Final Project Documentation (SRS)

Grid Social

(06/12/2024)

Ethan Mongelli, Hamza Ahmed, Raven Griffin

Table of Contents:

(Please note: There were somethings that used to work before, but did not work when we tried to implement security.)

Here are our contributions as well:

Hamza contributions: User model (all the classes associated with the user), Login, signup pages. Majority of the design doc. One part of the software requirements doc. Created the first Feed classes (controller, repository, etc.) until it was later refactored to Post. Tried to keep open and effective communication between group mates and ensure that deadlines were met on time. Also created the first likes and comments classes, but my computer refused to run the program no matter what troubleshooting I did so both Raven and Ethan reviewed this code and made modifications to it so that it would hopefully work. Raven: Took care of the group and admin side of things on both the front and backend stuff. Contribute to both the design document and software requirement documents. Also refactored some of Hamza's work while his laptop wasn't working properly. Ethan: Implemented Spring security, Nylas API and refactored some of Hamza's work while his laptop was not working properly and did the rest of the front end not mentioned in the previous 2 group mates contributions. Integrated everyone's parts together on his computer after Hamza and Raven pushed their parts together and was responsible for trying to get it to work on his laptop.

Software Requirements Specification (SRS) Document

Grid

05/19/2024

Version 1

Ethan Mongelli, Hamza Ahmed, Raven Griffin

1. Project General Description: **(Raven’s contribution to this document)**

   The goal of the Grid web application is to connect users to groups via a calendar. It is a new social media platform that makes making plans easy. This app will allow users to create and join groups according to their interests.

2. Product Features **(Raven’s contribution to this document)**

The Grid Social is an application is designed to revolutionize the way users make plans and find groups of people. Below are the key features and functions that define the essence of the application:

Feed: Every user has an individual feed unique to the user. The things shown on the feed are other events posted by the people the user follows.

Popup Scrapbook: Users apart of the same group attending the same event can upload pictures to a popup scrapbook. This scrapbook will appear whenever someone clicks on the event and is visible on an individual’s feed.

3. Functional Requirements: **(Hamza’s contribution to this document)**

- **FR0: Create/Modify User Profile:** Users can create their profiles, including personal information and calendar preferences.
- **FR1: View/Add Public Calendars:** Users can view and add public calendars of other users or groups to see their availability.
- **FR2: Establish Meeting Times:** Users can initiate a tool/sub-group feature to suggest overlapping meeting times with selected individuals or groups.
- **FR3: Automatic Calendar Updates:** When a meeting time is agreed upon, the creator of the group or sub-group can mark the time, and it will automatically update in each participant's calendar.
- **FR4: Group Management:** Users can create and manage groups, including mainboard calendars for organizations, and sub-groups for specific purposes.
- **FR5: Fun Social Features:** Optional inclusion of fun social features like theming to encourage interaction with others' calendars.

4. Non-Functional Requirements: **(Hamza’s contribution to this document)**

- **NFR0: Calendar Display Speed:** The system should display calendar availability and group information within 5 seconds to ensure a smooth user experience.
- **NFR1: User Interaction Time:** Users should be able to locate and interact with group features, such as creating or joining groups, in less than 10 seconds.

- **NFR2: Calendar Update Efficiency:** Automatic calendar updates should occur instantaneously or within 5 seconds of marking a meeting time to ensure prompt synchronization across all participants' calendars.

5. Scenarios: :** **(Ethan's contribution to this document)**

a. Users – **(Hamza Ahmed)**

i. GridSocial/Feed Scroll

· Initial Assumption: The user has access to the web app, is logged in and is on the main page (base directory) of GridSocial, being the feed page.

· Normal: The user will be able to see a feed focused on the day they currently on and see a public feed of social happenings on that given day

o the user can switch (via navbar) the feed to users the have followed/friended such that it only shows relevant events and "calendar nodes" that are publicized

o You can follow people you see in the public feed or request someone via search

· What Can Go Wrong: User may not have friends or followed people yet, can direct to where to do so.

· Other Activities: Client can click the refresh button to get a new set of feed recommendations. Can also switch over to personal view via the navbar at the top-left.

· System State on Completion: user can continue to scroll and find up to date feed

ii. Personal View

· Initial Assumption: The user has access to the web app, is logged in and directs to the personal view on the top left.

· Normal: The user will be able to see a weekly focused calendar that begins with their own calendar

o the user add dates (nodes) to their calendar, prompting to add details and make it private or public.

o You can also switch to view a specific person's calendar you have followed via the portion to the left of the screen

o switch between weeks focused (up and down)

· What Can Go Wrong: May be confused on which profile they are viewing and how to switch to and from their own calendar. May be resolved by vividly showing whether they are on their own account or another's. Prompt to return to own somewhere

· Other Activities: Join groups and add those calendar events to their own (they must join via request/invite). Add "Scrapbooking" to calendar node, in which you can add photos to (which is visible by feed)

· System State on Completion: user has new calendar events added to their calendar and is visible others based on parameters

a. Group Owner – a "collaborative" (**Ethan Mongelli)**

i. GridSocial/Group Management

· Initial Assumption: The user has access to the web app, is logged in and is on the personal view.

· Normal: The (currently a User) will be able to create a group calendar event.

o The now group owner can invite others to this group (they have followed/friended) and establish a meeting time (via a tool). Also can remove them.

o Once the date/time is agreed upon, the event can be added to all the peoples' calendars

· What Can Go Wrong: Misplace group events once they are made. Could be resolved by having a symbol or way to view all joined/ created group nodes

· Other Activities:

· System State on Completion: calendar event is added to all participants calendars

a.Organization Manager/Admin – a selected user of a certified organization (**Raven Griffin)**

i. Organization Calendar (houses groups)

· Initial Assumption: This Manager is provided via the GridSocial and is signed in and is on personal view.

