



CST-323 Design Report Template

Topic:	Milestone 4																																																											
Date:	12/12/2024																																																											
Revision:	4.0																																																											
Team:	1. Kaya Nelson																																																											
	2. Tyson Martin																																																											
	3.																																																											
	4.																																																											
	5.																																																											
	6.																																																											
Weekly Team Status Summary:	<table><thead><tr><th>User Story</th><th>Team Member</th><th>Hours Worked</th><th>Hours Remaining</th></tr></thead><tbody><tr><td><i>As a developer, I prefer utilizing sprints, a burndown chart, and an organized repository for effective project management.</i></td><td><i>Kaya</i></td><td><i>1</i></td><td><i>0</i></td></tr><tr><td><i>As a developer, I would appreciate having a centralized dashboard to monitor application alerts and errors.</i></td><td><i>Tyson</i></td><td><i>2</i></td><td><i>0</i></td></tr><tr><td><i>As a developer, I'm interested in implementing features that enhance my application's functionality.</i></td><td><i>Kaya</i></td><td><i>1</i></td><td><i>0</i></td></tr><tr><td><i>As a developer, I would like a comprehensive video overview that showcases the code, web design, and features to effectively present our current progress.</i></td><td><i>Kaya</i></td><td><i>1</i></td><td><i>0</i></td></tr><tr><td><i>Incorporating a documentation framework like JavaDoc, PhpDoc, nDoc, or JSDoc is essential for maintaining clear and organized project documentation.</i></td><td><i>Kaya/ Tyson</i></td><td><i>2</i></td><td><i>0</i></td></tr><tr><td><i>As a user, I would like to access this application via the public cloud.</i></td><td><i>Tyson</i></td><td><i>3</i></td><td><i>0</i></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></tbody></table>				User Story	Team Member	Hours Worked	Hours Remaining	<i>As a developer, I prefer utilizing sprints, a burndown chart, and an organized repository for effective project management.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>	<i>As a developer, I would appreciate having a centralized dashboard to monitor application alerts and errors.</i>	<i>Tyson</i>	<i>2</i>	<i>0</i>	<i>As a developer, I'm interested in implementing features that enhance my application's functionality.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>	<i>As a developer, I would like a comprehensive video overview that showcases the code, web design, and features to effectively present our current progress.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>	<i>Incorporating a documentation framework like JavaDoc, PhpDoc, nDoc, or JSDoc is essential for maintaining clear and organized project documentation.</i>	<i>Kaya/ Tyson</i>	<i>2</i>	<i>0</i>	<i>As a user, I would like to access this application via the public cloud.</i>	<i>Tyson</i>	<i>3</i>	<i>0</i>																												
User Story	Team Member	Hours Worked	Hours Remaining																																																									
<i>As a developer, I prefer utilizing sprints, a burndown chart, and an organized repository for effective project management.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>																																																									
<i>As a developer, I would appreciate having a centralized dashboard to monitor application alerts and errors.</i>	<i>Tyson</i>	<i>2</i>	<i>0</i>																																																									
<i>As a developer, I'm interested in implementing features that enhance my application's functionality.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>																																																									
<i>As a developer, I would like a comprehensive video overview that showcases the code, web design, and features to effectively present our current progress.</i>	<i>Kaya</i>	<i>1</i>	<i>0</i>																																																									
<i>Incorporating a documentation framework like JavaDoc, PhpDoc, nDoc, or JSDoc is essential for maintaining clear and organized project documentation.</i>	<i>Kaya/ Tyson</i>	<i>2</i>	<i>0</i>																																																									
<i>As a user, I would like to access this application via the public cloud.</i>	<i>Tyson</i>	<i>3</i>	<i>0</i>																																																									



GIT URL:		https://github.com/Kdeshun/CST323-Milestone			
Peer Review:		Y/N	We acknowledge that our team has reviewed this report and we agree to the approach we are all taking.		

Deployment URL: <http://cst323clc-env.eba-mm2xi3h5.us-east-1.elasticbeanstalk.com/>

Alternative Deployment URL (broken, in progress): <https://boiling-meadow-41866.herokuapp.com/>

Presentation URL: <https://go.screenpal.com/watch/cZloiVnnGPI>

To generate documentation for your React components using JSDoc, simply run npm run docs in your terminal.

