Bronco Bulletin

The Team

James Vong: Backend and Hypebeast

George Orloff: Frontend and Storyteller

Joseph Yu: Scrum Servant and Resident Philosopher

Ethan Shenassa: Scrum Master, and Menial Laborer

Eric Hicks: Frontend and Cheerer-Upper

Cole Fettkether: Frontend and Meme Master

Initial Goals

- An all-purpose posting board for any SCU student to post advertisements about whatever they want
- Sign in with SSO password*
- Content moderation done by GPT and superusers with a reporting system
- Different filtering methods to sort content by relevant type, tags, etc.

User Stories

As a club creator - I would like to post my club to a centralized location so people can find our club without needing to work through paperwork presented by SCU

As a tutor - I would like to advertise my hours so people can seek out my services and keep up with my current tutoring hours

As a student - I would like to find new clubs, activities, events, and tutors in one place so I can get more involved in extracurriculars

Bronco Bulletin

Search

J

General

- Events
- Clubs
- Activities
- Tutors
- Misc.

Create Post

ACM Competitive Programming

Join ACM Competitive Programming and ACM-G for our first ever on-campus coding competition on Friday, March 1st!

SCU Real Estate Association

The Santa Clara University Real Estate Association was established in 2016 and serves as the premier Real Estate...

Black Law Student Association

Our organization works with SCU law administration to encourage/ include cultural competency within the curriculum...

Tau Beta Pi

Beta Pi California Zeta is Santa Clara University's Chapter of the nationally renowned Engineering Honor Society Tau Beta Pi.

Women in STEM

Our goal is to cultivate an uplifting community where female undergraduate students majoring in STEM fields can...

Delta Epsilon Mu

Delta Epsilon Mu (DEM) is the nation's premier Professional Pre-Health Co-Ed Fraternity.

SCU Fight Club

We don't talk about Fight Club.

Join us if you like hand-to-hand combat. We host session every Tuesday and Thursday at 5 PM.

Blueprints for Pangaea

We are a 501(c)(3) nonprofit organization dedicated to addressing global healthcare inequalities by reallocating excess unused medical supplies...