- Normal: The Manager will be able to switch to the organization (via the friends tab on the left) in which a organization calendar is setup
    - Provides method of invite to join the organization calendar
    - Add and remove users within organization
    - Access to management controls of organization (on right side of personal view page)
    - allow groups (sent requests) to be made within organization
- What Can Go Wrong: not know where to allow requests of groups and to manage organization
- System State on Completion: a made calendar is made such that viewers can join and view the collective calendar of groups

Design Document (SRS) Document

Grid

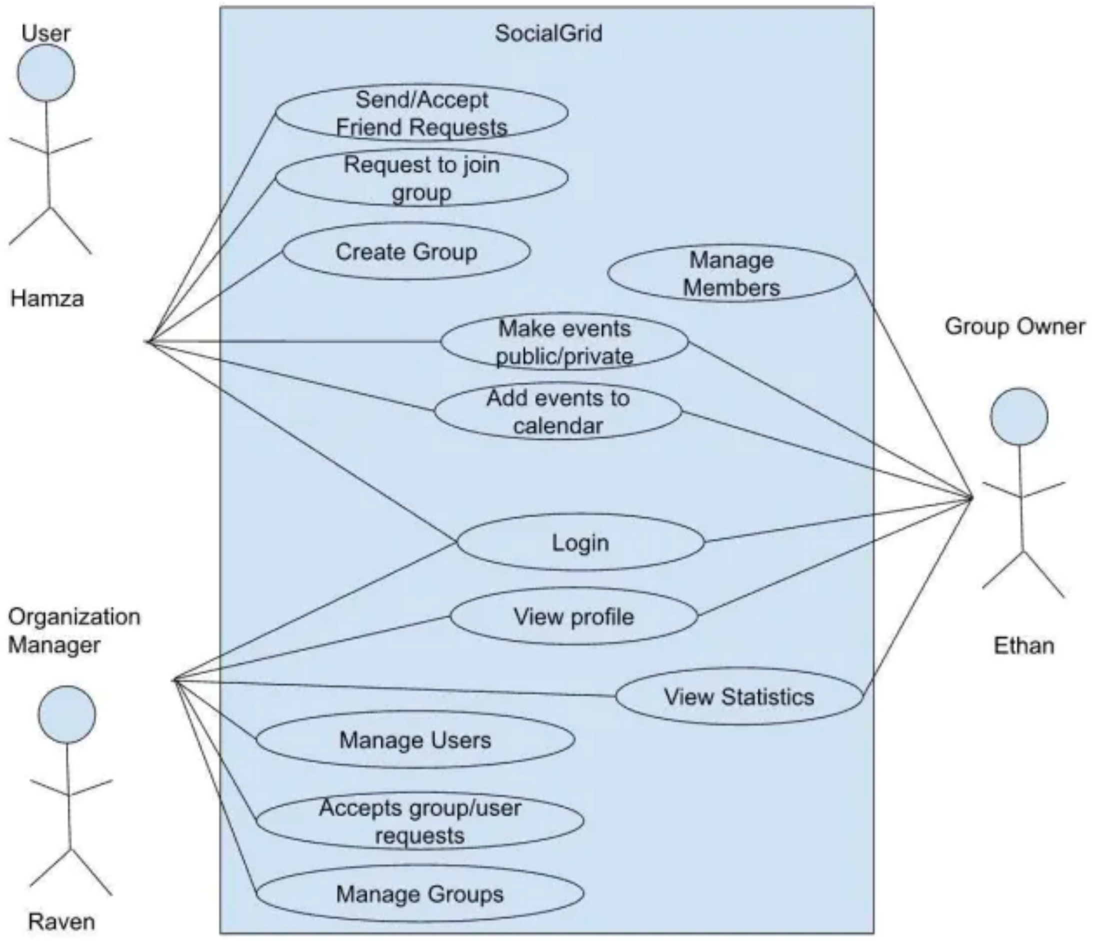
05/28/2024

Version 1

Ethan Mongelli, Hamza Ahmed, Raven Griffin

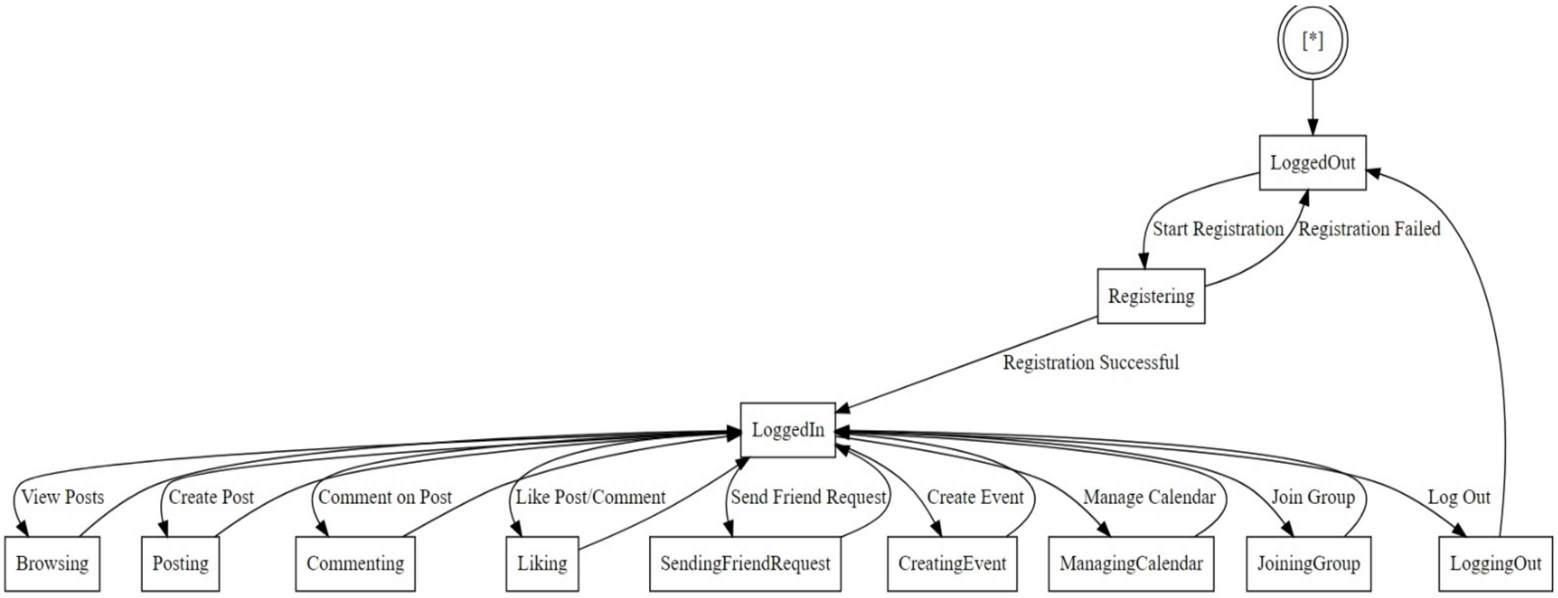
1. Project General Description (Raven)

The goal of the Grid web application is to connect users to groups via a calendar. It is a new social media platform that makes making plans easy. This app will allow users to create and join groups according to their interests.

