Web Application README

Overview

This Flask-based web application provides a user-friendly platform for event management. With features like user authentication, event creation, update, and visualization on a calendar, it's designed to enhance productivity and organization. The application leverages Flask extensions such as Flask-Login for handling user sessions, Flask-Migrate for database migrations, and Flask-Bcrypt for secure password hashing.

Features

- User Authentication: Sign up and Sign in.
- Event Management: Users can add, update, or delete events.
- Calendar View: Events are displayed on a monthly calendar, with the ability to view details for each day.

Technology Stack

- Flask: A micro web framework written in Python.
- Flask Extensions: Flask-Login, Flask-Migrate, Flask-Bcrypt.
- Database: SQLAlchemy ORM for data management.

Deployment on Render.com

Prerequisites

- A Render account.
- application source code hosted on a GitHub.

Steps

Log In to Render
- Navigate to Render.com and sign in.

Create a New Web Service
- From your Render dashboard, select the "New +" button, then choose "Web Service".

Connect Your Repository
- Connect to the Git provider and select the repository that contains the Flask application.

Configure Your Web Service
- Environment: Select `Python`.
- Branch: Choose the branch you want to deploy.
- Build Command: Enter the command to install your dependencies, typically `pip install -r requirements.txt`.
- Start Command: Use `gunicorn app:app` to start the Flask application.
- Environment Variables: Add all necessary environment variables, such as `SECRET_KEY`, `DATABASE_URL`, etc.

  DATABASE_URL is the internal URL from postgres instance created on render , make sure to replace postgres to postsql

Deploy
- Click the "Create Web Service" button. Render will automatically deploy your application.

Accessing the Application

- Once the deployment is successful, Render will provide a `.onrender.com` subdomain to access your web app.
- Navigate to the provided URL to access your Flask application live on Render.

Local Development

To run the application locally, ensure you have Python installed and follow these steps:

Clone the Repository

- `git clone <repository-url>`

Create a Virtual Environment

- `python -m venv venv`

Activate the Virtual Environment

- Windows: `venv\Scripts\activate`
- macOS/Linux: `source venv/bin/activate`

Install Dependencies

- `pip install -r requirements.txt`

Run the Application

- `flask run`

Visit `http://127.0.0.1:5000` in your web browser to access the application.

Setting Up PostgreSQL on Render.com

This application uses PostgreSQL as its database. Render allows you to set up and connect to a PostgreSQL database.

Create a PostgreSQL Database

Navigate to Your Render Dashboard

- Log in to your Render account and go to the dashboard.

Add a New Database

- Click on the "New +" button at the top of the dashboard, then select "Database".

Configure Your Database

- Name: Give your database a unique name.
- Region: Select the region closest to your web service to reduce latency.
- Render will automatically provision a PostgreSQL database for you.

Database Settings

- Once your database is created, Render will provide you with an internal connection string under the "Internal Connection String" section. This URL is used to connect your web application to the database.

Configure `DATABASE_URL`

- The application uses the `DATABASE_URL` environment variable to connect to the PostgreSQL database.
- The internal connection string provided by Render needs a slight modification before use:
    - Replace `postgres://` with `postgresql://` in the URL.
- Example:
    - If Render provides `postgres://user:password@internal-database-host:5432/database_name`, modify it to `postgresql://user:password@internal-database-host:5432/database_name`.

Setting the `DATABASE_URL` in Your Web Service

Go to Your Web Service Settings on Render
- Navigate to the settings of your web service that hosts your Flask application.

Add the `DATABASE_URL` Environment Variable
- In the "Environment" section, click on "Add Environment Variable".
- Key: Enter `DATABASE_URL`.
- Value: Paste the modified internal connection string (`postgresql://...`).

Redeploy Your Application
- After adding the `DATABASE_URL`, redeploy your application for the changes to take effect.

Connecting to Your Database with pgAdmin4

Launch pgAdmin4:
- Open pgAdmin4 on your computer. If you don’t have it installed, you can download it from the official pgAdmin website.

Add a New Server:
- Right-click on ‘Servers’ in the left sidebar and select ‘Create’ > ‘Server…’.
- In the ‘Create - Server’ window that opens, go to the ‘General’ tab and give your server a name.

Configure Connection:
- Switch to the ‘Connection’ tab.
- In the ‘Hostname/Address’ field, enter the external URL of your PostgreSQL instance.
- Fill in the ‘Username’ and ‘Password’ fields with your database credentials.
- Optionally, adjust the port if your database uses a non-default port (the default PostgreSQL port is 5432).
- Click ‘Save’. pgAdmin will attempt to connect to your PostgreSQL database.

