SET09103
Advanced Web Technologies

Coursework Part #2

Vinh Phat Tu
40507973

November 2023

Vinh Phat Tu
40507973

Contents

1 Introduction

This coursework part will discuss the implementation of recreating /r/place, an event and project made by Reddit.com, a discussion board website.

2 Implementation

Squarish is a React application that presents a 1000x1000 grid of squares which registered users can modify. The user selects a color from the preset colors on the left hand side and then select the square they wish to color. Each user can only change a square every five minutes, allowing for collaboration or competition amongst the player base.

3 Comparison from outlined plan

4 Additional features and future considerations

5 Reflection

Talk about

- Grid rendering using canvas, konvajs, pixijs
- Talk about JWT, authentication and how it relates to WEbsocket
- Talk about websockets
- Talk about decision to use PocketBase

6 Deliverable

The plan is to initially recreate the basic functionality of /r/place with its core elements.

6.1 Features

The website's main functionality centers around the ability for users to view and modify squares on a single canvas. The server (the Flask backend) handles the requests of the users as a central hub and using Websocket, transmits the changes on the board to everyone viewing.

Following is a list of core features (must-haves):

Feature	Description	Reason
Canvas	Implement a pixel canvas that will display the current state	Main feature of the website
Websocket	Using websocket protocol to allow for real-time updates	Give users instant feedback on changes to the canvas

Table 1: Main features

6.2 Additional features

As for additional features (nice-to-haves), these could be considered once the main functions have been implemented.

Feature	Description	Reason
Login system	System to track users	Potential personalisation
Timeout	Restrict canvas editing frequency	Reduce bot behavior

Table 2: Additional features

7 Navigation tree and wireframe

7.1 Navigation tree

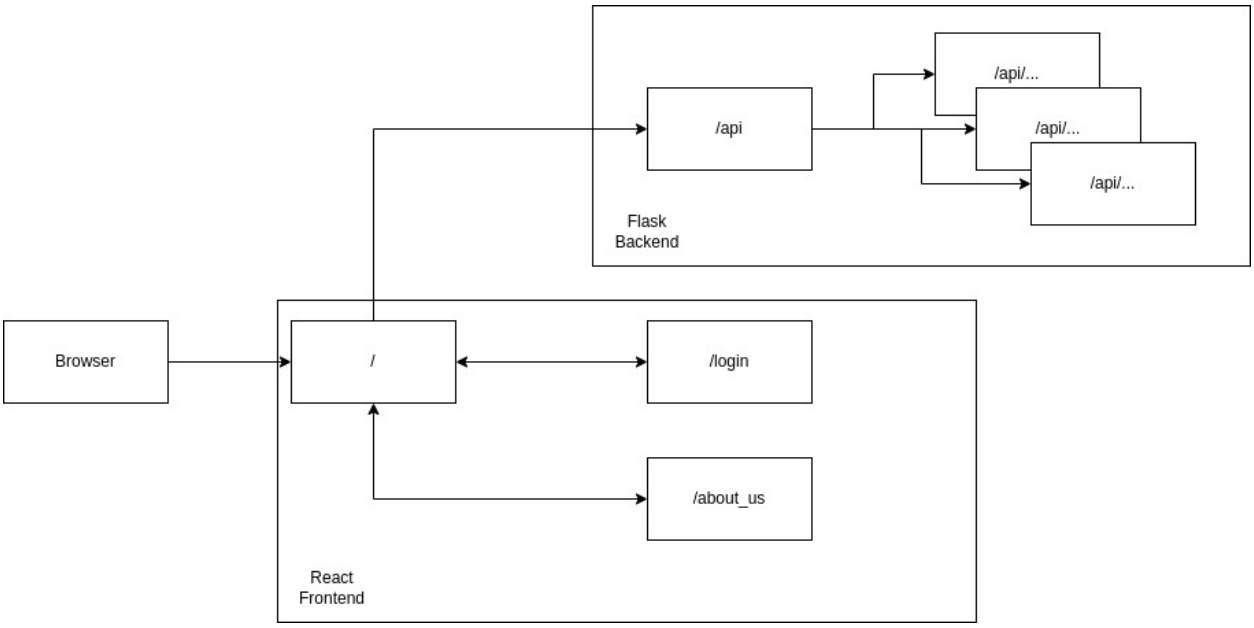

Figure 1: Navigation tree for single-page React application with Flask backend

The navigation tree for this application is as shown in 1. Deciding between a single-page application (SPA) and a multi-page application (MPA) came down to multiple concerns such as user experience and use case. SPAs allow the user to interact with the application without having to browse through pages for actions such as log-in. Single-page applications allow the users to view and interact with dynamic content while MPAs allow for simpler static content.

