GRAND CANYON UNIVERSITY™

# CST-323 Design Report Template

| | |
|---|---|
| **Topic:** | Milestone 5 |
| **Date:** | 12/20/2024 |
| **Revision:** | 5.0 |
| **Team:** | 1. Kaya Nelson |
| | 2. Tyson Martin |
| **Weekly Team Status Summary:** | |

| User Story | Team Member | Hours Worked | Hours Remaining |
|---|---|---|---|
| *As a dev I would like to have Sprints, burndown chart, and repo setup for project management.* | *Kaya* | *1* | *0* |
| *As a dev I would like to have a consistent CI/CD pipeline.* | *Tyson* | *3* | *0* |
| *As a user I would like to create and edit post data* | *Tyson* | *2* | *0* |
| *As a dev I would like to have all documentation in a zip file for submission* | *Kaya* | *1* | *0* |
| *As a dev I would like a powerpoint presentation for a final project presentation.* | *Kaya* | *2* | *0* |
| *As a dev I would like a high level video overview of the code, webdesign, and features for presenting the final project.* | *Kaya/Tyson* | *1* | *0* |
| *As a dev I would like to have a screencast of the current progress of the app for Milestone 5* | *Tyson* | *1* | *0* |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| | | |
|---|---|---|
| **GIT URL:** | *https://github.com/DanielCender/CST323CLC-App* | |
| **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: https://boiling-meadow-41866.herokuapp.com/

Gthub: https://github.com/Kdeshun/CST323-Milestone

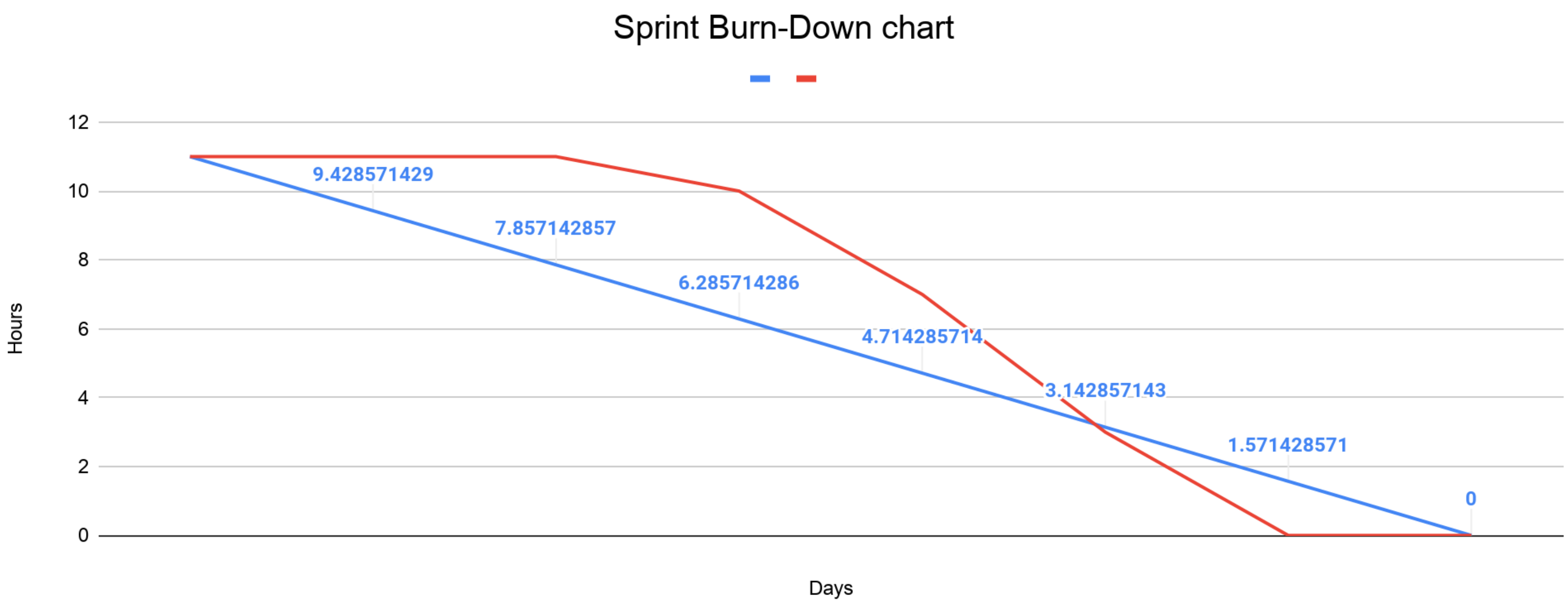
Presentation URL: https://go.screenpal.com/watch/cZlY61necwS

For React Component Documentation through JSDocs: run ‘npm run docs’ in terminal

# Planning Documentation

## 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 significantly enhanced our ability to convey team standards and expectations regarding the completion and submission of work. This approach provided a single platform where all team members could access crucial information, guidelines, and updates. By establishing a clear channel for communication, we ensured that everyone was aligned on project objectives, deadlines, and submission protocols. |

| |
|---|
| The initial setup with react/ node/ npm/ and base project file worked flawlessly |
| This centralized system not only streamlined our interactions but also fostered accountability among team members. It allowed us to document discussions, share resources, and clarify any uncertainties in real time. As a result, we minimized misunderstandings and ensured that everyone was on the same page regarding the quality and format of deliverables. |

*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** |
|---|---|---|
| | | |
| | | |
| | | |

## 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.*

1. **Clone the Repository**
   Begin by cloning the repository to your local machine. This can be done using the following command:

   bash

   ```
   git clone <repository-url>
   ```

2. **Change Directories**
   Navigate to the `/app` folder within the cloned repository:

   bash

   ```
   cd app
   ```

3. **Install Dependencies**
   Open a Terminal in the `/app` directory and run the following command to install all necessary dependencies:

   bash

   ```
   npm install
   ```

   This command will ensure that all required packages are installed and properly linked.

4. **Run the Development Server**
   To start the development server, execute the following command:

   bash

```
npm run dev
```

This command will initiate the server defined in `server.js` and start the React application using the react-script process.

## Building the Application for Deployment

5. **Authenticate with Heroku**
   Before deploying the application, you need to authenticate with Heroku. This involves connecting your session to the desired Heroku application.
6. **Deploy to Heroku**
   Once authenticated, you can deploy your application by running:

   bash

   ```
   git push heroku main
   ```

   Heroku will initiate a deployment process, which includes creating a build folder for the frontend application.

## Continuous Integration/Continuous Deployment (CI/CD)

To streamline the deployment process, continuous deployment has been configured on Heroku. This means that developers no longer need to manually push changes to the Heroku deployment. Whenever there is a change pushed to the main branch or a Pull Request is merged, Heroku will receive a webhook event and automatically start the build process using the latest code changes.