Bronco Bulletin

Search

J

Post Title

Description

Tags

Upload Image

Submit Post

Bronco Bulletin

Email Address

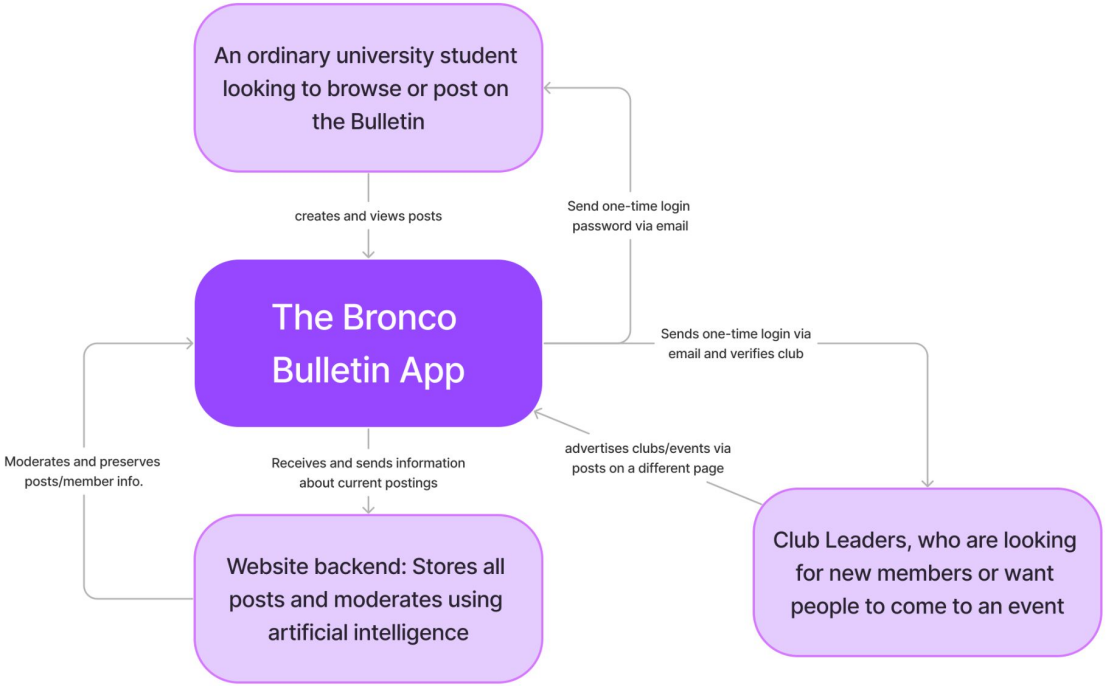
bbronco@scu.edu

Get Password

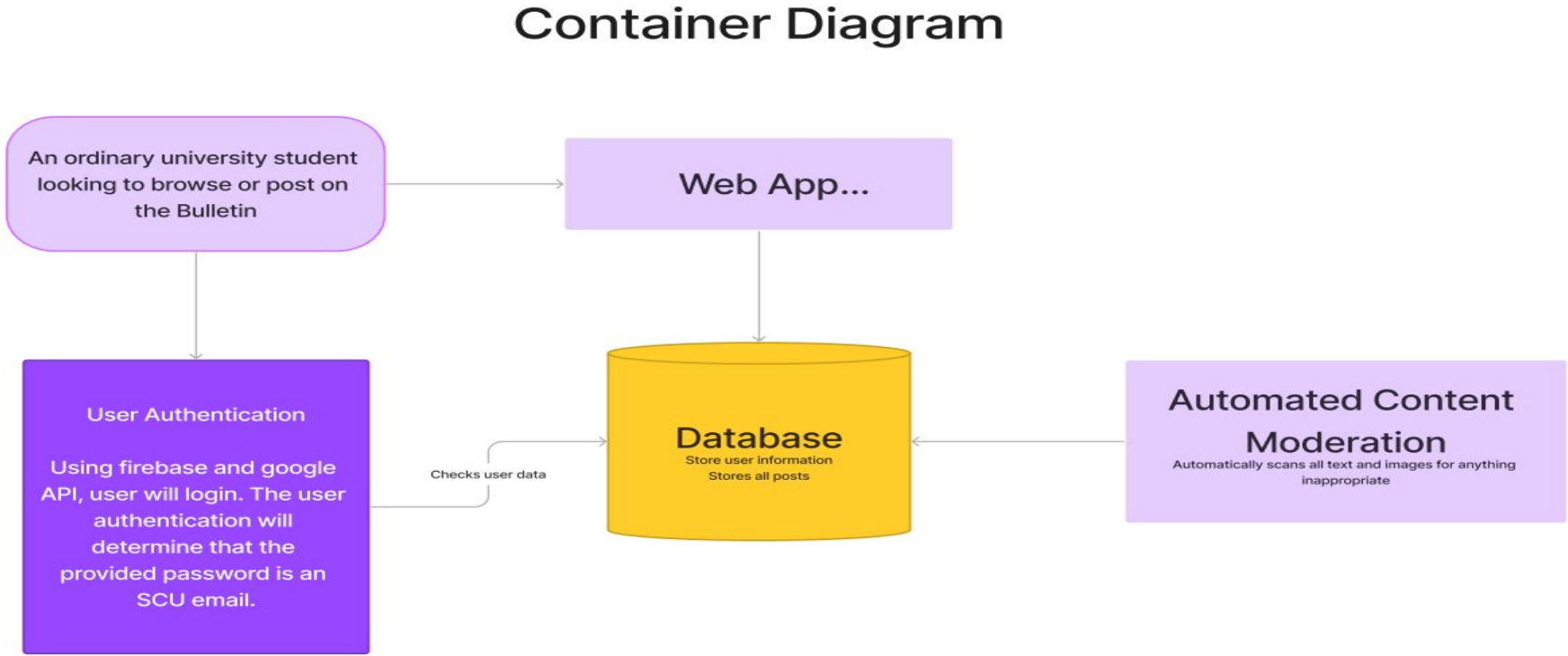
Minimum Viable Product

- A website that allows users to sign in, post ads, and view others posted ads
- 3 pages: sign in, create post, and explore