For this application, the main functionality can be displayed neatly within a single-page as shown in pre-existing applications. As the application will also be using Websocket, the content will be dynamic and thus a single-page application approach is preferred.

There is also a separation between the front-end and back-end, with the user-interface (UI) being built in Javascript and the React library and the server being built in Python and the Flask library. Although Flask allows to dynamically generate static HTML pages through Jinja2 Templates, there is minimal flexibility for dynamic content. In addition, Jinja2 Templates are primarily for creating static pages which again will not be the direction for this application. And therefore, a more suitable library such as React will be used to create the UI.

7.2 Wireframe

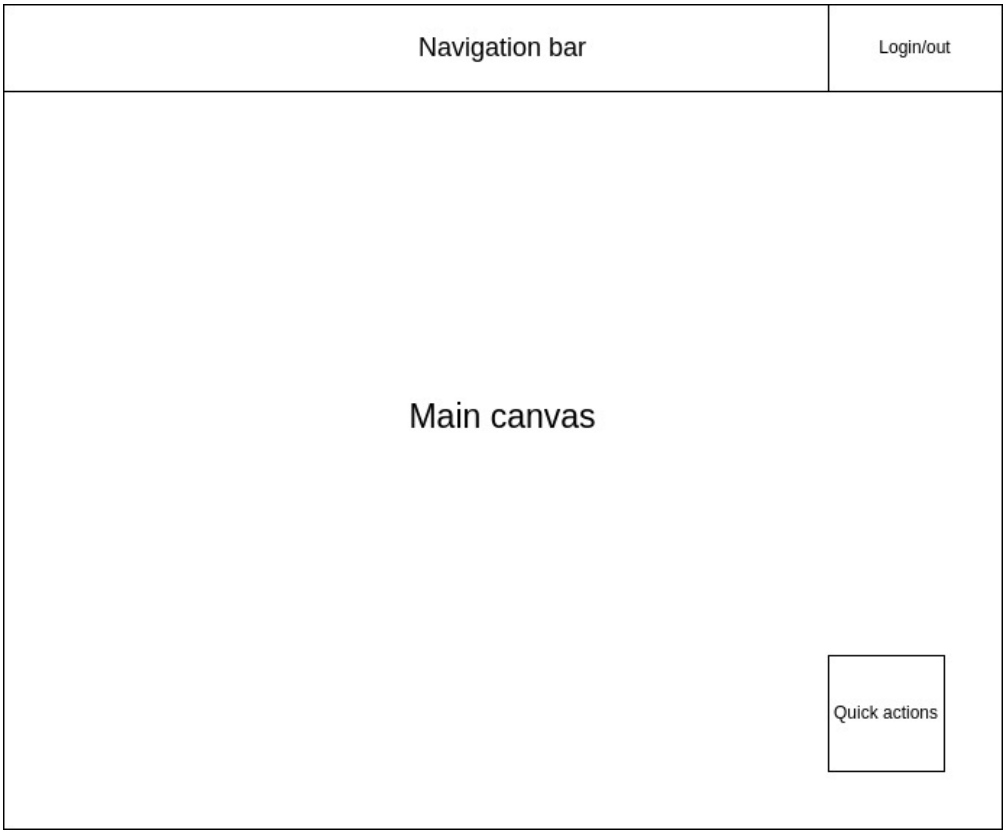

Figure 2: Wireframe for application

8 Research Spikes

8.1 Static (Jinja2) versus dynamic (React) webpages

Flask allow for variables and thus data to be injected into Jinja2 Templates. And whenever a user tries to reach an endpoint, the server generates a HTML page based of the template filling in all the values. Therefore, for a large portion of static content and applications with a traditional content-driven website, this approach is sufficient for most use cases.

