SW Engineering CSC648-848-05 Summer 2024

TeamMate
Personnel Management
Team 05 - Concat

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Frontend Dev
		Backend Lead
		Backend Dev
		Database Lead
		Database Dev
		Github Master

Milestone 4
July 30 2024

Version History

Revision	Date Submitted
M1	13 June 2024
M1V1	20 June 2024
M1V2	09 July 2024
M2V1	09 July 2024
M2V2	23 July 2024
M3V1	23 July 2024
M3V2	30 July 2024
M4V1	30 July 2024

Table of Contents

Product Summary

TeamMate is a comprehensive personnel management solution designed for small to mid-sized groups, such as non-profits and community organizations. It consolidates essential management functions into a single, user-friendly platform, enabling easy registration, member tracking, event management, and award issuance.

TeamMate's standout and superior feature is its Automated Recognition and Progress Tracking System. This system automatically tracks member participation, awards achievements, and updates progress, providing timely recognition and broad overviews of individual and group accomplishments. This holistic view is valuable for performance reviews, planning, setting future goals along with motivating the members of the organization.

Discord focuses on communication and lacks recognition and structured progress tracking, **Airtable** and **Google sheets** require extensive manual setup to achieve similar functionality, whereas **TeamMate** offers an out-of-the-box solution that simplifies administrative tasks. **PERSCOM** provides good tracking but demands more manual effort for recognition, while **Rippling HCM** caters to larger organizations with complex and costly systems, making it less suitable for smaller groups.

Committed Functions

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- An owner can assign other members to be admins.
- An organization shall have a roster page that shows all members in the organization separated by groups.
- Users shall click on any member on the roster to access their member page.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have a title.
- Awards shall have a description.
- Admins shall directly assign awards.
- Award instances shall have a date received.
- Award instances shall state the issuing member.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.
- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.

Events

- Admins shall create events.
- Group leaders shall create events.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall be able to join an event.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

Application

http://ec2 compute-1.amazonaws.com:3000/

Usability Test Plan

Use/Test Case 1 (Organization Creation)

Objective:
Create an organization. While conducting this test, the user will also go through the registration and login process. This feature is being tested to determine if the user is intuitively able to navigate the home page and sidebar of the dashboard.

Description:
The horizontal prototype already supports these features, the user will start from the website's home page. The intended user is a first time user of the website, with no prior context as to what it is. We are measuring the time and mouse movements of the user, to see how long it takes the user to complete the test, and where all they navigate while completing the test, in order to determine if the website design makes intuitive sense to a first time user.
http://ec2 compute-1.amazonaws.com:3000/

Use/Test Case 2 (Group Creation)

Objective:
Create a group. This feature is being tested to determine if the user is intuitively able to navigate the header and corresponding pages intuitively.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a new user that has just created an organization. We are measuring the time taken and mouse movements of the user to determine if they can quickly navigate to the required pages intuitively, to validate our dashboard design.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 3 (Award Creation)

Objective:

Create an award. This feature is being tested to determine if the user is able to find the awards page and create an award.

Description:

The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created an organization. We are measuring the time taken and mouse movements & page navigation to determine if they are able to figure out that awards are part of the recognitions tab, and can use the creation form to make an award instead of certification.

http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 4 (Event Creation)

Objective:

Create an event. This feature is being tested to determine if the user is able to navigate to the events page and follow through the event creation form.

Description:

The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created and organization. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate from the dashboard to the event page and the events creation form and complete it.

http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 5 (Member Operations)

Objective:
Add a member. This feature is being tested to determine if the user can figure out how to add members to their organization and assign them attributes.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after having created a few attributes. The intended user is a user that has explored the dashboard for some time and created various attributes. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate to the members page and invite another user to their organization.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Test Table