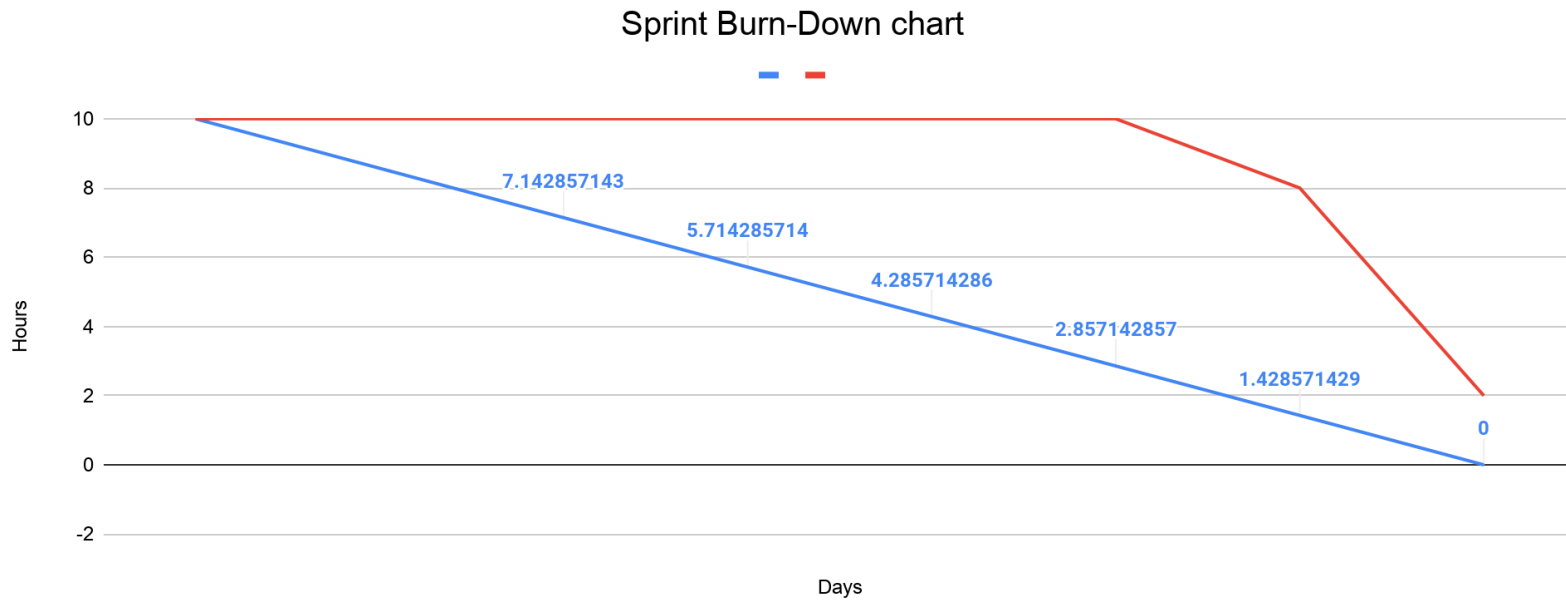
Planning Documentation

Agile Scrum Product Backlog:

Agile Scrum Sprint Backlog:

Agile Scrum Burn Down Chart:

This needs to contain a URL to Bitbucket Scrum Burn Down Chart Artifact.



Agile Retrospective Results:



The following table should be completed after each Retrospective on Things That Went Well (Keep Doing). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool, you must include a URL or Image File.

What Went Well
Having a centralized communication method facilitated the sharing of team standards, guiding the group on how to complete and submit their work effectively.
The initial setup with React, Node, npm, and the base project files went smoothly without any issues.

The following table should be completed after each Retrospective on Things That Didn't Go Well (Stop Doing) and What Would Be Done Differently Next Time with an Action Plan to Improve (Try Doing and Continuous Improvement). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool, you must include a URL or Image File.