## Generating and Viewing Documentation

7. **Generate Documentation**
   In the base directory of your project, run the following command to generate a `docs/` directory containing the most recent changes:

   bash

   ```
   npm run docs
   ```

8. **View Documentation**
   There are a couple of ways to view the generated documentation:

- **Using Serve**: If you have the `serve` package installed, you can host the documentation locally by running:

  bash

  ```
  serve docs/
  ```

  This will typically host the documentation at a static local port (usually 5000).

- **Directly in a Browser**: If `serve` is not installed, you can manually open the documentation by dragging the `docs/index.html` file into a web browser like Chrome. The browser will then read and display the site data.

## 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 is focused on creating a blogging platform designed for the open web. To facilitate effective collaboration and streamline our development process, we will utilize a variety of tools:

1. **Communication: Discord**
   We will use Discord as our primary communication channel. This platform will enable us to stay connected throughout the week, discuss project updates, and collaborate in real time.
2. **Source Code Management: GitHub**
   For source code management and version control, we will leverage GitHub. This will allow us to track changes, manage branches, and collaborate on code effectively, ensuring that everyone is aligned and can contribute seamlessly.
3. **Documentation: Google Docs**
   We will utilize Google Docs to create and maintain sprint and burndown charts, as well as to provide updates on our design reports. This collaborative documentation will help us keep track of our progress and ensure transparency within the team.
4. **Design and Wireframing: Draw.io**
   For wireframing and UML designs, we will use Draw.io. This tool will assist us in visually mapping out our platform's structure and functionality, fostering a clearer understanding of the project as a whole.
5. **Database Design: MySQL Workbench**
   To generate Entity-Relationship (ER) diagrams, we will employ MySQL Workbench. This software will enable us to create detailed database designs that accurately reflect the relationships and structures necessary for our blogging platform.

By integrating these tools into our workflow, we will enhance collaboration and maintain an organized approach to developing our blogging platform. This structured methodology will help us achieve our goals efficiently while ensuring that all team members are engaged and informed throughout the project.

## 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**

- **Communication: Discord**
- **Documentation: Google Docs**
- **Version Control: GitHub**
- **Code Editor: Visual Studio Code (alternative editors allowed)**
- **Video Presentations: Loom**

**Deployment/Monitoring**

- **Deployment & CI/CD: Heroku**
- **Database Management: JawsDB (Heroku plugin)**
- **Service Uptime Monitoring: UptimeRobot**
- **Error Tracking: Sentry**

**Framework/Language**

- **Frontend: React**
- **Backend: Node.js**
- **Database: MySQL**

**Libraries/Packages**

**Utilizing various packages to enhance functionality:**

- **MySQL: MySQL npm package**
- **Editor.js: Editor.js Documentation**
- **Axios: Axios npm package**
- **UUID: UUID npm package**
- **React Bootstrap: React Bootstrap Documentation**
- **JSDocs for documentation generation**

## Known Issues:

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

**User Registration**

- **Issue**: After registering as a new user, users are not redirected to the login page.
  - **Planned Fix**: Implement a redirect to the login page post-registration. Target for resolution in the next milestone.

**Account Page Layout**

- **Issue**: The "write/edit posts" button does not aesthetically match the change password form.
  - **Planned Fix**: Update the styling of the button to ensure it fits better with the overall design of the Account page.

**Post Creation**

- **Issue**: When a post is created, the user who created the post is not linked to the Author field in the Post database record.
  - **Planned Fix**: Modify the post creation logic to ensure that the current user is correctly associated with the Author field in the database.

These updates will enhance user experience and ensure better functionality within the application.

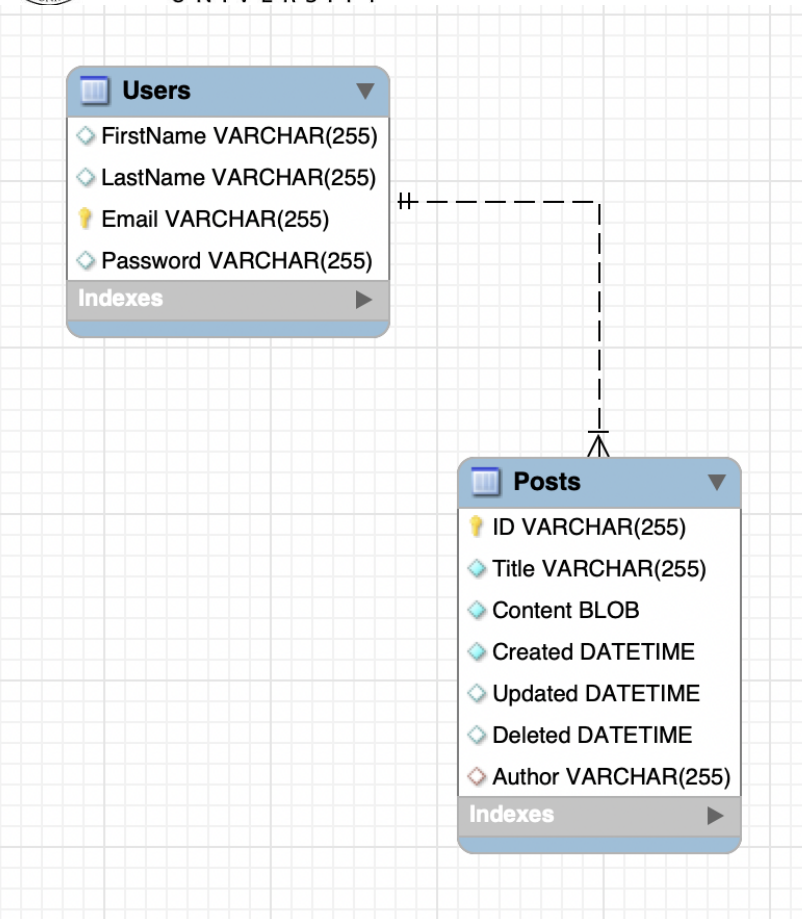
## Risks:

*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.



Users
- FirstName VARCHAR(255)
- LastName VARCHAR(255)
- Email VARCHAR(255)
- Password VARCHAR(255)

Indexes

Posts
- ID VARCHAR(255)
- Title VARCHAR(255)
- Content BLOB
- Created DATETIME
- Updated DATETIME
- Deleted DATETIME
- Author VARCHAR(255)