Creating the Database Tables

Once connected, you can create the necessary tables for your application. Here's how to create the `users`, `events`, and `last_viewed` tables:

Select Your Database:

- In the pgAdmin sidebar, navigate to your server, then Databases. Select the database you'll be working with.

Open the Query Tool:

- Right-click on your database and choose 'Query Tool…'.

Create the Tables:

- Copy and paste the SQL commands below into the Query Tool's editor.

```
-- Create Users Table

CREATE TABLE users (

 id SERIAL PRIMARY KEY,

 email VARCHAR 120  UNIQUE NOT NULL,

 password_hash VARCHAR 128  NOT NULL

);


-- Create Events Table

CREATE TABLE events (

 id SERIAL PRIMARY KEY,

 user_id INTEGER NOT NULL,

 title VARCHAR 100  NOT NULL

 date DATE NOT NULL,

 start_time TIMESTAMP WITHOUT TIME      NOT NULL,

 end_time TIMESTAMP WITHOUT TIME      NOT NULL,

 FOREIGN KEY (user_id) REFERENCES users (id));
```

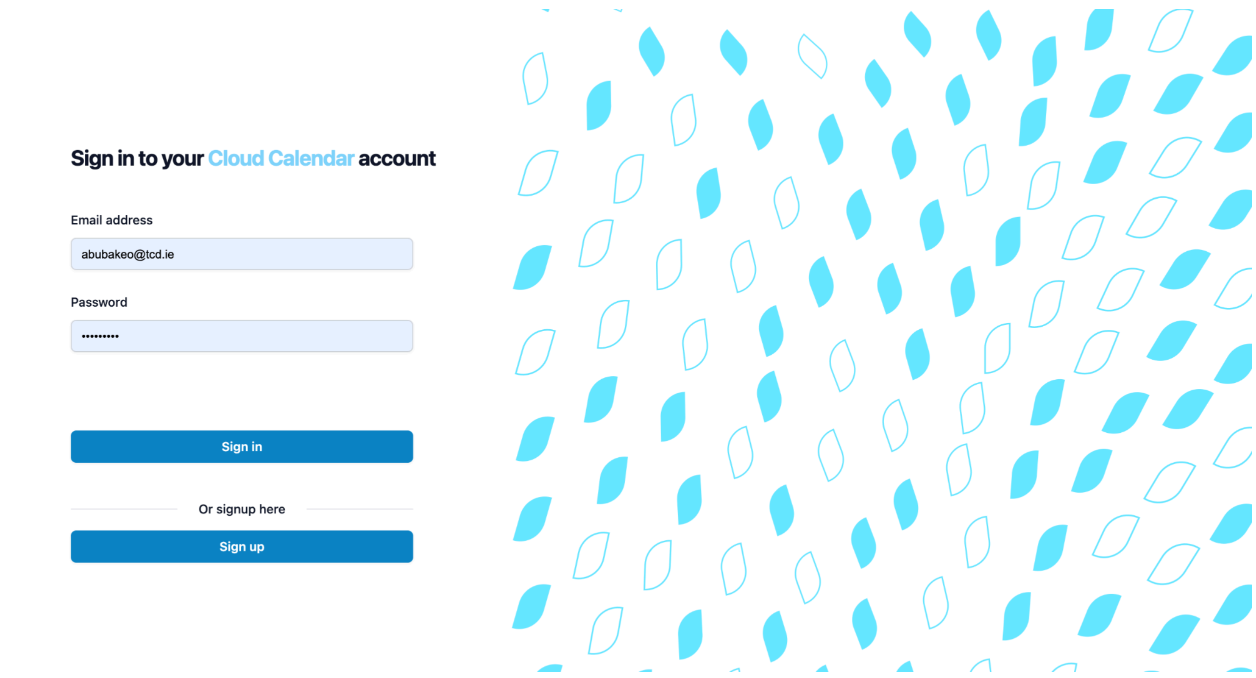
```sql
-- Create LastViewed Table

CREATE TABLE last_viewed (

 id SERIAL PRIMARY KEY,

 user_id INTEGER NOT NULL,

 month INTEGER NOT NULL,

 year INTEGER NOT NULL,

 FOREIGN KEY (user_id) REFERENCES users (id)

);
```

Execute the Query:

- Click the ‘Execute/Refresh’ button or press F5 to run the commands.
- You should see messages indicating that the tables have been successfully created.

By following these steps, you will have created the necessary tables in your database for the Flask application to function correctly.

