result with total score and correct answers
- Uses charts (bar/pie) to show:
    - Questions attempted
    - Correct vs incorrect answers
    - Time taken (optional)

## 3.2 UML DIAGRAMS:

## USE CASE DIAGRAM



## CLASS DIAGRAM :-

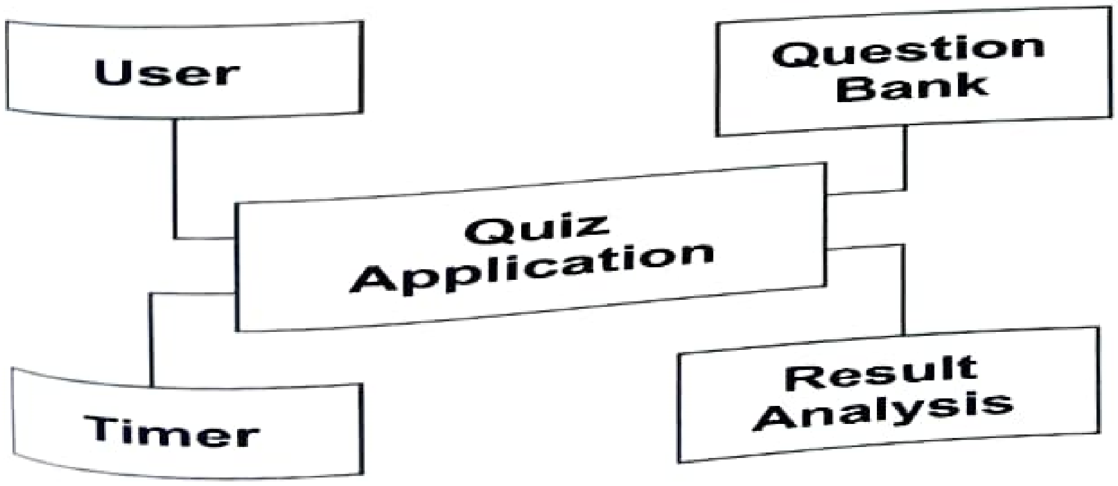


## COMPONENT DIAGRAM

**Main Components:**

1. **User Interface (UI) Component**
   - Responsible for interacting with users.
   - Handles login, quiz display, and result visualization.
   - Sends/receives data from the application core.

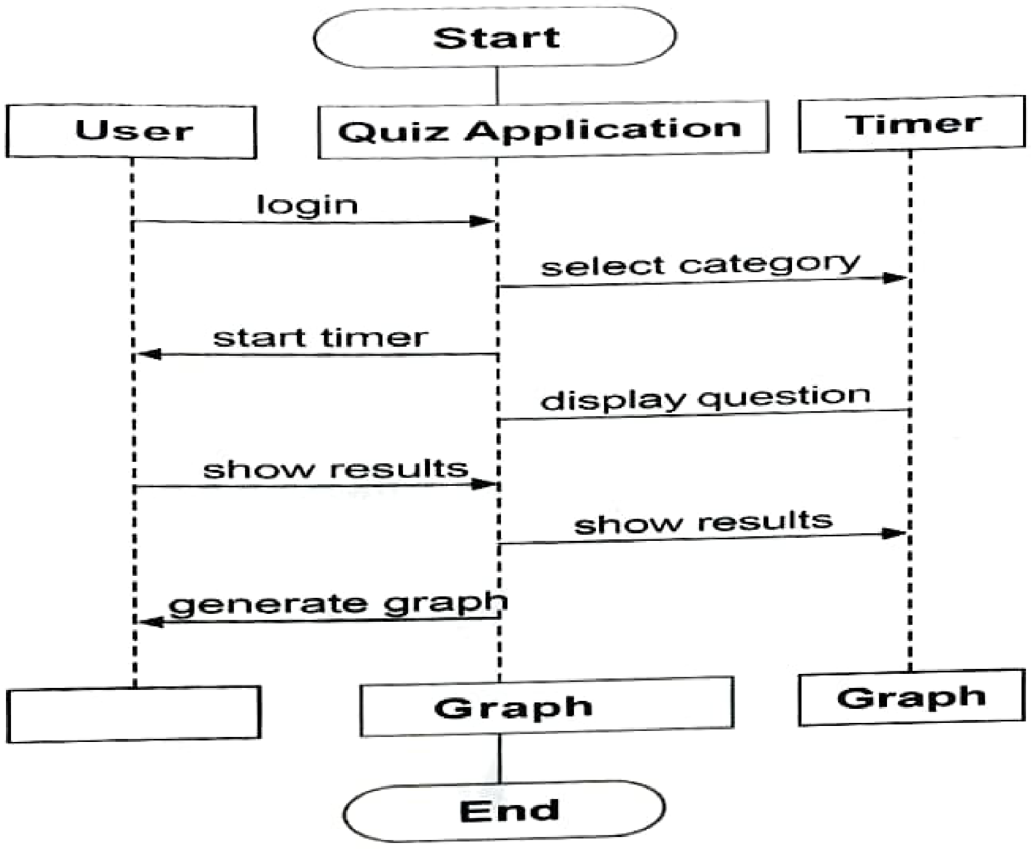
2. Quiz Manager Component
   - Central control unit of the application.
   - Handles quiz initiation, question selection, and timer control.
   - Connects the UI with the backend logic.
3. Question Bank Component
   - Stores categorized questions based on topics like Java, Python, and C.
   - Supplies questions when requested by the Quiz Manager.
4. Timer Component
   - Tracks time for each question or the entire quiz.
   - Sends timeout events to Quiz Manager to auto-submit or skip questions.
5. Result Processor Component
   - Evaluates user answers.
   - Calculates total score, correct/wrong answers.
   - Passes data to the Graph Generator.
6. Graph Generator Component
   - Converts result data into graphical formats (bar chart, pie chart).
   - Displays performance visually to the user.



