CODE FUN

A Smart Coding Skills Development Framework

A dissertation submitted in the partial fulfilment of the academic requirements for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

Charkaman Khushi **(100520733086)**

CS Koushik **(100520733087)**

Samreen **(100520733096)**

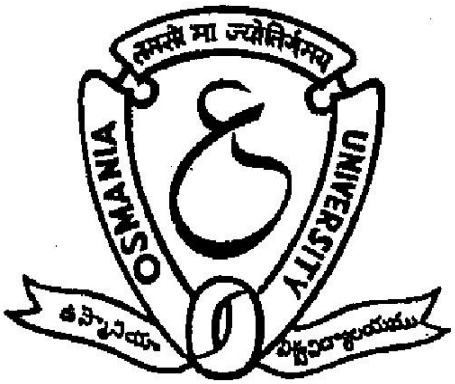
Thuppari Pranay **(100520733097)**

Under the guidance of
Mrs. V. Sukanya

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)
Osmania University, Hyderabad – 500007, Telangana , India



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,

UNIVERSITY COLLEGE OF ENGINEERING,

OSMANIA UNIVERSITY

CERTIFICATE

This is to certify that the project work entitled " Codefun - A Smart Coding Skills Development Framework"submitted by CS koushik(100520733087),Samreen (100520733099) ,Thuppari pranay(100520733103),Charkaman khushi (100520733086), a students of DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY COLLEGE OF ENGINEERING, OSMANIA UNIVERSITY in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering is a record of the bonafide work carried out by him during the academic year 2023-2024.

Signature of the Supervisor

Mrs. V. Sukanya (Asst. Prof.)
Dept. of CSE, OU

Signature of Head of the Dept.

Prof. P V. Sudha
Dept. of CSE, OU

DECLARATION

I declare that the project work report entitled “Codefun - A Smart Coding Skills Development Framework” is a record of the work done by me in the **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY COLLEGE OF ENGINEERING, OSMANIA UNIVERSITY**. No part of the report is copied from books/journals/internet and wherever referred, the same has been duly acknowledged in the text. The reported data are based on the project work done entirely by me and not copied from any other source.

Signature of the Student

ACKNOWLEDGEMENT

I am greatly indebted to the Management of University College of Engineering, Osmania University, Hyderabad, for providing us the necessary facilities to successfully carry out this work titled **"**CODEFUN - A SMART CODING SKILLS DEVELOPMENT FRAMEWORK**".**

Apart from the effort of me, the success of the work mainly depends on encouragement and guidelines of others. I take this opportunity to express my gratitude to the people who had given me continuous support and cooperation in successful completion of work.

I would like to express our sincere gratitude to our Principal **Prof. P. Chandra Shekar** for providing the necessary infrastructure to complete.

I thank and express my solicit gratitude to **Prof. P V. Sudha**, HOD-CSE department, University College of Engineering, for her invaluable help and support which helped us a lot in successfully completing our work.

I also express my gratitude to **Mrs. V. Sukanya,** Assistant Professor, internal guide, University College of Engineering, for her suggestion and encouragement which helped us in the successful completion of our project.

I also extend my thanks to the entire faculty of the Department of Computer Science and Engineering, University College of Engineering, Osmania University who encourages us throughout the course of our Bachelor degree and allows me to use the many resources present in the department.

Finally, I would like to express our heartfelt gratitude to our parents and all our peers who were very supportive and for their encouragement to achieve our goals.

Cs koushik(100520733087)

Samreen(100520733099)

Thupaari pranay(100520733103)

Charkaman khushi(100520733086)

ABSTRACT

CODEFUN : Smart Coding Skills Development Framework” is an exciting and interactive web application designed to transform the way you learn programming. Dive into a world of captivating coding challenges and puzzles that turn education into an adventure. With real-time feedback and a user-friendly interface, you'll conquer concepts like loops, conditionals, and functions while having a blast. Whether you're a budding programmer or an enthusiast seeking to refine your skills, CODEFUN's gamified learning path offers something for everyone. Embark on a journey where curiosity meets code and become a programming master in a fun and engaging way.

"CODEFUN: Smart Coding Skills Development Framework" is an innovative and interactive web application aimed at revolutionizing the programming learning experience. Designed to make coding education engaging and enjoyable, CODEFUN integrates captivating coding challenges and puzzles that transform traditional learning into an adventurous journey. With a focus on real-time feedback and a user-friendly interface, the platform ensures learners grasp fundamental programming concepts such as loops, conditionals, and functions effectively while having fun.

CODEFUN caters to a wide range of users, from aspiring programmers to seasoned enthusiasts looking to hone their skills. The gamified learning path offered by CODEFUN encourages continuous engagement and motivation, making complex topics accessible and enjoyable. By combining curiosity and code, CODEFUN creates an environment where learners can develop their programming expertise in a structured yet entertaining manner. This approach not only enhances learning outcomes but also fosters a passion for coding, ultimately guiding users to become proficient programmers through an immersive and interactive experience.

S.NO	TOPICS	PAGE NO
1.	**CHAPTER I**	1
	1.1 MOTIVATION	1
	1.2 PROBLEM DEFINITION	1
	1.3 OBJECTIVES OF THE PROJECT	2
	1.4 NEED ANALYSIS	2
	1.5 PROJECT OUTCOMES	2
	1.6 PROPOSED ARCHITECTURE	3
2.	**CHAPTER II**	5
	2.1 POPULAR MODEL	5
	2.2 PROPOSED SYSTEM	6
	2.3 SOFTWARE REQUIREMENTS	7
	2.4 HARDWARE REQUIREMENTS	7
3.	**CHAPTER III**	8
	3.1 NODEJS INSTALLATION	9
	3.2 REACTJS INSTALLATION	9
	3.3 FIREBASE API SETUP	9
	3.4 VERCEL	11
	3.5 CODE IMPLEMENTATION	12
4.	**CHAPTER IV**	41
	4.1 RESULT	41
5.	**REFERENCES**	46

LIST OF FIGURES

S.NO	CONTENT	PAGE. NO
1	MVC architecture	3
2	Website outlook	6
3	User profile	41
4	Java module interface	41
5	Java intro module	42
6	Coding intro	42
7	Quiz	43
8	Output interface	43
9	Quiz	44
10	Quiz challenge	44
11	Coding activity	45
12	Activity result	45

CHAPTER I

INTRODUCTION

1.1 MOTIVATION

Coding holds immense importance for computer science students as it serves as the foundational language of their discipline. Firstly, coding facilitates a deep understanding of computational concepts such as algorithms, data structures, and problem-solving strategies. This understanding is crucial for students to navigate complex systems and develop efficient solutions to real-world problems. Secondly, coding fosters critical thinking and logical reasoning skills. Through writing code, students learn to break down problems into smaller, more manageable tasks, develop systematic approaches to problem-solving, and debug errors effectively. These skills are invaluable not only within the realm of computer science but also in various other domains where analytical thinking is required.

Moreover, coding promotes creativity and innovation. Students are encouraged to explore different programming paradigms, experiment with new technologies, and develop novel solutions to emerging challenges. Furthermore, proficiency in coding opens doors to a wide range of career opportunities. Whether students pursue careers in software development, data science, cybersecurity, or artificial intelligence, coding proficiency is a prerequisite for success in these fields. Additionally, coding skills are transferable, enabling students to excel in diverse professional settings where technology plays a crucial role.

In essence, coding is not just a technical skill; it is a fundamental aspect of computer science education that equips students with the knowledge, skills, and mindset necessary to thrive in the digital age.

1.2 PROBLEM DEFINITION

The “CodeFun - A Student friendly Web Application” aims to help students and reduce their hardship. The problems faced by students while learning coding are defined as follows:

1. Computational concepts such as algorithms, data structures, and problem-solving strategies should be understand clearly.
2. Students aren’t able to choose the right platform based on their requirements like where to start and how to start the coding journey.
3. Students are generally unaware about the skills to learn as per the industry requirements. Due to inadequate and imbalanced information, students are lacking technical skills, which leads to inadequate skills for industry.

1.3 OBJECTIVES OF THE PROJECT

Corresponding to problems cited above, following are objectives that "CodeFun – A Student friendly Web Application" is trying to solve:

1. To implement algorithms, data structures and problem-solving strategies to develop efficient solutions to real-world problems.
2. To solve the complex problems using algorithms and data structures efficiently.
3. To recommend the trending technologies on the basis of user intrest.
4. To provide the information of required skill set information what to learn for a specific role.
5. To design a web application for achieving above objectives.

1.4 NEED ANALYSIS

Coding plays a pivotal role in a computer science student's life due to several critical reasons. Firstly, it serves as the backbone of their education, enabling them to grasp fundamental concepts like algorithms and data structures. Secondly, coding fosters skill development, enhancing critical thinking and problem-solving abilities. Thirdly, it nurtures innovation and creativity by encouraging experimentation with various programming languages and paradigms. Fourthly, proficiency in coding expands career horizons, opening doors to lucrative opportunities in technology-driven fields. Additionally, coding cultivates technological literacy, empowering students to navigate and contribute to an increasingly digital world. Furthermore, it instills resilience and perseverance, as students learn to debug and overcome coding challenges. Overall, the need for coding in a computer science student's life is undeniable, shaping their education, skills, career prospects, and adaptability in a rapidly evolving technological landscape.