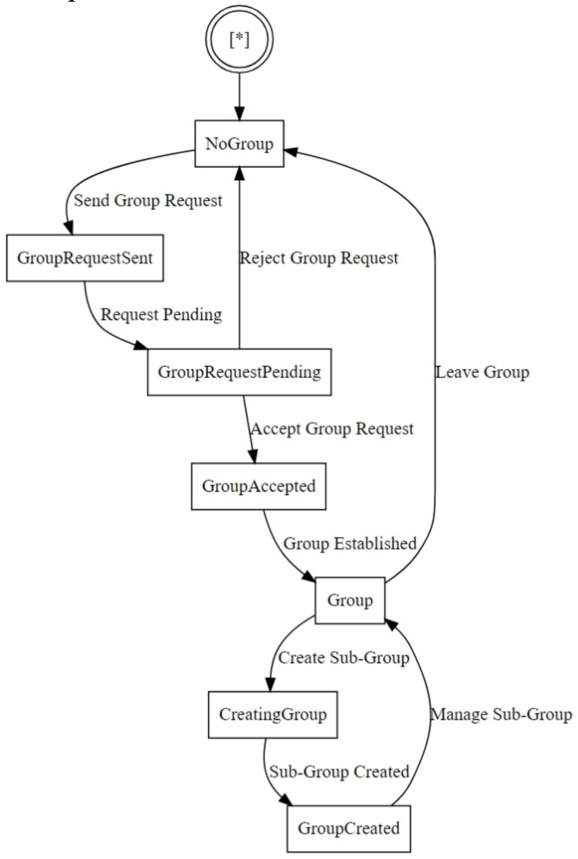


State Use Machine Diagrams (User, Group, Admin): (Originally Ethan, but then Hamza for these updated versions)

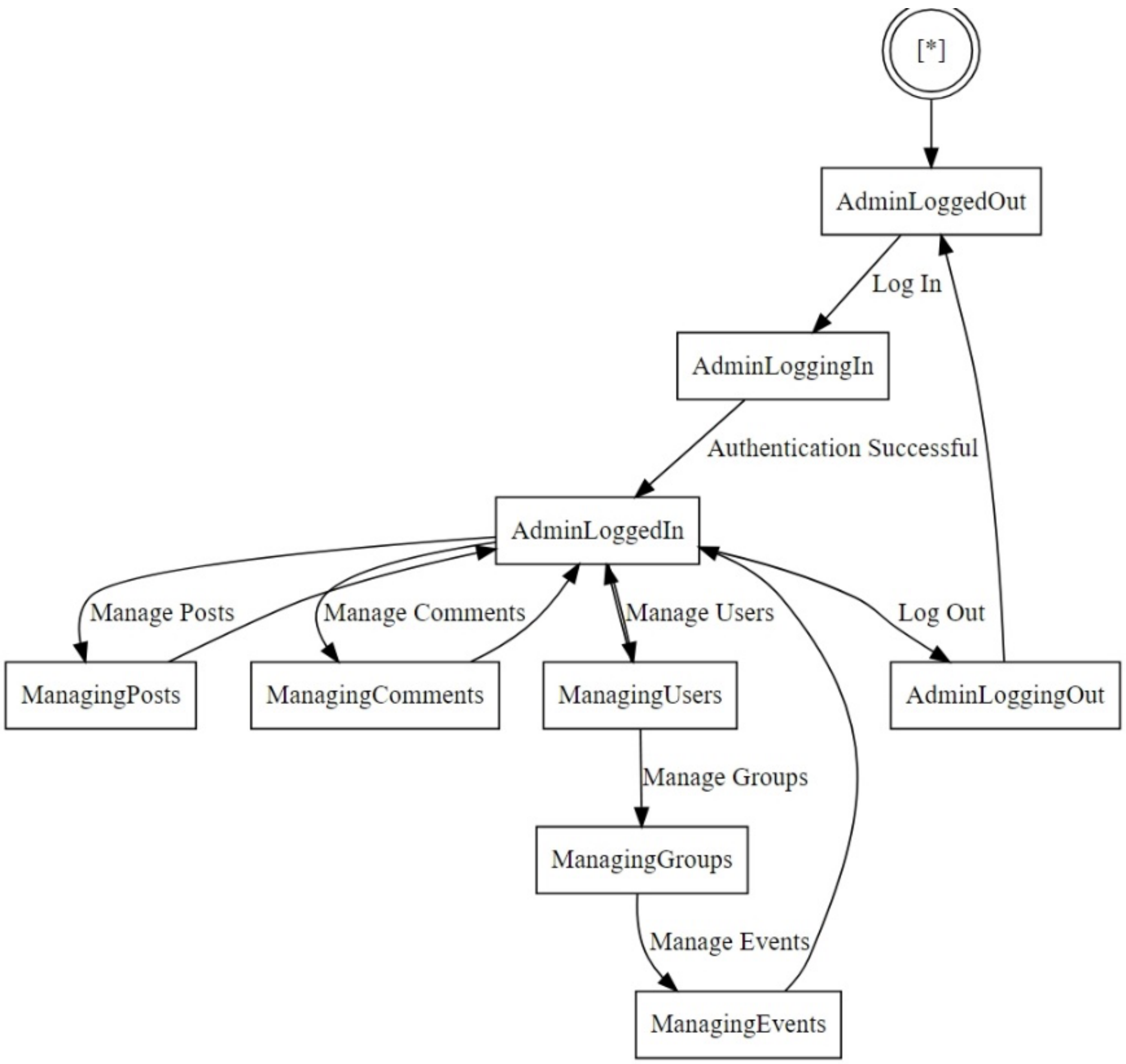
User:



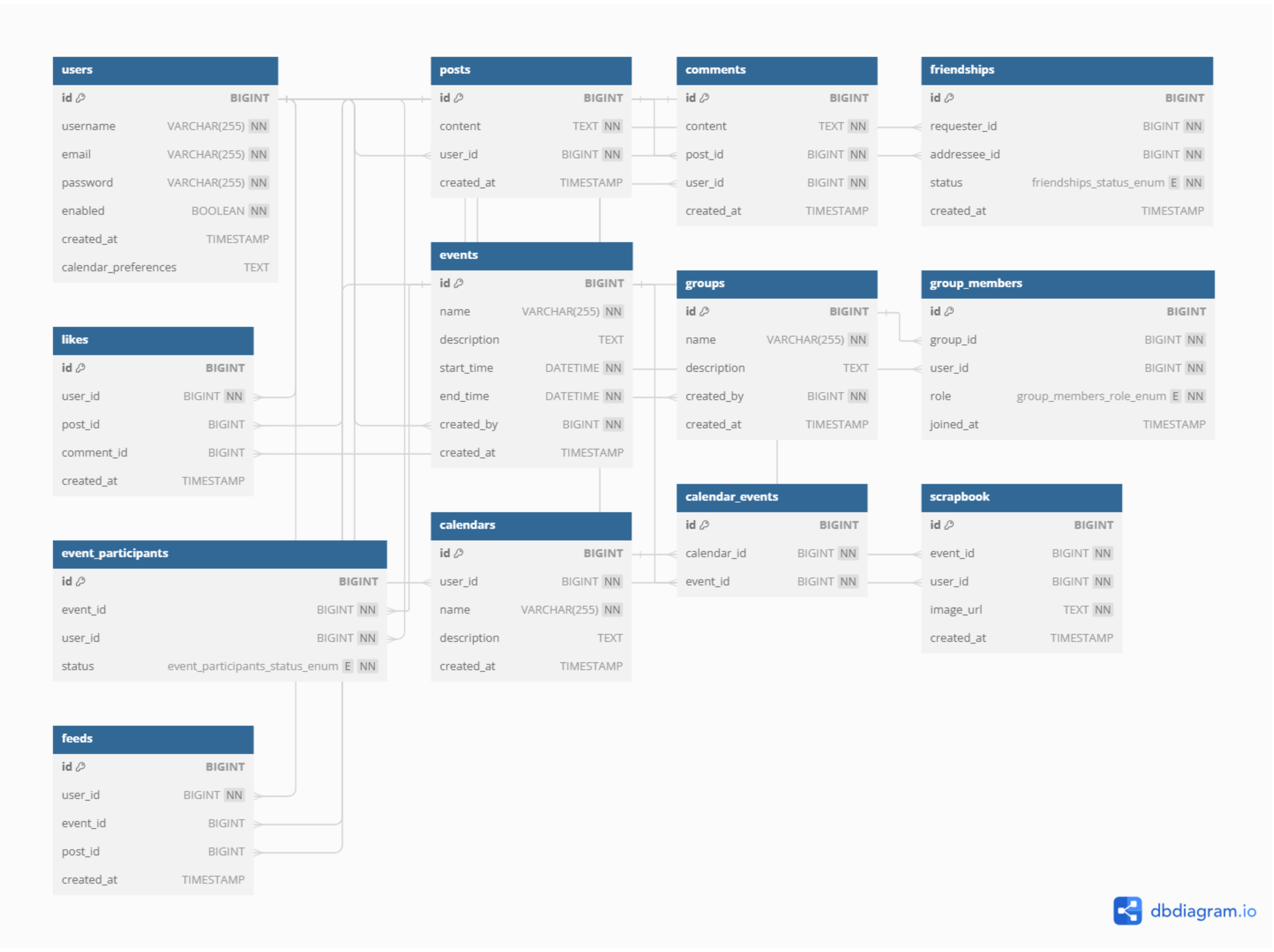
Group:



Admin:


[*]
AdminLoggedOut
Log In
AdminLoggingIn
Authentication Successful
AdminLoggedIn
Manage Posts
Manage Comments
Manage Users
Log Out
ManagingPosts
ManagingComments
ManagingUsers
AdminLoggingOut
Manage Groups
ManagingGroups
Manage Events
ManagingEvents

Database Schema (Hamza):



users

id	BIGINT
username	VARCHAR(255) NN
email	VARCHAR(255) NN
password	VARCHAR(255) NN
enabled	BOOLEAN NN
created_at	TIMESTAMP
calendar_preferences	TEXT

posts

id	BIGINT
content	TEXT NN
user_id	BIGINT NN
created_at	TIMESTAMP

comments

id	BIGINT
content	TEXT NN
post_id	BIGINT NN
user_id	BIGINT NN
created_at	TIMESTAMP

friendships

id	BIGINT
requester_id	BIGINT NN
addressee_id	BIGINT NN
status	friendships_status_enum E NN
created_at	TIMESTAMP

events

id	BIGINT
name	VARCHAR(255) NN
description	TEXT
start_time	DATETIME NN
end_time	DATETIME NN
created_by	BIGINT NN
created_at	TIMESTAMP

groups

id	BIGINT
name	VARCHAR(255) NN
description	TEXT
created_by	BIGINT NN
created_at	TIMESTAMP

group_members

id	BIGINT
group_id	BIGINT NN
user_id	BIGINT NN
role	group_members_role_enum E NN
joined_at	TIMESTAMP

likes

id	BIGINT
user_id	BIGINT NN
post_id	BIGINT
comment_id	BIGINT
created_at	TIMESTAMP

calendar_events

id	BIGINT
calendar_id	BIGINT NN
event_id	BIGINT NN

scrapbook

id	BIGINT
event_id	BIGINT NN
user_id	BIGINT NN
image_url	TEXT NN
created_at	TIMESTAMP

event_participants

id	BIGINT
event_id	BIGINT NN
user_id	BIGINT NN
status	event_participants_status_enum E NN

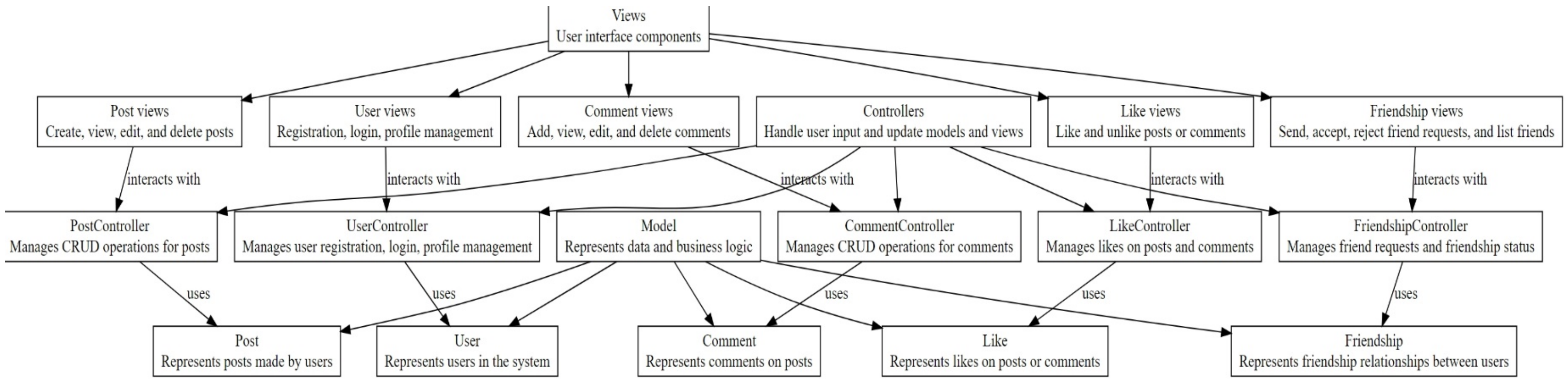
calendars

id	BIGINT
user_id	BIGINT NN
name	VARCHAR(255) NN
description	TEXT
created_at	TIMESTAMP

feeds

id	BIGINT
user_id	BIGINT NN
event_id	BIGINT
post_id	BIGINT
created_at	TIMESTAMP

Software Architecture-MVC (Hamza, Originally Ethan's part but I had to do this at the last minute)


