Software design

Code-Now

System Design Document

CSC301 - Introduction to software engineering

Collaborators: Abdullah, Sana, Kyle, Amelia, Melissa, Mohammad, Tamseel

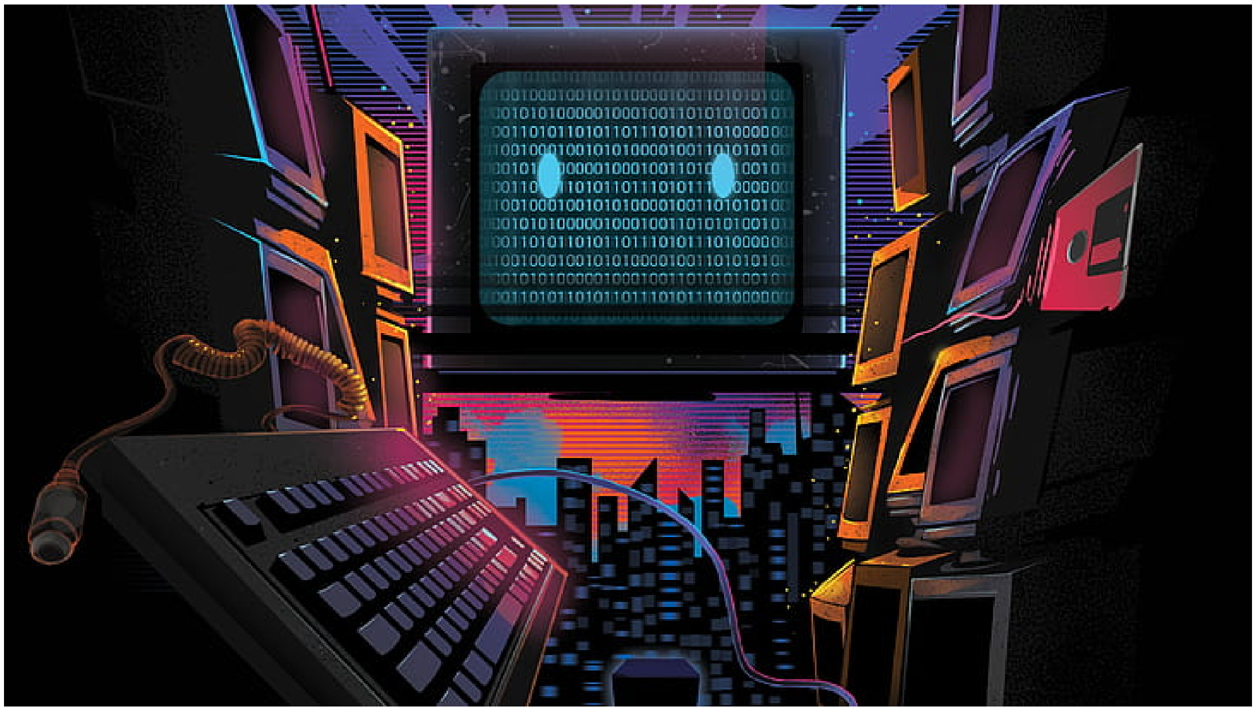
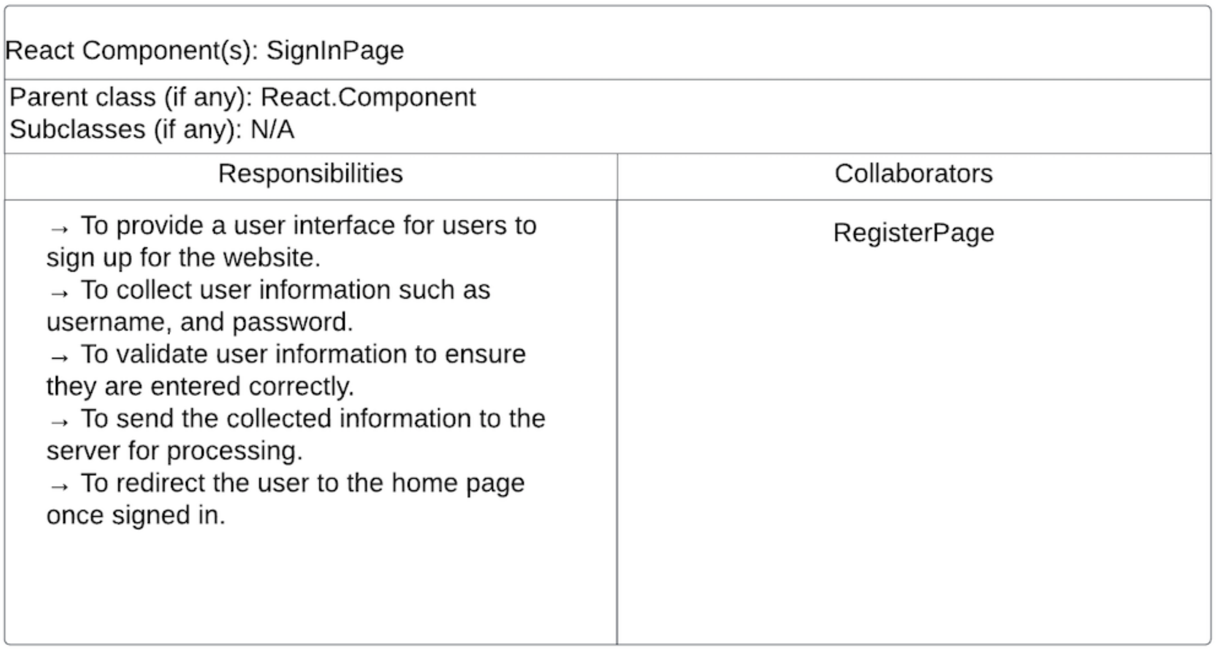