However, when considering dynamic content, using templating becomes complex and in many cases suboptimal to other approaches. Javascript frameworks such as React and Angular focus on generating dynamic applications for dynamic content.

A framework such as React also allows for a layer of abstraction from HTML and CSS, which allows for a quicker development cycle. The Javascript landscape has a huge variety of libraries and dependencies that can eliminate tedious tasks such as layout, alignment and even offers complete building blocks for pages. In essence, with all these tools available and also actively used in industry, developing a user interface in React reduces the tedious and low-level work, thereby providing more time for server-side development.

8.2 Websocket protocol

Websocket is a two-way communication protocol that allow a client to have a interactive session with the server [1].

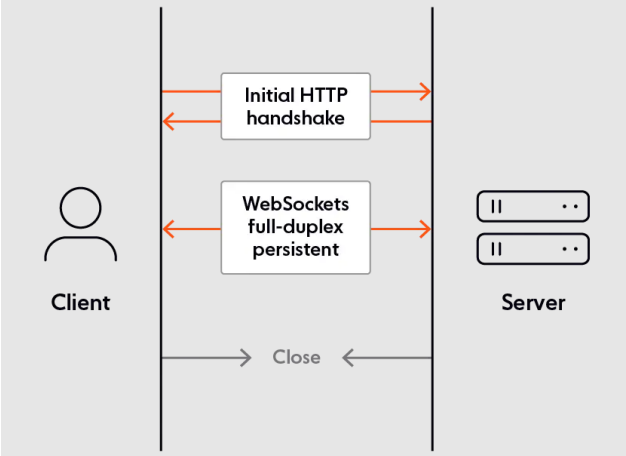

Figure 3: Websocket communication [2]

As shown in 3, the connection is established with an initial HTTP handshake. Once connected, the client and server can exchange data through the Websocket connection. The session can be closed on either side with a close message [2].

8.2.1 Why Websocket?

The application should be able to take many changes from many users. For example, if a user changes the color on square (0,0) on the grid, other users that are currently viewing the canvas should be able to receive the change without having to refresh the page.

Websocket allow real-time communication and updates between the server and the client, without needing many HTTP requests. The protocol allow for low latency and therefore a more responsive application. However, Websocket unlike HTTP is a stateful protocol which

introduces complexity to the server-side [2].

This protocol fulfills the real-time requirement for the current use cases of the application. It will be an added complexity that will have to be dealt with during development, especially when considering a possible authentication system.

8.2.2 Socket.IO

Socket.IO is a library that builds on top of the WebSocket protocol that adds additional capabilities such broadcasting capabilities and compatibility concerns [3]. The library is available for both Javascript and Python Flask and can thus be used.

There was a consideration for using WebSockets without Socket.IO. Javascript and modern browsers do have native support for the protocol through browser APIs [1]. However when looking at Flask and its implementations, the most up-to-date examples and approaches use SocketIO rather than raw WebSockets. This is not to say that other implementations or libraries don't exist (see flask-sockets)but SocketIO seems to be the most popular solution with an active community and developers behind it.

9 Technologies

9.1 Client-side (front-end)

For the client-side/front-end, React will most likely be used to design the UI.

As for Websocket communication, the Socket.io library will be used configure and interact with the sessions between the server and client.

9.2 Server-side (back-end)

On the back-end a Python Flask server will be used to handle requests from the front-end. To create a Websocket route in Flask, the Flask-SocketIO library will be used.

10 Data

In the initial version of the application, the only persistent data is the canvas. Storing a $n \times n$ grid can be done easily with an n-dimensional array, that can then either be serialized and stored in a discrete file or in a database.

For further extensions such as the login system, a database will likely be needed to track the users credentials. This would also raise a security concern as credentials are sensitive information that need to be handled correctly, both in transit and at rest. Complexities such as how to store the data and how to encrypt the data will have to be decided and implemented at that stage.

References

[1] MozDevNet. *The Websocket API (WebSockets).* 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[2] Alex Diaconu. *What is WebSocket?* 2023. URL: https://ably.com/topic/websockets?utm_campaign=evergreen.

