CS307 Design Document

Team 16

Ancil Trent, Brandon Xu, Charlie Hyun, Lucy Cheng, Mark Lim, Sarah Mi

Index

Purpose

In the modern job market there is a high demand for “personal portfolio websites” , custom web pages meant to show potential employers your skills, history, and personality, in order to stand out with other possible hires. For many prominent and growing career fields, having a personal portfolio is either expected or a valuable tool to gain a leg up on the competition. However making your own from scratch requires familiarity with front end web design. Learning these tools can be difficult even for those familiar with programming languages, and for those without any programming experience it can feel like it's not worth the effort.

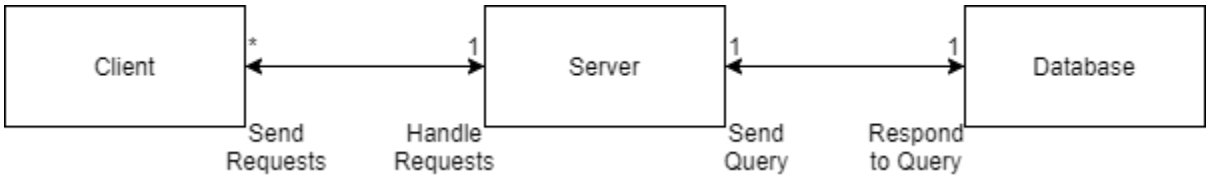
Dragon Drop is a website for creating web-based personal portfolios built on HTML and CSS. We provide novel portfolio-making using drag and drop components to simplify the process of making a portfolio. While sites like Wix and Squarespace can be used for making portfolios, their primary purpose is general web pages whereas Dragon Drop is focused on portfolio making and also totally free. Dragon Drop provides higher levels of customizability for your portfolio pages and additional tools to help make a portfolio that stands out to employers. Finally, for those who need a portfolio but aren’t looking to spend time crafting their own or feel their creativity is lacking, Dragon Drop allows you to pick from a variety of User templates to apply to your own website.

Design Outline

High Level Overview

Our software architecture pattern for our web application will lend from the client-server architecture. This program will have multiple connected clients who will send server requests to view projects, save projects, and to go into edit mode. The server sends requested projects to the client, writes projects to the database, and sends requested page editing tools. The database stores the project information and the login information.

High Level UML Diagram



1. Client
    a. Client will run the application interface through a web browser.
    b. Client sends requests for things the user wants
    c. Client sends data for changes the user has made on a project that affect database content.
2. Server
    a. Server acts as the “negotiator” between the client and the database.
    b. Server accepts login requests, existing projects requests, and edit requests.
    c. When the client makes a request, the server determines if the request is valid and then sends data from the database to the client if it is.
    d. Server will generate appropriate response to send to targeted client
3. Database

a. Database will store all of the login credentials and existing user projects.
b. Database will respond to server queries for matching login credentials and send a response back to the server to allow the user to log in or try again.
c. Database will respond to server queries for existing projects that a user has created
d. Database either hands server the requested information or an error if the request cannot be resolved.

Interaction Between Systems

Users enter the site as a client and are immediately sent to the home page unless they specifically request a different page. From here the user can either login or request a different page. When a user sends login information to the server, it is checked against the database to make sure that A: there is an account registered under the given email and B: the password is correct. If both these conditions are met then the server notifies the user they have logged in and the user now has access to additional user specific features.

Regardless of whether or not the user logs in, they can request additional pages such as public templates and Dragon Drop default pages (home page, about page, account creation, etc.). Page requests are handled on the server by simply passing along the page request to the database and then either serving the correct page or giving an error response if the page does not exist or is not accessible to the user.

Portfolio pages are editable and users can request permission to edit a page as long as they are the owner of the page. When edit access is requested the server first confirms with the database that the user owns the page, if this is true the server then sends the user an editable version of the page including edit tools. When users are content with the changes that are made,

they can hit the “save” button. Doing this will pass the client-side page info to the server which then uses it to replace the database contents for that page.

Design Issues

Functional Issues

- How should we handle the storage and transfer of passwords?
    - Option 1: Plaintext passwords
        - Unsecure
        - Easy to implement
    - Option 2: Manual encryption
        - Passwords are encrypted using MD5 by client script and then sent to server for decryption
        - More control over password encryption/decryption
    - Option 3: HTTPS Certificate
        - Most secure
        - Requires the most work to setup
        - Some older OS such as Windows XP, and Android pre 3.0 can't handle Server Name Identification.
    - Decision: Option 3 - HTTPS certification offers the most security out of our options, which is a top priority as we are dealing with sensitive information. This is why we believe it is worth it to work through the additional overhead it requires. Moreover, HTTPS is the standard for modern website design.
- Should incomplete projects (websites) be removed after a certain amount of time?
    - Option 1: Save indefinitely until the user reaches the max number of projects.
    - Option 2: After X amount of time since the project was last edited, the project will be removed.