Table of Contents

Front-end CRC models

SignInPage



React Component(s): SignInPage

Parent class (if any): React.Component
Subclasses (if any): N/A

Responsibilities	Collaborators
→ To provide a user interface for users to sign up for the website. → To collect user information such as username, and password. → To validate user information to ensure they are entered correctly. → To send the collected information to the server for processing. → To redirect the user to the home page once signed in.	RegisterPage

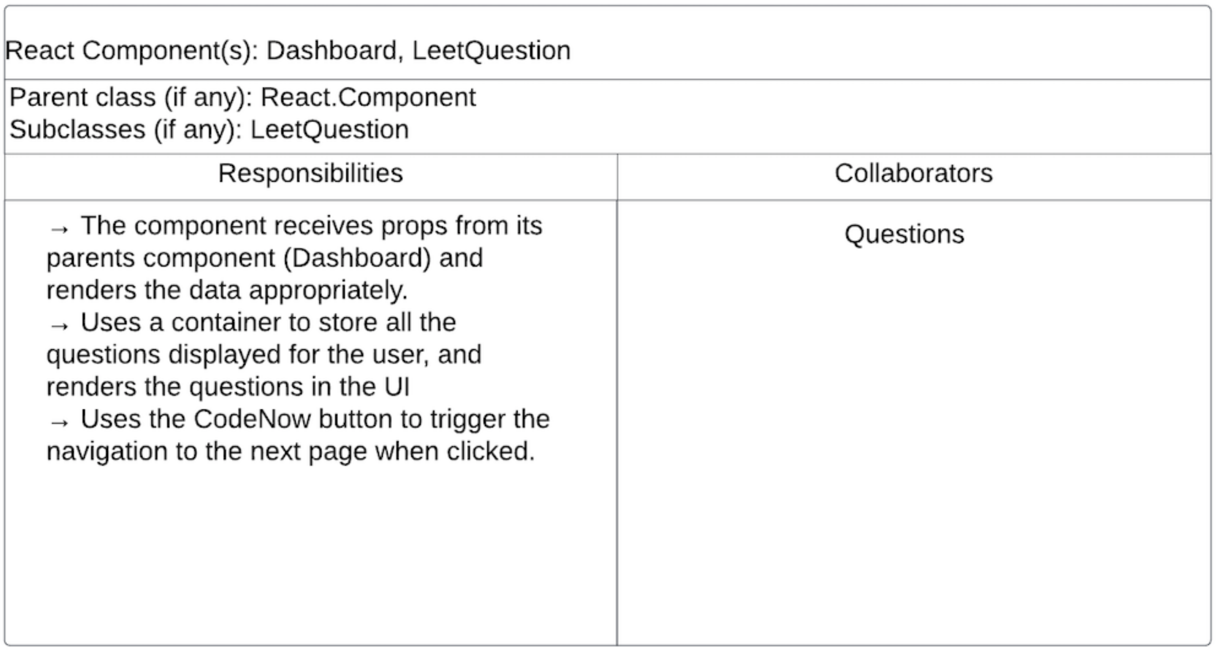
RegisterPage

React Component(s): RegisterPage	
Parent class (if any): React.Component Subclasses (if any): N/A	
Responsibilities	Collaborators
→ To provide a form for users to enter their personal information, such as username, email, and password → To validate user information to ensure they are entered correctly. → To send the collected information to the server for registration. → To receive and handle the registration response from the server → To redirect the user to the home page once signed in.	SignInPage

NavBar

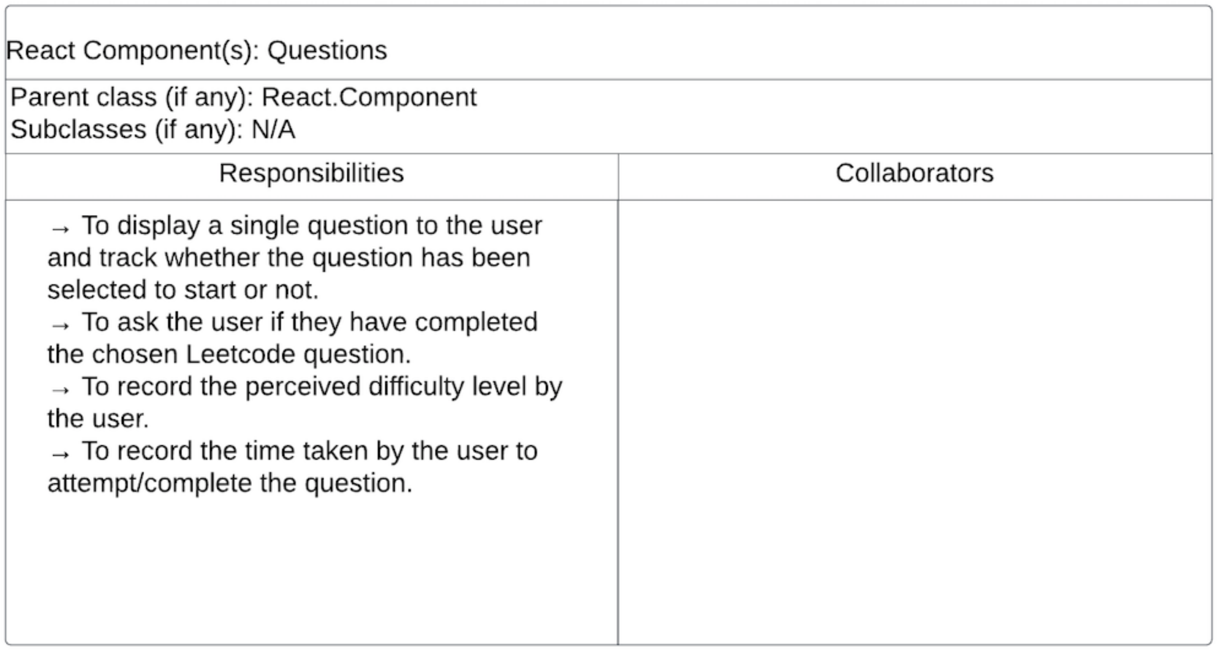
React Component(s): NavBar	
Parent class (if any): React.Component Subclasses (if any): N/A	
Responsibilities	Collaborators
→ To display navigation links → To handle user interactions with navigation links (e.g. redirects to specific page)	

Dashboard questions and the
individual Leet-code question



React Component(s): Dashboard, LeetQuestion	
Parent class (if any): React.Component Subclasses (if any): LeetQuestion	
Responsibilities	Collaborators
→ The component receives props from its parents component (Dashboard) and renders the data appropriately. → Uses a container to store all the questions displayed for the user, and renders the questions in the UI → Uses the CodeNow button to trigger the navigation to the next page when clicked.	Questions

Leetcode questionnaire form



React Component(s): Questions	
Parent class (if any): React.Component Subclasses (if any): N/A	
Responsibilities	Collaborators
→ To display a single question to the user and track whether the question has been selected to start or not. → To ask the user if they have completed the chosen Leetcode question. → To record the perceived difficulty level by the user. → To record the time taken by the user to attempt/complete the question.	