[3] Priya Pedamkar. *WebSocket vs Socket.io.* 2023. URL: https://www.educba.com/websocket-vs-socket-io/.

A r/place canvas

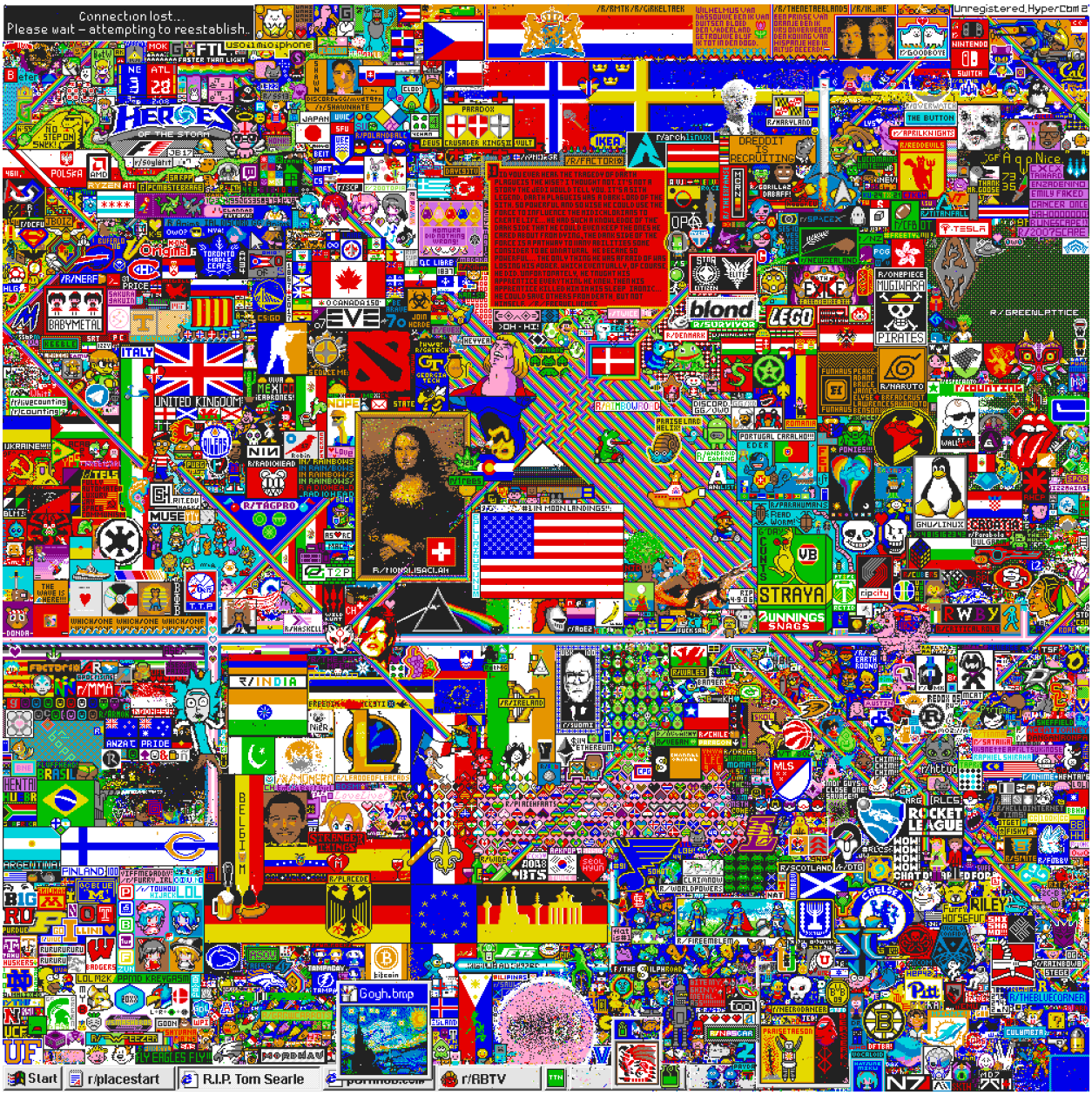

Figure 4: Final state of canvas in 2017