    - Decision: Option 1 - Between the two options, there is a tradeoff between space occupied in the database versus user experience. We believed it would be detrimental to the user experience if a user were to come back to a website after a long period of time to find that it didn't exist anymore. However, since we likely would not have enough space in our database for users to maintain their projects indefinitely, we would need to implement a limit on how many projects a user can have.
- What information does the user need to input in order to make an account?
    - Option 1: Create account through Facebook/Google
    - Option 2: Input username, email, password
    - Option 3: Input email, password
    - Option 4: Input username, password
    - Decision: Option 1 and 3 - Users will have the convenient option of making an account through Facebook or Google like many other websites provide the option for. If the user doesn't want to do that, they can input an email and password. We decided usernames are not necessary since emails will already be unique and easier to remember for users.
- What requirements will users need to include for their passwords?
    - Option 1: No requirements.
    - Option 2: Length requirement.
    - Option 3: Require at least one uppercase and a number.
    - Option 4: Require special characters.

    - Decision: Options 2, 3, and 4 - We want the users to have secure enough passwords to ensure personal information is not stolen.

Non-Functional Issues

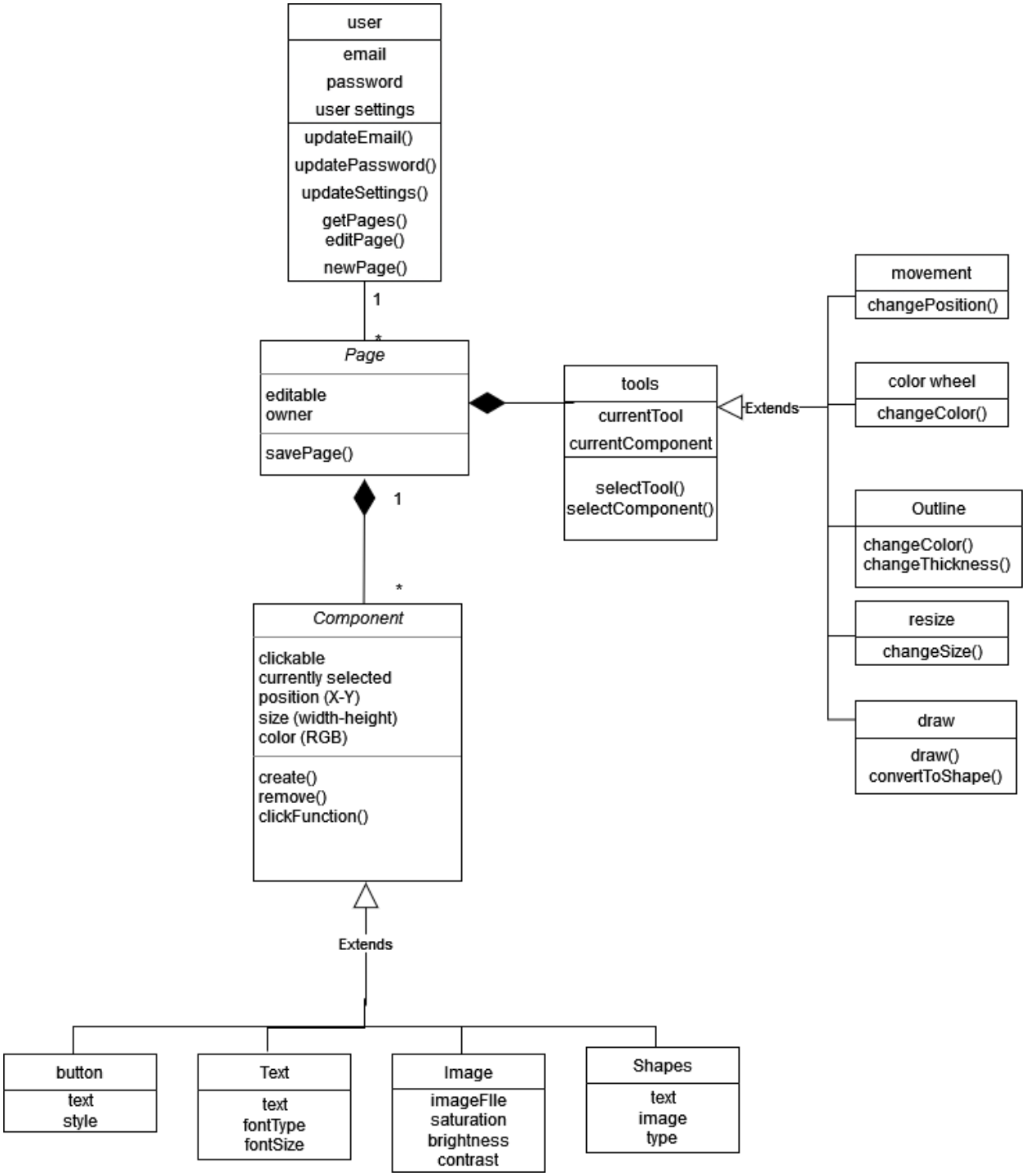
- Will we host websites that users make?
    - Option 1: We host.
    - Option 2: User downloads generated script and hosts themself or finds a third party.
    - Decision: Option 2 - If Dragon Drop were to become popular in the future, web hosting is a feature we might consider. However, the effort and resources required to host the users' websites is outside the scope of this project
- Which language/frameworks should we use to implement our front-end services?
    - Option 1: JavaScript (React)
    - Option 2: JavaScript/Typescript (Angular)
    - Decision: Option 1 - React is considered to have more flexibility and a shallower learning curve compared to Angular. This is beneficial for our team since we have limited front-end experience. Additionally, React is one of the most popular front end web development frameworks in the tech industry.
- Which language should we use to implement our back-end services?
    - Option 1: JavaScript (Node.js)
    - Option 2: SQL
    - Option 3: PHP
    - Decision: Option 1 - Since we are using javascript for our front end, implementing the backend in Node.js provides consistency throughout the project.