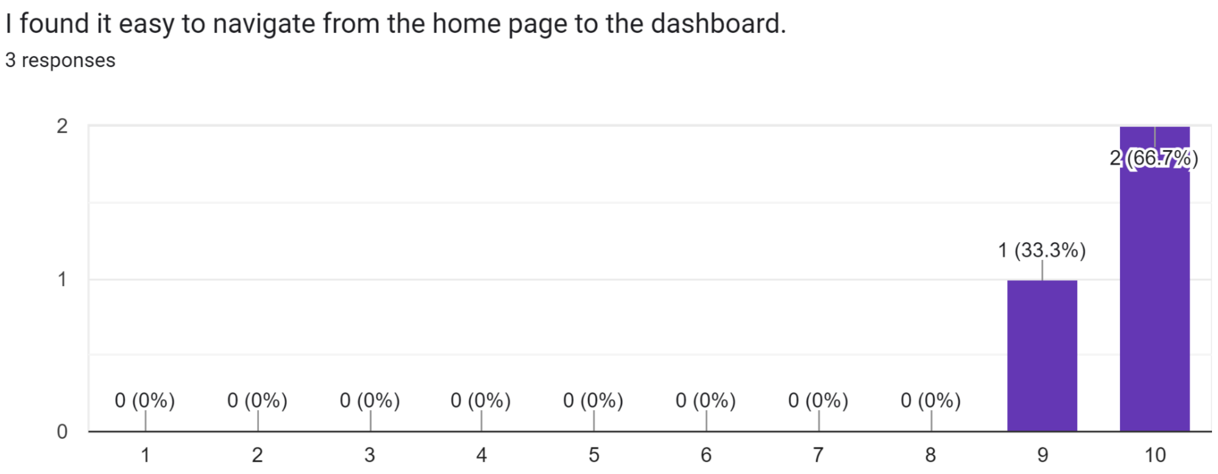
Test/Use Case	% Completed	Efficiency	Errors	Comments
Org Creation	100%	4-5 Minutes	None	Older users unable to figure out that the create bar on the sidebar is how you make an org. Inflated efficiency due to users coming up with registration info.
Group Creation	100%	<1 Minute	None	Creation complete.
Award Creation	100%	<1 Minute	None	Creation complete.
Event Creation	100%	<1 Minute	None	Creation complete.
Member Add	100%	1-2 Minutes	None	It's not clear how to find a user's ID to add into the org. Can also add duplicate members to org.

Questionnaire

1. I found it easy to navigate from the home page to the dashboard.
2. I found it easy to create an account and login.
3. I was able to intuitively navigate the dashboard to create an organization.
4. The dashboard header was difficult to understand.
5. The group's page layout confused me.
6. I was able to intuitively navigate the dashboard to create a group.
7. I struggled to find the awards page.
8. The layout of the awards page was consistent with the other pages.
9. I was able to intuitively navigate the dashboard to create an award.
10. The event page's layout was consistent with the rest of the application.
11. The event details page layout contained all the information I wanted it to.
12. I was able to intuitively navigate the dashboard to create an event.
13. The layout of the members page made intuitive sense.
14. Using the add button in the members page was confusing.
15. I was able to intuitively navigate the dashboard to add a member.

https://forms.gle/

Responses

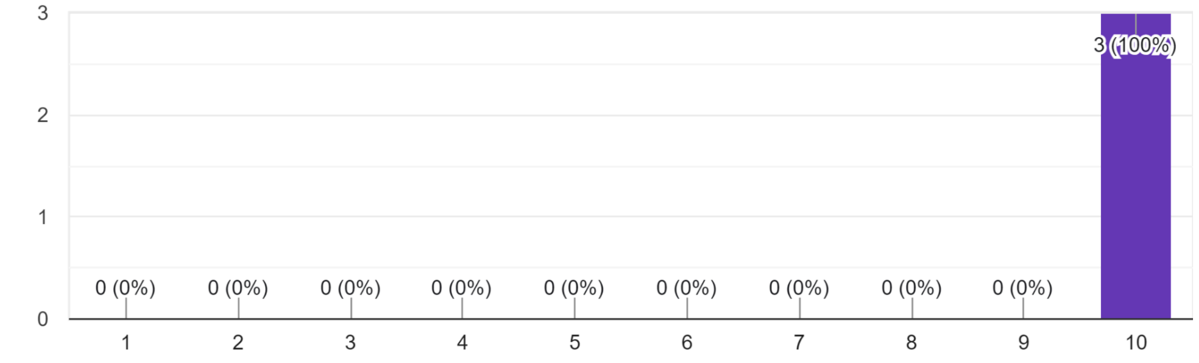


I found it easy to navigate from the home page to the dashboard.