Views
User interface components
Post views
Create, view, edit, and delete posts
User views
Registration, login, profile management
Comment views
Add, view, edit, and delete comments
Controllers
Handle user input and update models and views
Like views
Like and unlike posts or comments
Friendship views
Send, accept, reject friend requests, and list friends
interacts with
interacts with
interacts with
interacts with
interacts with
PostController
Manages CRUD operations for posts
UserController
Manages user registration, login, profile management
Model
Represents data and business logic
CommentController
Manages CRUD operations for comments
LikeController
Manages likes on posts and comments
FriendshipController
Manages friend requests and friendship status
uses
uses
uses
uses
uses
Post
Represents posts made by users
User
Represents users in the system
Comment
Represents comments on posts
Like
Represents likes on posts or comments
Friendship
Represents friendship relationships between users

Software Architecture: MVC Model Breakdown and Summary

The Model-View-Controller (MVC) architecture pattern is a widely used design pattern in software development. It divides an application into three interconnected components, each with distinct responsibilities: Models, Views, and Controllers.

Model

The Model represents the data and the business logic of the application. It directly manages the data, logic, and rules of the application.

- **User**: Represents users in the system.
- **Post**: Represents posts made by users.
- **Comment**: Represents comments on posts.
- **Friendship**: Represents friendship relationships between users.

Views

The Views are the components that display the data provided by the Model in a specific format. They are responsible for the presentation layer of the application.

- **User Views**:
  - Registration
  - Login
  - Profile management
- **Post Views**:
  - Create posts
  - View posts
  - Edit posts
  - Delete posts
- **Comment Views**:
  - Add comments

    - View comments
    - Edit comments
    - Delete comments
- **Friendship Views**:
    - Send friend requests
    - Accept friend requests
    - Reject friend requests
    - List friends

Controllers

Controllers act as intermediaries between Models and Views. They handle the user input, manipulate the data model, and update the view accordingly.

- **UserController**: Manages user registration, login, and profile management.
- **PostController**: Manages CRUD (Create, Read, Update, Delete) operations for posts.
- **CommentController**: Manages CRUD operations for comments.
- **FriendshipController**: Manages friend requests and friendship status.

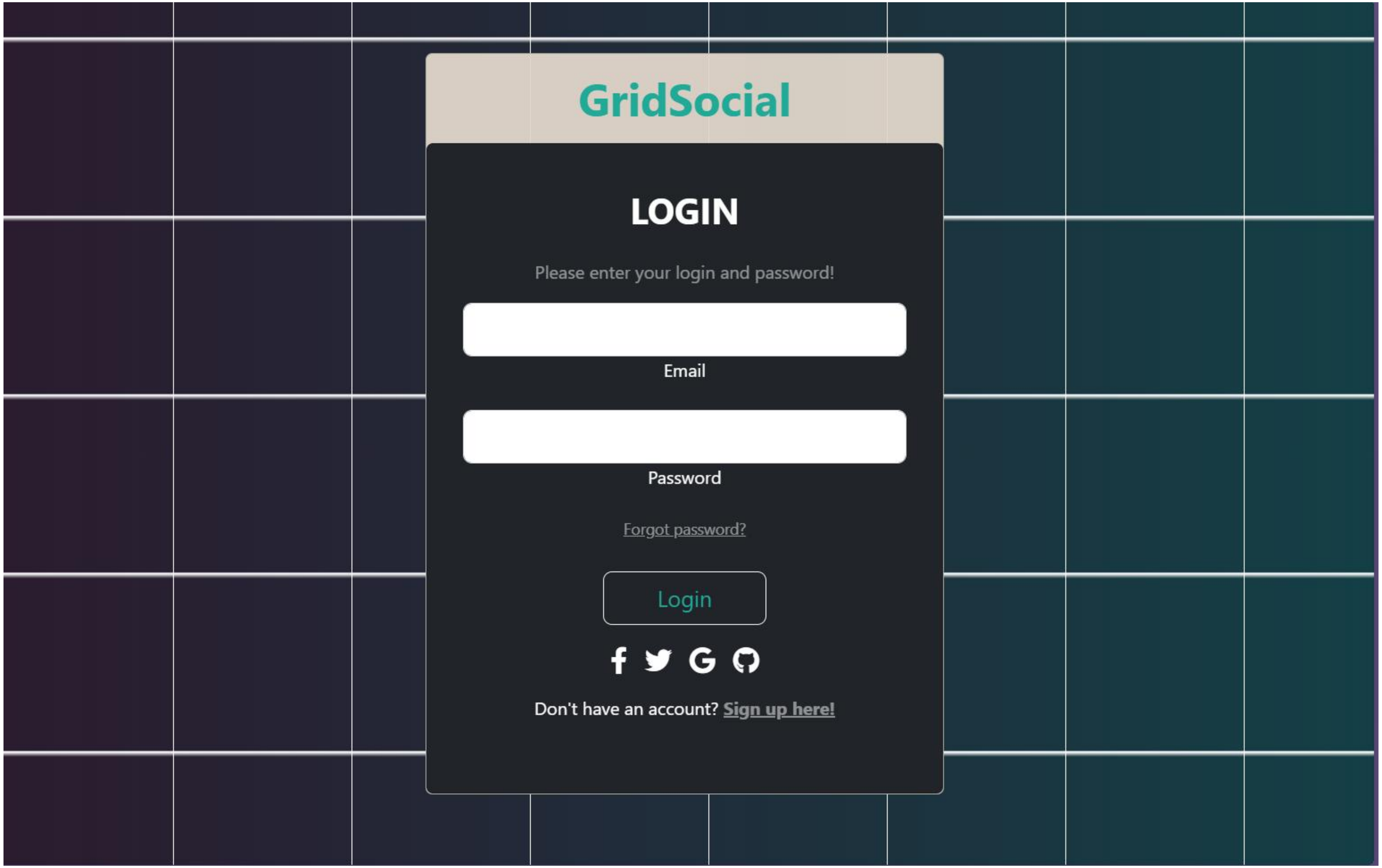
Summary

The MVC architecture separates an application into three main logical components:

1. **Models** handle the data and business logic.
2. **Views** are responsible for the user interface and presentation.
3. **Controllers** manage the interaction between the Model and the View.

In this particular implementation:

- **Models** represent core entities such as users, posts, comments, friendships, and likes.
- **Views** provide the user interface for various operations related to these entities, such as managing user profiles, posts, comments, friendships, and likes.
- **Controllers** handle the business logic and user input, performing operations on the models and updating the views accordingly.