1.5 PROJECT OUTCOMES

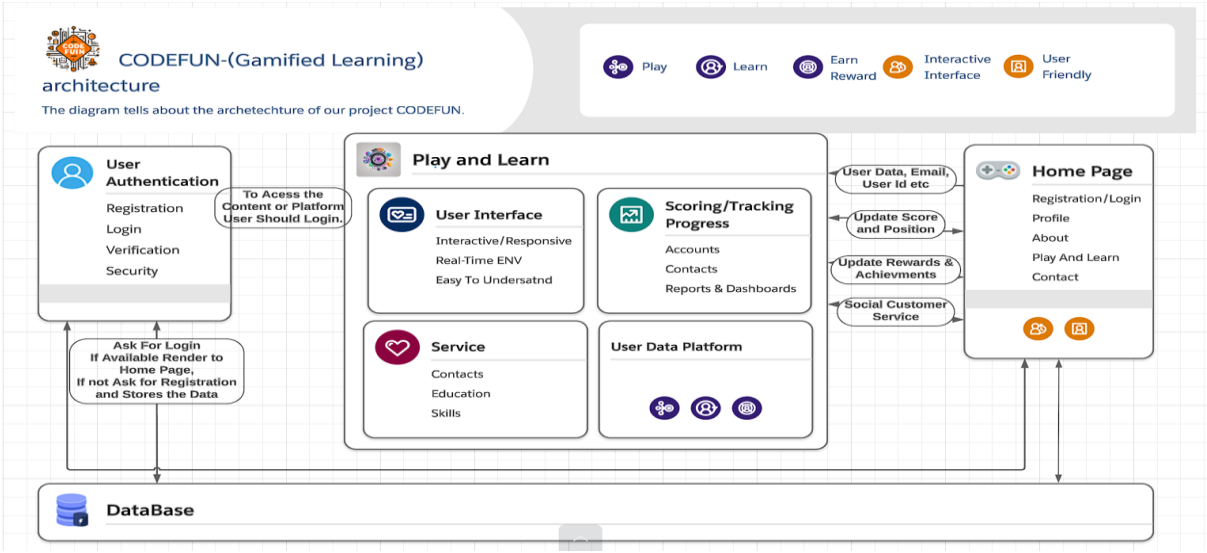
Enhanced Problem-Solving Skills: Coding teaches students how to approach complex problems systematically. This skill is valuable not only in programming but also in everyday life and various professional fields.

Career Opportunities: Proficiency in coding opens up a wide range of career opportunities in various industries, including tech, finance, healthcare, and more. Roles like software developer, data scientist, and system analyst are in high demand.

Critical Thinking: Coding encourages critical thinking by requiring students to break down problems into smaller, manageable parts and devise logical solutions.

Creativity: Learning to code allows students to bring their ideas to life, whether it's developing a new app, building a website, or automating tasks. This creative outlet can be incredibly fulfilling.

1.6 PROPOSED ARCHITECTURE



The MVC (Model-View-Controller) architecture is a software design pattern commonly used in web development to organize and structure code in a clear and maintainable way. Here's an overview of the MVC architecture:

1. Model:

- The Model represents the data and business logic of the application. It encapsulates the data-related operations and behavior, such as data validation, retrieval, manipulation, and storage.

- The Model is independent of the user interface and communicates with the database or external services to perform CRUD (Create, Read, Update, Delete) operations on the data.

- Examples of the Model in a web application include database tables, ORM (Object-Relational Mapping) models, business logic classes, and data access objects.

2. View:

- The View represents the presentation layer of the application and is responsible for rendering the user interface. It displays the data from the Model to the user and captures user input.

- The View receives data from the Controller and presents it to the user in a format that is suitable for display, such as HTML, CSS, and JavaScript.

- In web development, the View is typically composed of templates, markup files, stylesheets, and client-side scripts.

3. Controller:

- The Controller acts as an intermediary between the Model and the View. It handles user input, processes requests, and updates the Model accordingly.

- The Controller receives input from the user via the View, interprets the input, and invokes appropriate actions on the Model to fulfill the user's request.

- After processing the request and updating the Model, the Controller selects the appropriate View to render and passes the necessary data to the View for display.

- In web applications, the Controller is responsible for routing incoming HTTP requests to the corresponding actions and generating HTTP responses.

Key principles and benefits of the MVC architecture include:

- Separation of Concerns: MVC separates the concerns of data, presentation, and user interaction, making it easier to maintain, modify, and test each component independently.

- Modularity and Reusability: MVC promotes modularity and reusability by organizing code into distinct components (Model, View, Controller) that can be reused across different parts of the application.

- Scalability: MVC facilitates scalability by allowing developers to add new features, modify existing functionality, or replace components without affecting other parts of the application.

- Code Organization: MVC provides a clear and structured organization of code, making it easier for developers to understand the architecture of the application and collaborate effectively.

- Flexibility: MVC allows for flexibility in choosing different technologies and frameworks for each component e.g., different database technologies for the Model, different front-end frameworks for the View.

CHAPTER II

2.EXISTING SYSTEMS

2.1 POPULAR MODEL

GeeksforGeeks is a well-known online platform that provides educational resources, tutorials, coding challenges, interview preparation materials, and articles related to computer science, programming, data structures, algorithms, and various other technical topics

1.Educational Content: It offers a vast collection of tutorials, articles, and courses covering a wide range of computer science and programming topics. The content is organized into categories such as algorithms, data structures, programming languages (e.g., C++, Java, Python), web development, machine learning, competitive programming, and more.

2. Coding Practice: The platform provides coding practice opportunities through its "Practice" section, where users can solve coding problems across different difficulty levels and topics. It hosts coding challenges, contests, and competitions to help users improve their coding skills and prepare for technical interviews.

3. Interview Preparation: idely used by software engineers and students as a resource for interview preparation. The website offers curated collections of interview questions, coding problems, company-wise interview experiences, and preparation tips for popular tech companies like Google, Amazon, Microsoft, and Facebook.

4. Algorithms and Data Structures: It covers a comprehensive range of algorithms, data structures, and problem-solving techniques commonly used in software development and competitive programming. Users can learn about topics such as sorting algorithms, searching algorithms, dynamic programming, graph algorithms, and more.

6. Career Guidance: The platform provides career guidance and resources for students and professionals looking to pursue careers in software development, data science, machine learning, and other technology fields. This platform offers tips, advice, and insights into career paths, job opportunities, and industry trends.

2.2 PROPOSED SYSTEM

CODEFUN provides all the features provided by the existing system as well as short and crisp content for every concept.

After every concept CODEFUN provides interactive way of learning like drag and drop of the missing statements, puzzles related to specific concepts. User can experience fun way of experience in learning.

The CODEFUN will track the progress of the user according to the logical and analytical thinking power of the user.

Fig 1: website out look

2.3 SOFTWARE REQUIREMENTS:

2.3.1 USER INTERFACES

- **Operating system**: Windows 10.
- **Coding Language**: HTML, CSS, JavaScript ,ReactJs and nextJS.
- **Front-End:** HTML, CSS, JavaScript.
- **Back-End:** DataBase (FireBAse)
- **Tools:** VS code.

2.4 HARDWARE REQUIREMENTS:

- **Processor:** Intel i3 or above and above with 2.5 GHz
- **Ram:** 4GB.

CHAPTER-3

IMPLEMENTATION AND ENVIORNMENTAL SETUP

3.1 NODEJS INSTALLATION:

The installation process for node.js depends on your operating system. here are the general steps for installing node.js on different platforms:

For windows:

1. Download installer:

   - visit the official node.js website: [node.js downloads](https://nodejs.org/en/download/).

   - download the windows installer (msi) package for your system architecture (32-bit or 64-bit).

2. Run installer:

   - double-click the downloaded msi file to run the installer.

   - follow the prompts in the installation wizard.

   - accept the license agreement, choose the installation directory, and select additional components if desired.

3. Complete installation:

   - click "install" to begin the installation process.

   - once the installation is complete, you can close the installer.

4. Verify installation:

   - open command prompt or powershell.

   - run the following command to verify that node.js and npm (node package manager) are installed:

     node -v

     npm -v

3.2 REACTJS INSTALLATION:

1. Node.js and npm installation:

- before installing react.js, ensure that you have node.js and npm (node package manager) installed on your system. you can download and install node.js from the official website: [node.js downloads](https://nodejs.org/).

2. Create react app:

- react provides a command-line tool called create react app for quickly setting up new react projects. to install create react app globally on your system, open your terminal or command prompt and run the following command:

```
npm install -g create-react-app
```

3. Create a new react project:

- once create react app is installed, you can create a new react project by running the following command in your terminal or command prompt:

```
npx create-react-app my-react-app
```

- replace `my-react-app` with the name of your project.

4. Navigate to project directory:

- after creating the react project, navigate to the project directory using the `cd` command:

```
cd my-react-app
```

5. Start development server:

- once you're inside the project directory, you can start the development server by running the following command:

```
npm start
```

- this command will start the development server and open your default web browser to display the react application.

6. Verify installation:

- if the development server starts successfully, you should see the default react application running in your web browser. you can verify that react is installed correctly by checking the displayed webpage.

3.3 FIREBASE API SETUP:

To use the firebase api in javascript project, first need to set up a firebase project in the firebase console and obtain firebase configuration settings. then, can import the firebase sdk into project and initialize firebase with configuration.

1. Set up firebase project:

- go to the [firebase console](https://console.firebase.google.com/) and create a new project or select an existing one.

- follow the instructions to set up your project and choose the services you want to use (e.g., authentication, firestore, realtime database, storage).

- once your project is created, click on the settings icon (gear icon) and select "project settings". go to the "general" tab, and you'll find your firebase configuration object.

2. Install firebase sdk:

- if you're using npm (node package manager) for managing dependencies, you can install the firebase sdk by running the following command in your project directory:

```
npm install firebase
```

3. Import firebase sdk:

- in your javascript or typescript file where you want to use firebase, import the firebase sdk at the top of the file:

```
import firebase from 'firebase/app';
import 'firebase/auth'; // import specific firebase services as needed
import 'firebase/firestore';
import 'firebase/storage';
```

4. Initialize firebase:

- initialize firebase with your firebase project's configuration settings. you can do this at the beginning of your javascript file or in a separate configuration file:

```
const firebaseconfig = {
  apikey: "your_api_key",
  authdomain: "your_auth_domain",
  projectid: "your_project_id",
  storagebucket: "your_storage_bucket",
  messagingsenderid: "your_messaging_sender_id",
  appid: "your_app_id"
};
// initialize firebase
firebase.initializeapp(firebaseconfig);
```

5. Use firebase services:

- once firebase is initialized, you can use firebase services such as authentication, firestore, realtime database, and storage by calling the corresponding methods or apis provided by the firebase sdk.

```
const auth = firebase.auth();

const db = firebase.firestore();

const storage = firebase.storage();
```

3.4 VERCEL

VERCEL is a cloud platform for static sites and serverless functions. formerly known as zeit, vercel simplifies the deployment process for web applications, making it easier for developers to deploy and scale their projects.