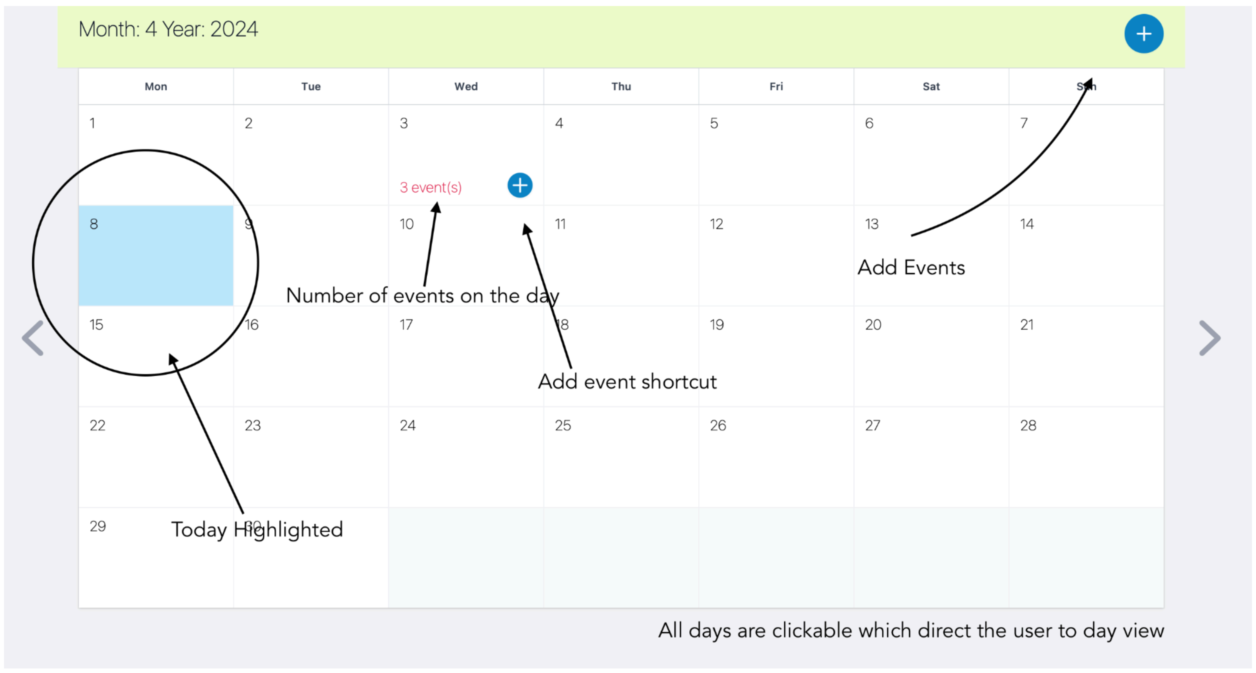
UI design and navigation



Sign in to your Cloud Calendar account

Email address: abubakeo@tcd.ie

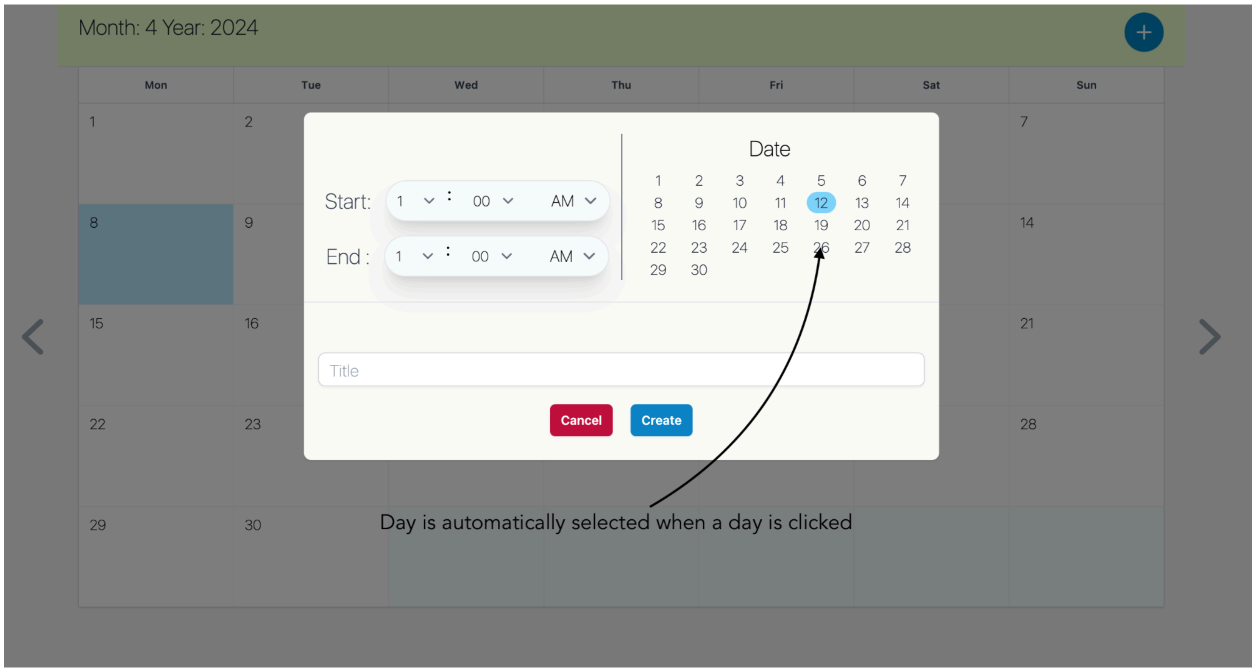
Password: •••••••••