GridSocial

LOGIN

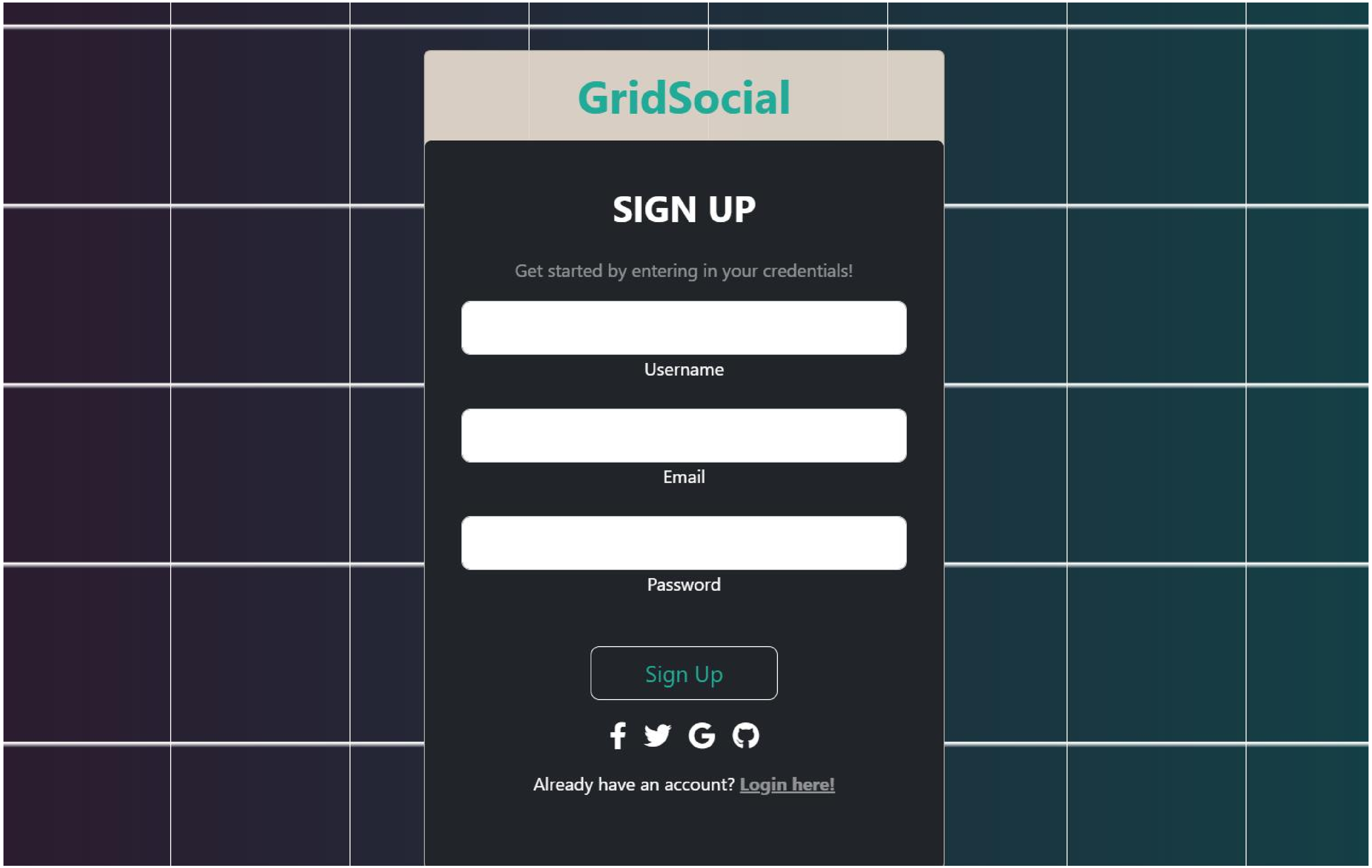
Please enter your login and password!

Email

Password

Forgot password?

Don't have an account? Sign up here!



GridSocial

SIGN UP

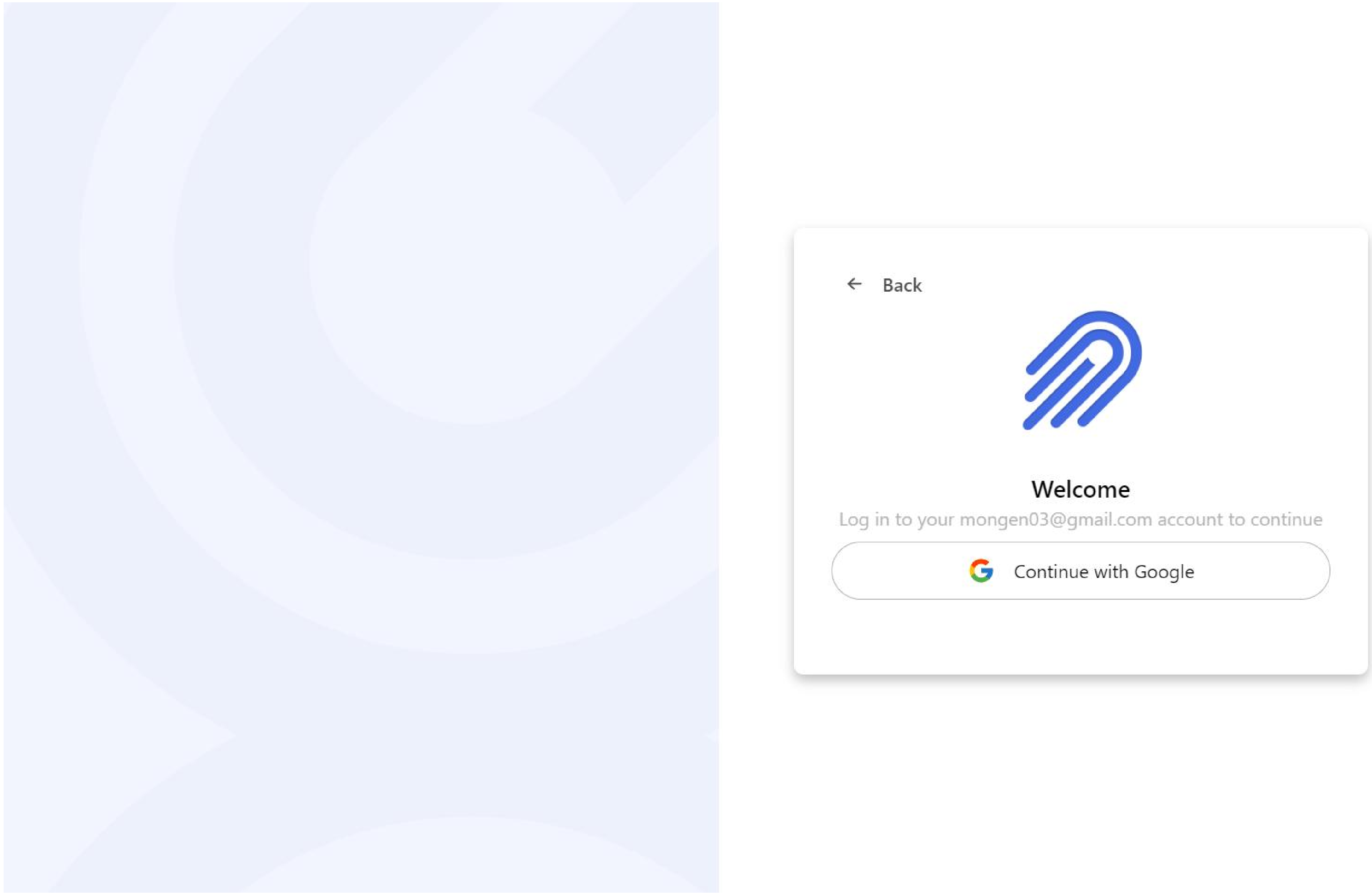
Get started by entering in your credentials!