1. Static site hosting: vercel specializes in hosting static websites and single-page applications (spas) with ease. developers can deploy their front-end projects directly from their git repositories without any complex configurations.

2. Serverless functions: vercel supports serverless functions, allowing developers to create backend logic using popular serverless frameworks like next.js and deploy them seamlessly alongside their front-end code.

3. Automatic deployments: vercel provides continuous deployment (cd) pipelines that automatically deploy changes to production whenever you push code to your git repository. this ensures that your website is always up-to-date without manual intervention.

4. Global cdn: vercel leverages a global content delivery network (cdn) to serve your website content from edge locations closest to your users, ensuring low latency and fast loading times regardless of geographical location.

5. Custom domains and ssl: vercel allows you to use custom domains for your projects and provides automatic ssl certificate provisioning, ensuring secure connections for your users.

6. Environment variables: vercel supports environment variables, allowing you to securely manage sensitive information such as api keys and database credentials across different deployment environments.

7. Collaboration and teams: vercel offers collaboration features that enable multiple team members to work on the same project simultaneously. you can also manage permissions and access controls for different team members.

8. Analytics and monitoring: vercel provides built-in analytics and monitoring tools that allow you to track website performance, monitor traffic, and debug issues in real-time.

3.5 CODE IMPLEMENTATION

Index

```
import Head from "next/head";
import Image from "next/image";
import { Inter } from "next/font/google";
import styles from "@/styles/Home.module.css";
import Topbar from "@/components/Topbar/Topbar";
import Link from "next/link";
import ProblemsTable from "@/components/ProblemsTable/ProblemsTable";
import { useState } from "react";
import { setDoc, doc } from "firebase/firestore";
import { firestore } from "@/firebase/firebase";
import HeroSec from "@/components/HomePage/HeroSec";

import Footer from "@/components/Footer/Footer";
import Sections from "@/components/Sections/Sections";
import { DndProvider } from "react-dnd";
import { HTML5Backend } from "react-dnd-html5-backend";
import MemoryGame from "@/components/MemoryGame/MemoryGame";
const inter = Inter({ subsets: ["latin"] });

export default function Home() {
  return (
    <>
      <main className="bg-dark-layer-2 min-h-screen">
        <Topbar />
        <HeroSec />
        <Sections />
        <br /><br /><br />
        <MemoryGame />
        <Footer />
      </main>
    </>
  );
}
```

Application

```
import "@/styles/globals.css";
import type { AppProps } from "next/app";
import Head from "next/head";
import { RecoilRoot } from "recoil";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import { DndProvider } from "react-dnd";
import { HTML5Backend } from "react-dnd-html5-backend";
```

```
export default function App({ Component, pageProps }: AppProps) {
  return (

      <RecoilRoot>
        <Head>
          <title>CodeFun</title>
          <meta name="viewport" content="width=device-width, initial-scale=1" />
          <link rel="icon" href="/logo-full1.png" />
          <meta
            name="description"
            content="Web application that contains way to learn coding in gamified
way"
          />
        </Head>
        <ToastContainer />
        <Component {...pageProps} />
      </RecoilRoot>

  );
}
```

The provided code defines two key components for a Next.js application

1.Home Component:

- It imports various modules and components, including `Topbar`, `HeroSec`, `Sections`, `MemoryGame`, and `Footer`.

- The `Home` function renders these components inside a `main` tag with specific styling, creating the page layout.

- It ensures the user sees a top bar, hero section, different content sections, an interactive memory game, and a footer.

2. App Component:

- It wraps the entire application with the `RecoilRoot` to manage state using Recoil.

- It uses the `Head` component to set the page title, viewport settings, and meta description.

- The `ToastContainer` is included for displaying toast notifications.

- The `DndProvider` from `react-dnd` is imported but not used in this snippet.

- The `App` function renders the component corresponding to the current route (`Component`) and passes the `pageProps` to it.

JAVA Module

```
import JavaGame from "@/components/Games/JavaGame/JavaGame";
import Topbar from "@/components/Topbar/Topbar";
import FooterComponent from "@/components/Footer/Footer";
import Link from "next/link";
import * as React from "react";
import { useRef } from "react";
import { TypeAnimation } from "react-type-animation";
import { SteppedLineTo } from "react-lineto";
import Levels from "@/components/Levels/Levels";
import { useAuthState } from "react-firebase-hooks/auth";
import { auth } from "@/firebase/firebase";
import AllUsers from "@/components/AllUsers/AllUsers";

export interface IJavaHomePageProps {}

export function JavaHomePage(props: IJavaHomePageProps) {
  const [user, loading] = useAuthState(auth);
  const scrollToLanguagesSection = () => {
    const languagesSection = document.getElementById("levels-section");

    if (languagesSection) {
      console.log("Scrolling to languages section");
      languagesSection.scrollIntoView({ behavior: "smooth" });
    }
  };

  return (
    <div className="bg-dark-layer-2 min-h-screen ">
      <Topbar />
      {/* Hero Section */}
      <section className="container items-center px-4 pb-12 mx-auto lg:flex md:px-8
md:py-16 lg:px-20 ">
        <div className="lg:w-1/2 md:w-full pr-4 md:pr-8 mb-8 lg:mb-0 sm:mt-4 md:mt-
10">
          <img
            src="./java-image (2).png"
            className="w-full mx-auto lg:w-10/12"
            alt="Java Image"
          />
        </div>
        <div className="flex-1 space-y-4 sm:text-center lg:text-left">
          <h1 className="text-3xl sm:text-4xl md:text-5xl lg:text-6xl font-bold mb-
4 sm:mb-6 text-center sm:text-left text-orange-500">
            JAVA
          </h1>
          <div className="java-details h-24 text-center sm:text-left">
            <span className="primary-text">
              <h1 className="text-xl sm:text-2xl md:text-3xl lg:text-4xl">
                <TypeAnimation
                  sequence={[
```

```
                "Unlock the Power of Java Programming.",
                2000,
                "Java: Where Innovation Meets Coding.",
                2000,
                "Master Java, Master Possibilities.",
                2000,
                "Transform Your Ideas into Code with Java.",
                2000,
                "Dive into Java, Unleash your Coding Potential.",
                2000,
              ]}
              speed={80}
              style={{
                fontFamily: "ShadowIntoLight",
                fontWeight: "bold",
                letterSpacing: "1px",
                color: "white",
              }}
              repeat={Infinity}
            />
          </h1>
        </span>
      </div>
      <div
        className="mb-7 md:text-2xl sm:p-5 md:p-0"
        style={{
          fontFamily: "ShadowIntoLight",
          fontWeight: "bold",
          letterSpacing: "1px",
          color: "white",
        }}
      >
        <p className="hidden sm:block">import java.util.*;</p>
        <p className="hidden sm:block">
          public static void main(Strings[] args) {"{"}
        </p>
        <p className="hidden sm:block">
          {/* follow indentation for below print statements  */}
                    
          <TypeAnimation
            sequence={[
              'System.out.println("Welcome to CodeFun!😍");',
              2000,
              'System.out.println("Get ready to learn Java!🚀");',
              2000,
              'System.out.println("Let\'s get started!👇");',
              2000,
            ]}
            speed={10}
            style={{
              fontFamily: "ShadowIntoLight",
```

```
                fontWeight: "bold",
                letterSpacing: "1px",
                color: "white",
              }}
              repeat={Infinity}
            />
          </p>
          <p className="hidden sm:block">{"}"}</p>

          {/* Responsive Code for Small Screens */}
          <div className="sm:hidden">
            <p>import java.util.*;</p>
            <p>public static void main(Strings[] args) {"{"}</p>
            <p>
              {/* follow indentation for below print statements  */}
                        
              <TypeAnimation
                sequence={[
                  'System.out.println("Welcome to CodeFun!😍");',
                  2000,
                  'System.out.println("Get ready to learn Java!🚀");',
                  2000,
                  'System.out.println("Let\'s get started!👇");',
                  2000,
                ]}
                speed={10}
                style={{
                  fontFamily: "ShadowIntoLight",
                  fontWeight: "bold",
                  letterSpacing: "1px",
                  color: "white",
                }}
                repeat={Infinity}
              />
            </p>
            <p>{"}"}</p>
          </div>
        </div>

        <br />
        <div className="flex flex-col items-center sm:flex-row sm:justify-center
lg:justify-start space-y-4 sm:space-y-0 sm:space-x-6">
          <a
            href="#level-section"
            className="text-xl px-8 py-3 text-white bg-orange-500 rounded-md mb-4
sm:mb-0"
            onClick={scrollToLanguagesSection}
          >
            Get Started
          </a>
          <a
```

```
              href="https://docs.oracle.com/en/java/"
              className="text-xl px-8 py-3 text-gray-500 bg-white rounded-md"
            >
              See More
            </a>
          </div>
        </div>
      </section>
      <br />
      {/* Levels Section */}
      <div id="level-section">
        <h3 className="text-2xl sm:text-3xl md:text-2xl lg:text-
5xl  underline  font-bold text-center sm:text-left text-white md:ml-72 mb-10">
          Start Learning Java
        </h3>
        <div className="md:flex ">
          <div className=" md:mt-20">
            <Levels />
          </div>
          <div className="p-4 md:p-0 md:ml-[500px] ">
            <AllUsers />
          </div>
        </div>
      </div>

      {/* footer */}
      <FooterComponent/>
    </div>
  );
}

export default JavaHomePage;
```

This code defines a React component, `JavaHomePage`, which serves as a landing page for a Java programming course. It incorporates several imported components, including `Topbar`, `FooterComponent`, `Levels`, and `AllUsers`. The `useAuthState` hook from Firebase manages user authentication state, ensuring certain content is only accessible to logged-in users. The hero section features a Java-themed introduction with animated text, encouraging users to start learning. The `scrollToLanguagesSection` function smoothly scrolls the page to the levels section when a button is clicked. The levels section allows users to explore various Java learning levels, while the `AllUsers` component displays user information. Styling and animations enhance user engagement, and the layout adapts responsively for different screen sizes.