What Did Not Go Well	Action Plan	Due Date
This week has been challenging, particularly as we reach the fourth milestone with only half of our CLC team participating. While we managed to achieve some progress, it came at the cost of Tyson's and my personal activity homework.	Focus on eliminating unnecessary features and complexities from the application to ensure we meet the final deadline.	12/15

Design Documentation

Install Instructions:

Include step-by-step instructions for setting up your database, configuring, and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from BitBucket and deploy the application to an externally hosted site.

Setting Up the Application

1. **Clone the Repository:** Start by cloning the repository to a local machine.
2. **Navigate to the App Directory:** Change directories to the `/app` folder.
3. **Install Dependencies:**
 - o Open a terminal and run:

```
bash
```

```
npm install
```

4. **Install Client Dependencies:** Change to the `/client` directory and execute:

```
bash
```

```
npm install
```

or simply:

```
bash
```

```
npm i
```

5. **Run the Development Server:** Return to the `/app` directory and start the development server with:

```
bash
```



```
npm run start
```

This command will run the server defined in `server.js` and initiate the React script process in the `/client` directory.

Building the Application for Deployment

1. **Change to Client Directory:** Navigate to the `/client` directory.
2. **Build the Application:** Run the following command and wait for the build process to complete:

```
bash
```

```
npm run build
```

3. **Deploying the Application:** If deploying as a Node.js application on Heroku or another platform, the service's build system will install dependencies at the root `/app` level and start the application server with:

```
bash
```

```
npm run start
```

Installing JSDoc with Better-Docs Plugin

1. **Navigate to App Directory:** Open your terminal and go to the `/app` directory.
2. **Install JSDoc:** Run the following command:

```
bash
```

```
npm i jsdoc
```

3. **Install Better-Docs:** In the same `/app` directory, execute:

```
bash
```

```
npm i better-docs
```



GRAND CANYON
UNIVERSITY™

General Technical Approach:

In your own words describe your approach and design here. You should also summarize any meeting notes, brain storming sessions, and so forth that you want to retain thru the design of your project.

Our team's approach is to develop a blogging platform accessible on the open web. We will collaborate using a variety of tools to enhance our teamwork:

- **Discord:** This will be our main communication channel to stay connected throughout the week and coordinate on projects.
- **GitHub:** We will utilize GitHub for source code management and version control, ensuring effective tracking of changes.
- **Google Docs:** For managing sprint and burndown charts, as well as updating design reports, we'll rely on Google Docs.
- **Draw.io:** We will use Draw.io for wireframing and creating UML diagrams to visualize our design.
- **MySQL Workbench:** ER diagrams will be generated using MySQL Workbench to support our database design.

This collaborative approach will help us streamline our development process and keep everyone aligned on project goals.

Key Technical Design Decisions:

Any final technical design decisions, (e.g., framework decisions) should be documented here. List the technology/framework, its purpose in the design, and why it was chosen.

Non-Code/Team Tools

Our team will develop a basic blogging platform using the following tools:

- **Discord:** This will serve as our primary communication tool.
- **Google Docs:** We will utilize this for documentation purposes.
- **GitHub:** This platform will handle version control for our project.
- **Visual Studio Code:** We will use this as our main code editor, although any suitable editor can be substituted as needed.
- **Loom:** This tool will be used for creating video presentations.

Framework/Language

Our application will be built using a pure JavaScript stack, which includes:

- **React:** A frontend development framework that facilitates building user interfaces.
- **Node.js:** A backend server-side development framework that enables JavaScript to run on the server.
- **MySQL:** A web-oriented database engine employed by high-profile companies like Uber.

This stack will provide a solid foundation for our blogging platform, ensuring efficiency and scalability.

Libraries/Packages:

Certainly! Here's a brief overview of each package and library you mentioned, highlighting their functionality and integration within a React/Node.js stack:

1. MySQL

- **URL:** [MySQL NPM Package](#)
- **Description:** A Node.js driver for MySQL, allowing you to connect to and interact with MySQL databases. It supports connection pooling, error handling, and query execution.

2. EditorJS

- **URL:** [EditorJS](#)
- **Description:** A block-style editor that outputs clean JSON data. It allows you to create rich content using customizable blocks (like paragraphs, headers, images) and is easy to integrate into web applications.