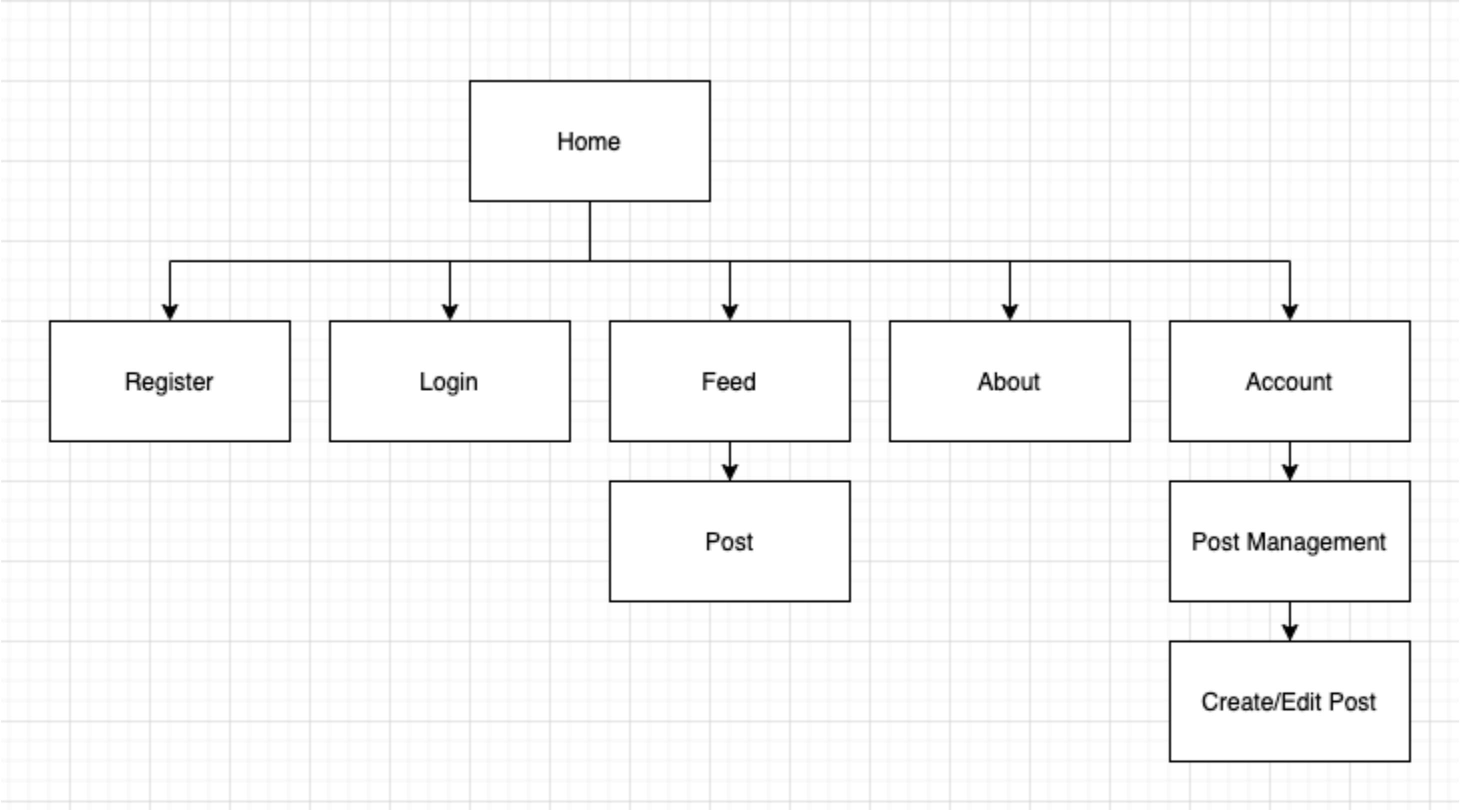
Indexes

## DDL Scripts:

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

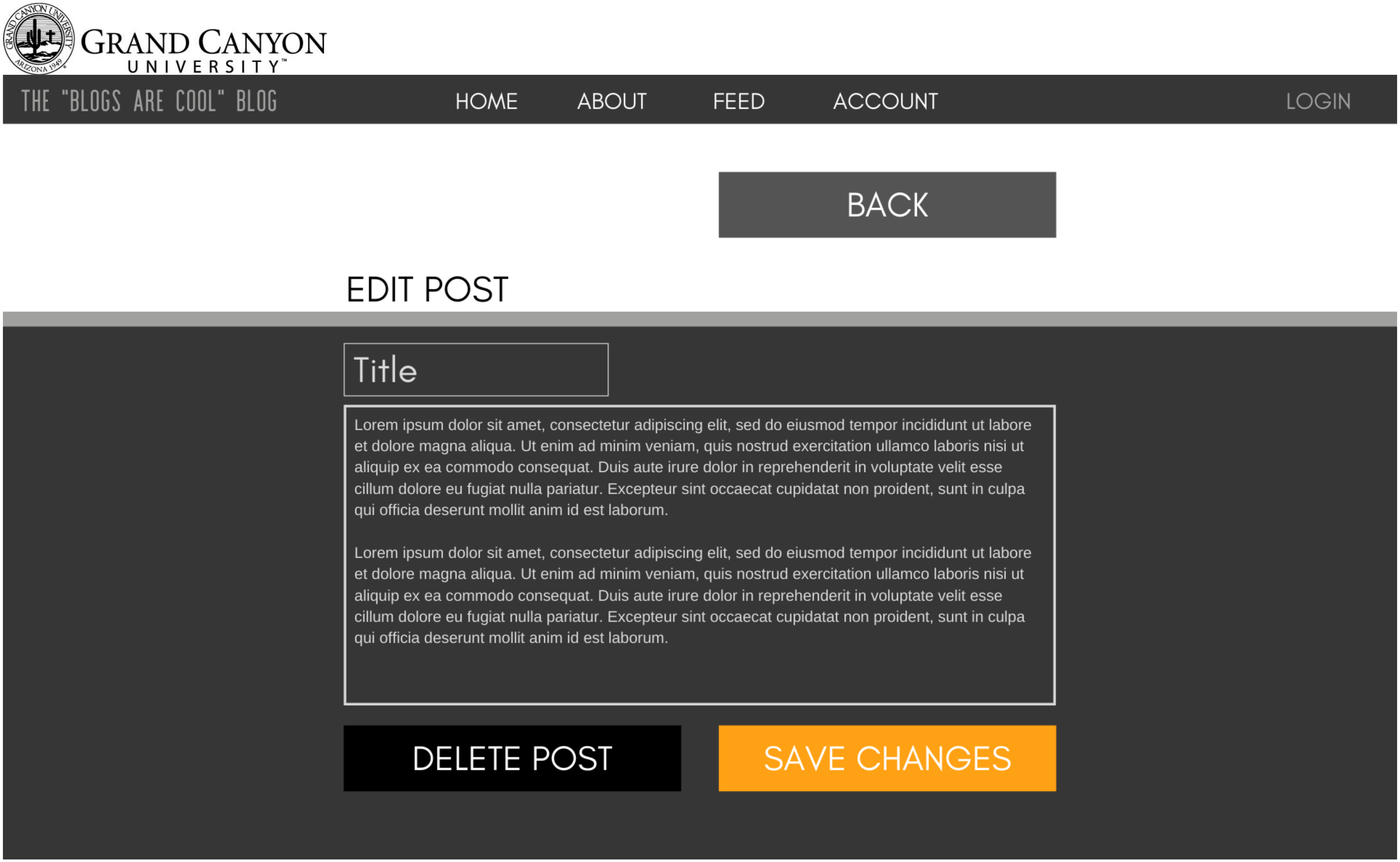
## Sitemap Diagram:

*Include an image file of your Sitemap diagram.*


Home
Register
Login
Feed
About
Account
Post
Post Management
Create/Edit Post

## 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.*

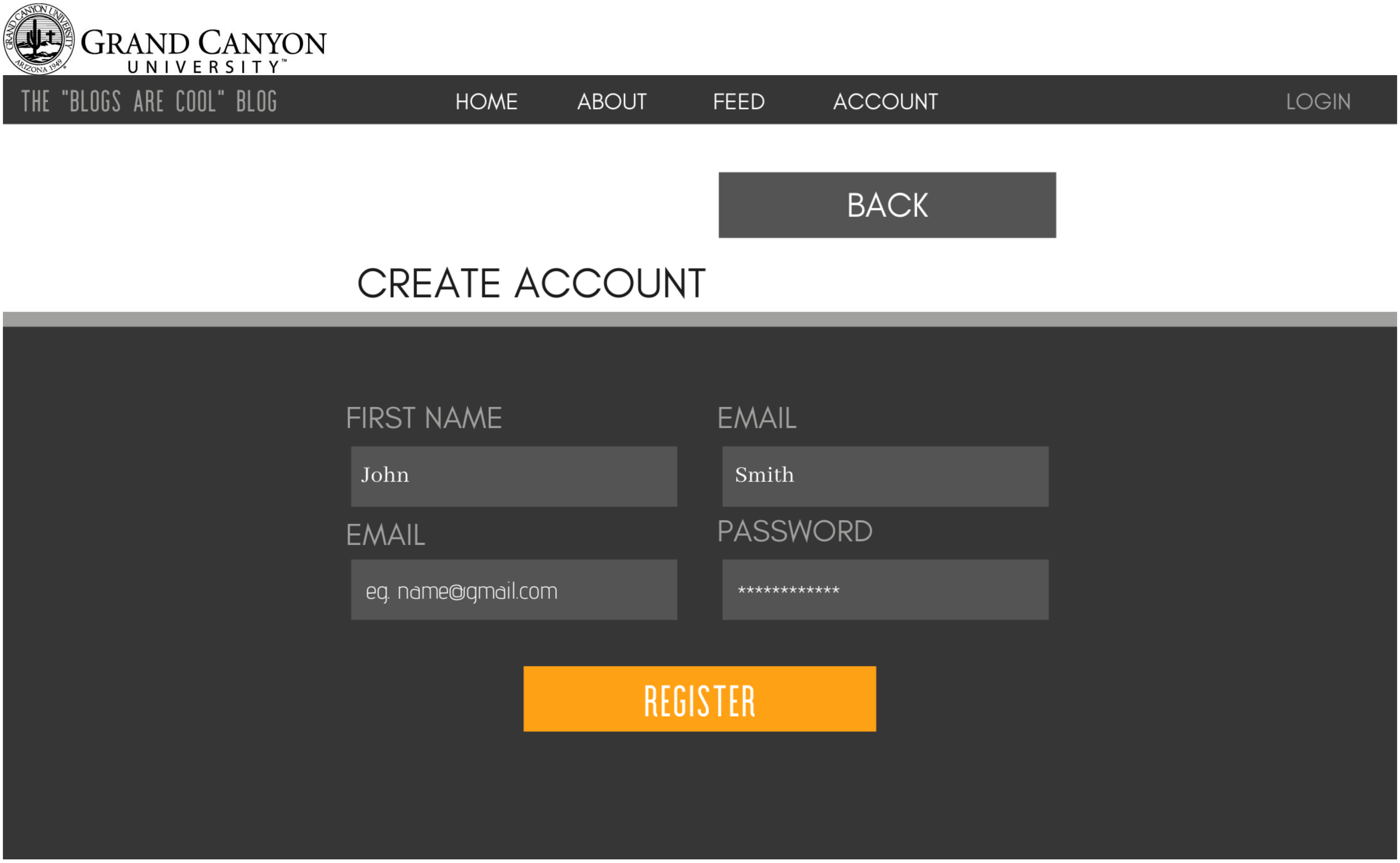


EDIT POST

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.



