Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Second 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.

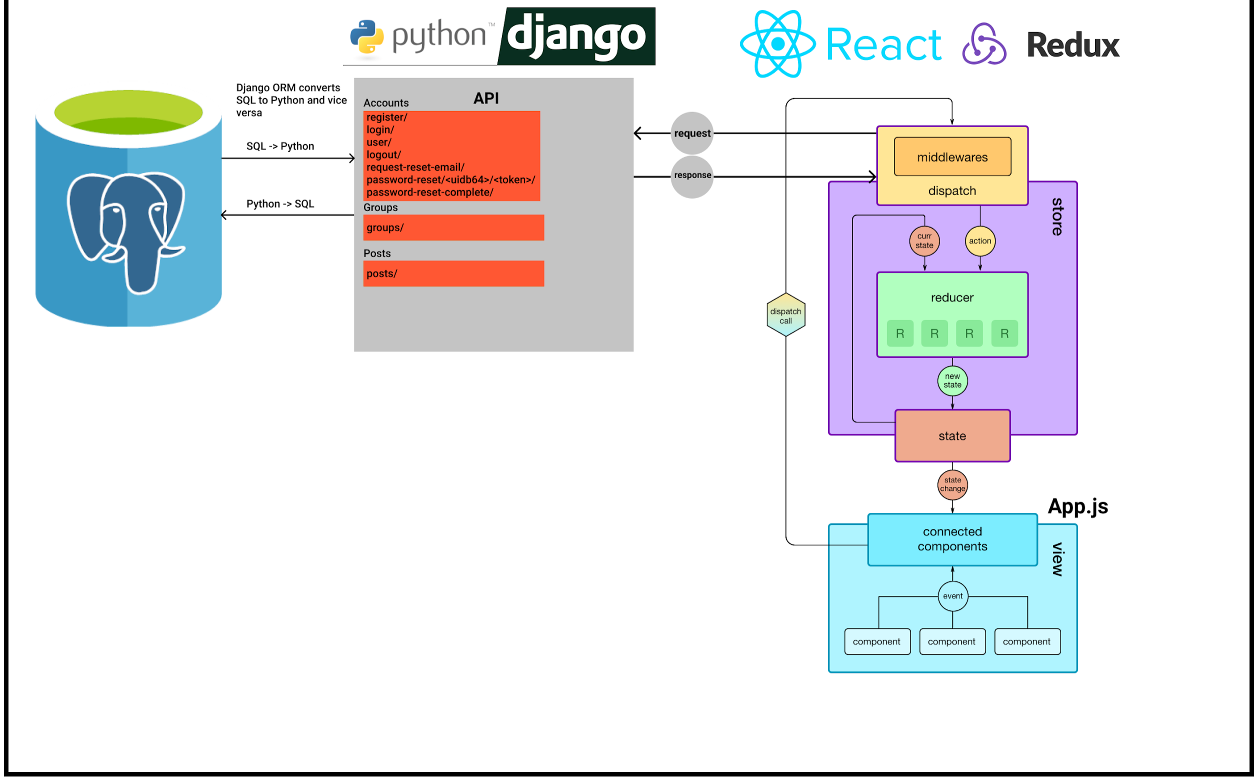

Figure 1: Stod application architecture

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. Create a new user and login as that user

Groups:

1. Navigate to home page and click “Find Groups” to view all existing groups
2. Click “subscribe” on a group
3. Navigate back to the home page see list of groups in the sidebar
4. Click on a subscribed group in the sidebar and view all the posts to that group

Posts:

1. Navigate to home page and view all posts loaded on the home page
2. Navigate to a subscribed group to view all the posts within that group
3. Create a new post from within a subscribed group
4. Edit a post created by the current user, and view updates
5. Delete post created by a user, and view updates

Comments:

1. Comment a single comment under each post within a group
2. Choose a post and add multiple comments
3. View each comment under their respective posts

Direct Messaging:

1. Check on user sidebar, make sure previous messages load
2. Send a single message and make sure it gets sent
3. Makes sure message gets sent to correct side

Measurable Criteria

Login:

1. A new user can be created and logged in as that user at any point

Groups:

1. All available groups are displayed to the user after clicking “Find Groups”
2. User is able to subscribe to new groups
3. Subscribed groups are displayed in sidebar
4. Upon clicking on a subscribed group in the sidebar, user can view posts from that group

Posts:

1. Navigation to the home page results in only posts for that group being displayed
2. Navigation to a subscribed groups results in only posts for that group being displayed
3. Creation of a post is reflected accurately in both the front and back end after submission
    a. Creation of a post also accurately shows date of creation and poster username
4. Editing and Deleting a post is available to the user who created them
5. Edits to a post are reflected accurately in both the front and back end after submission
6. Deletion of a post is reflected accurately in both the front and back end after submission

Commenting:

1. Able to create a comment under a previously created post
2. Comments should be updated and visible automatically without refresh
3. The comments should be connected to the id of the post
4. Only comments pertaining to a certain post should be loaded under that post
5. All previously created comments under a post should always load under that post

Direct Messaging:

1. Websocket connection starts between two users
2. Messages from one user directly shows up on another
3. Message list shows in correct order after refresh