3. Axios

- **URL:** [Axios NPM Package](#)
- **Description:** A promise-based HTTP client for the browser and Node.js. Axios simplifies making HTTP requests and handling responses, making it ideal for interacting with APIs.

4. UUID

- **URL:** [UUID NPM Package](#)
- **Description:** A library for generating unique identifiers (UUIDs). Useful for creating unique keys for database entries or other purposes where unique identification is needed.

5. React Bootstrap

- **URL:** [React Bootstrap](#)
- **Description:** A popular front-end framework rebuilt for React. It offers a wide range of components that are accessible by default and styled with Bootstrap, enabling you to build responsive, mobile-first applications.

6. JSDocs

- **Description:** While not linked to a specific URL, JSDoc is a documentation generator for JavaScript code. It helps in creating clear documentation for your codebase, enhancing maintainability and collaboration.

Integration Suggestions

- **MySQL:** Use it in your Node.js backend to handle database operations.
- **EditorJS:** Integrate it in your React front end to allow users to create and edit content.
- **Axios:** Use it to make API calls from your React components to your Node.js server or other APIs.
- **UUID:** Utilize it for generating unique keys for database entries or components.
- **React Bootstrap:** Leverage its components to build a responsive UI quickly.
- **JSDocs:** Document your code to maintain clarity and ease of use for future developers.

These tools together create a powerful stack for developing modern web applications with rich features and a great user experience.

Known Issues:

Any anomalies or known issues in the code or functionality should be documented here.



Known Issues

1. Redirection After Registration

- **Issue:** After registering a new user, the user is not redirected to the login page.
- **Proposed Solution:** Implement a redirect function in the registration logic. After successful registration, use a routing method (like `history.push` in React Router) to navigate to the login page.
- **Next Steps:** This will be prioritized for the next milestone.

2. Styling of the “Write/Edit Posts” Button

- **Issue:** The button for writing/editing posts does not match the aesthetic of the change password form.
- **Proposed Solution:** Review the CSS styles applied to the button and the form. You may want to adjust padding, margins, background color, font size, or border radius to create a more cohesive design.
- **Next Steps:** Update the styles in the relevant CSS or styling component files. Consider using consistent design tokens or classes to ensure uniformity across the page.

Action Items

- **For Registration Redirect:**
 - Update the registration handler to include a redirect to the login page after successful registration.
- **For Button Styling:**
 - Review and modify the CSS styles for the “Write/Edit Posts” button to ensure it aligns with the design of the change password form.

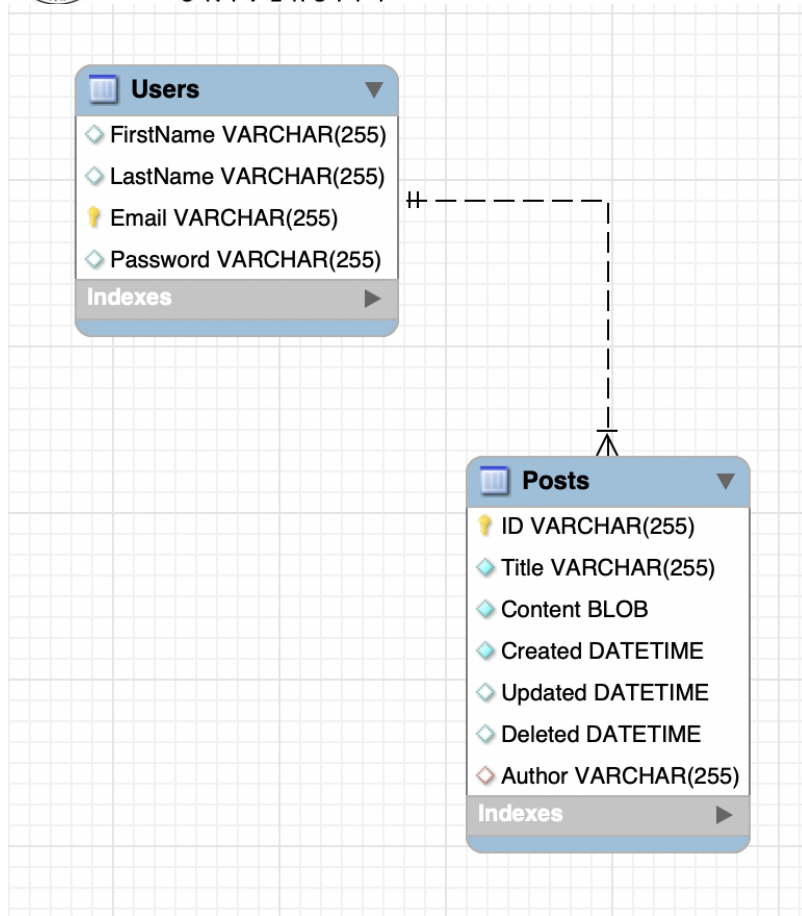
Risks:

Any risks, unknowns, or general project elements that should be tracked for risk management should be documented here.

No known Risks at this time.

ER Diagram:

Below is an image of the current database ER diagram.

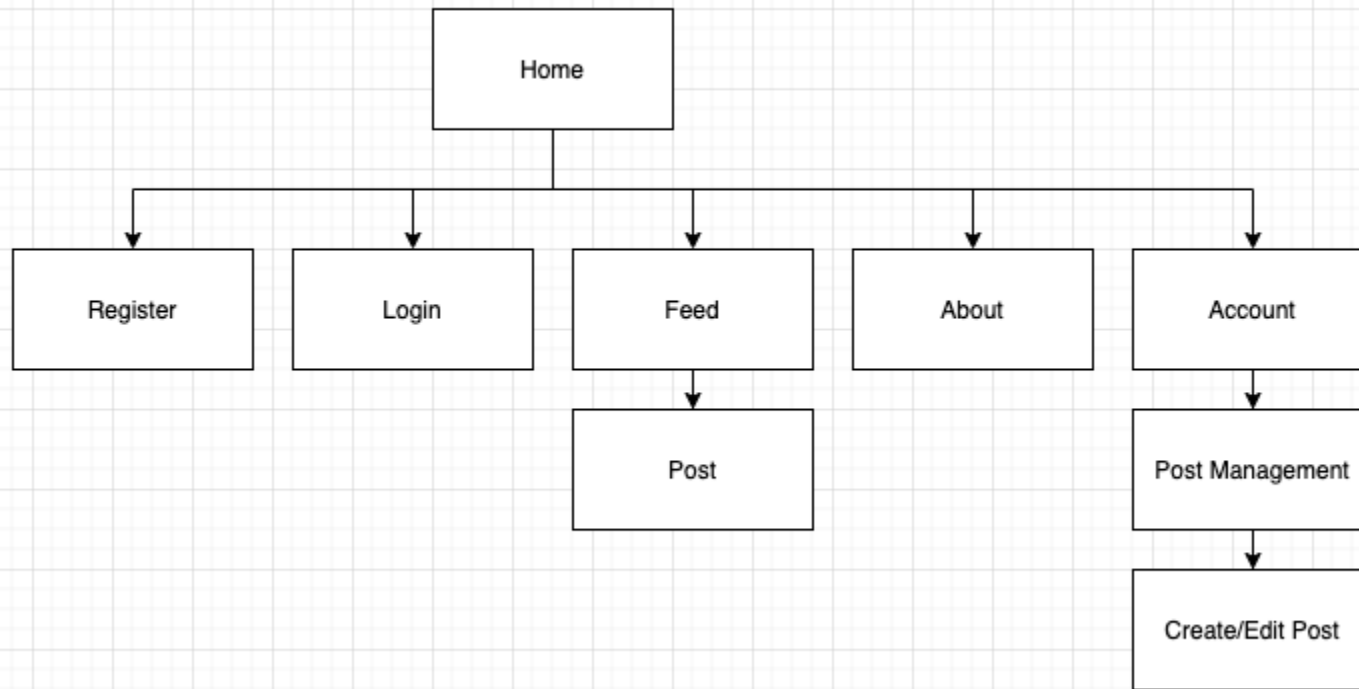


DDL Scripts:

The current DDL Create script can be found at the following link: <https://github.com/Kdeshun/CST323-Milestone>

Sitemap Diagram:

Include an image file of your Sitemap diagram.





GRAND CANYON
UNIVERSITY™

User Interface Diagrams:

You should insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for labeling this section N/A.