Additionally, Node.js works well for real-time applications, which is beneficial since we would like to implement an autosave feature.

- What database service should we use?
    - Option 1: MySQL
    - Option 2: MongoDB
    - Option 3: OrientDB
    - Decision: Option 1- Some of our group members have experience with SQL already, so there will be less time spent on learning how to use new services. MySQL is also relatively simple and fast, making it a good all-purpose database for our project.

Design Details

Class Design



Interactions Between Classes

A user class with a unique ID, SHA-256 encrypted password, and associated page classes.

Page classes have component classes and tool classes.

Component classes have a multitude of components, like text, media, shapes, and decorative elements.

Components should be able to be placed along guidelines (like when centering things in powerpoint). Components should be able to keep a certain ratio when resizing.

Tool classes include undo, redo, color wheel, movement, and more.

User:

- Has a unique ID, SHA-256 encrypted password, associated username and e-mail, first and last name, and associated pages.
- Users store a number of page classes or references to them in order to keep track of which ones they own.
- Users should have functionality to change their password, change their name, and edit / remove pages

Page:

- Contains a reference to which user owns the page
- Contains a variable describing if they're private or public
- Represents the user's personal portfolio page
- Page classes have component classes within them, as well as metadata that helps with placement or order of component classes.
- Users use this class to design their portfolio and add components

Component:

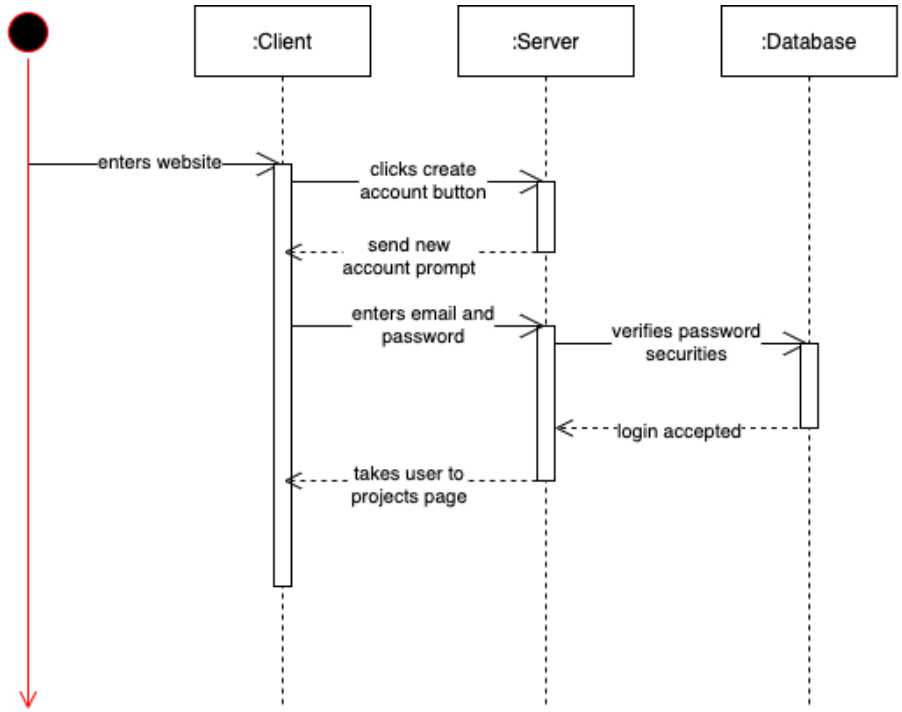
- Users use this class in order to add buttons, text, images, shapes, and more.
    - Buttons can have various functionalities:
        - Navigate to external website
        - Navigate to different page
        - Can be different states when loading, change upon mouse-over, etc.
    - Text can be multiple fonts and styles (italicized, bolded, underlined), as well as can be aligned center, right, or left.
    - Images can be uploaded from the user's files or searched online.
    - Shapes will be a number of preset shapes, including squares, triangles, stars, arrows, and more.
    - Other components could include more functional components, like dropdown menus, switches, icons, or other React components.
- This class has properties:
    - Clickable: This component can be attached to a hyperlink to any component.
    - Currently selected: determines whether the UI element to edit a component is on this current component.
    - Size information: determines the relative size of the component, i.e. dimensions in x and y.
    - Location information: determines the location on the page of the component with similar x and y coordinates.
    - Color: components can have different colors.

