Boston University
Electrical & Computer Engineering
EC463 Senior Design Project

First Prototype Testing Plan

Stöd

By

Team 11
Stöd

Team Members:
Sharith Godamanna | sharithg@bu.edu
Camden Kronhaus | kronhaus@bu.edu
Quinn Meurer | quinnyyy@bu.edu
Alexander Trinh | aktrinh@bu.edu

Required Materials (Technologies)

Frontend

- ReactJS
- Redux
- Material UI
- TypeScript

Backend

- Django
- Django Rest Framework
- Python3

Setup

The Stod application architecture is split into two parts: the backend and frontend, which communicate through HTTP requests. This allows for modularity and can easily be migrated to a microservices architecture. To start with a backend template, we create a Python virtual environment and install Django within it. To create a new Django application we run `django-admin startproject stod-backend` which creates an empty django application. For the frontend, we create a new React app running `yarn create stod-frontend --typescript` specifying a typescript template. We decided to use typescript to be able to use type safe variables which makes debugging much simpler. To handle global React state, we are using Redux which manages a global store where we can store and access global variables. To connect our frontend to the backend, we created asynchronous redux actions and used the axios library to make HTTP requests to our server.

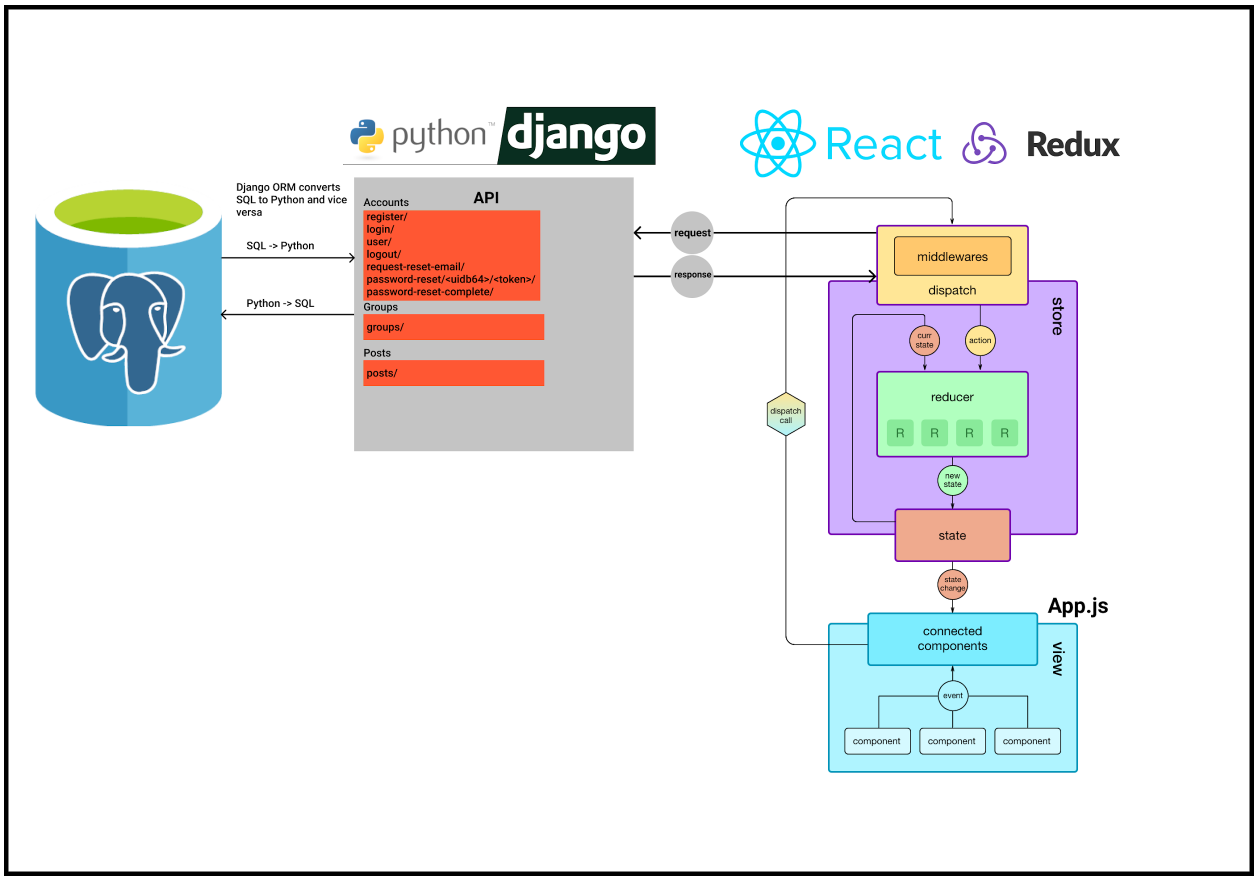

Figure 1: Stod application architecture

Pre-testing Setup Procedure:

- Run the back end server:
  `python stod-backend/stod/manage.py runserver`
- Run the front end server:
  `cd stod-frontend && yarn start`
- Add dummy posts to database
- Add dummy groups to database
- Add dummy comments to database

Testing Procedure:

Login:

1. Register a new user with:
    a. Username: testuser99
    b. Email: testuser99@bu.edu
    c. Password: SecurePW@54367
2. Login with newly created user account
    a. Username: testuser99
    b. Password: SecurePW@54367
3. Attempt to login with a user that does not exist:
4. Error checking, (we would try all these errors to see if the application handles them correctly).
    a. Register errors:
        - Field was left empty
        - Username and/or email already exists
        - Email is invalid
        - Password too common
        - Password not long enough
        - After successfully registering, page does not get redirected to /home page.
    b. Login errors:
        - Field was left empty
        - Username and password combination invalid
        - After successfully logging in, page does not get redirected to /home page

Posts:

1. Navigate to the /posts route
2. View all posts currently available on the back end server and confirm all posts in the back end database are visible on the posts page
3. Create a new post using Django REST framework.
4. Refresh posts page and view all posts

Comments:

1. Create multiple posts available for commenting
2. Comment on the newly created posts
3. View all of the comments

Groups:

1. View the list of groups on the frontend. The name and description for each group should be visible.
2. Delete a group on the frontend. The frontend should be updated with the group removed.
3. Add a new group by specifying a name and a description. The frontend should be updated with the new group visible.

Measurable Criteria

Login:

1. After registering a new user, the user data is successfully stored in the Django database
2. After a new user is created, login successfully takes new user to the homepage
3. Login does not work with a user that doesn't exist
4. All errors to be check checked are processed normally without crash

Posts:

1. New posts added to back end database appear in order on front end route
2. All posts appear with appropriate data fields in front end (poster, title, body, group)

Groups:

1. The groups displayed in the frontend should be consistent with the data in the database.
2. After deleting a group from the frontend, the deletion should be reflected in the database.
3. After adding a group from the frontend, the new group should appear in the database.

Commenting:

1. Able to create a comment under a previously created post
2. The comments should be connected to the id of the post
3. Able to view all previously created comments