Username

Email

Password

Already have an account? Login here!

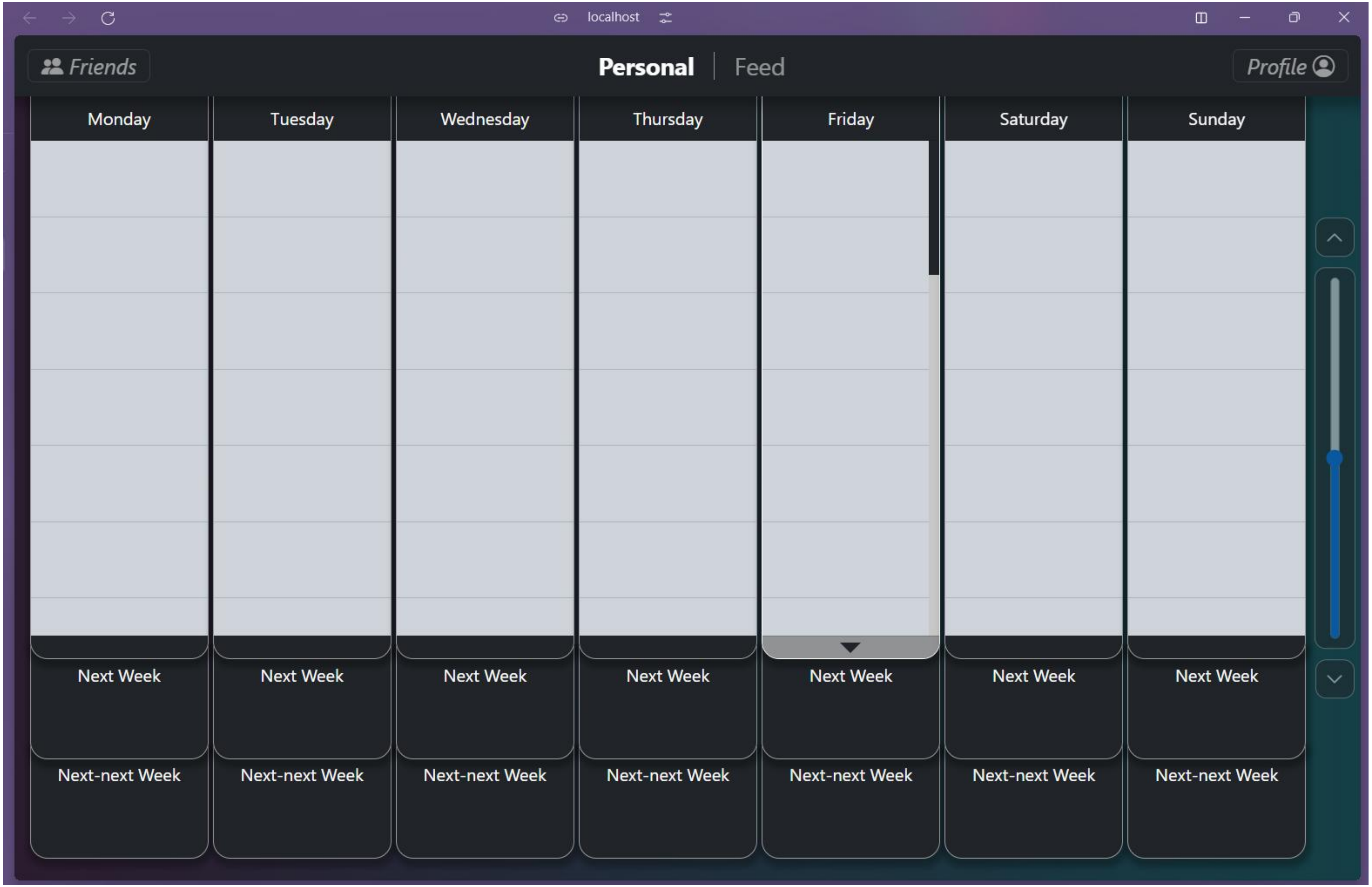


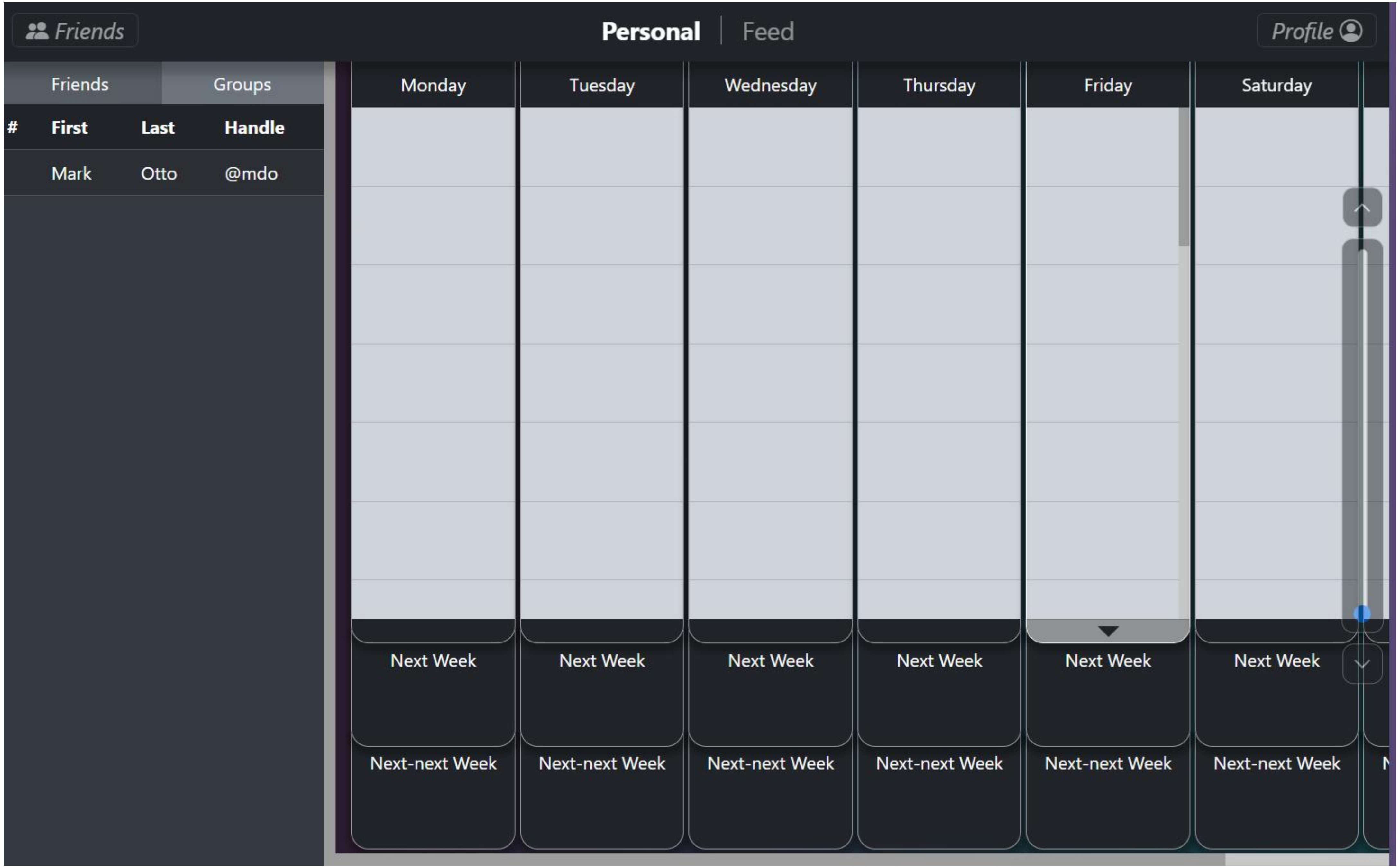
← Back

Welcome

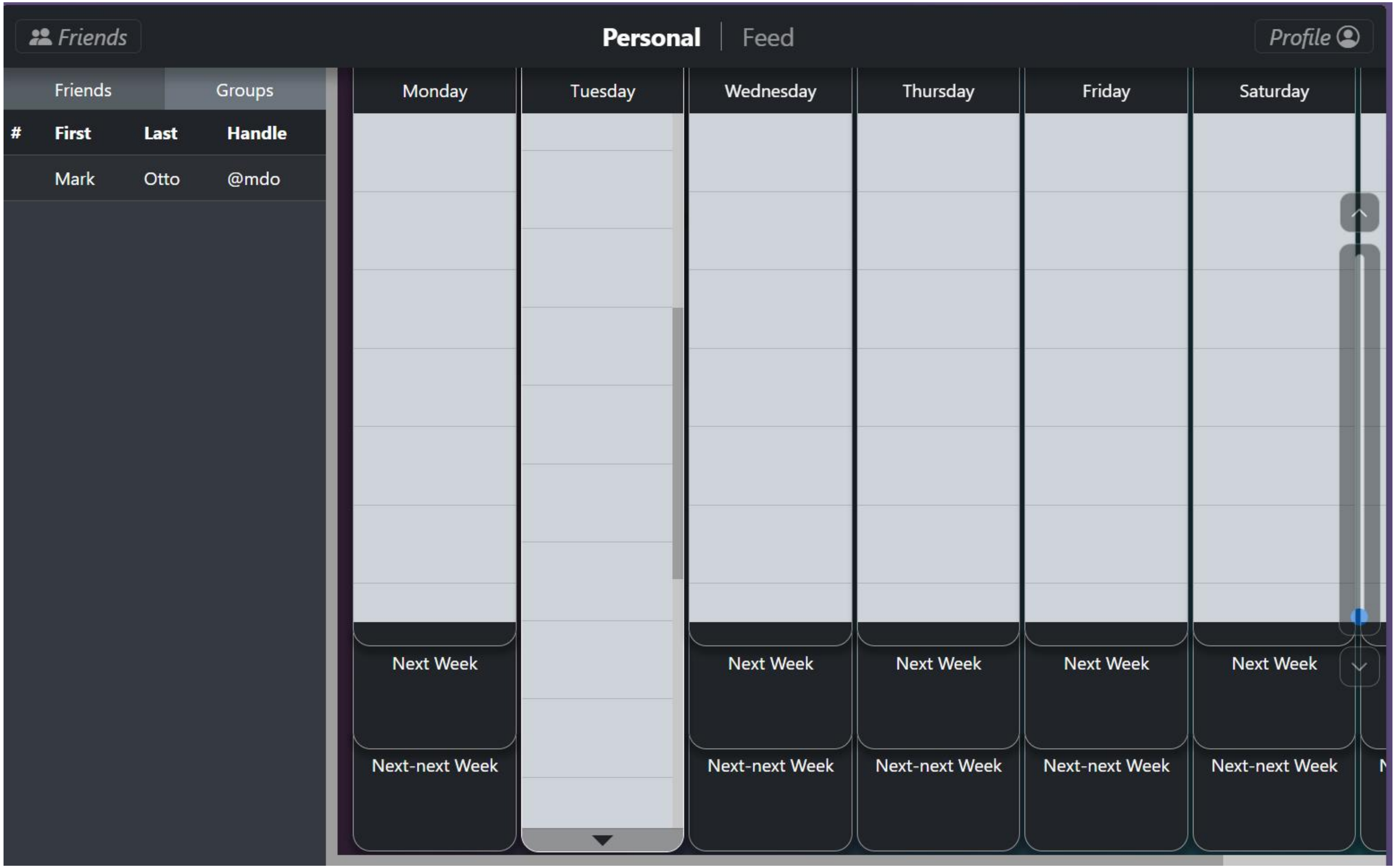
Log in to your mongen03@gmail.com account to continue

Continue with Google


localhost
Friends
Personal
Feed
Profile
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Next Week
Next Week
Next Week
Next Week
Next Week
Next Week
Next Week
Next-next Week
Next-next Week
Next-next Week
Next-next Week
Next-next Week
Next-next Week
Next-next Week



#	First	Last	Handle
	Mark	Otto	@mdo



Friends

Personal | Feed

Profile

Friends	Groups

#	First	Last	Handle
	Mark	Otto	@mdo