CSCC01

System Design

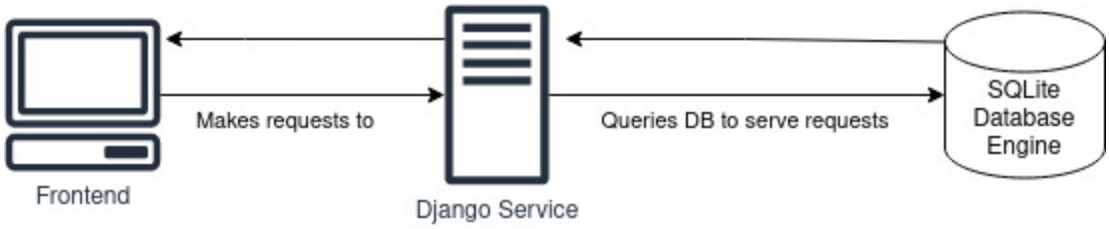
Aydin, Ben, Skandan, Howard, Tina, Pedram, Mohammad

Table of Contents

System Architecture

Our app follows a three tiered framework . The frontend is in javascript using React and is served by one server; this is the presentation layer. The backend is made up of a Django service which is the logic layer. The service uses a SQLite database to store data which forms the data layer.

Our service uses the REST protocol. All requests from the frontend are served with JSON responses from our REST endpoints. The service is responsible for making calls to the database engine when serving requests.



For further reference about three tiered architectures see the following link:

https://www.ibm.com/cloud/learn/three-tier-architecture

System Decomposition

A user is able to retrieve the frontend by making a request to the frontend server. The frontend can then be used to make requests to the service. The service will serve requests by using relevant inputs from the frontend to query the database and then return a relevant response.

The frontend is built so that unauthorized requests are only possible to our login/register routes. On registration or login, the frontend will make a request to the backend; if valid, the backend will create a session and send back a session cookie to the frontend. Using this session cookie the user will be able to make further requests for other useful functionality.

The frontend takes the approach of limiting opportunities for bad requests by constraining user input. For example, many of our forms will not allow the user to make a request until they have properly filled out the form. In the case where a user is able to make a request with bad input, the Django service will try to serialize the input or find the requested resource in the database. If the backend fails to validate the input or find the requested resource, it will return an error 400 or 404 in which case the frontend will display a relevant error message to the user.

System Interaction Description

The frontend requires the user to have Node(^18.0.0) and npm(^8.19.1) installed in order to run and install its dependencies present in package.json. Furthermore, while in development the frontend is served over localhost and runs on port 3000. The backend assumes the frontend is running on port 3000 in order to whitelist it for CORS. The backend requires Python 3.10.7 and we have provided a requirements.txt file that contains all of its dependencies.

Assumptions about having Node, npm and Python installed can be eliminated by using the docker-compose file we have provided. In this case, the user must have a recent version of docker and docker-compose installed.

The full stack can be run on Linux, Mac or Windows. Users will need a web browser such as Firefox or Chrome to access the application.

CRC Cards:

Frontend:

The root of all files is src.

For frontend components, we consider a child component to be a collaborator. It does not have an “is-a” relationship with its parent. Rather, it’s parent “has-a” child component of that type.

Apart from the index.tsx file at the root; there are many index.tsx files in sub-directories. These are used to consolidate named exports and offer a simple name space for other components to import from.

Class Name	index.tsx
Parent/Subclass	None
Responsibilities:	• Provides routing, query client for react query and localization for dates and times and user auth context for all child components • Provides basic styling of the application • Hosts the App
Collaborators:	• App • contexts/UserContext.tsx

Class Name	contexts/UserContext.tsx
Parent/Subclass	None
Responsibilities:	• Determines whether the user is logged in or not • Provides user data such as username and role to child components • The Contexts folder additionally exports a utility function that makes using the UserContext easier (simplifies the types); for classification/CRC purposes we consider it to be a part of UserContext

Collaborators:	• Index.tsx • App.tsx • pages/Login/Login.tsx

Class Name	App.tsx
Parent/Subclass	None
Responsibilities:	• Does not allow pages to be rendered until user auth state is determined • Provides routing to all other pages of the application
Collaborators:	• components/PrivateRoute.tsx • pages/CreateProjectPage/CreateProjectPage.tsx • pages/Login/.Login.tsx • pages/Register/Register.tsx • contexts/UserContext.tsx

Class Name	components/PrivateRoute.tsx
Parent/Subclass	None
Responsibilities:	• Redirects users to login page if they try to access a page that they are not authorized to access
Collaborators:	• App.tsx • pages/NavBar/NavBar.tsx

Class Name	pages/Login/Login.tsx
Parent/Subclass	None
Responsibilities:	• Allows users to login to the application
Collaborators:	• App.tsx • contexts/UserContext.tsx • api/api-client.tsx

Class Name	api/api-client.tsx
Parent/Subclass	None
Responsibilities:	• Provides all calls that will be made to the service
Collaborators:	• pages/Login/.Login.tsx • pages/Register/Register.tsx • pages/CreateProjectPage/CreateProjectPage.tsx • pages/Projects/ProjectsPage.tsx