Month: 4 Year: 2024

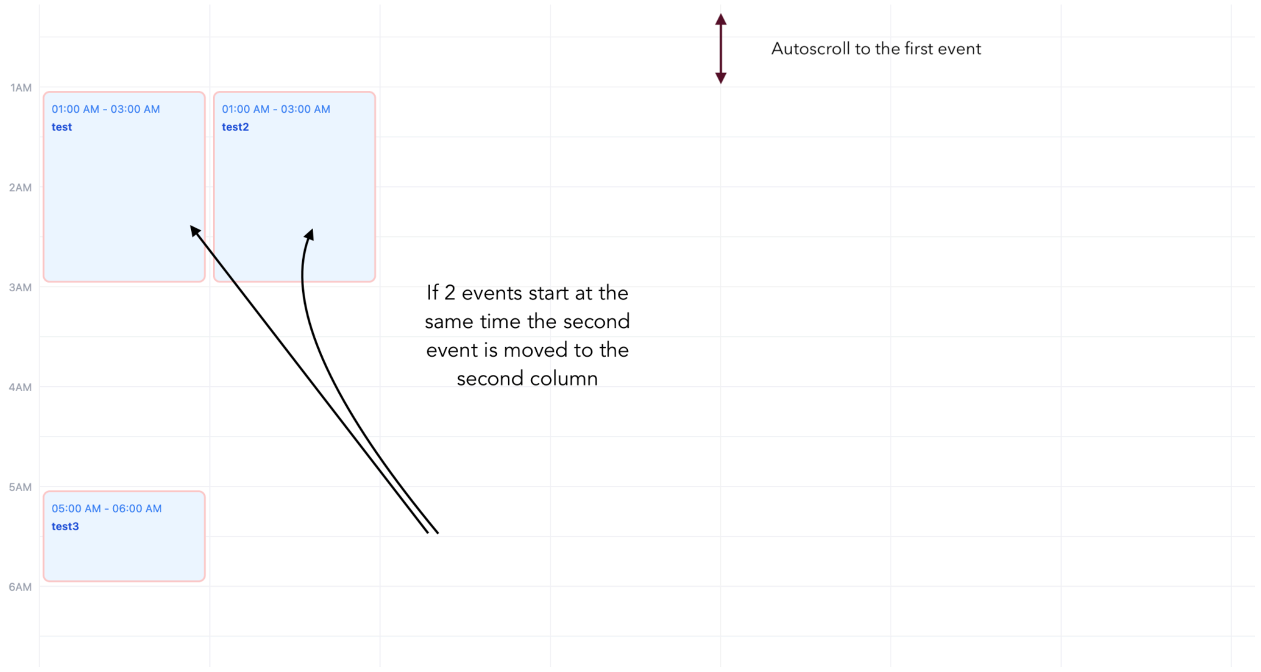
Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3 3 event(s)	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