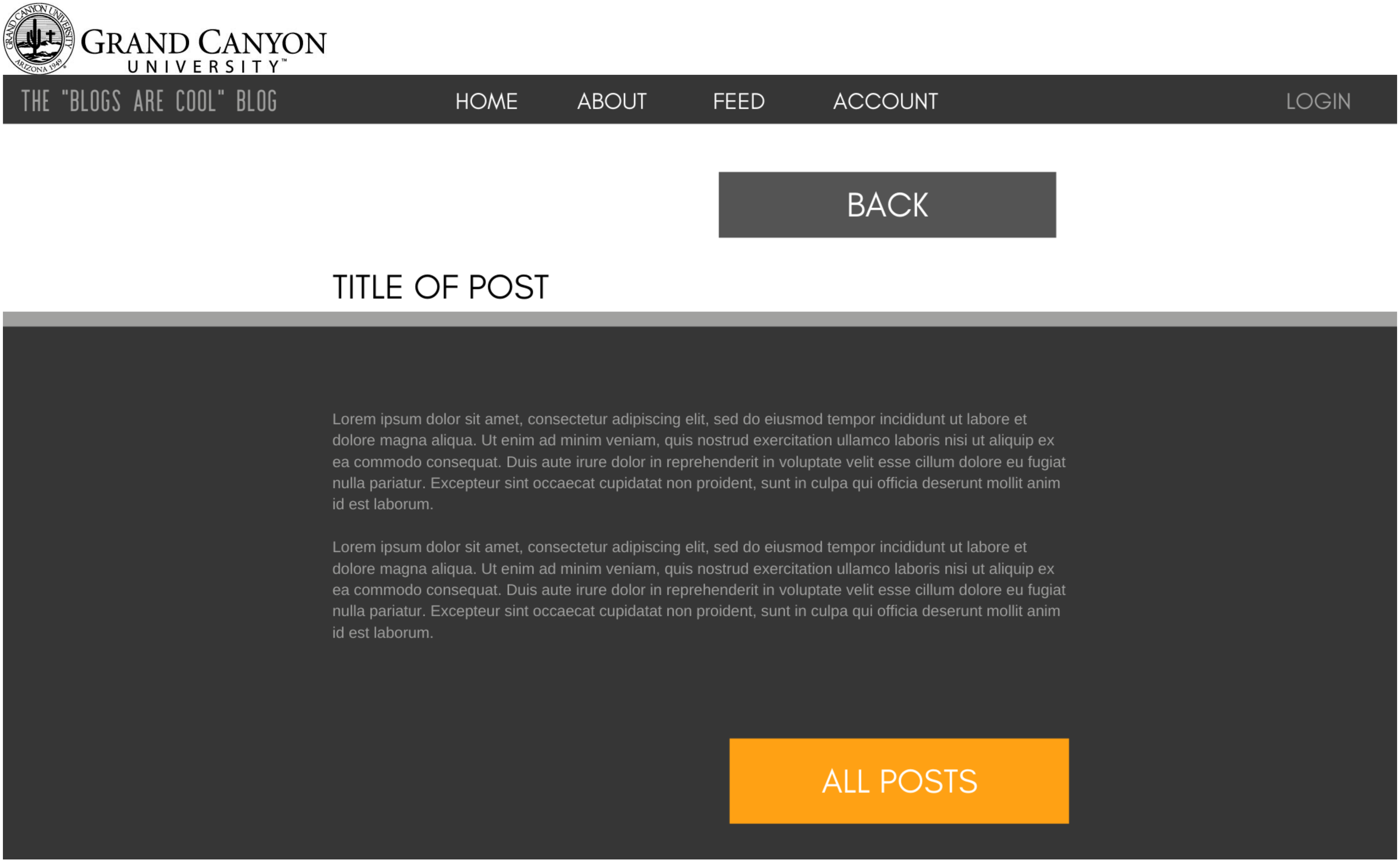
## CREATE ACCOUNT

FIRST NAME: John

EMAIL: Smith

EMAIL: eg. name@gmail.com

PASSWORD: ************



GRAND CANYON UNIVERSITY™

THE "BLOGS ARE COOL" BLOG

HOME ABOUT FEED ACCOUNT LOGIN

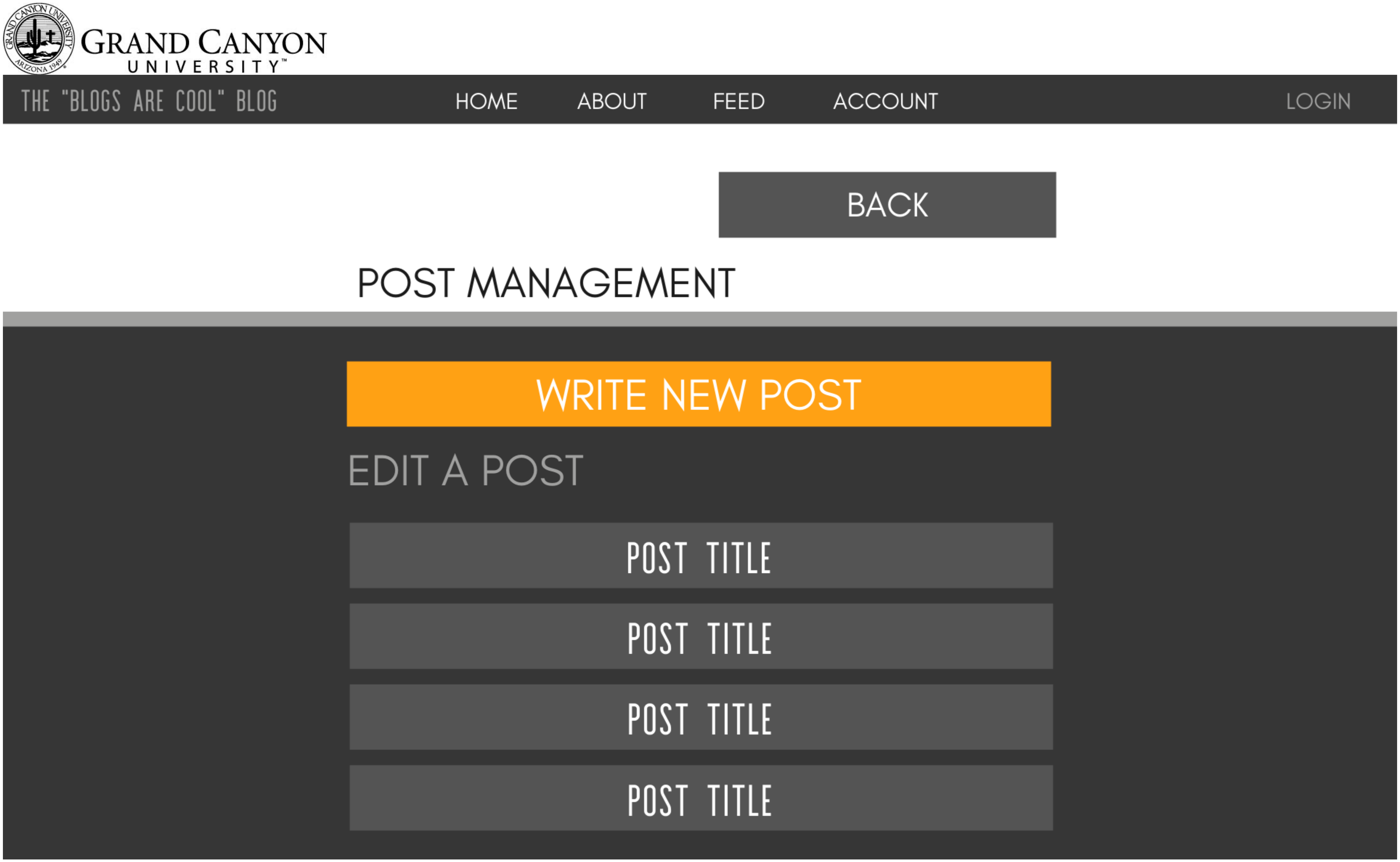
BACK

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.

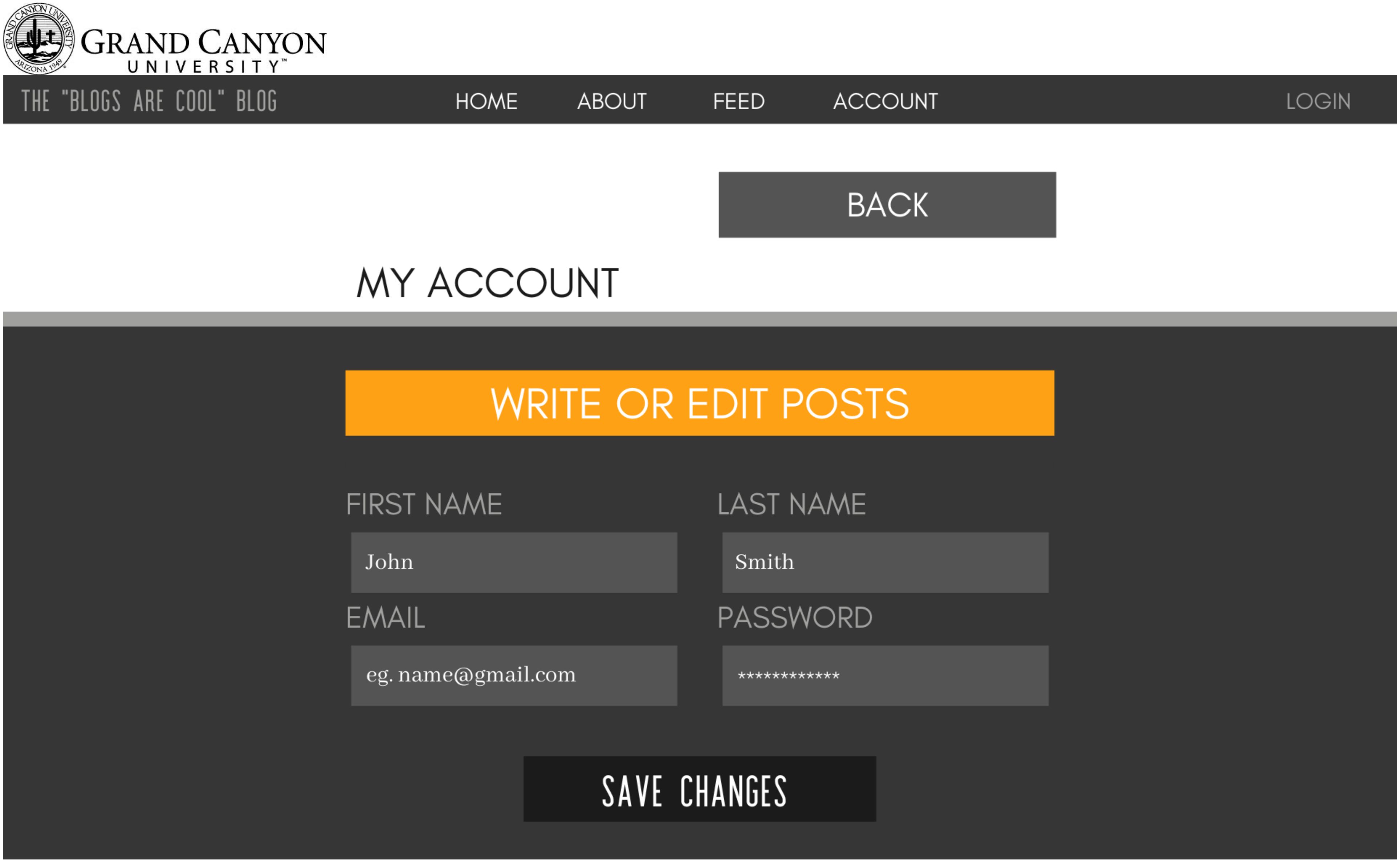
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum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