Class Name	api/api-client-types.tsx
Parent/Subclass	None
Responsibilities:	• Provides the types for all requests/responses made to the service
Collaborators:	• api/api-client.tsx

Class Name	api/constants.tsx
Parent/Subclass	None
Responsibilities:	• Provides configuration variables to the api-client
Collaborators:	• api/api-client.tsx

Class Name	Pages/Register/Register.tsx
Parent/Subclass	None
Responsibilities:	• Allows the user to register for an account • Sets user auth state once they have registered
Collaborators:	• App.tsx • api/api-client.tsx

	• api/api-client-types.tsx • contexts/UserContext.tsx

Class Name	pages/CreateProjectPage/CreateProjectPage.tsx
Parent/Subclass	None
Responsibilities:	• Manages the request to create a project • Renders project creation form or resulting invite code depending on whether the request has been made yet
Collaborators:	• App.tsx • api/api-client.tsx • api-client-types.tsx • pages/CreateProjectPage/components/CreateProjectForm.tsx

Class Name	pages/CreateProjectPage/components/CreateProjectForm.tsx
Parent/Subclass	None
Responsibilities:	• Renders the form that allows professors to create projects
Collaborators:	• api-client-types.tsx • pages/CreateProjectPage/components/interfaces.tsx • pages/CreateProjectPage/components/styles.tsx • pages/CreateProjectPage/components/GroupSlider.tsx • pages/CreateProjectPage/components/EndDatePicker.tsx • hooks/useOnMount.tsx

Class Name	pages/CreateProjectPage/components/interfaces.tsx
Parent/Subclass	None
Responsibilities:	• Provides common interfaces for CreateProjectPage related components to use

Collaborators:	• pages/CreateProjectPage/components/CreateProjectForm.tsx • pages/CreateProjectPage/components/EndDatePicker.tsx • pages/CreateProjectPage/components/GroupSlider.tsx

Class Name	pages/CreateProjectPage/components/styles.tsx
Parent/Subclass	None
Responsibilities:	• Provides common styles used by CreateProjectPage related components
Collaborators:	• pages/CreateProjectPage/components/CreateProjectForm.tsx • pages/CreateProjectPage/components/InviteCode.tsx

Class Name	pages/CreateProjectPage/components/InviteCode.tsx
Parent/Subclass	None
Responsibilities:	• Renders invite code and allows user to copy it once project has been created
Collaborators:	• pages/CreateProjectPage/components/CreateProjectForm.tsx • pages/CreateProjectPage/components/CopyIcon.tsx • pages/CreateProjectPage/components/styles.tsx

Class Name	pages/CreateProjectPage/components/CopyIcon.tsx
Parent/Subclass	None
Responsibilities:	• Allows a user to copy text by clicking on it
Collaborators:	• pages/CreateProjectPage/components/InviteCode.tsx

Class Name	pages/CreateProjectPage/components/EndDatePicker.tsx
Parent/Subclass	None
Responsibilities:	• Allows the user to pick a date and converts it to UTC time

Collaborators:	• pages/CreateProjectPage/components/CreateProjectForm.tsx • pages/CreateProjectPage/components/interfaces.tsx

Class Name	pages/CreateProjectPage/components/GroupSlider.tsx
Parent/Subclass	None
Responsibilities:	• Allows user to specify the group size on create project form
Collaborators:	• pages/CreateProjectPage/components/CreateProjectForm.tsx • pages/CreateProjectPage/components/interfaces.tsx

Class Name	pages/Projects/components/JoinProject
Parent/Subclass	None
Responsibilities:	• Allows a user to join a project by entering an invite code • Shows an error if user enters incorrect code
Collaborators:	• pages/Projects/ProjectsPage

Class Name	pages/Projects/components/CreateProjectButton
Parent/Subclass	None
Responsibilities:	• Redirects user to Create Project Page (does not directly collaborate with page, since React router handles redirects)
Collaborators:	• pages/Projects/ProjectsPage

Class Name	hooks/useOnMount.tsx
Parent/Subclass	None
Responsibilities:	• Exports a hook that runs an effect only once
Collaborators:	• pages/CreateProjectPage/components/EndDatePicker.tsx

Class Name	api/constants.tsx
Parent/Subclass	None
Responsibilities:	• Exports constants used by the api client
Collaborators:	• api/api-client.tsx

Class Name	pages/Projects/ProjectsPage
Parent/Subclass	None
Responsibilities:	• Renders the ProjectList, JoinProject, and CreateProjectButton components
Collaborators:	• pages/Projects/components/JoinProject • pages/Projects/components/CreateProjectButton • pages/Projects/components/ProjectList • pages/Projects/components/ProjectCard • api/api-client.tsx

Class Name	pages/Projects/components/ProjectList
Parent/Subclass	None
Responsibilities:	• Displays a list of project cards for each project a user is involved with
Collaborators:	• pages/Projects/ProjectsPage • pages/Projects/components/ProjectCard

Class Name	pages/Projects/components/ProjectCard
Parent/Subclass	None
Responsibilities:	• Displays the name, professor, and due date for a single project
Collaborators:	• pages/Projects/components/ProjectList