## SEQUENCE DIAGRAM

Flow of Interaction:

1. User → Login Page:
   The user initiates the interaction by providing login credentials.
2. Login Page → User Manager:
   The login page sends the credentials to the user manager for authentication.
3. User Manager → DB:
   The user manager queries the database to verify credentials.
4. User Manager → Login Page:
   On success, login confirmation is sent back, and the user is directed to the quiz section.
5. User → Quiz Manager:
   The user selects a topic and starts the quiz.
6. Quiz Manager → Question Bank:
   The quiz manager fetches questions for the selected topic.
7. Quiz Manager → Timer:
   A timer starts as each question is presented to the user.
8. User → Quiz Manager:
   The user submits answers, which are evaluated in real-time.
9. Quiz Manager → Result Processor:
   At the end of the quiz, the system calculates the score and prepares the result.
10. Result Processor → Graph Generator:
    Graphs (correct/wrong answers) are generated and sent back to the user interface.
11. User → Logout:
    The user logs out or exits after viewing the result.


Start
User
Quiz Application
Timer
login
select category
start timer
display question
show results
show results
generate graph
Graph
Graph
End

# CHAPTER-04 IMPLEMENTATION

## IMPLEMENTATION

### SAMPLE CODE

**HTML :**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Quiz Application</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header><h1>General Knowledge Quiz</h1></header>
  <div class="quiz-container">
    <div id="question"></div>
    <div id="options"></div>
    <button id="next" onclick="nextQuestion()">Next Question</button>
    <div id="result"></div>
  </div>
  <footer><p>&copy; 2023 Quiz Application</p></footer>
  <script src="script.js"></script>
</body>
</html>
```

**CSS:**

```
body {
    font-family: Arial, sans-serif;
    background: #f4f4f9;
    margin: 0;
}

header {
    background: #007BFF;
    color: #fff;
    padding: 10px 20px;
    text-align: center;
}

.quiz-container {
    padding: 20px;
    text-align: center;
}

button {
    margin-top: 20px;
    padding: 10px 20px;
    background: #28a745;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
```

```
 button:hover {
   background: #218838;
 }


footer {
   background: #333;
   color: #fff;
   text-align: center;
   padding: 10px;}
```

**JAVA SCRIPT :**

```
const questions = [
  {
      question: "What is the correct syntax to output 'Hello World' in C?",
      options: ["print('Hello World');", "echo 'Hello World';", "printf('Hello World');", "cout << 'Hello
World';"],
      answer: "printf('Hello World');"
  },
  {
      question: "Which of the following is a Python data type?",
      options: ["int", "float", "list", "All of the above"],
      answer: "All of the above"
  },
  {
      question: "Which keyword is used to create a function in Java?",
      options: ["function", "def", "method", "void"],
      answer: "void"
  }
];
```