BACK

EDIT POST

Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

DELETE POST

SAVE CHANGES



BACK

CREATE ACCOUNT

FIRST NAME

John

EMAIL

Smith

EMAIL

eg. name@gmail.com

PASSWORD

REGISTER



BACK

TITLE OF POST

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

ALL POSTS



BACK

POST MANAGEMENT

WRITE NEW POST

EDIT A POST

POST TITLE

POST TITLE

POST TITLE

POST TITLE



BACK

MY ACCOUNT

WRITE OR EDIT POSTS

FIRST NAME

John

LAST NAME

Smith

EMAIL

eg. name@gmail.com

PASSWORD

SAVE CHANGES



BACK

LOGIN

EMAIL

eg. name@gmail.com

PASSWORD

LOG IN

Don't have an account?

REGISTER



GRAND CANYON
UNIVERSITY™

THE "BLOGS ARE COOL" BLOG

[HOME](#)

[ABOUT](#)

[FEED](#)

[ACCOUNT](#)

[LOGIN](#)

Read interesting things, even if you don't want to.

[CREATE ACCOUNT](#)

[LOGIN](#)



BACK

ALL ARTICLES

TITLE OF ARTICLE

TITLE OF ARTICLE

TITLE OF ARTICLE

TITLE OF ARTICLE

Class Diagrams:

You should insert any class diagrams here. Your class diagrams should be drawn correctly with the three appropriate class compartments, + and – minus to indicate accessibility, and the data types for the state/properties as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for labeling this section N/A.

UML Diagram:

No UML diagram at this time, while we sort out how we will organize and structure our code layers between the front/back ends.

Service API Design:

This section should fully document any Third Party Service Interface API's being consumed or application specific Service API's being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third party developer to integrate with the service and API.

- **Port:** 3001
- **Purpose:** This API layer will serve as the intermediary between the web client and the database, providing RESTful routes to manage Users and Posts records.

User Routes

1. **Get User by Email**
 - **Route:** /user/:email
 - **Method:** GET
 - **Description:** Retrieve user information based on the provided email.
2. **Get All Users**
 - **Route:** /users
 - **Method:** GET
 - **Description:** Retrieve a list of all registered users.
3. **Create User**
 - **Route:** /user/create
 - **Method:** POST
 - **Description:** Create a new user account.
 - **Expected Parameters:** (to be documented)
4. **Update User**
 - **Route:** /user/update
 - **Method:** POST
 - **Description:** Update user information.
 - **Expected Parameters:** (to be documented)
5. **Authenticate User**
 - **Route:** /user/auth



GRAND CANYON UNIVERSITY™

- **Method:** POST
- **Description:** Authenticate a user for login.
- **Expected Parameters:** (to be documented)

Post Routes

1. **Get Post by ID**
 - **Route:** /post/:id
 - **Method:** GET
 - **Description:** Retrieve a post based on its ID.
2. **Get All Posts**
 - **Route:** /posts
 - **Method:** GET
 - **Description:** Retrieve a list of all posts.
3. **Get Posts by User**
 - **Route:** /posts/:user
 - **Method:** GET
 - **Description:** Retrieve all posts created by a specific user.
4. **Create Post**
 - **Route:** /post/create
 - **Method:** POST
 - **Description:** Create a new post.
 - **Expected Parameters:** (to be documented)
5. **Update Post**
 - **Route:** /post/update
 - **Method:** POST
 - **Description:** Update an existing post.
 - **Expected Parameters:** (to be documented)

Next Steps

- **Documentation:** Additional documentation detailing the expected POST request parameters and the return objects for each route will be necessary to provide clarity for developers.
- **Implementation:** Begin implementing the API routes, ensuring proper error handling and response formatting.

This structured API will facilitate interactions between the client and the database, allowing for effective management of user accounts and posts.

Security Design:

This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.

Password Management Strategy

1. User Registration

- **Process:**
 - Users will provide their email and a unique password during registration.
 - Before storing the password, it will be salted and hashed to enhance security.
- **Implementation:**
 - Use a library like bcrypt to handle salting and hashing.
 - Store the hashed password in the database, never the raw password.