	• pages/Projects/ProjectsPage

Class Name	pages/NavBar/NavBar.tsx
Parent/Subclass	None
Responsibilities:	• Displays a drawer with different options such as logout and navigating to other pages
Collaborators:	• components/PrivateRoute.tsx

BACKEND

Models

Class Name	accounts.models.User
Parent/Subclass	AbstractBaseUser, Permissions Mixin

Responsibilities:	• Contain:
	  ○ Email
	  ○ Name
	  ○ Is_active
	  ○ Is_student
	  ○ Is_professor
	  ○ Is_admin
	  ○ Is_staff
	  ○ Is_superuser
	• Hashing passwords

Collaborators:	• Student • Professor

Class Name	accounts.models.UserManager
Parent/Subclass	BaseUserManager
Responsibilities:	• Add a custom create_user method to the Users
Collaborators:	• User

Class Name	accounts.models.Professor
Parent/Subclass	Model
Responsibilities:	• Store all the professor related fields
Collaborators:	• User

Class Name	accounts.models.Student
Parent/Subclass	Model
Responsibilities:	• Store all the student related fields
Collaborators:	• User

Class Name	projects.models.Project
Parent/Subclass	Model

Responsibilities:	• Contain:
	  ○ Project_name
	  ○ Min_group_size
	  ○ Max_group_size
	  ○ End_date
	  ○ Description
	  ○ Join_code
	  ○ Students
	  ○ Professor
	• Generate a join code on object creation

Collaborators:	• Student • Professor

Class Name	groupr.CsrfExemptSessionAuthentication
Parent/Subclass	SessionAuthentication
Responsibilities:	• Enable sessions authentication for the app but disable CSRF protections
Collaborators:	•

Class Name	accounts.admin.CustomUserAdmin
Parent/Subclass	UserAdmin
Responsibilities:	• Customize the form used to edit User database entries in django admin
Collaborators:	• User

Class Name	accounts.admin.CustomUserAdmin
Parent/Subclass	UserAdmin
Responsibilities:	• Customize the form used to edit User database entries in django admin
Collaborators:	• User

Class Name	accounts.serializers.UserRetrievalSerializer
Parent/Subclass	Serializer
Responsibilities:	• Serialize a User object to JSON
Collaborators:	• User

Class Name	accounts.serializers.UserRegisterSerializer
Parent/Subclass	Serializer
Responsibilities:	• Deserialize the User json to validate the fields and create a user object
Collaborators:	• User

Class Name	accounts.serializers.ProfessorSerializer
Parent/Subclass	Serializer
Responsibilities:	• Serialize a professor object, discarding any sensitive user information before returning a JSON
Collaborators:	• UserRetrievalSerializer

Class Name	accounts.serializers.ProfessorSerializer
Parent/Subclass	Serializer
Responsibilities:	• Serialize a professor object, discarding any sensitive user information before returning a JSON
Collaborators:	• UserRetrievalSerializer

Class Name	projects.serializers.ProjectSerializer
Parent/Subclass	Serializer
Responsibilities:	• Serialize and deserialize the Projects for JSON • Create a project object with a generated join code
Collaborators:	• ProfessorSerializer

Class Name	projects.serializers.ProjectSerializer
Parent/Subclass	Serializer
Responsibilities:	• Serialize and deserialize the Projects for JSON • Create a project object with a generated join code
Collaborators:	• ProfessorSerializer

Views

Class Name	accounts.views.LoginView
Parent/Subclass	APIView
Responsibilities:	• Accept POST requests • Authenticate the user email and password • Attach a session cookie if the user is authenticated • Return a response body with the user object if authenticated
Collaborators:	• UserRetrievalSerializer

Class Name	accounts.views.RegisterView
Parent/Subclass	APIView
Responsibilities:	• Accept POST requests • Validate the user register data • Create a new user if the data is valid • Attach a session cookie after create • Return the response body with the user object if valid
Collaborators:	• UserRetrievalSerializer

Class Name	accounts.views.RetrieveUserView
Parent/Subclass	APIView
Responsibilities:	• Accept GET requests • If a valid session cookie is present, return the user information in JSON format
Collaborators:	• UserRetrievalSerializer

Class Name	projects.views.ProjectView
Parent/Subclass	APIView

Responsibilities:	• Accept GET and POST requests
	• If the user is authenticated
	  ○ GET
	  ■ Return a list of all projects that the user has joined
	  ○ POST
	  ■ Deserialize the request data
	  ■ Only allow professors to create projects

	■ Create a new project database entry if the data is valid ■ Return the generated join_code in the response body
Collaborators:	● ProjectSerializer

Class Name	projects.views.ProjectJoinView
Parent/Subclass	APIView
Responsibilities:	● Accept GET requests ● If the user is authenticated ○ Only allow students to retrieve and join project ○ Join the project corresponding to the join code in the URL ○ If the user is already joined, don't do anything ○ Return a serialized project object JSON in the response body and whether the student has just joined the project as a result of the request
Collaborators:	● ProjectSerializer