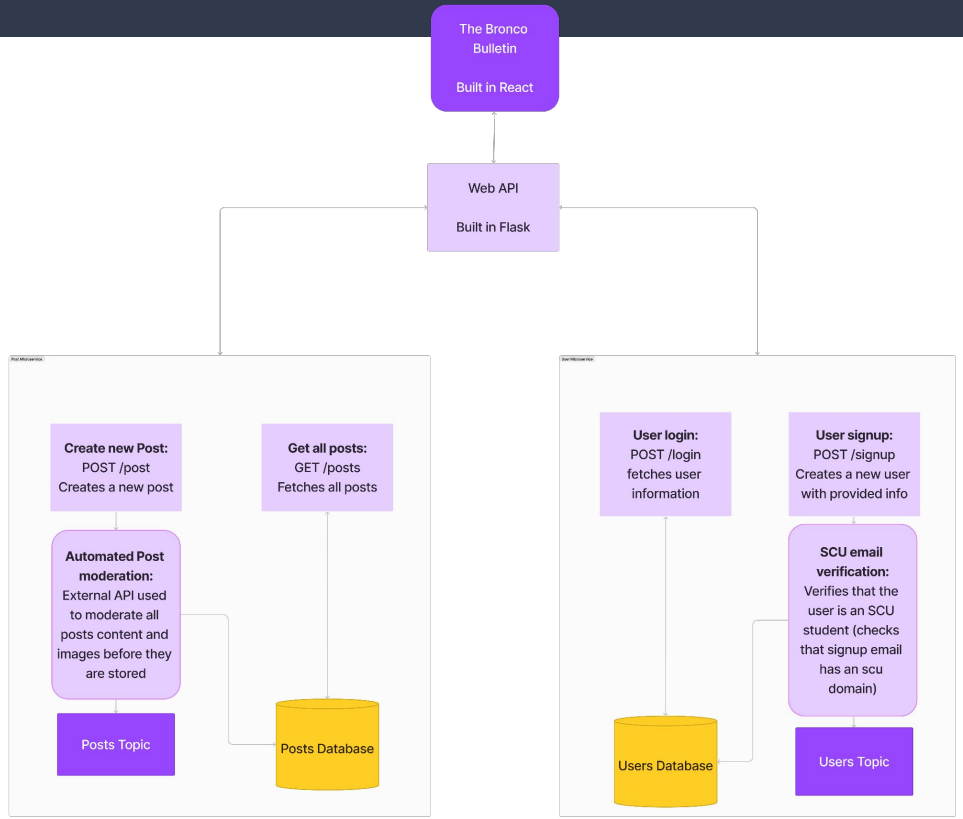
System Context Diagram



Initial Container Diagram



Updated Container Diagram



Tools

Frontend

- React
- NodeJS

Backend

- Python, Flask, SQLAlchemy
- PostgreSQL (Hosted on Neon)
- Render (Host)

Live Demo

Incomplete and Future Work

- Superusers
- GPT moderation
- Sorting system

Successes

- **Organization** - We were able to delegate roles in a way that made it clear what everyone's job on the project was.
- **Development -** We were able to create most of the features for our MVP within the timeframe.
- **Hosting** - We were able to host the backend allowing us to have a universal database for development.

```
import sys
import os
from dotenv import load_dotenv
from flask import Flask, jsonify, request
from flask_cors import CORS, cross_origin

# Adjust the PYTHONPATH to include the parent directory of Backend
sys.path.append(os.path.dirname(os.path.abspath(__file__)))

from modules.database.dbHandler import PGManager
from modules.database.post import Tag

load_dotenv()  # Load environment variables from .env file

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
manager = PGManager("csen174_owner", os.getenv("PGPASSWORD"))

@cross_origin
@app.route('/', methods=['GET'])
def index():
    return "Hello, world!"

# Route to get all posts
@cross_origin
@app.route('/posts', methods=['GET'])
def get_all_posts():
    try:
        posts = manager.get_posts()
        posts_list = [{
            'post_id': post.post_id,
            'author': post.author,
            'title': post.title,
            'description': post.description,
            'current_time': post.current_time,
            'tags': [tag.name for tag in post.tags],
            'image': post.image
        } for post in posts]
        return jsonify({'posts': posts_list})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

# Route to add a new post
@cross_origin
@app.route('/posts', methods=['POST'])
def add_post():
```

```
    try:
        data = request.get_json()
        author = data.get('author')
        title = data.get('title')
        description = data.get('description')
        tag_names = data.get('tags', [])
        image = data.get('image')

        if not author or not title or not description:
            return jsonify({'error': 'Missing required fields'}), 400

        post = manager.insert_post(author, title, description, tag_names,
image)
        return jsonify({
            'post_id': post.post_id,
            'author': post.author,
            'title': post.title,
            'description': post.description,
            'current_time': post.current_time,
            'tags': [tag.name for tag in post.tags],
            'image': post.image
        }), 201
    except ValueError as ve:
        return jsonify({'error': str(ve)}), 400
    except Exception as e:
        return jsonify({'error': str(e)}), 500

# Route to get posts by a specific tag
@cross_origin
@app.route('/posts/tag/<string:tag_name>', methods=['GET'])
def get_posts_by_tag(tag_name):
    try:
        session = manager.Session()
        tag = session.query(Tag).filter_by(name=tag_name).first()
        if not tag:
            return jsonify({'error': 'Tag not found'}), 404

        posts = tag.posts
        posts_list = [{
            'post_id': post.post_id,
            'author': post.author,
            'title': post.title,
            'description': post.description,
            'current_time': post.current_time,
            'tags': [tag.name for tag in post.tags]
        } for post in posts]
        session.close()
        return jsonify({'posts': posts_list})
    except Exception as e:
```

```
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port, debug=True)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <!-- CSS only -->
            <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
    rel="stylesheet"

integrity="sha384-iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+
fzT"
    crossorigin="anonymous"
    />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
```

```
href="https://fonts.googleapis.com/css2?family=Heebo:wght@100;200;300;400;500;6
00;700;800;900&family=Noto+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0
,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=
swap"
      rel="stylesheet"
    />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
```

```
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