Tool:

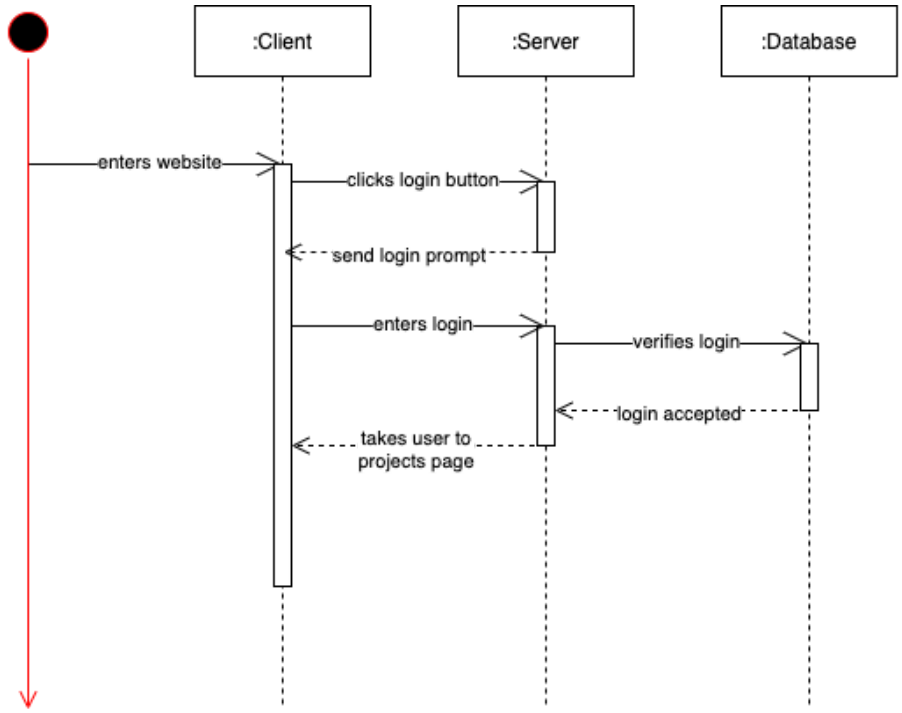
- Tools are used to edit components or the page overall.

- Users may use this class in order to implement movement, color, outlining, and resizing of components:
    - Movement:
        - Users may move components by arrow keys, dragging, or editing x- and y-coordinates
    - Color:
        - Users may choose a component's color from a color wheel
    - Outline:
        - Users may alter the thickness of a component's outline
    - Resize:
        - Users may alter the size of a component by inputting a value or dragging
    - Draw:
        - Users can create their own shapes by hand using the draw tool

Sequence Diagram (#1, #2)