Introduction Page

```
import React, { useEffect, useState } from 'react';
import { DndProvider } from 'react-dnd';
import { HTML5Backend } from 'react-dnd-html5-backend';
import { useAuthState } from "react-firebase-hooks/auth";
import { auth, firestore } from "@/firebase/firebase";
import { toast } from 'react-toastify';

import Topbar from '@/components/Topbar/Topbar';
import Level1 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level1';
import Level10 from
'@/components/GameLevels/JavaLevels/IntroductionLevels/Level10';
import Level11 from
'@/components/GameLevels/JavaLevels/IntroductionLevels/Level11';
import Level2 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level2';
import Level3 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level3';
import Level4 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level4';
import Level5 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level5';
import Level6 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level6';
import Level7 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level7';
import Level8 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level8';
import Level9 from '@/components/GameLevels/JavaLevels/IntroductionLevels/Level9';
import next from 'next';


const IntroductionLevel = () => {
  const [user, loading] = useAuthState(auth);
  const [currentLevel, setCurrentLevel] = useState(1);

  useEffect(() => {
    const storedLevel = localStorage.getItem('currentLevel');
    if (storedLevel) {
      setCurrentLevel(parseInt(storedLevel, 10));
    }
  }, []);

  const handleLevelComplete = () => {
    const nextLevel = currentLevel + 1;

    localStorage.setItem('currentLevel', String(nextLevel));

    setCurrentLevel(nextLevel);
  };

  const handlePreviousLevel = () => {
    const previousLevel = Math.max(1, currentLevel - 1);

    localStorage.setItem('currentLevel', String(previousLevel));

    setCurrentLevel(previousLevel);
  };
```

```
  const renderLevelComponent = () => {
    if (loading) {
      return <div>Loading...</div>; // You can replace this with a loading
indicator
    }

    switch (currentLevel) {
      case 1:
        return <Level1 onComplete={handleLevelComplete} />;
      case 2:
        return <Level2 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 3:
        return <Level3 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 4:
        return <Level4 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 5:
        return <Level5 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 6:
        return <Level6 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 7:
        return <Level7 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 8:
        return <Level8 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 9:
        return <Level9 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 10:
        return <Level10 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 11:
        return <Level11 onPrevious={handlePreviousLevel} />;

      default:
        return null;
    }
  };


  return (
    <DndProvider backend={HTML5Backend}>
      <div className=' bg-dark-layer-2 min-h-screen'>
        <Topbar />
        { user ? (
```

```
        <div>
          <h1 className=' text-2xl md:text-4xl font-bold mb-4 md:ml-[550px] text-
white mt-7'>
            Introduction to Java Programming
          </h1>
          {/* Render components based on the current level */}
          {renderLevelComponent()}
          `<>{currentLevel }</>`
        </div>
        ) : (

          <div className='bg-dark-layer-2 min-h-screen flex items-center justify-
center text-white'>

          <p>Please login to access the introduction levels.</p>
        </div>
        )}
      </div>
    </DndProvider>
  );
};

export default IntroductionLevel;
```

This code defines a React component, `IntroductionLevel`, which offers a gamified learning experience for Java programming. It uses Firebase for user authentication and state management to track the current level. Upon mounting, it retrieves the user's current level from local storage and updates it as the user progresses through the levels. Each level is rendered conditionally based on the current level state. The `DndProvider` with `HTML5Backend` enables drag-and-drop functionality for interactive elements within the levels. A `Topbar` is included for navigation, and if the user is not authenticated, a prompt is displayed to log in. The component ensures a responsive and engaging interface with Tailwind CSS for styling.

Java Multiple statements

```
import Level1 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level1';
import Level10 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level10';
import Level11 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level11';
import Level2 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level2';
import Level3 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level3';
import Level4 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level4';
import Level5 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level5';
import Level6 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level6';
import Level7 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level7';
import Level8 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level8';
import Level9 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level9';
import Topbar from '@/components/Topbar/Topbar';
import React, { useEffect, useState } from 'react';
import { DndProvider } from 'react-dnd';
import { HTML5Backend } from 'react-dnd-html5-backend';
import { useAuthState } from "react-firebase-hooks/auth";
import { auth, firestore } from "@/firebase/firebase";
import Level12 from '@/components/GameLevels/JavaLevels/MultipleStatements
Levels/Level12';


const MultipleStatements = () => {
  const [user, loading] = useAuthState(auth);
  const [currentLevel1, setCurrentLevel1] = useState(1);

  useEffect(() => {
    const storedLevel = localStorage.getItem('currentLevel1');
    if (storedLevel) {
      setCurrentLevel1(parseInt(storedLevel, 10));
    }
  }, []);

  const handleLevelComplete = () => {
    const nextLevel = currentLevel1 + 1;

    localStorage.setItem('currentLevel1', String(nextLevel));
```

```
    setCurrentLevel1(nextLevel);
  };

  const handlePreviousLevel = () => {
    const previousLevel = Math.max(1, currentLevel1 - 1);

    localStorage.setItem('currentLevel1', String(previousLevel));

    setCurrentLevel1(previousLevel);
  };

  const renderLevelComponent = () => {
    if (loading) {
      return <div>Loading...</div>; // You can replace this with a loading
indicator
    }

    switch (currentLevel1) {
      case 1:
        return <Level1 onComplete={handleLevelComplete} />;
      case 2:
        return <Level2 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 3:
        return <Level3 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 4:
        return <Level4 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 5:
        return <Level5 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 6:
        return <Level6 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 7:
        return <Level7 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 8:
        return <Level8 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 9:
        return <Level9 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 10:
        return <Level10 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete} />;
      case 11:
        return <Level11 onPrevious={handlePreviousLevel}
onComplete={handleLevelComplete}/>;
```

```
      case 12:
        return <Level12 onPrevious={handlePreviousLevel}/>;
      default:
        return null;
    }
  };


  return (
    <DndProvider backend={HTML5Backend}>
      <div className=' bg-dark-layer-2 min-h-screen'>
        <Topbar />
        { user ? (
        <div>
          <h1 className=' text-2xl md:text-4xl font-bold mb-4 text-center text-
white mt-7'>
            Multiple Statements
          </h1>
          {/* Render components based on the current level */}
          {renderLevelComponent()}
        </div>
        ) : (

          <div className='bg-dark-layer-2 min-h-screen flex items-center justify-
center text-white'>

          <p>Please login to access the introduction levels.</p>
        </div>
        )}
      </div>
    </DndProvider>
  );
};

export default MultipleStatements;
```

Python Module

```
import Topbar from "@/components/Topbar/Topbar";
import { auth, firestore } from "@/firebase/firebase";
import { arrayUnion, doc, getDoc, updateDoc } from "firebase/firestore";
import { FooterComponent } from "flowbite-react/lib/esm/components/Footer";
import router from "next/router";
import * as React from "react";
import { useAuthState } from "react-firebase-hooks/auth";
import { toast } from "react-toastify";
import { TypeAnimation } from "react-type-animation";

interface IPythonHomePageProps {}

export function PythonHomePage(props: IPythonHomePageProps) {
  const [user] = useAuthState(auth);

  const handleStartLearning = () => {
    // Logic to navigate to the Python learning section
    toast.info("The Python Course will be added pretty soon 🧐, Stay Tuned!", {
        position: "top-center",
        autoClose: 3000,
        theme: "dark",
        });
  };

  return (
    <div className="bg-dark-layer-2 min-h-screen">
      {/* Topbar Component */}
      <Topbar />

      {/* Hero Section */}
      <section className="container items-center px-4 pb-12 mx-auto lg:flex md:px-8
md:py-16 lg:px-20">
        <div className="lg:w-1/2 md:w-full pr-4 md:pr-8 mb-8 lg:mb-0 sm:mt-4 md:mt-
10">
          <img
            src="./python-image.png"
            className="w-full mx-auto lg:w-10/12  rounded-3xl mt-5"
            alt="Python Image"
          />
        </div>
        <div className="flex-1 space-y-4 sm:text-center lg:text-left">
          <h1 className="text-3xl sm:text-4xl md:text-5xl lg:text-6xl font-bold mb-
4 sm:mb-6 text-center sm:text-left text-orange-500">
            Python
          </h1>
          <div className="python-details h-24 text-center sm:text-left">
            <span className="primary-text">
              <h1 className="text-xl sm:text-2xl md:text-3xl lg:text-4xl">
                <TypeAnimation
                  sequence={[
```

```
                  "Unlock the Power of Python Programming.",
                  2000,
                  "Python: Where Simplicity Meets Efficiency.",
                  2000,
                  "Master Python, Master Automation.",
                  2000,
                  "Transform Your Ideas into Code with Python.",
                  2000,
                  "Dive into Python, Unleash your Coding Potential.",
                  2000,
                ]}
                speed={80}
                style={{
                  fontFamily: "ShadowIntoLight",
                  fontWeight: "bold",
                  letterSpacing: "1px",
                  color: "white",
                }}
                repeat={Infinity}
              />
            </h1>
          </span>
        </div>
        <br />
        <div
          className="mb-7 md:text-2xl sm:p-5 md:p-0"
          style={{
            fontFamily: "ShadowIntoLight",
            fontWeight: "bold",
            letterSpacing: "1px",
            color: "white",
          }}
        >

          {/* Responsive Code for Small Screens */}
          <div >
            <p>Python is a versatile programming language...</p>
          </div>
        </div>
        <br />
        <div className="flex flex-col items-center sm:flex-row sm:justify-center
lg:justify-start space-y-4 sm:space-y-0 sm:space-x-6">
          <button
            onClick={handleStartLearning}
            className="text-xl px-8 py-3 text-white bg-orange-500 rounded-md mb-4
sm:mb-0 hover:bg-green-600 transition duration-300"
          >
            Get Started
          </button>
          <a
            href="https://www.python.org/doc/"
```

```
              className="text-xl px-8 py-3 text-gray-500 bg-white rounded-md
hover:text-gray-700 hover:bg-gray-200 transition duration-300"
            >
              See More
            </a>
          </div>
        </div>
      </section>

      {/* Footer Component */}

    </div>
  );
}

export default PythonHomePage;
```

The `PythonHomePage` React component is designed as the landing page for an upcoming Python programming course. It utilizes Firebase authentication to manage user state with the `useAuthState` hook.

