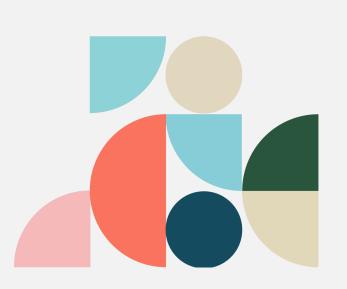Devankar Raj
devankarraj@gmail.com
Github
LinkedIn

High-Level Design
Online Judge

The main aim is to provide an overview of the system architecture. The task is to build an Online Judge using Django from scratch. The final product will allow users to submit their code in multiple programming languages, execute them and generate a verdict in real-time. The system will be constructed with a focus on ensuring robust security and scalability.

Main Components

1) **Frontend:** HTML, CSS, Javascript and Python Templates
2) **Backend:** Django framework to handle the logic
3) **Database:** Postgresql of django
4) **Other components**: Custom compiler, Docker for isolation and AWS EC2 instance to host the app

Key Functionality

1. **User Authentication and Authorization:**
   a) User registration and login page for authentication
   b) Permissions assigned based on role (e.g., Students, Admin and Employee)

2. **Database Design:**
   a. User_id to identify which user is currently active
   b. Customized Home Page based on gender and role
   c. Problem Table:
      i. Problem Id
      ii. Problem name
      iii. Problem description
      iv. Difficulty level → Easy, Medium, Hard
      v. Input and Output (base case)

d. Testcase Table:
  i. Problem Id → to which this testcases corresponds to
     1. **Use foreign key to connect the problems and test case**
  ii. Test cases → array of actual Output corresponding to each Input
     1. Test case number
     2. Input and Output → pre-defined

e. Attempt Table:
  i. Attempt number
  ii. Problem Id
  iii. User Id
  iv. Language used
  v. Date and Time of submission
  vi. Code
  vii. Final Verdict
     1. Returns "**Accepted**" if the output from user's code matches with the pre-defined output array
     2. Make changes in the attempts database
     3. Supports TLE and Runtime ERROR

f. View common stats in profile section

g. Custom Input feature

h. Can copy submitted codes in past submissions

3. **WebServer Design:**

a. UI Screens Involved:
  i. Login Page
  ii. Registration Screen
  iii. Home Page (before & after)
  iv. Dashboard with list of problems & difficulty
  v. Profile
  vi. Problem Screen → Shows problem statement with Input and Output (Base Case) with difficulty of problem
  vii. Create Testcase
  viii. Create Question
  ix. Leaderboard

b. Editor to write code along with submit, run button and Custom Input option.

    i. After clicking the submit button, feed the Input to the user's code and compare the output received with the pre-defined Output in the solution table. Generate the verdict and update the attempts table.

    ii. Run Button - Compile code and check for any error or not

    iii. Custom Input - to test code on input and code provided by user

c. Syntax highlighting and line numbers feature for code-editor

d. Shortcut keys support - Tab, Shift + Tab and Enter key in code-editor

e. Code-editor supports both HTML & Markdown language.

4. Security:

a. Custom made Compiler using python subprocess
b. Docker and Docker-Compose to maintain 2 containers → one for postgresql database and other for whole django application

5. Deployment on AWS

a. Using EC2 instance provided by AWS to deploy our whole file and host our website on internet to be accessed globally.
b. Using Git Student's Developer's Pack to have domain name and SSL certificate to by-pass chrome security layer.