Super Awesome Team

TaskTogether

By: Sophia Turnbow, Alex Carl, Ziqian Zhang (Andy), Gavin Landry, Jincheng Xu (Mike)

The idea behind TaskTogether

What application combines a personal planner, group planner, and social media? None that we could find!

And the idea for TaskTogether was born...

Our vision is a single application that includes:

- Personal Planner
- Group Planner
- Social Media

This way, people can get motivated to complete their goals and succeed TOGETHER!

Who uses our site?

TaskTogether is geared towards students and young professionals who want a place to organize their tasks, collaborate with people, and share their progress.

Main features

- Set calendar events on a specific date
- Create tasks for a personal planner, share it with friends
- Collaborate with group planners
- Post updates about your daily life

Home

My Tasks

Group Tasks

Social

Calendar

Client Side

Located on the Home Page

Users can add events by clicking on the date they wish to add them to.

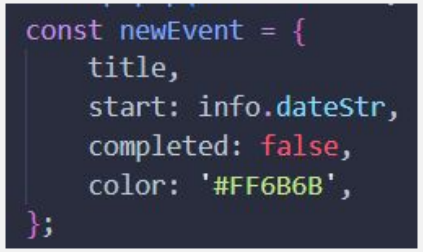
Events can be deleted by clicking on them directly.

Events for today's date are neatly displayed where they can be marked as complete.

FullCalendar

Client Side

Made using the FullCalendar API



```
const newEvent = {
    title,
    start: info.dateStr,
    completed: false,
    color: '#FF6B6B',
};
```

Events are handled as json objects.

Connected to php using AJAX and the Fetch API

https://fullcalendar.io/

Server Side

Requires users to be signed in. Once signed in, the users events will be fetched from the database, then displayed.

Events are tied to the user id.

Creating, Sharing, & Posting Tasks.

Client Side

Once a user signs in, they can navigate to the My Tasks page. From there, click on Add Planner and start adding tasks to it.

To share a planner, the owner clicks on Settings → Share, and that copies a link that they can send to their friends. This goes into the Group Tasks page.

When making a planner public, this adds it to their posts on the Social page.

Law of Proximity: *tasks for one planner are grouped together*

Server Side

When the user creates a task, it goes into the task table. Then, when they add blocks to that schedule it goes into a different table called blocks which has a foreign key from the schedule linking them.

When the schedule is shared with someone, it takes the id from the link and the puts the person who the schedule was shared with and the schedule id in a table so they can do basic edits of the schedule.

Social Page.

similar to Twitter/X!
***Jakob's Law** of familiarity*

Client Side

When a user visits the Social page, they can see the most recent posts. If they are signed in, they can create a post, comment on, and like other posts.

If a post on the page belongs to them, it can be edited or deleted.

All posts a users creates can also be viewed in their profile page.

Server Side

If the user is signed in, they can create a post.

When they make a post, it gets stored in the database.

When they do a GET request for all the posts. In the back end we compare the profile id of the person who posted to the person who is signed in to make it so they can edit that post.

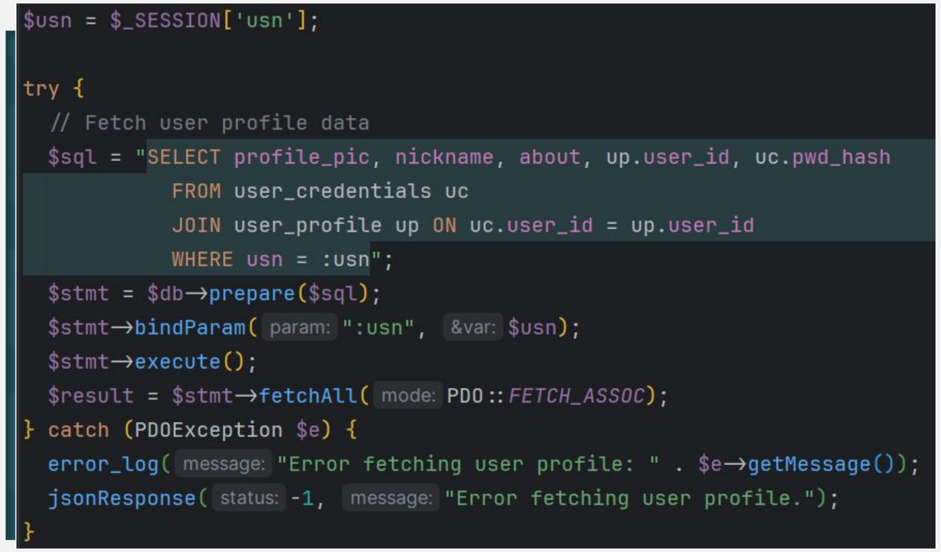
Profile Page.