ALL POSTS



POST MANAGEMENT

EDIT A POST



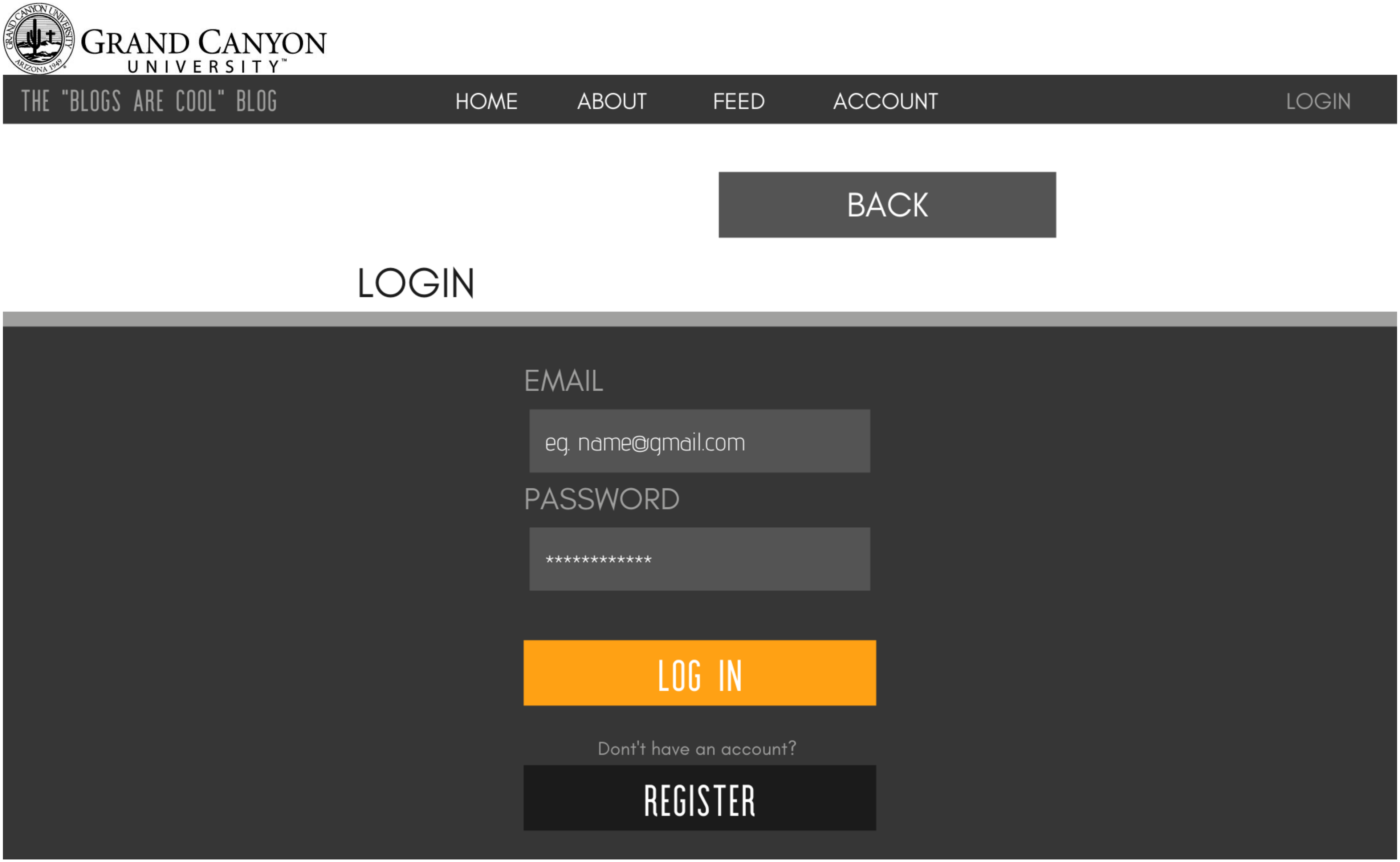
## MY ACCOUNT

FIRST NAME: John

LAST NAME: Smith

EMAIL: eg. name@gmail.com

PASSWORD: ************



LOGIN

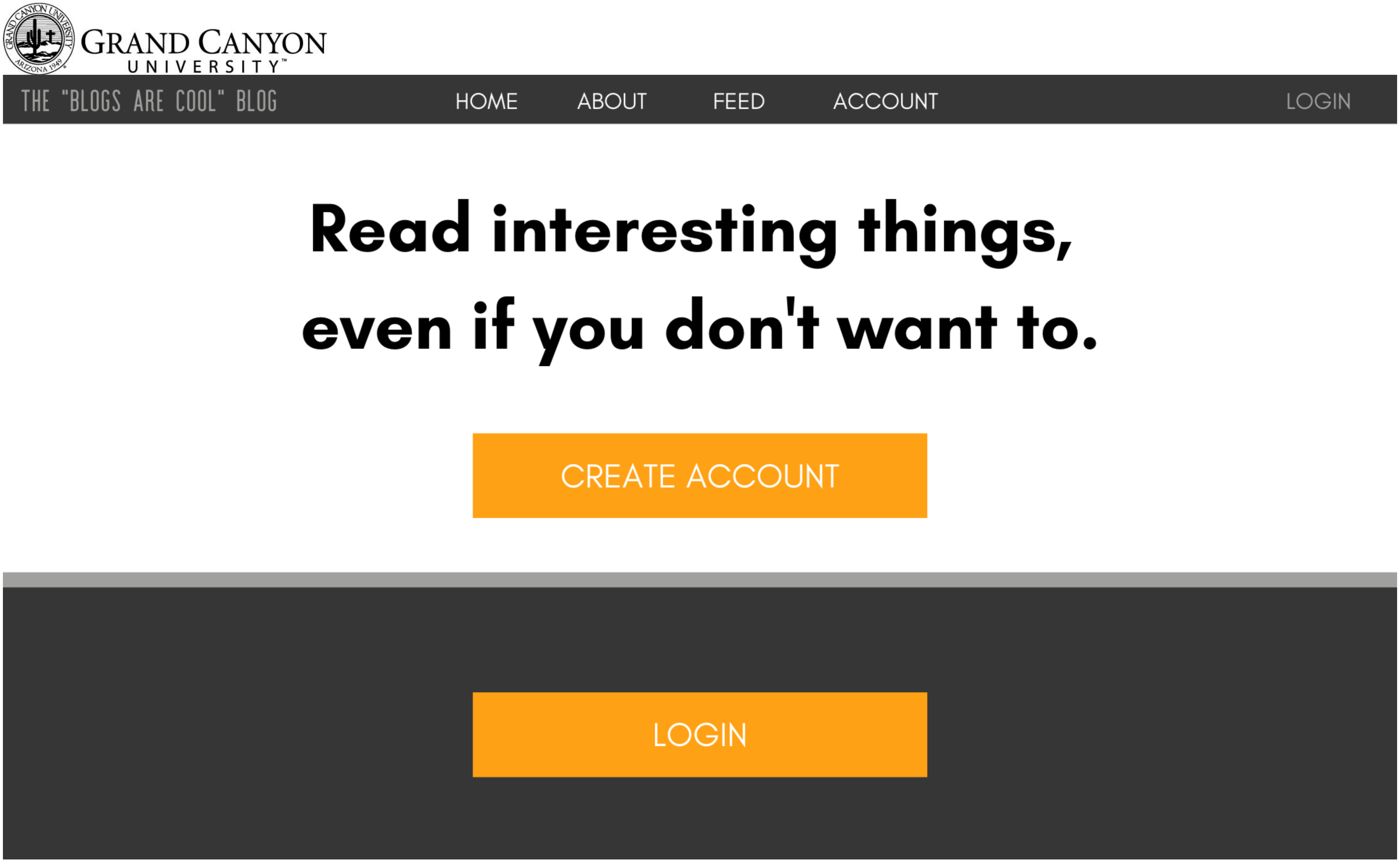
EMAIL

eg. name@gmail.com

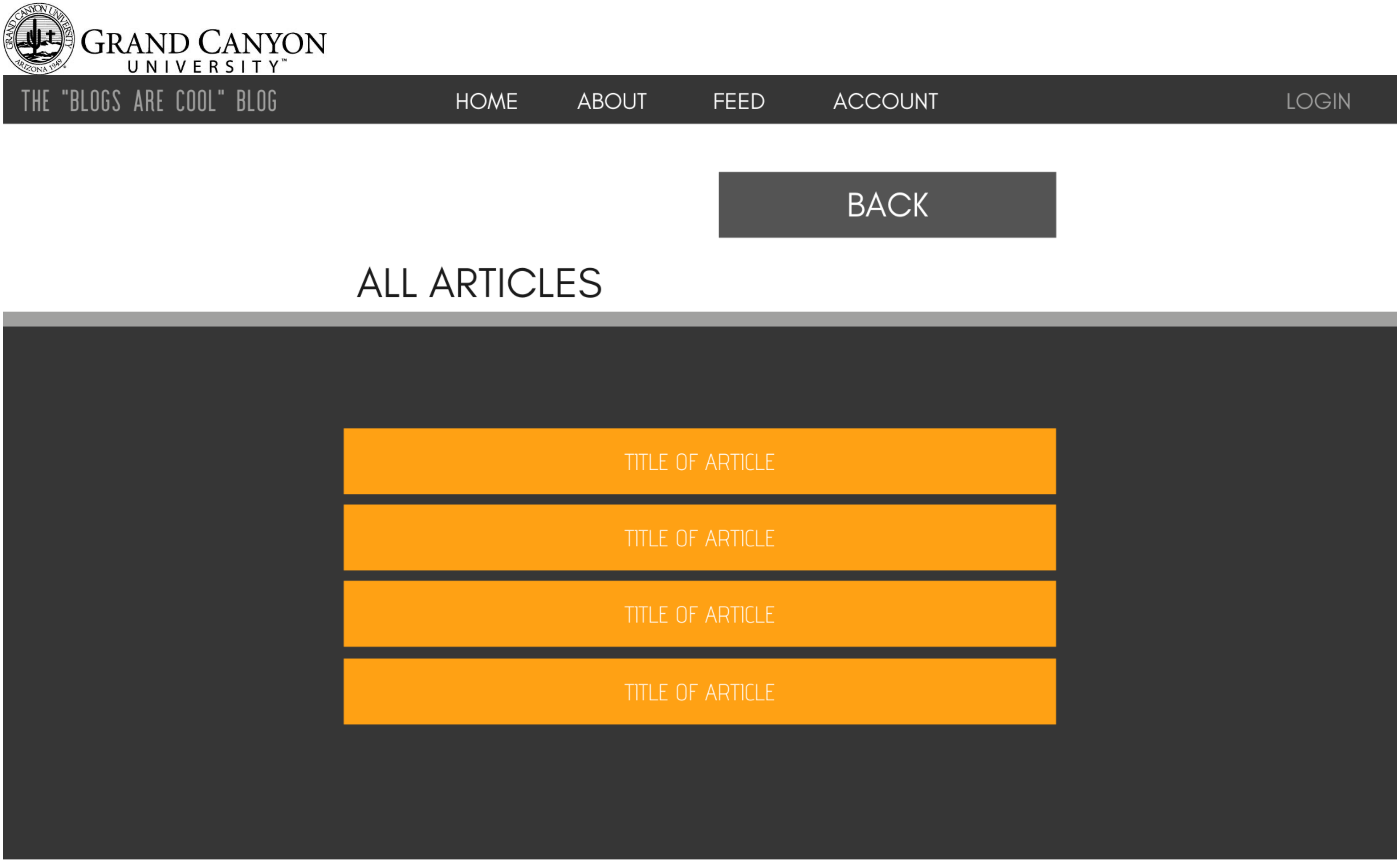
PASSWORD

************

Don't have an account?



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



**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.*

**Base URL**

- **Endpoint**: /api
- **Port**: 3001

**User Routes**

1. **GET /user/:email**
   - Retrieve user information by email.
2. **GET /users**
   - Fetch a list of all users.
3. **POST /user/create**
   - Create a new user.
4. **POST /user/update**
   - Update an existing user.
5. **POST /user/auth**
   - Authenticate a user (login).