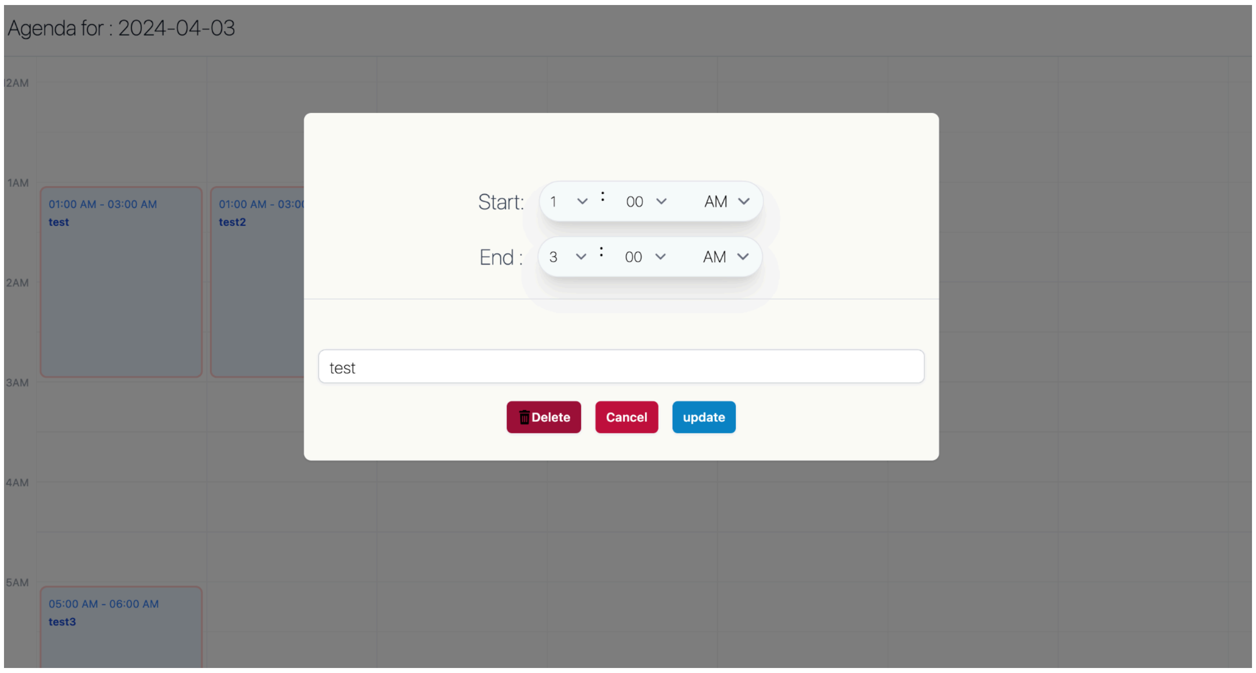
All days are clickable which direct the user to day view



Calendar View UI

- Date Navigation: The calendar provides a clear view of the current month and year at the top, which aids in orienting the user.
- Daily Events: Days with scheduled events show a numeric indicator, informing the user at a glance which dates are populated with activities. This immediate visual cue is helpful for planning and reviewing the user's schedule.
- Interactive Elements:
  - Adding Events: The plus symbol (+) on the upper right of the calendar suggests a quick and intuitive way to add new events.
  - Day View: Each calendar date is interactive, allowing the user to click on a day to view its detailed schedule. This feature is indicated by the change in background color upon hovering or selecting a date.
- Visual Highlights: The current day is highlighted, which helps users quickly find the current date.
- Navigation Arrows: Users can navigate between months using the arrows on the side, providing an easy way to traverse the calendar.


1AM
2AM
3AM
4AM
5AM
6AM
01:00 AM - 03:00 AM
test
01:00 AM - 03:00 AM
test2
05:00 AM - 06:00 AM
test3
Autoscroll to the first event
If 2 events start at the same time the second event is moved to the second column


Agenda for : 2024-04-03

Day View UI

- Concurrent Events: Events are presented in blocks along a timeline, with overlapping or concurrent events arranged in adjacent columns. This layout helps to visualize potential scheduling conflicts and manage time more effectively.

- Auto-scrolling: Upon opening the day view, the interface automatically scrolls to the first event of the day, sparing users from manual scrolling and providing immediate access to their upcoming activities.

- Modifying Events: events can be edited by clicking on them, which brings up a modal window with the event details.Users have the option to delete the event or cancel any changes. If the event details are modified, users can save the changes by clicking the 'update' button.

Resources used

- Tailwind Components: The UI design incorporates Tailwind CSS for styling, utilizing its utility-first components to achieve the clean and responsive design .

- fffuel.co : This resource has been utilized for the background image for the login page.

Additional Links

- GitHub Repository: For more details on the application's codebase, updates, and version control, visit the GitHub repository at github.com/omarbaker8/calendar.
- Live Application: To see the application in action and test its functionality, access the live version hosted on Render at calendar-1-qcai.onrender.com.