Back-end CRC models

React Component(s): Attempts	
Parent class (if any): models.Model Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class represents the database model that will store the user-submitted feedback after a problem is attempted. → To track which problems have been attempted → To track when those attempts occurred and to store a brief survey on the user's thoughts of the question.	

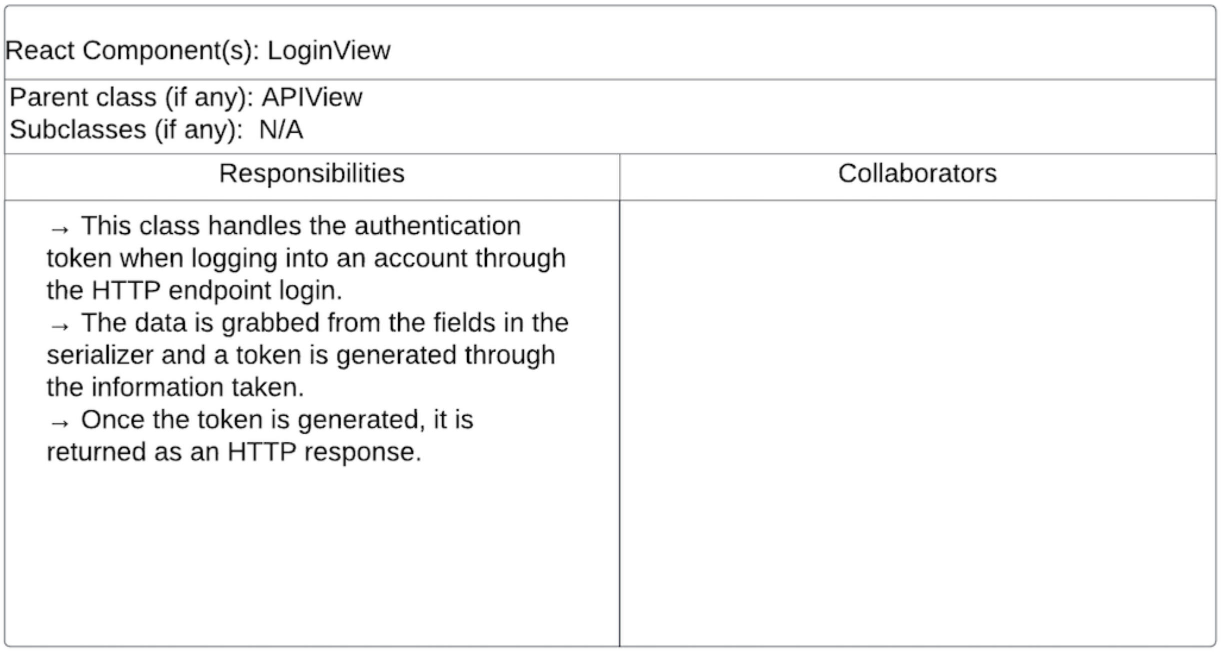
React Component(s): AttemptView	
Parent class (if any): APIView Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class handles the http get and post endpoint `/problems/attempt` and will create a new attempt record in the database when posted to. → When a get request is received, if the problem_id parameter is provided, then it will return all of a user's attempts at that particular problem. → If a problem ID is not provided, all of the user's attempts will be returned.	Attempts class for the database model.

React Component(s): Categories	
Parent class (if any): models.Model Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class represents the database model that will store the categories that questions on CodeNow will be grouped into. → Instead of categorizing questions on our own, Codenow like Coursera, will survey users after completing a question to determine which category a question should be in.	