1. Topbar: Provides navigation at the top of the page.

2. Hero Section:

   - Displays a Python-related image and an animated text banner with the `TypeAnimation` component highlighting Python's benefits.

   - Features a prominent "Python" title with dynamic styling.

   - Includes a brief, responsive introductory snippet about Python's versatility.

3. Buttons:

   - Get Started: Shows a toast notification that the course is coming soon.

   - See More: Links to the official Python documentation.

4. Responsive Design: Ensures content is visually appealing across various screen sizes.

5. Footer Component: Mentioned in imports but not displayed, indicating future use.

Problems

```
import Topbar from "@/components/Topbar/Topbar";
import Workspace from "@/components/Workspace/Workspace";
import useHasMounted from "@/hooks/useHasMounted";

import { problems } from "@/utils/problems";
import { Problem } from "@/utils/types/problem";
import React from "react";

type IProblemPageProps = { problem: Problem; };

const ProblemPage: React.FunctionComponent<IProblemPageProps> = ({ problem }) => {
    const hasMounted = useHasMounted();

    if (!hasMounted) return null;
    return <div>
        <Topbar problemPage />
        <Workspace problem={problem} />
    </div>;
};

export default ProblemPage;

// fetch the local data
//  SSG
// getStaticPaths => it create the dynamic routes
export async function getStaticPaths() {
    const paths = Object.keys(problems).map((key) => ({
        params: { pid: key },
    }));

    return {
        paths,
        fallback: false,
    };
}

// getStaticProps => it fetch the data

export async function getStaticProps({ params }: { params: { pid: string } }) {
    const { pid } = params;
    const problem = problems[pid];

    if (!problem) {
        return {
            notFound: true,
        };
    }
    problem.handlerFunction = problem.handlerFunction.toString();
    return {
        props: {
```

```
            problem,
        },
    };
}
```

Problem solving Interface

```
import ProblemsTable from "@/components/ProblemsTable/ProblemsTable";
import Topbar from "@/components/Topbar/Topbar";
import useHasMounted from "@/hooks/useHasMounted";
import FooterComponent from "@/components/Footer/Footer";
import * as React from "react";
import { useState } from "react";
import { TypeAnimation } from "react-type-animation";
interface IProblemsProps {}

const Problems: React.FunctionComponent<IProblemsProps> = (props) => {
  const [loadingProblems, setLoadingProblems] = useState(true);
  const hasMounted = useHasMounted();

  if (!hasMounted) return null;

  return (
    <>
      <main className="bg-dark-layer-2 min-h-screen">
        <Topbar />
        <section className="container items-center px-4 pb-12 mx-auto lg:flex
md:px-8 md:py-16 lg:px-20 ">
          <div className="lg:w-1/2 md:w-full pr-4 md:pr-8 mb-8 lg:mb-0 sm:mt-4
md:mt-10">
            <img
              src="./problem-solving1.png"
              className="w-full mx-auto lg:w-10/12"
              alt="Java Image"
            />
          </div>
          <div className="flex-1 space-y-4 sm:text-center lg:text-left">
            <h1 className="text-3xl sm:text-4xl md:text-5xl lg:text-6xl font-bold
mb-4 sm:mb-6 text-center sm:text-left text-orange-500">
              Problem Solving
            </h1>
            <div className="java-details h-24 text-center sm:text-left">
              <span className="primary-text">
                <h1 className="text-xl sm:text-2xl md:text-3xl lg:text-4xl">
                  <TypeAnimation
                    sequence={[
```

```
                    "Unlock the Power of Problem Solving.",
                    2000,
                    "Problem Solving: Where Innovation Meets Critical Thinking.",
                    2000,
                    "Master Problem Solving, Master Possibilities.",
                    2000,
                    "Transform Your Ideas into Solutions with Problem Solving.",
                    2000,
                    "Dive into Problem Solving, Unleash your Creative
Potential.",
                    2000,
                  ]}
                  speed={80}
                  style={{
                    fontFamily: "ShadowIntoLight",
                    fontWeight: "bold",
                    letterSpacing: "1px",
                    color: "white",
                  }}
                  repeat={Infinity}
                />
              </h1>
            </span>
          </div>
          <div className="flex flex-col items-center sm:flex-row sm:justify-
center lg:justify-start space-y-4 sm:space-y-0 sm:space-x-6">
            <a
              href="#problems-section"
              className="text-xl px-8 py-3 text-white bg-orange-500 rounded-md
mb-4 sm:mb-0"
            >
              Get Started
            </a>
            <a
              href="https://workat.tech/problem-solving"
              className="text-xl px-8 py-3 text-gray-500 bg-white rounded-md"
            >
              See More
            </a>
          </div>
        </div>
      </section>
      <div id="problems-section">
        <h1
          className="text-2xl text-center text-gray-700 dark:text-gray-400 font-
medium
          uppercase mt-10 mb-5"
          >
          &ldquo; QUALITY OVER QUANTITY &rdquo; 👇
        </h1>
        <div className="relative overflow-x-auto mx-auto px-6 pb-10">
```

```
            {loadingProblems && (
              <div className="max-w-[1200px] mx-auto sm:w-7/12 w-full animate-
pulse">
                {[...Array(10)].map((_, idx) => (
                  <LoadingSkeleton key={idx} />
                ))}
              </div>
            )}
            <table className="text-sm text-left text-gray-500 dark:text-gray-400
sm:w-7/12 w-full max-w-[1200px] mx-auto">
              {!loadingProblems && (
                <thead className="text-xs text-gray-700 uppercase dark:text-gray-
400 border-b ">
                  <tr>
                    <th scope="col" className="px-1 py-3 w-0 font-medium">
                      Status
                    </th>
                    <th scope="col" className="px-6 py-3 w-0 font-medium">
                      Title
                    </th>
                    <th scope="col" className="px-6 py-3 w-0 font-medium">
                      Difficulty
                    </th>

                    <th scope="col" className="px-6 py-3 w-0 font-medium">
                      Category
                    </th>
                    <th scope="col" className="px-6 py-3 w-0 font-medium">
                      Solution
                    </th>
                  </tr>
                </thead>
              )}
              <ProblemsTable setLoadingProblems={setLoadingProblems} />
            </table>
          </div>
        </div>
        <br />
        <FooterComponent />
      </main>
    </>
  );
};

const LoadingSkeleton = () => {
  return (
    <div className="flex items-center space-x-12 mt-4 px-6">
      <div className="w-6 h-6 shrink-0 rounded-full bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52  w-32  rounded-full bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52  w-32 rounded-full bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52 w-32 rounded-full bg-dark-layer-1"></div>
```

```
      <span className="sr-only">Loading...</span>
    </div>
  );
};

export default Problems;
```

The `Problems` component is a React page component responsible for displaying a list of problems for solving. Key features include:

1. Topbar : Displays a navigation bar at the top of the page.

2. Hero Section: Features an image and an animated text banner promoting problem-solving skills.

3. Buttons: Provides buttons to get started with problem-solving and to explore more problems.

4. Problem Section: Displays a list of problems in a table format, including their status, title, difficulty, category, and solution links.

5. Loading Skeleton: Renders a loading skeleton while the problems are being fetched.

6. Responsive Design: Ensures the layout adjusts appropriately for different screen sizes.

7. Footer Component: Displays a footer at the bottom of the page.

8. Firebase Integration: Utilizes Firebase hooks for authentication.

9. Dynamic Content: Content is dynamically loaded and displayed based on user interaction.

10. Animation: Utilizes the `TypeAnimation` component for animated text banners.

MAP.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HashMap Animation</title>
    <link rel="stylesheet" href="map.css">
</head>
<body>
   <div>
    <img src="HashMapInternalStructure.png" style="height: 370px; width: 600px;"
alt="">
   </div>
    <button onclick="addElement()">Add Element</button>
    <button onclick="removeElement()">Remove Element</button>

    <button onclick="resizeHashMap()">Resize HashMap</button>
    <div id="hashMapContainer">
        <!-- Buckets will be added dynamically here -->
    </div>
    <p id="operationMessage"></p>
    <script src="map.js"></script>
</body>
</html>
```

MAP.css

```
#hashMapContainer {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin-top: 20px;
}

.bucket {
    width: 100px;
    height: 100px;
    border: 2px solid black;
    margin: 5px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 18px;
    transition: background-color 0.5s;
}