**Post Routes**

1. **GET /post/:id**
   - Retrieve a post by ID.
2. **GET /posts**
   - Fetch a list of all posts.
3. **GET /posts/:user**
   - Fetch posts created by a specific user.
4. **POST /post/create**
   - Create a new post.
5. **POST /post/update**

  - Update an existing post.

**Future Documentation**

- Detailed documentation for expected POST request parameters and return objects for each route will be provided soon.

**Notes**

- Currently, there is no expectation to consume any third-party APIs or services, which simplifies the development focus on your internal API.

## 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.*

**Login Process**

- **Credentials**: Users will log in using their email and a unique password.
- **Password Security**:
  - **Salting and Hashing**:
    - Before storing passwords, they will be salted (adding a random string) and hashed (using a secure hashing algorithm).
    - This ensures that raw passwords are never stored in the database.

**Login Attempt**

- When a user attempts to log in:
  - The inputted password will be salted and hashed using the same method.
  - The hashed value will then be compared to the stored hash in the database.
  - If they match, the user is authenticated; if not, the login fails.

**Password Update**

- Users can update their password:
  - They must provide their current password to verify their identity.
  - The new password will also be salted and hashed before being stored.

**Security Considerations**

- This approach enhances security by protecting user passwords and ensuring that even if the database is compromised, raw passwords are not exposed.

## Other Documentation:

**Integration with Sentry**

- **Error Tracking: Sentry will be used to monitor and track application errors in real-time.**
- **Notifications: Sentry provides alerts and notifications for exceptions and performance issues, allowing for quick responses to potential issues.**

**Benefits of Using Sentry**

- **Comprehensive Monitoring: Sentry supports tracking errors across different environments (development, staging, production).**
- **APIs and SDKs: It offers APIs and SDKs compatible with various programming languages and frameworks, making integration straightforward.**
- **Dashboard: Sentry provides a user-friendly dashboard that visualizes errors, performance data, and user feedback.**

**Next Steps**

- **Ensure that Sentry is properly configured in your application.**
- **Capture relevant error information and context to facilitate debugging.**
- **Regularly review the Sentry dashboard for insights and trends in application performance.**

**Uptime Tracking**

**Integration with Uptime Robot**

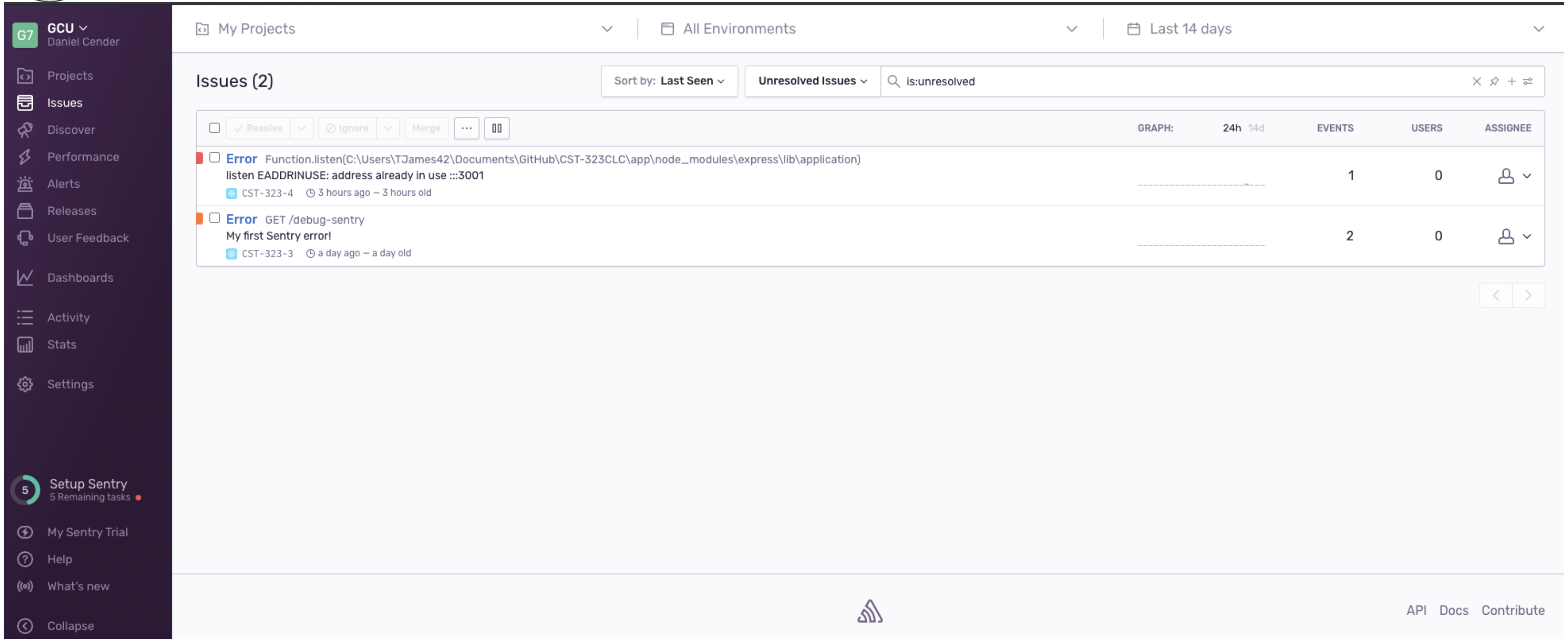
- **Service Monitoring: Uptime Robot will monitor the availability of your web service, checking specified endpoints to ensure they are operational.**
- **Outage Notifications: It provides alerts via email or SMS when downtime is detected, allowing for quick responses to service interruptions.**

**Benefits of Using Uptime Robot**

- **Generous Free Tier: The free plan allows you to monitor multiple services without incurring costs, making it ideal for startups and small projects.**
- **Protocol Support: Uptime Robot supports various protocols, including HTTP(s), ping, and TCP, allowing flexibility in what you monitor.**
- **User-Friendly Dashboard: The dashboard provides a clear overview of your service’s uptime history, response times, and any incidents.**

**Next Steps**

- **Set up monitoring for all critical services and endpoints.**
- **Configure alert settings to ensure timely notifications during outages.**
- **Regularly review the dashboard to analyze uptime trends and performance metrics.**



Issues (2)

Sort by: Last Seen | Unresolved Issues | is:unresolved

| | GRAPH: 24h 14d | EVENTS | USERS | ASSIGNEE |
|---|---|---|---|---|
| **Error** Function.listen(C:\Users\TJames42\Documents\GitHub\CST-323CLC\app\node_modules\express\lib\application)<br>listen EADDRINUSE: address already in use :::3001<br>CST-323-4 3 hours ago – 3 hours old | | 1 | 0 | |
| **Error** GET /debug-sentry<br>My first Sentry error!<br>CST-323-3 a day ago – a day old | | 2 | 0 | |