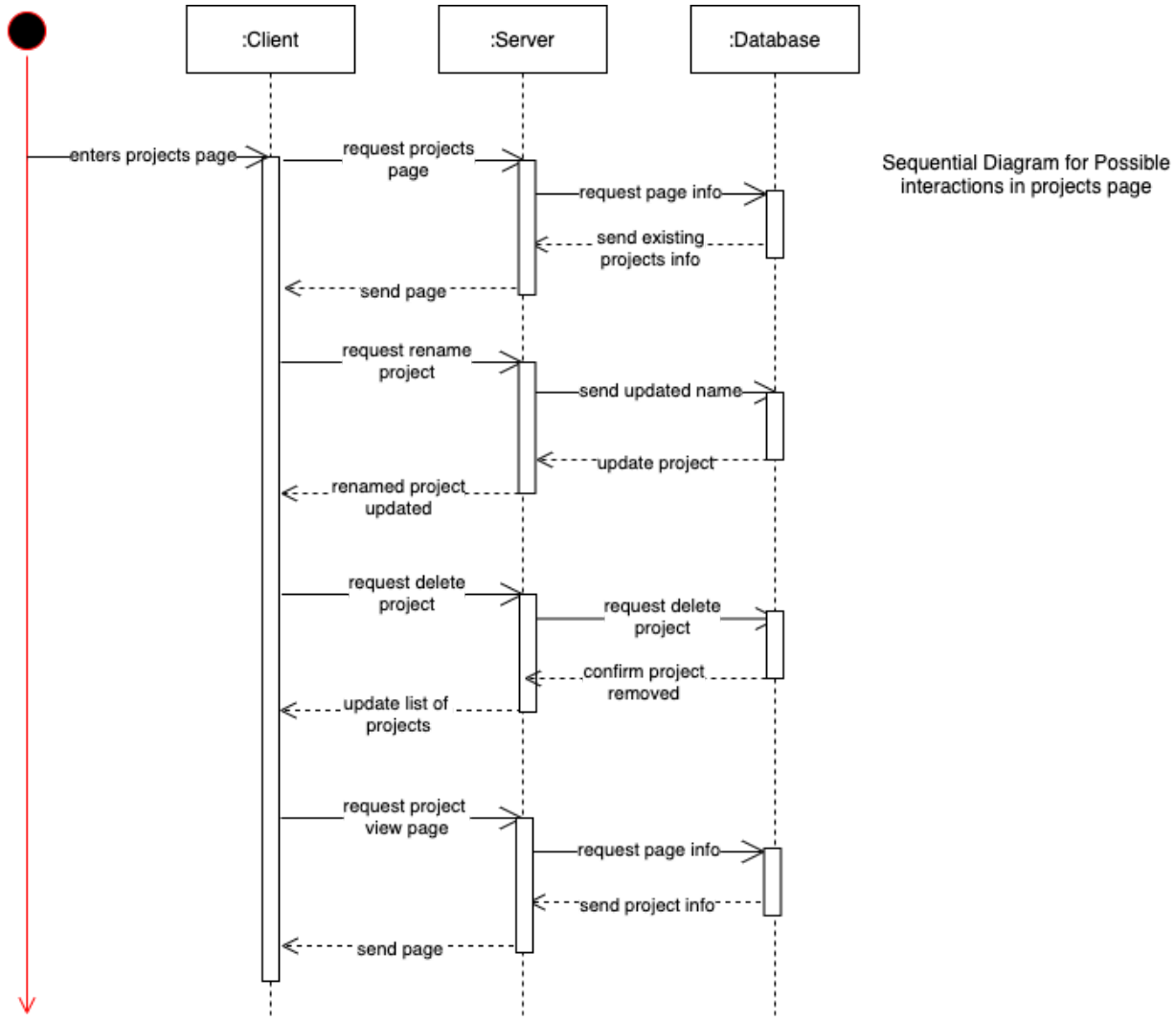


Sequential Diagram for New User Account Creation

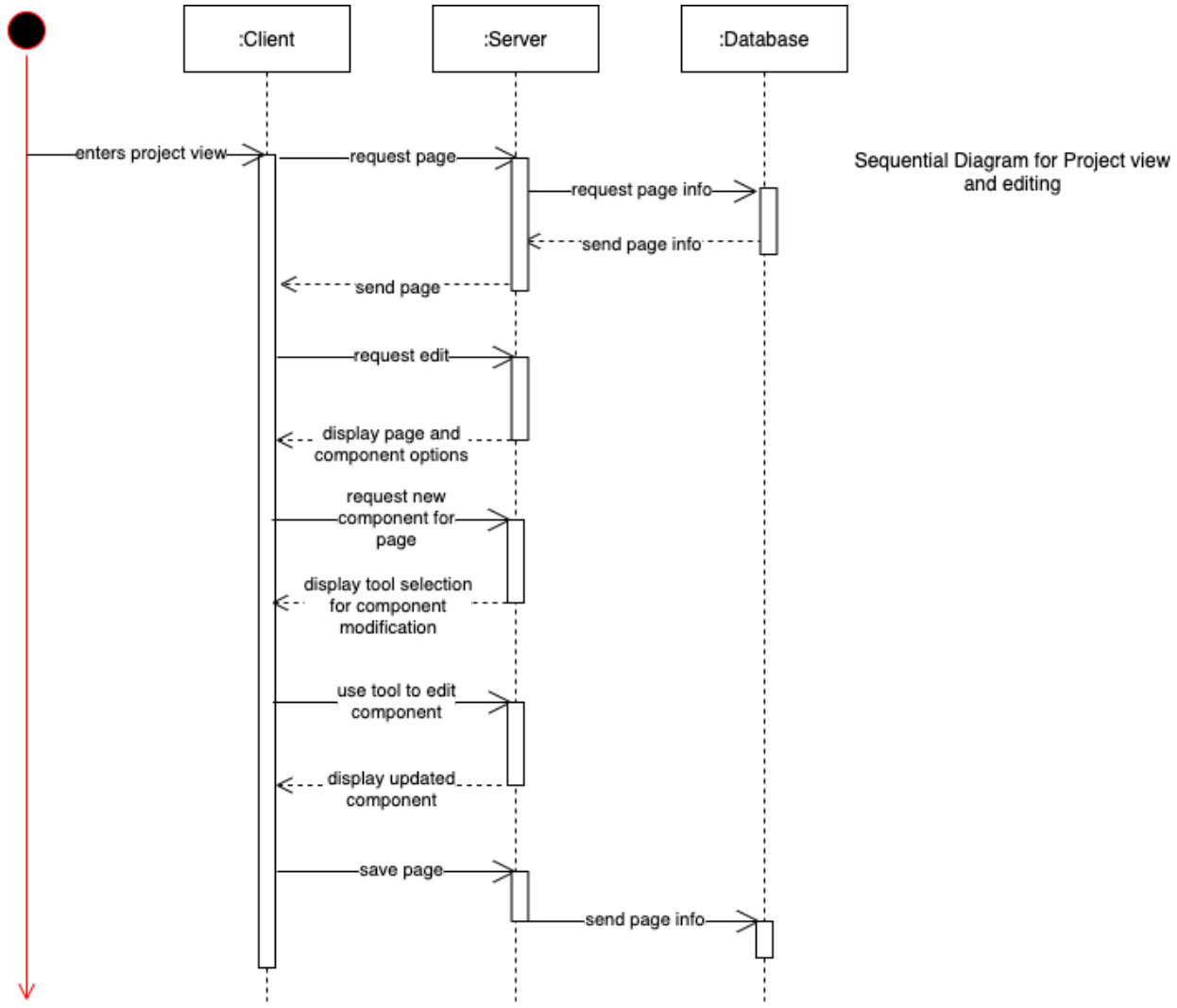