.added {
```

```
    background-color: lightgreen;
}

.removed {
    background-color: tomato;
}

.searched {
    background-color: lightblue;
}
```

Map.js

```
let hashMap = new Map();

function addElement() {
    let key = Math.floor(Math.random() * 100); // Generate a random key
    let value = Math.random().toString(36).substring(7); // Generate a random value
    hashMap.set(key, value);
    visualizeHashMap('add', key, value);
}

function removeElement() {
    if (hashMap.size > 0) {
        let keys = Array.from(hashMap.keys());
        let indexToRemove = Math.floor(Math.random() * keys.length); // Choose a
random index to remove
        let keyToRemove = keys[indexToRemove];
        let valueToRemove = hashMap.get(keyToRemove);
        hashMap.delete(keyToRemove);
        visualizeHashMap('remove', keyToRemove, valueToRemove);
        updateSizeDisplay();
    }
}



function resizeHashMap() {
    // For demonstration purposes, let's just log the resizing operation
    console.log("HashMap resized");
}

function visualizeHashMap(operation, key, value) {
    let container = document.getElementById('hashMapContainer');

    if (operation === 'remove') {
        let buckets = document.querySelectorAll('.bucket');
        let bucketToRemove = Array.from(buckets).find(bucket =>
bucket.textContent.includes(`${key}: ${value}`));
```

```
        // Remove the element from the container
        container.removeChild(bucketToRemove);

        document.getElementById('operationMessage').textContent = `Removed element:
${key}: ${value}`;
    } else {
        let bucket = document.createElement('div');
        bucket.classList.add('bucket');
        bucket.textContent = `${key}: ${value}`;

        container.appendChild(bucket);

        // Highlight the action
        if (operation === 'add') {
            bucket.classList.add('added');
            document.getElementById('operationMessage').textContent = `Added
element: ${key}: ${value}`;
        } else if (operation === 'search') {
            bucket.classList.add('searched');
            document.getElementById('operationMessage').textContent = `Searched
element: ${key}: ${value}`;
        }

        setTimeout(() => {
            clearHighlight(bucket);
        }, 1500);
    }
}


function clearHighlight(bucket) {
    bucket.classList.remove('added', 'removed', 'searched');
    document.getElementById('operationMessage').textContent = '';
}

function updateSizeDisplay() {
    let sizeDisplay = document.getElementById('sizeDisplay');
    sizeDisplay.textContent = `HashMap size: ${hashMap.size}`;
}
```

Queue.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Queue Animation</title>
<link rel="stylesheet" href="queueAni.css">
</head>
<body>


<div id="container">

    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
    <div>

        <img src="barr.jpeg" style="height: 40px; width: 100px; margin-top: 45%;"
id="bar">

    </div>


</div>
<div id="controls">
  <button id="pushBtn" style="margin-left: 28%;">Push</button>
  <button id="popBtn">Pop</button>
</div>
<div>
    <p class="heading" style="margin-top:-153px; margin-left: 12%; font-
family:cursive;">Front</p>
</div>

<div>
    <img src="arr.jpeg" style="height: 40px; width: 100px; margin-left:
10%;  margin-top: -1%;" id="arrow">

</div>


<script src="queueAni.js"></script>
</body>
</html>
```

Queue.css

```
#container {
    display: flex;
    overflow-x: hidden; /* Hide overflow to prevent horizontal scrollbar */
    margin-bottom: 50px;
    margin-left: 20%;
    margin-top: 5%;
  }

  .box {
    width: 50px;
    height: 50px;
    background-color: #ccc;
    margin: 5px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 20px;
    transition: transform 0.5s ease;
    animation-duration: 1s;
    animation-timing-function: ease;
    animation-fill-mode: both;
    margin-top: 5%;
  }

  @keyframes slideIn {
    0% {
      opacity: 0;
      transform: translateX(-100%);
    }
    100% {
      opacity: 1;
      transform: translateX(0);
    }
  }

  @keyframes slideOut {
    0% {
      opacity: 1;
      transform: translateX(0);
    }
    100% {
      opacity: 0;
      transform: translateX(100%);
    }
  }

  .slide-in {
    animation-name: slideIn;
  }
```

```
  .slide-out {
    animation-name: slideOut;
  }

  #controls {
    margin-top: 20px;
  }

  button {
    margin-right: 10px;
  }
```

Queue.js

```
document.addEventListener("DOMContentLoaded", function() {
    const pushBtn = document.getElementById("pushBtn");
    const popBtn = document.getElementById("popBtn");
    const container = document.getElementById("container");
    const arrow=document.getElementById("bar")
    let currentIndex = 6; // Starting index for newly added elements

    pushBtn.addEventListener("click", function() {
      const newBox = document.createElement("div");
      newBox.textContent = currentIndex++;
      newBox.classList.add("box");
      container.appendChild(newBox);
      animatePush(newBox);
      moveArrowLeft();
    });

    function moveArrowLeft() {
        const boxes = document.querySelectorAll(".box");
        if (boxes.length > 0) {
          const lastBox = boxes[boxes.length - 1];
          const boxRect = lastBox.getBoundingClientRect();
          arrow.style.left = `${boxRect.right}px`; // Move arrow to the right of
the last added element
        } else {
          arrow.style.left = "0"; // Move arrow back to initial position if queue
is empty
        }
      }




    popBtn.addEventListener("click", function() {
      const firstBox = document.querySelector(".box");
      if (firstBox) {
```

```
        animatePop(firstBox);
      } else {
        console.log("Queue is empty!");
      }
    });

    function animatePush(element) {
      element.classList.add("slide-in");
      setTimeout(() => {
        element.classList.remove("slide-in");
      }, 1000);
      moveArrow();
    }

    function animatePop(element) {
      element.classList.add("slide-out");
      setTimeout(() => {
        element.remove();
      }, 1000);
    }
  });
```

Set.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HashSet Animation</title>
    <link rel="stylesheet" href="set.css">
</head>
<body>
    <div id="hashSetContainer">
        <!-- Elements will be added dynamically here -->
    </div>

    <button onclick="addElement()">Add Element</button>
    <button onclick="removeElement()">Remove Element</button>
    <button onclick="getHashSetSize()">Get Set Size</button>
    <p id="operationMessage"></p>
    <p id="setSizeDisplay"></p>

    <div>
        <img src="HowHashSetWorks.png" style="height: 400px; width: 700px; margin-
left: 26%;" alt="">
    </div>
    <script src="set.js"></script>
</body>
</html>
```

Set.js

```
let hashSet = new Set();

function addElement() {
    let element = Math.floor(Math.random() * 100); // Generate a random number as
the element
    hashSet.add(element);
    visualizeHashSet('add', element);
}

function removeElement() {
    if (hashSet.size > 0) {
        let elements = Array.from(hashSet);
        let indexToRemove = Math.floor(Math.random() * elements.length); // Choose
a random index to remove
        let elementToRemove = elements[indexToRemove];
        hashSet.delete(elementToRemove);
        visualizeHashSet('remove', elementToRemove);
        getHashSetSize(); // Update HashSet size after removing an element
    }
```

```
}

function getHashSetSize() {
    let sizeDisplay = document.getElementById('setSizeDisplay');
    if (hashSet.size === 0) {
        sizeDisplay.textContent = "HashSet is empty";
    } else {
        sizeDisplay.textContent = `HashSet size: ${hashSet.size}`;
    }
}

function visualizeHashSet(operation, element) {
    let container = document.getElementById('hashSetContainer');
    container.innerHTML = ''; // Clear previous visualization

    hashSet.forEach(val => {
        let div = document.createElement('div');
        div.classList.add('hashSetElement');
        div.textContent = val;
        container.appendChild(div);
    });

    if (operation === 'add') {

        addedElement.classList.add('added');
        document.getElementById('operationMessage').textContent = `Added element:
${element}`;
    } else if (operation === 'remove') {
        let removedElement = document.querySelector(`.hashSetElement:nth-
child(${elements.indexOf(element) + 1})`);
        removedElement.classList.add('removed');
        document.getElementById('operationMessage').textContent = `Removed element:
${element}`;
    }

    setTimeout(() => {
        clearHighlight();
    }, 1500);