React Component(s): CategoryView	
Parent class (if any): APIView Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class handles the http get request for `/problem/categories` and will return a Json object containing a list of all problem category names, descriptions and ids.	Categories class for the database model

React Component(s): Problem	
Parent class (if any): models.Model Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class represents the database model that will store problems. → The problems themselves are hosted on leetcode, this database model stores a link to that problem along with necessary problem metadata such as the problem name, category and id.	

React Component(s): ProblemsView	
Parent class (if any): APIView Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class handles HTTP get and post requests for our `problem` endpoint. → Get requests will retrieve a list of all available questions and relevant metadata such as name, description and category. → A post request to this endpoint will add a new question to our database. → All requests and responses will be done in JSON.	Problem class for the database model.



React Component(s): LoginView	
Parent class (if any): APIView Subclasses (if any): N/A	
Responsibilities	Collaborators
→ This class handles the authentication token when logging into an account through the HTTP endpoint login. → The data is grabbed from the fields in the serializer and a token is generated through the information taken. → Once the token is generated, it is returned as an HTTP response.	

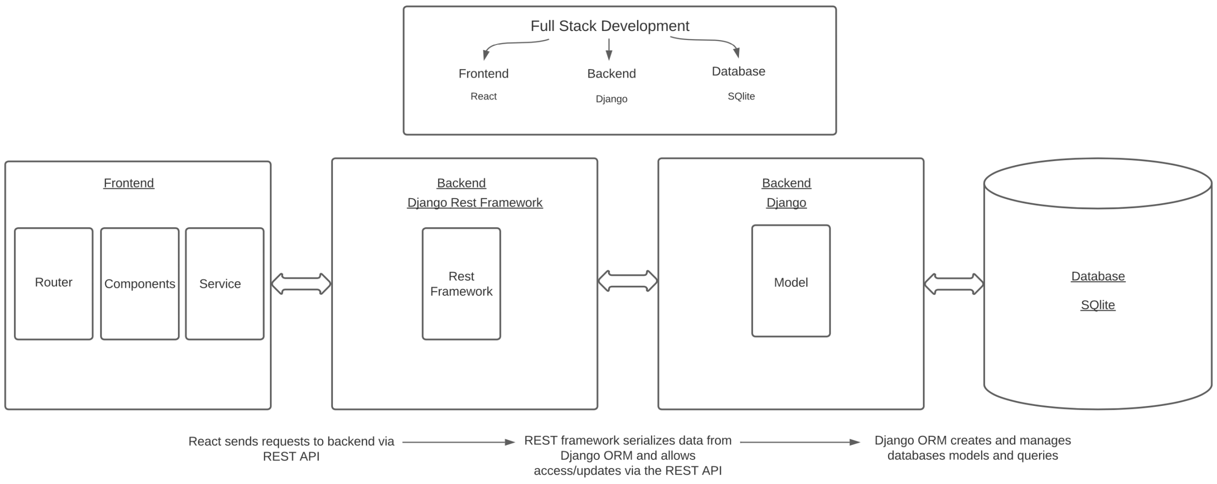
Software Architecture Diagram

This project adheres to a Three Tier architecture with React, Django, and SQLite.

The architecture diagram for using React, Django, and SQLite for a web application typically looks something like this:

1. Front-end: React is used to build the user interface of the web application. It is a JavaScript library for building user interfaces, and it will allow us for building reusable UI components and managing the state of the application.
2. Back-end: Django is used as the back-end framework for the web application. It provides an easy way to handle HTTP requests and responses, manage the database, and implement the business logic of the application.
3. Database: SQLite is used as the database management system for the web application. It is a serverless, file-based database that provides reliable and efficient data storage and retrieval capabilities.

These three technologies, when combined, offer a powerful and efficient solution for developing this website, making it a great choice for the project.


Full Stack Development
Frontend
React
Backend
Django
Database
SQlite
Frontend
Router
Components
Service
Backend
Django Rest Framework
Rest Framework
Backend
Django
Model
Database
SQlite
React sends requests to backend via REST API
REST framework serializes data from Django ORM and allows access/updates via the REST API
Django ORM creates and manages databases models and queries