Sequential Diagram for User Login

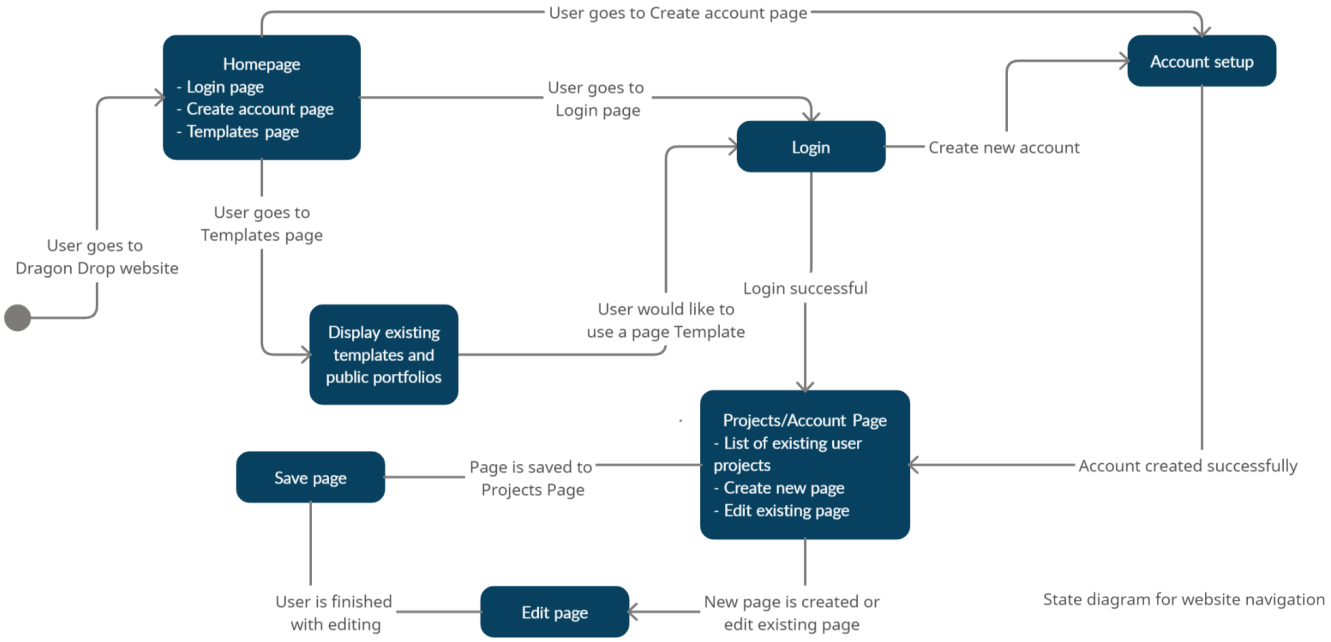
Sequence Diagram (#3)



Sequence Diagram (#4)