    // Update size display
    getHashSetSize();
}

function clearHighlight() {
    let elements = document.querySelectorAll('.hashSetElement');
    elements.forEach(element => {
        element.classList.remove('added', 'removed');
    });
    document.getElementById('operationMessage').textContent = '';
}
```

CHAPTER IV

4.RESULT

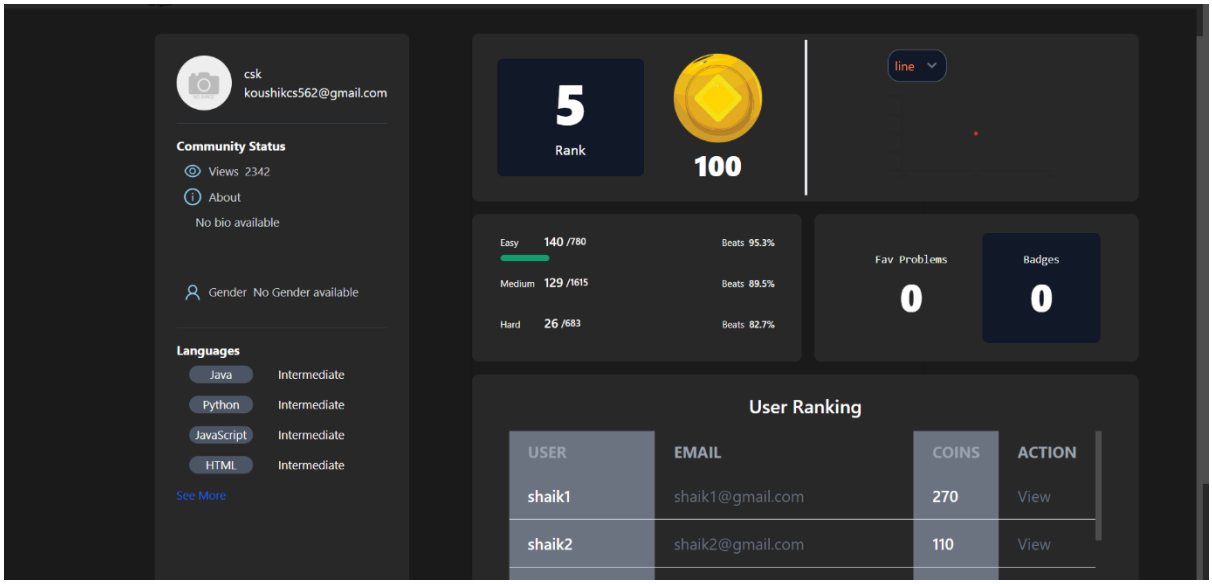


Fig 5 : User profile

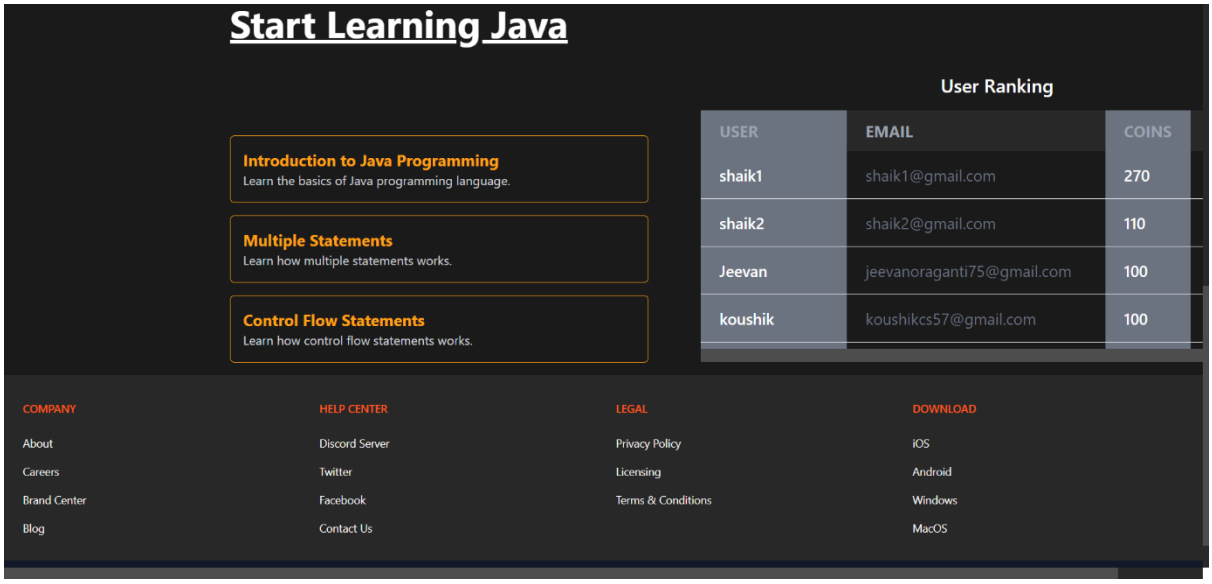


Fig 6: Java module interface

Fig 7 : Java intro module

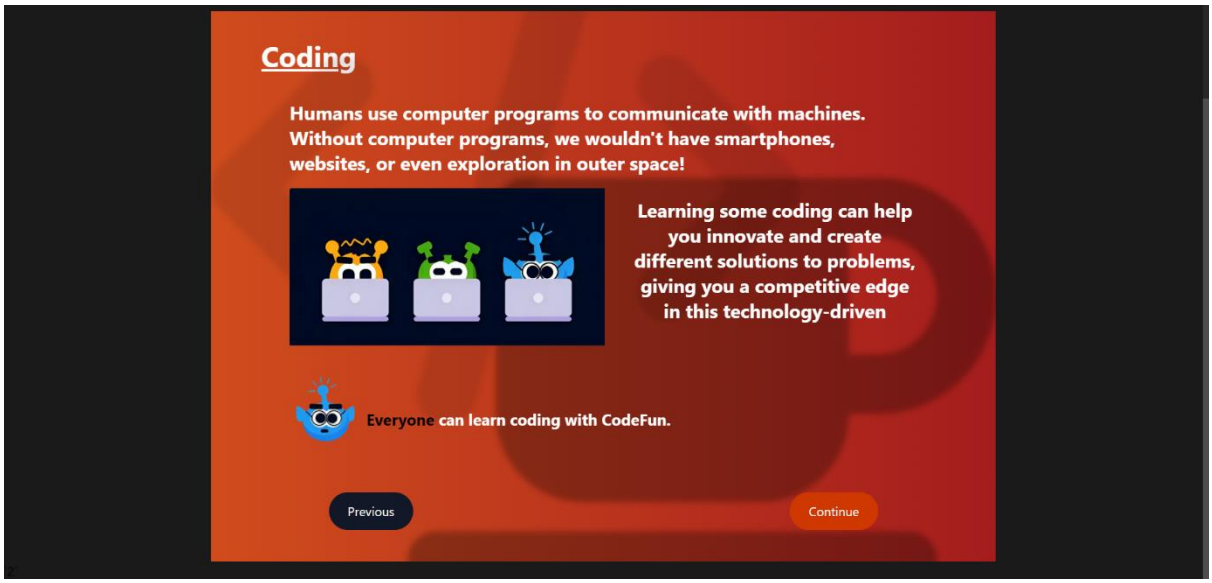


Fig 8 : Coding intro

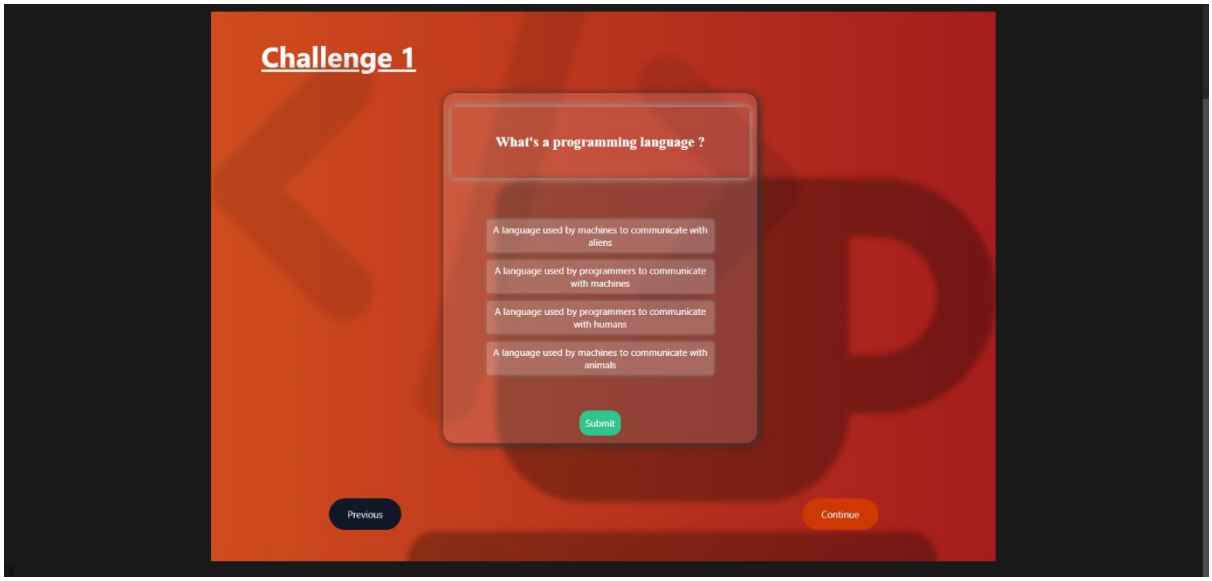


Fig 9 : Quiz

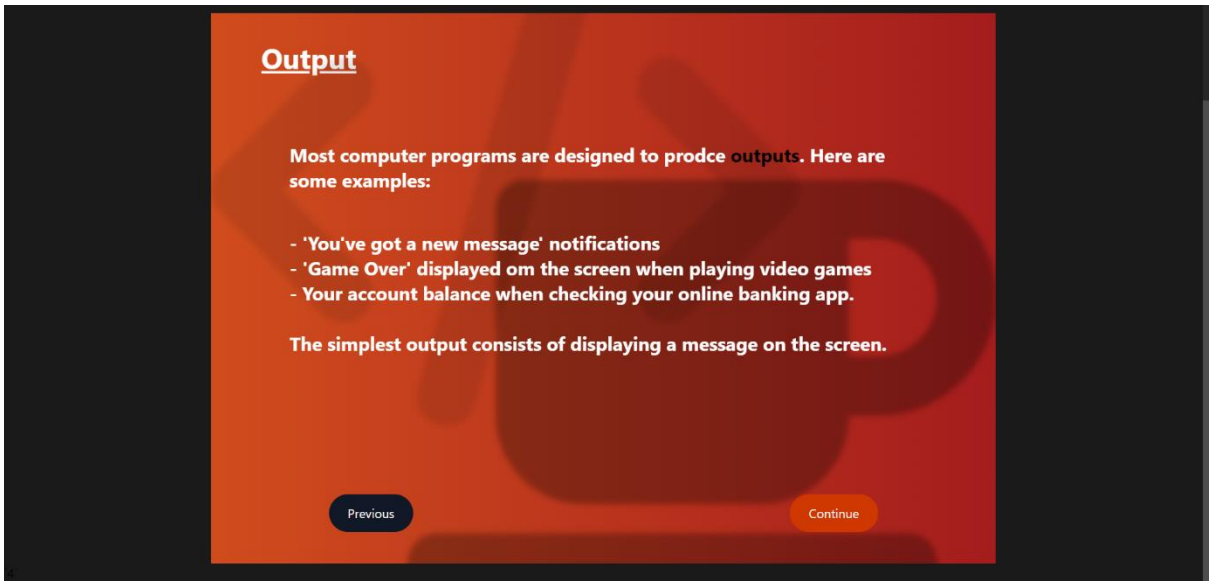


Fig 10 : Output interface

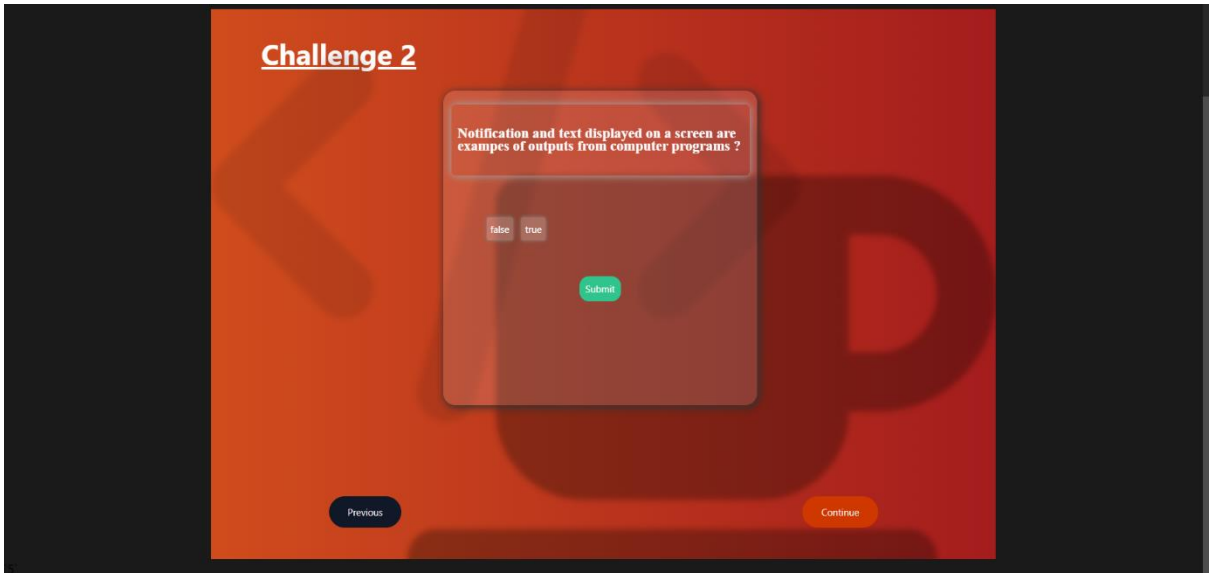


Fig 11 : Quiz

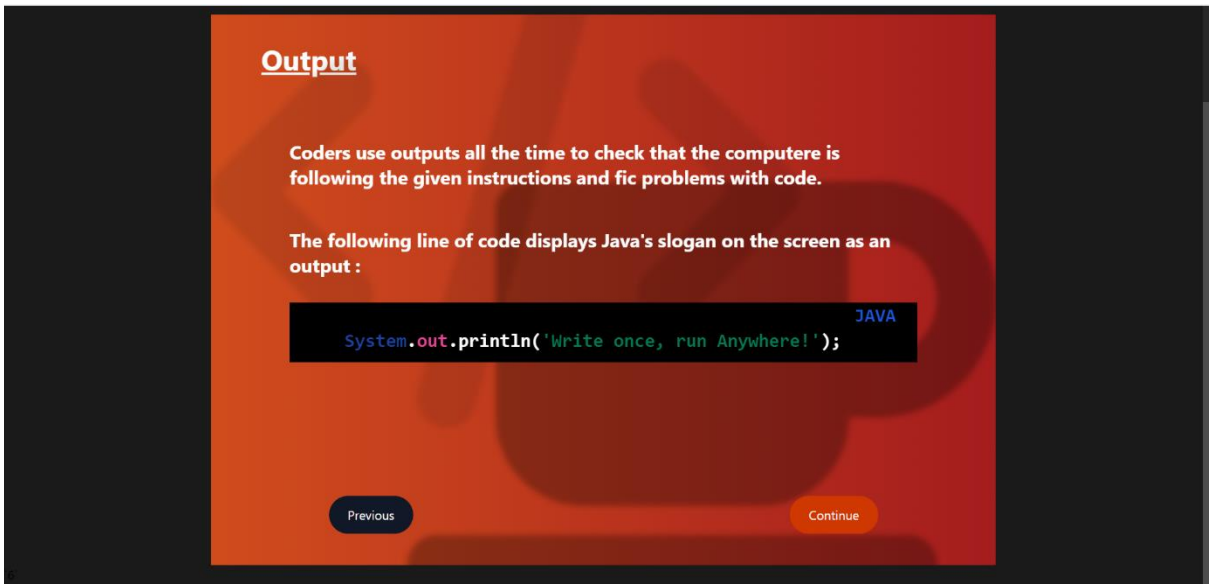


Fig 12 : Challange

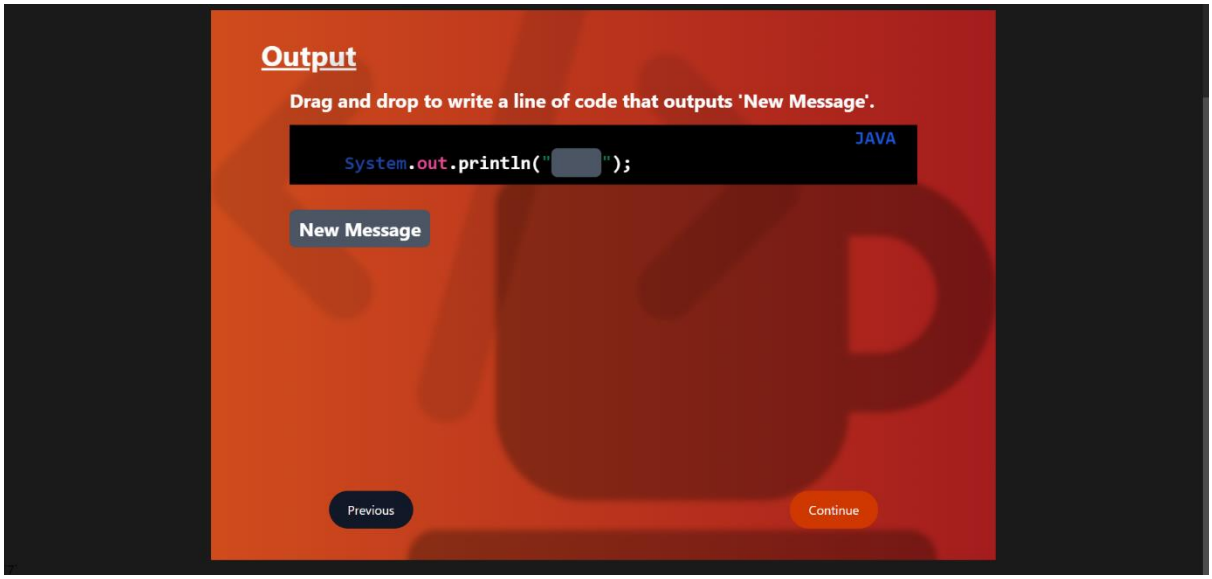


Fig 13 : Coding activity

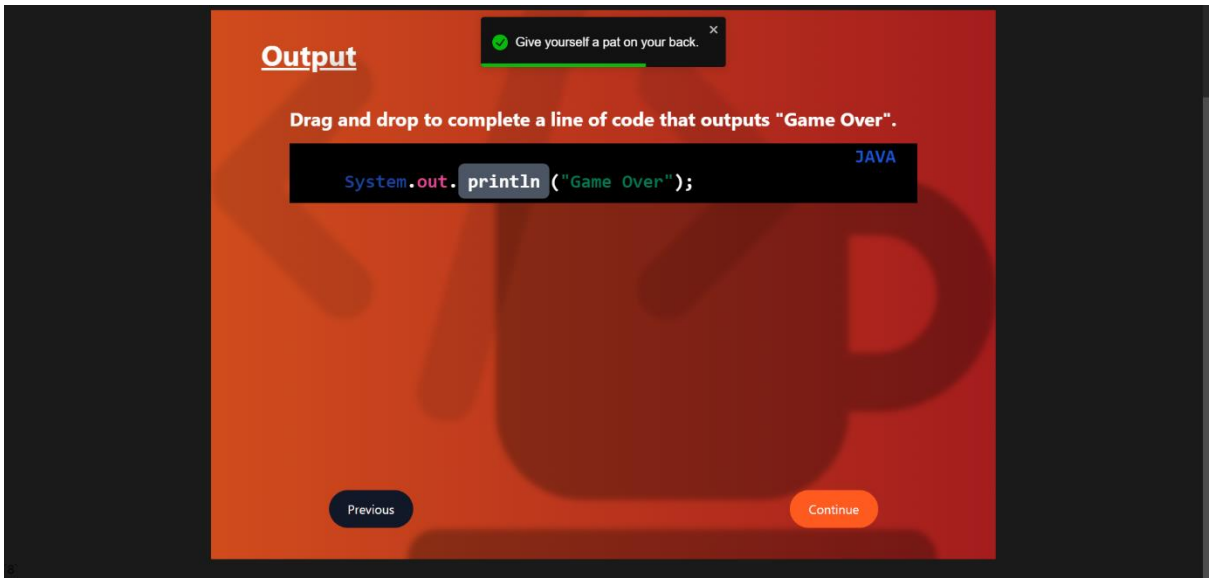


Fig 14 : Activity result

REFERENCES

REACT TYPESCRIPT DOCUMENTATION:

official react documentation: https://reactjs.org/docs/getting-started.html

typescript documentation: https://www.typescriptlang.org/docs/

tailwind css documentation:

official tailwind css documentation: https://tailwindcss.com/docs

tailwind css cheat sheet: https://nerdcave.com/tailwind-cheat-sheet

firebase documentation:

firebase documentation: https://firebase.google.com/docs

firestore documentation (for database): https://firebase.google.com/docs/firestore

firebase authentication documentation: https://firebase.google.com/docs/auth

firebase hosting documentation: https://firebase.google.com/docs/hosting

gamified learning concepts:

gamification in education: a comprehensive guide: https://elearningindustry.com/gamification-in-education-comprehensive-guide

the gamification of learning: https://www.edutopia.org/classroom-student-participation-tips

gamification in education: what, how, why bother?: https://www.sciencedirect.com/science/article/pii/s1877042813053607