```
import React from "react";
import { Link } from "react-router-dom";
import "./styles/header.css";

const Header = () => {
      return (
            <header>
      <nav className="navbar shadow-sm p-3">
        <Link className="logo navbar-brand navbar mx-2" to="/">
          Bronco Bulletin
        </Link>
      </nav>
            </header>
      );
};

export default Header;
```

```
.App {
      height: 100%;
      width: 100%;
      display: flex;
      flex-direction: column;
}

body {
      font-family: "Heebo", sans-serif;
}

.text-white {
      color: var(--white-color) !important;
}

#page {
  height: calc( 100vh - 72px );
  display: flex;
}
```

```
import { Route, createRoutesFromElements, createBrowserRouter, RouterProvider }
from "react-router-dom";
import { Home } from './pages/Home';
import { Create } from './pages/Create';
import './App.css';

function App() {
  const RoutesJSX = (
    <Route path="/">
      <Route index element={<Home />} />
      <Route path="create" element={<Create />} />
    </Route>
  );

  const routes = createRoutesFromElements(RoutesJSX);
  const router = createBrowserRouter(routes);

  return (
    <div className="App">
      <RouterProvider router={router}/>
    </div>
  );
}

export default App;
```

```
*,
*::after,
*::before {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
}

#root,
html,
body {
      position: absolute;
      height: 100%;
      width: 100%;
      margin: 0;

      /* Tints */
      --tint-1: 0.9;
      --tint-2: 0.7;
      --tint-3: 0.5;
```

```
      --tint-4: 0.1;

      /* Colors */
      --white-color: #FFFFFF;
      --white-color-rgb: 255, 255, 255;
      --white-color-1: rgba(var(--white-color-rgb), var(--tint-1));
      --white-color-2: rgba(var(--white-color-rgb), var(--tint-2));
      --white-color-3: rgba(var(--white-color-rgb), var(--tint-3));
      --white-color-4: rgba(var(--white-color-rgb), var(--tint-4));

      --light-grey-color: #D9D9D9;
      --light-grey-color-rgb: 217, 217, 217;
      --light-grey-color-1: rgba(var(--light-grey-color-rgb), var(--tint-1));
      --light-grey-color-2: rgba(var(--light-grey-color-rgb), var(--tint-2));
      --light-grey-color-3: rgba(var(--light-grey-color-rgb), var(--tint-3));
      --light-grey-color-4: rgba(var(--light-grey-color-rgb), var(--tint-4));

  --grey-color: #C8C8C8;
  --grey-color-rgb: 52, 52, 52;
  --grey-color-1: rgba(var(--grey-color-rgb), var(--tint-1));
      --grey-color-2: rgba(var(--grey-color-rgb), var(--tint-2));
      --grey-color-3: rgba(var(--grey-color-rgb), var(--tint-3));
      --grey-color-4: rgba(var(--grey-color-rgb), var(--tint-4));

      --dark-grey-color: #484747;
      --dark-grey-color-rgb: 72, 71, 71;
      --dark-grey-color-1: rgba(var(--dark-grey-color-rgb), var(--tint-1));
      --dark-grey-color-2: rgba(var(--dark-grey-color-rgb), var(--tint-2));
      --dark-grey-color-3: rgba(var(--dark-grey-color-rgb), var(--tint-3));
      --dark-grey-color-4: rgba(var(--dark-grey-color-rgb), var(--tint-4));

      --black-color: #030406;
      --black-color-rgb: 3, 4, 6;
      --black-color-1: rgba(var(--black-color-rgb), var(--tint-1));
      --black-color-2: rgba(var(--black-color-rgb), var(--tint-2));
      --black-color-3: rgba(var(--black-color-rgb), var(--tint-3));
      --black-color-4: rgba(var(--black-color-rgb), var(--tint-4));

  --red-color: #9D1534;
  --red-color-rgb: 157, 21, 52;
  --red-color-1: rgba(var(--red-color-rgb), var(--tint-1));
      --red-color-2: rgba(var(--red-color-rgb), var(--tint-2));
      --red-color-3: rgba(var(--red-color-rgb), var(--tint-3));
      --red-color-4: rgba(var(--red-color-rgb), var(--tint-4));
}

#root {
      display: flex;
}
```

```
body {
      background-color: var(--light-grey-color);
}

.text-md {
  font-size: 1.2em;
}
```

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) =>
{
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```