2. User Login

- **Process:**
 - Users log in by providing their email and password.
 - The inputted password will be salted and hashed using the same method as during registration.
 - Compare the hashed input to the stored hash in the database.
- **Implementation:**
 - If the hashes match, grant access to the user's account.
 - If they do not match, return an error indicating invalid credentials.

3. Password Update

- **Process:**
 - Users can update their password by providing their current password along with the new password.
 - Similar to the login process, the current password will be salted and hashed for verification.
- **Implementation:**
 - Compare the hashed current password with the stored hash.
 - If they match, proceed to hash the new password and update the stored hash in the database.
 - Ensure to validate the new password (e.g., minimum length, complexity requirements).

Security Considerations

- **Salting:** Each password should be salted with a unique salt before hashing to prevent rainbow table attacks.
- **Hashing Algorithm:** Use a strong hashing algorithm like bcrypt, Argon2, or PBKDF2, which are designed for password storage and include built-in salting.
- **Error Handling:** Provide generic error messages for failed login attempts to avoid disclosing whether the email or password was incorrect.

Example Implementation

Here's a brief code snippet showcasing how you might implement the password hashing and comparison using bcrypt:

javascript

Copy

```
const bcrypt = require('bcrypt');
```



// Registering a user

```
async function registerUser(email, password) {  
  const saltRounds = 10;  
  const hashedPassword = await bcrypt.hash(password, saltRounds);  
  // Store email and hashedPassword in the database  
}
```

// Logging in a user

```
async function loginUser(email, inputPassword) {  
  const user = await getUserByEmail(email); // Fetch user from DB  
  if (user) {  
    const match = await bcrypt.compare(inputPassword, user.hashedPassword);  
    if (match) {  
      // Successful login  
    } else {  
      // Invalid password  
    }  
  } else {  
    // User not found  
  }  
}
```

// Updating a user's password

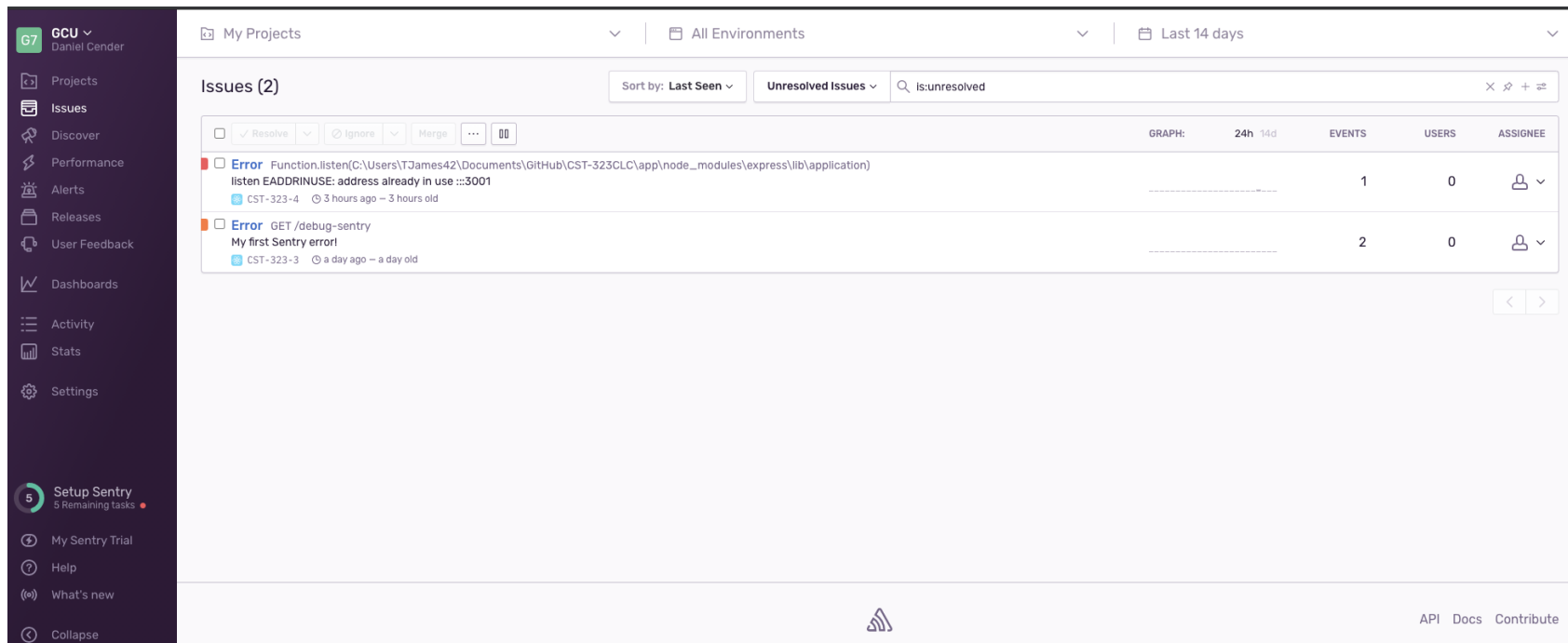
```
async function updatePassword(email, currentPassword, newPassword) {  
  const user = await getUserByEmail(email);  
  if (user) {  
    const match = await bcrypt.compare(currentPassword, user.hashedPassword);  
    if (match) {  
      const hashedNewPassword = await bcrypt.hash(newPassword, 10);  
      // Update the user's password in the database  
    } else {  
      // Current password is incorrect  
    }  
  }  
}
```