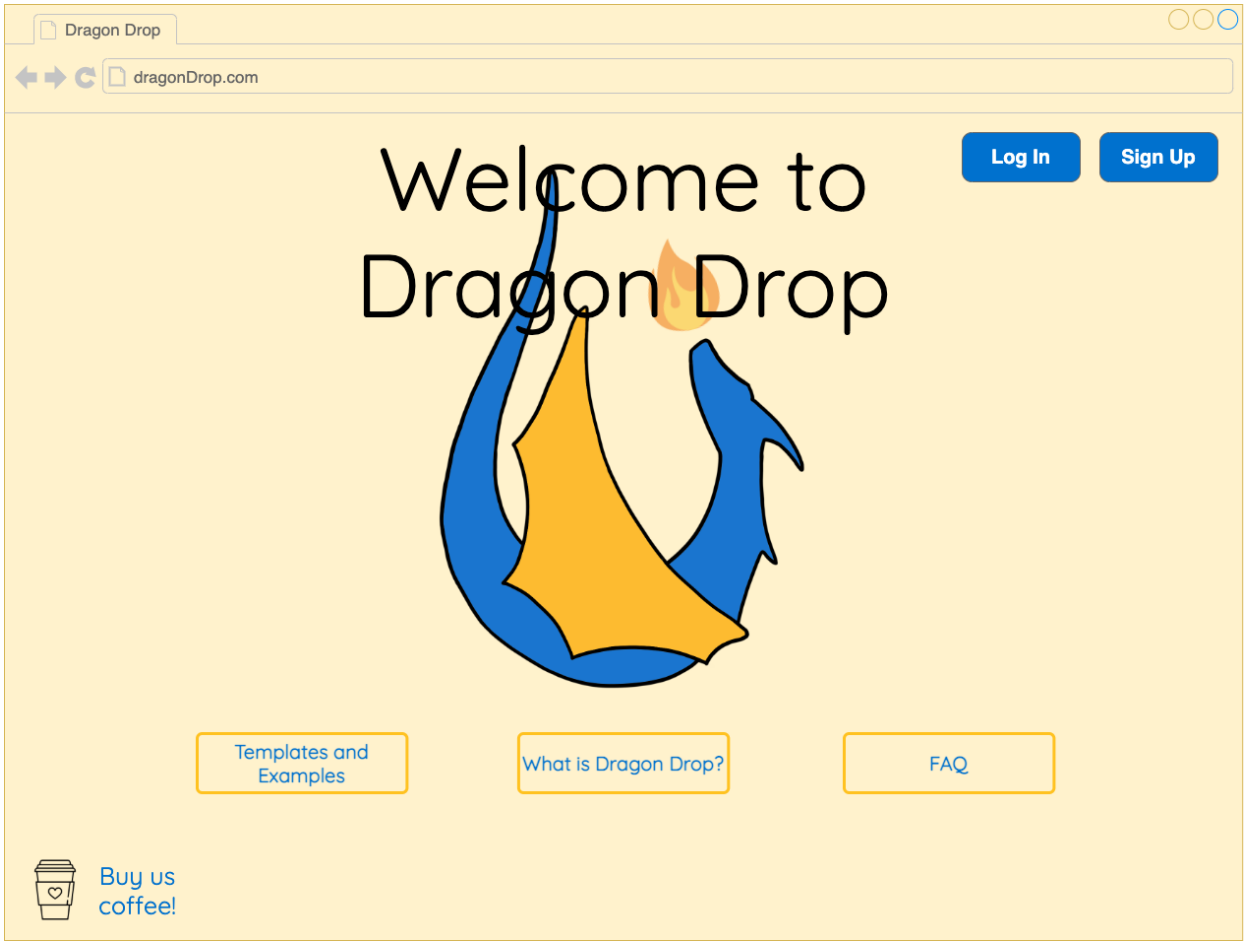


State Diagram (#1)



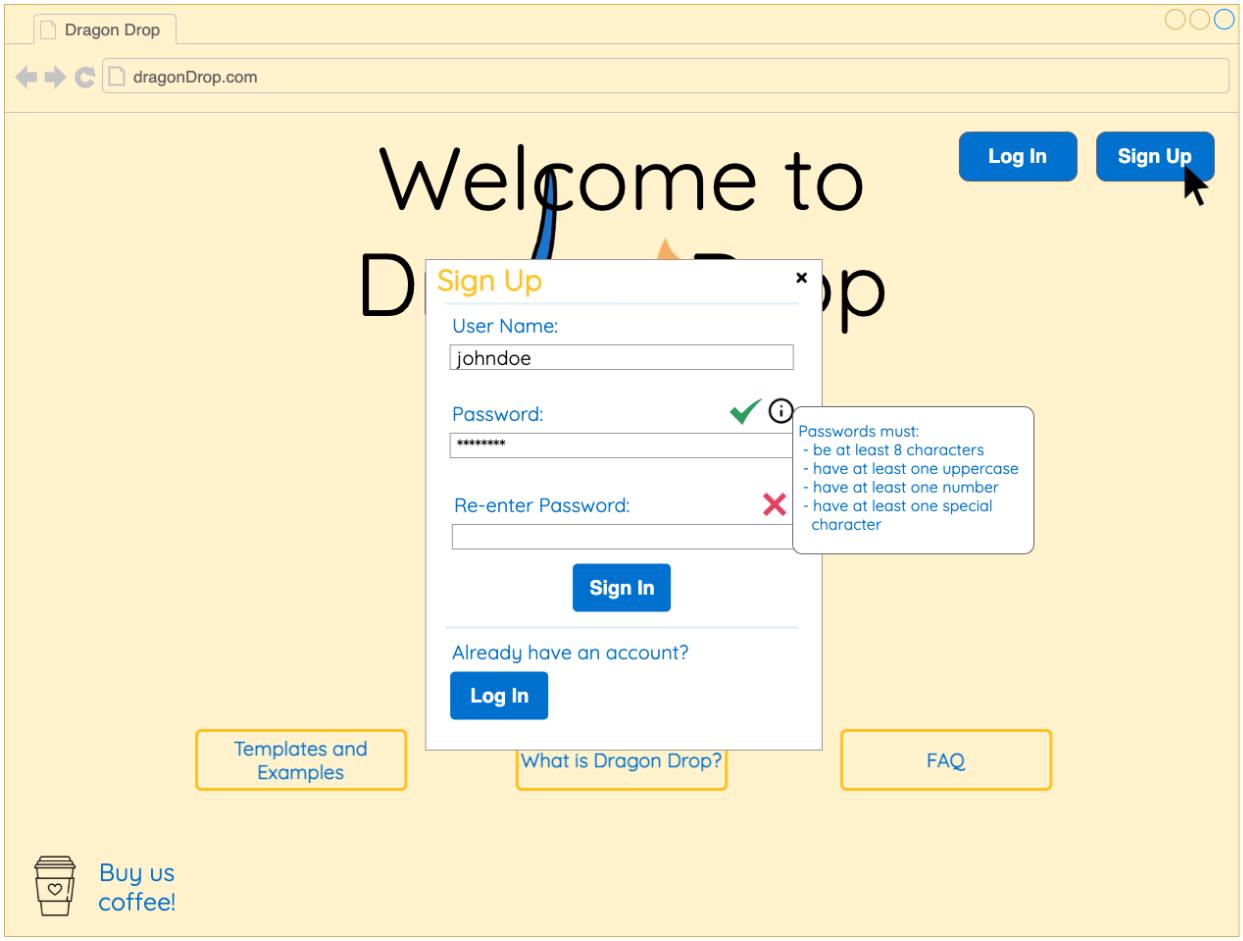
<u>UI Mockup</u> (#1)