```
let currentQuestionIndex = 0;
let score = 0;

function loadQuestion() {
  const questionElement = document.getElementById('question');
  const optionsElement = document.getElementById('options');
  questionElement.innerText = questions[currentQuestionIndex].question;
  optionsElement.innerHTML = '';
  questions[currentQuestionIndex].options.forEach(option => {
    const button = document.createElement('button');
    button.innerText = option;
    button.onclick = () => checkAnswer(option);
    optionsElement.appendChild(button);
  });
}

function checkAnswer(selectedOption) {
  if (selectedOption === questions[currentQuestionIndex].answer) {
    score++;
  }
  currentQuestionIndex++;
  if (currentQuestionIndex < questions.length) {
    loadQuestion();
  } else {
    displayResult();
  }
}

function displayResult() {
  const quizContainer = document.querySelector('.quiz-container');
```

```
    quizContainer.innerHTML = `<h2>Your score: ${score}/${questions.length}</h2>`;
}

function nextQuestion() {
    if (currentQuestionIndex < questions.length) {
        loadQuestion();
    }
}

window.onload = loadQuestion;
```

## 4.2 Test Cases

1. **User Registration:**
    - Test if the user can register successfully.
    - Test if the user receives an error message for duplicate usernames.
2. **Invitation Creation:**
    - Test if the user can create a quiz invitation.
    - Test if the invitation contains the correct quiz details.