Client Side

You can edit your profile in many ways:

- Set your nickname (display name)
- Write a bio
- Choose a profile picture
- Change username
- Change password

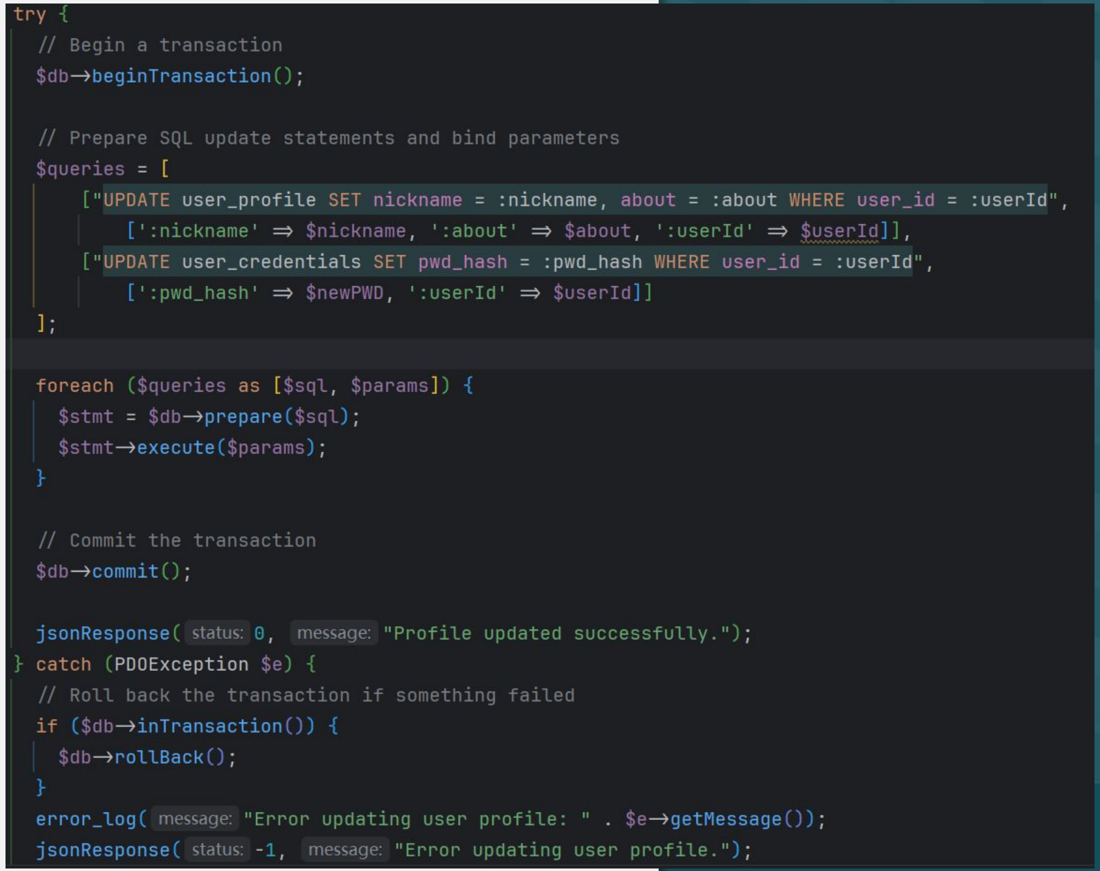
Server Side



```
$usn = $_SESSION['usn'];

try {
  // Fetch user profile data
  $sql = "SELECT profile_pic, nickname, about, up.user_id, uc.pwd_hash
            FROM user_credentials uc
            JOIN user_profile up ON uc.user_id = up.user_id
            WHERE usn = :usn";
  $stmt = $db→prepare($sql);
  $stmt→bindParam( param: ":usn",  &var: $usn);
  $stmt→execute();
  $result = $stmt→fetchAll( mode: PDO::FETCH_ASSOC);
} catch (PDOException $e) {
  error_log( message: "Error fetching user profile: " . $e→getMessage());
  jsonResponse( status: -1,  message: "Error fetching user profile.");
}
```

username stored in session to retrieve user id



```
try {
  // Begin a transaction
  $db→beginTransaction();

  // Prepare SQL update statements and bind parameters
  $queries = [
      ["UPDATE user_profile SET nickname = :nickname, about = :about WHERE user_id = :userId",
          [':nickname' ⇒ $nickname, ':about' ⇒ $about, ':userId' ⇒ $userId]],
      ["UPDATE user_credentials SET pwd_hash = :pwd_hash WHERE user_id = :userId",
          [':pwd_hash' ⇒ $newPWD, ':userId' ⇒ $userId]]
  ];

  foreach ($queries as [$sql, $params]) {
    $stmt = $db→prepare($sql);
    $stmt→execute($params);
  }

  // Commit the transaction
  $db→commit();

  jsonResponse( status: 0,  message: "Profile updated successfully.");
} catch (PDOException $e) {
  // Roll back the transaction if something failed
  if ($db→inTransaction()) {
    $db→rollBack();
  }
  error_log( message: "Error updating user profile: " . $e→getMessage());
  jsonResponse( status: -1,  message: "Error updating user profile.");
```

- When we create a user, an associated user profile is also created

Demo Time.

Thank You

Any questions?