Welcome Page



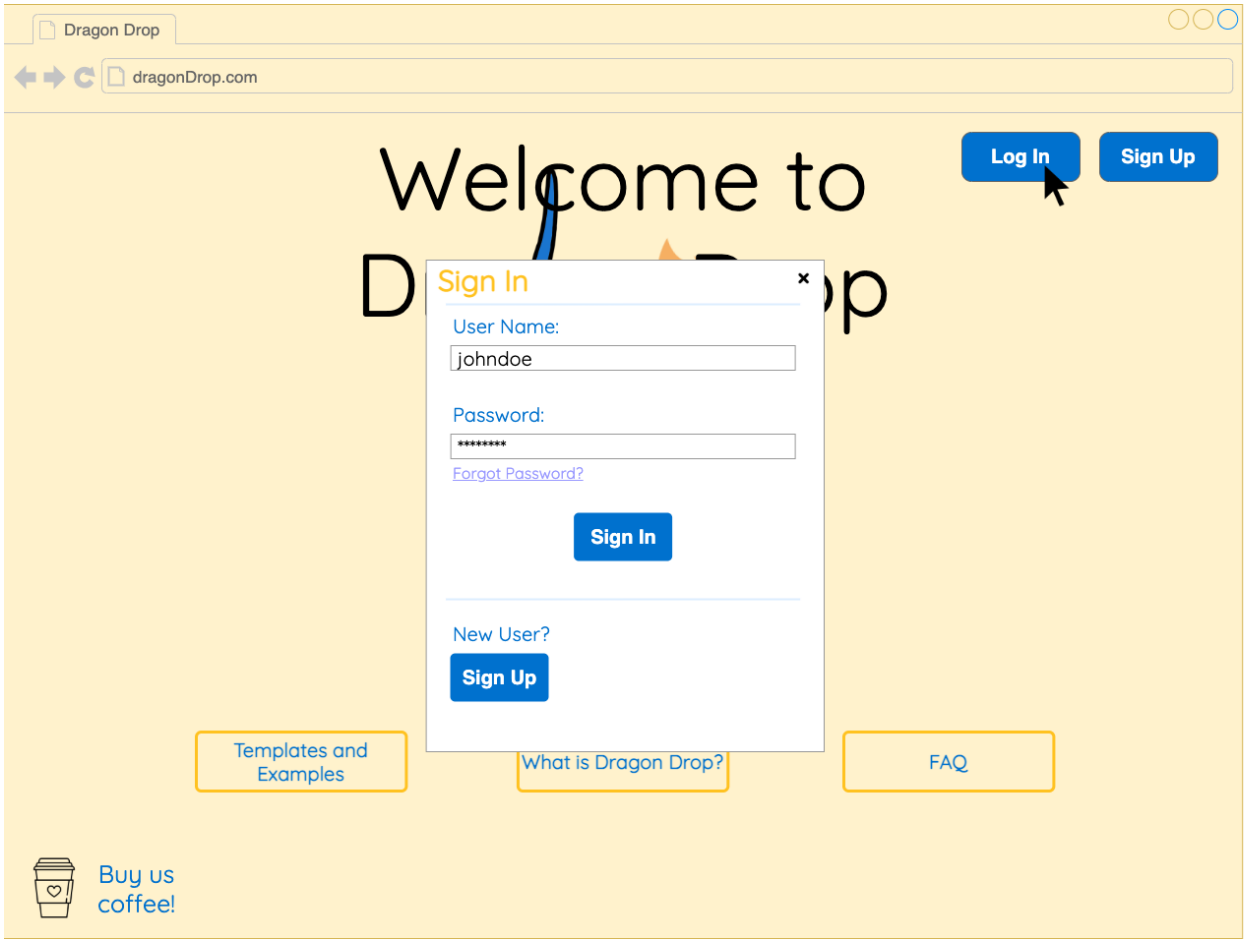
<u>UI Mockup</u> (#2)

Sign up



<u>UI Mockup</u> (#3)

Log in



<u>UI Mockup</u> (#4)

Projects page