This approach ensures that user passwords are handled securely, minimizing the risk of data breaches and unauthorized access.

Other Documentation:



Logging

Error tracking and notification integrations are being handled by Sentry, a great service which provides APIs and SDKs for most software development toolsets. A screenshot of this application’s dashboard is below:



The screenshot shows the Sentry dashboard for a project named 'GCU'. The left sidebar contains navigation links: Projects, Issues, Discover, Performance, Alerts, Releases, User Feedback, Dashboards, Activity, Stats, and Settings. Below these is a 'Setup Sentry' section with 5 remaining tasks, and links for 'My Sentry Trial', 'Help', 'What's new', and 'Collapse'.

The main area displays 'Issues (2)' with a search bar and filters. The issues are listed in a table with columns: GRAPH, 24h, 14d, EVENTS, USERS, and ASSIGNEE.

GRAPH	24h	14d	EVENTS	USERS	ASSIGNEE
<p>Error Function.listen(C:\Users\TJJames42\Documents\GitHub\CST-323CLC\app\node_modules\express\lib\application) listen EADDRINUSE: address already in use :::3001</p> <p>CST-323-4 3 hours ago - 3 hours old</p>			1	0	 v
<p>Error GET /debug-sentry My first Sentry error</p> <p>CST-323-3 a day ago - a day old</p>			2	0	 v

At the bottom right of the dashboard, there are links for 'API', 'Docs', and 'Contribute'.

Uptime Tracking

Site uptime and outage notifications are being handled by Uptime Robot. It has a generous free tier that can be used to verify a web service is basically functional and available on specified ports or protocols. Below is this app’s dashboard:



UptimeRobot

Upgrade

Dashboard

Status pages

My Settings

TV Mode

+ Add New Monitor

(Bulk Actions)

(Export Monitors - Expand Monitor Names)

Sort Monitors

Last 24 Hours

3

5

7

9

11

13

15

17

19

21

23

1

0%

http

CST-323-CLC

CST-323-CLC

<http://cst323clc-env.eba-mm2xi3h5.us-east-1.elasticbeanstalk.com/>

A place to find all the details about your monitors

Uptime last 24 hours

Response Time last 24 hours (0.00ms av.)

Shows the "Instant" that the monitor started returning a response in ms (and average for the displayed period is 0.00ms).

1.0

0.5

0.0

-0.5

-1.0

Milliseconds

Current Status

Up

Since 0 hrs, 0 mins (2020-12-07 03:20:26)

Uptime

0.000% (last 24 hours)

0.000% (last 7 days)

0.000% (last 30 days)

Latest downtime

No downtime recorded.

Latest Events (up, down, start, pause)

Event	Date-Time	Reason	Duration
Up	2020-12-07 03:20:26	OK (200)	0 hrs, 0 mins
Started	2020-12-07 03:17:51	Started	0 hrs, 2 mins

Export Logs