3. **RSVP Tracking:**
    - Test if the user can RSVP to the quiz invitation.
    - Test if the RSVP status is updated correctly in the system.
4. **Quiz Functionality:**
    - Test if the quiz loads the first question correctly.
    - Test if the user can select an answer and proceed to the next question.

Test if the final score is calculated and displayed correctly

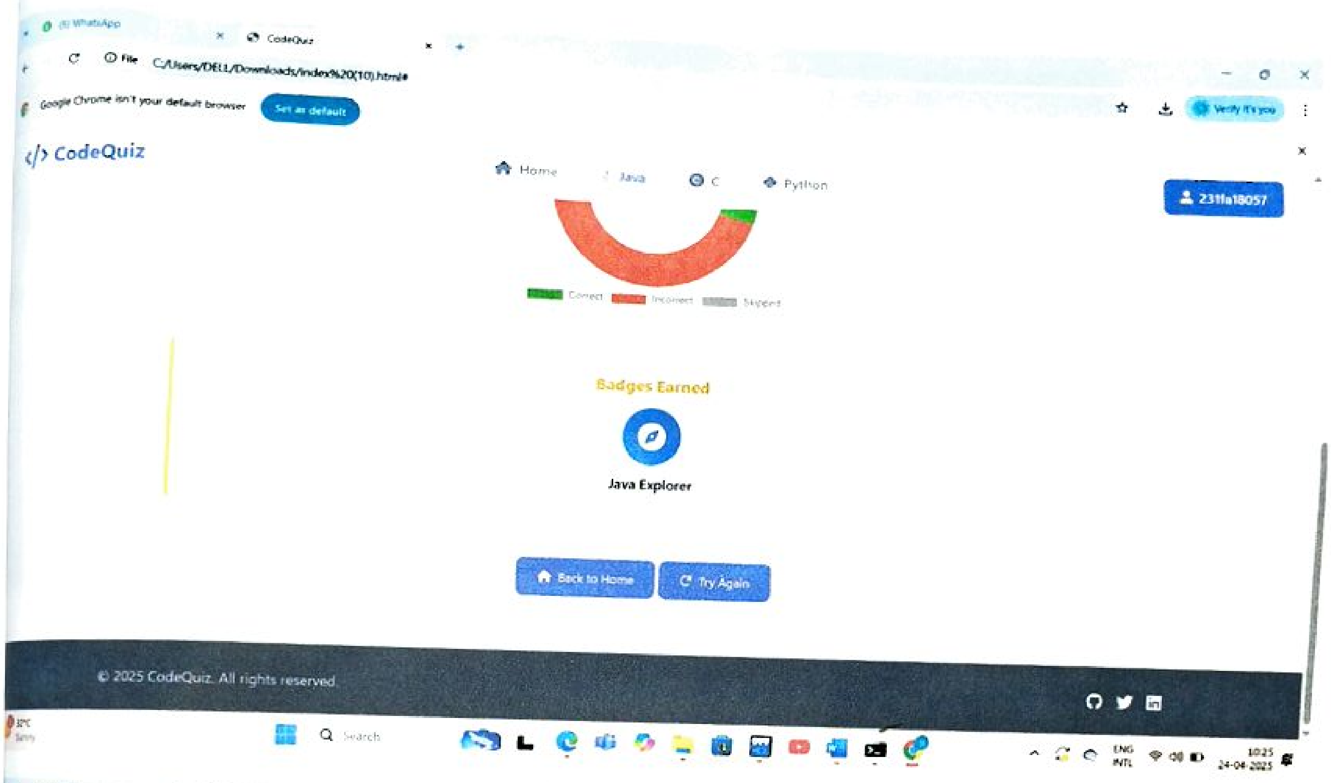
# CHAPTER-05

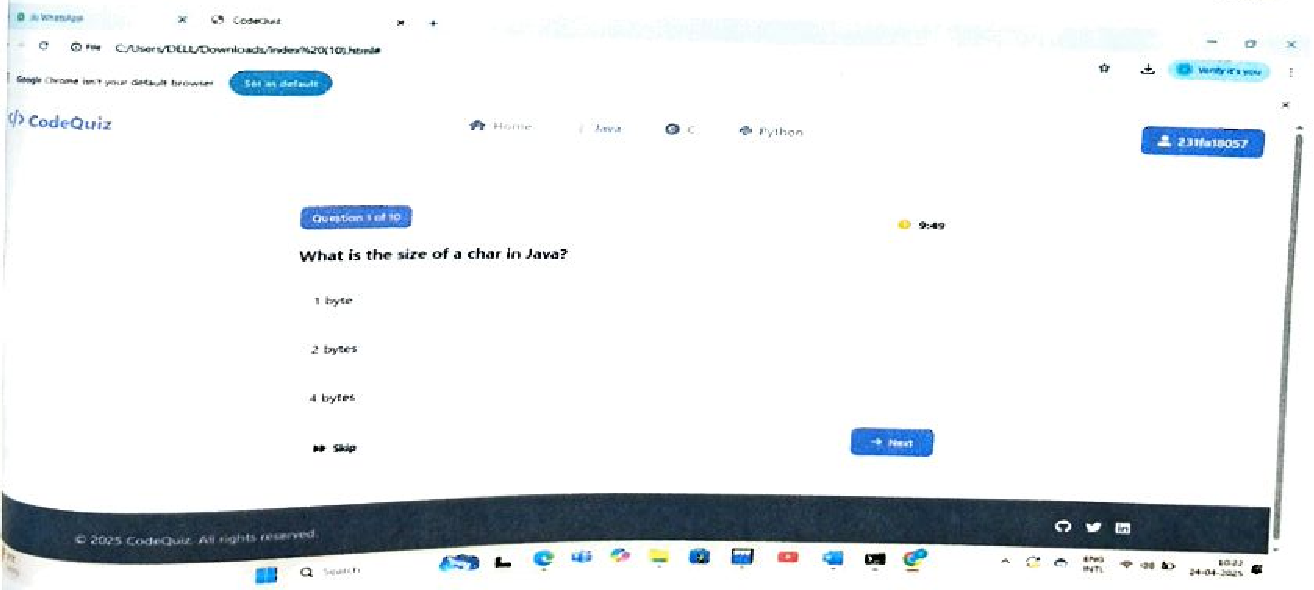
# RESULTS

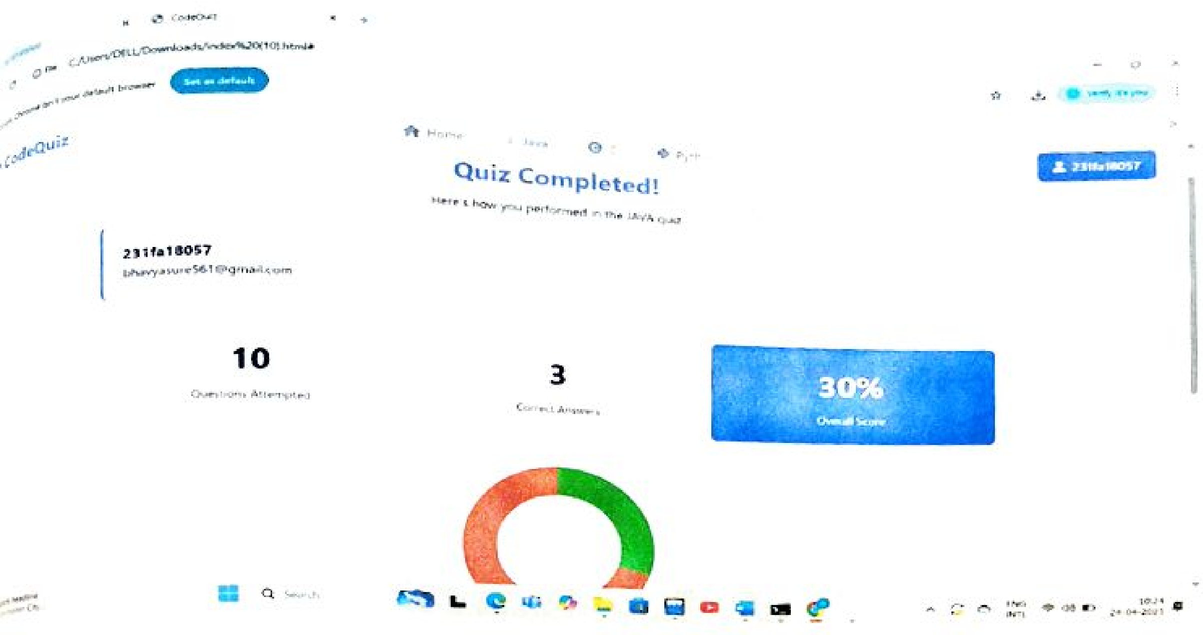
# RESULTS

## 5.1 Output Screens

Screenshots of the system's user interface






C:/Users/DELL/Downloads/index%20(10).html#

Quiz Completed!

Here's how you performed in the JAVA quiz

231fa18057
bhavyasure561@gmail.com

10
Questions Attempted

3
Correct Answers

30%
Overall Score

# CHAPTER-06
# CONCLUSION

## CONCLUSION

The Code Quiz project successfully delivers an interactive, user-friendly platform that helps users enhance their programming knowledge through quizzes in Java, C, and Python. With a visually engaging interface, progress tracking, a badge reward system, and intuitive navigation, it provides a motivating environment for learning and self-assessment. Features like a login system, quiz analytics, and personalized feedback make it suitable for both individual learners and educational environments. This project not only demonstrates effective web development skills but also emphasizes user engagement and educational impact.

**In addition to its core functionalities**, the project also integrates adaptive learning features that ensure quizzes become progressively more challenging as users improve, offering a personalized learning experience tailored to each individual's proficiency level. The **real-time feedback mechanism** is a key element, allowing users to learn from their mistakes and immediately improve their coding skills. This approach not only builds confidence but also reinforces correct problem-solving techniques, which is crucial for mastering programming languages.

## REFERENCE

- **HTML and CSS:** W3Schools
- **JavaScript:** MDN Web Docs
- **Icons:** Font Awesome
- **Charts:** Chart.js

## Research Papers on QUIZ APPLICATION FOR GENERAL KNOWLEDGE

- **Google Scholar** (scholar.google.com)
- **IEEE Xplore** (ieeexplore.ieee.org)
- **SpringerLink** (link.springer.com)

## Web Development & Quiz Application Platforms

1. **Edureka – Creating an Online Quiz Application Using JSP Servlet**
   Edureka Tutorial
2. **FreeProjectz – Online Quiz System Project**
   FreeProjectz – Online Quiz System
3. **GeeksforGeeks – Technical Wizard Quiz Application Project**
   GeeksforGeeks – Technical Wizard
4. **GitHub – Online Quiz Management System**
   GitHub Repository

**Github link : https://github.com/231FA04B83/quiz-application-for-general-knowledge**