3 responses

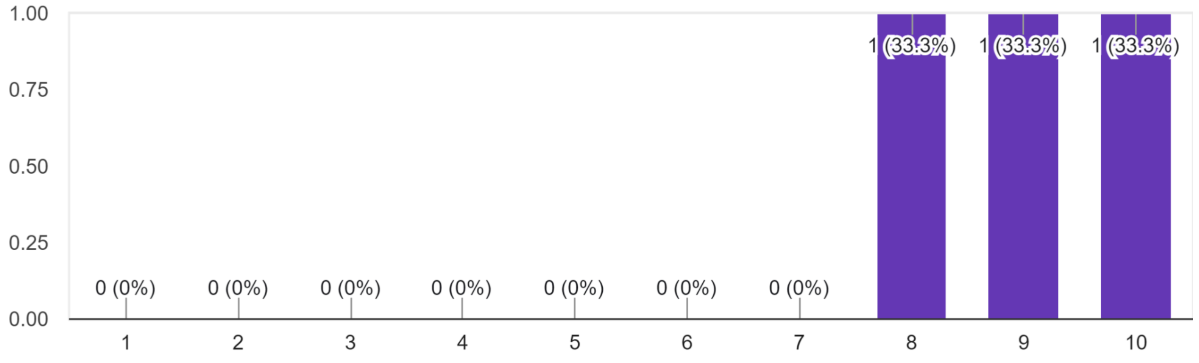
I found it easy to create an account and login.

3 responses



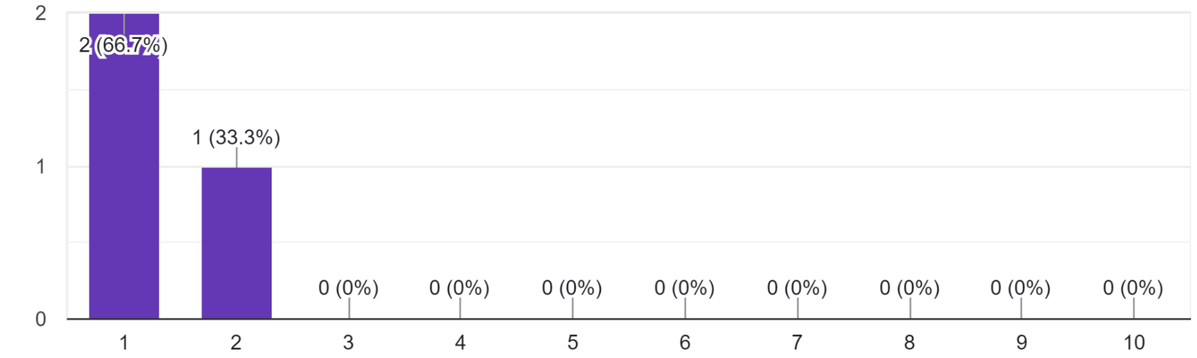
I was able to intuitively navigate the dashboard to create an organization.

3 responses



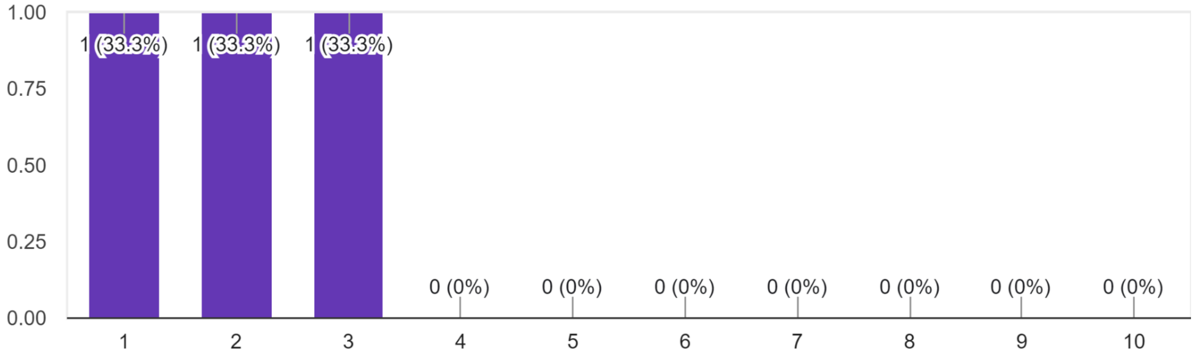
The dashboard header was difficult to understand.

3 responses



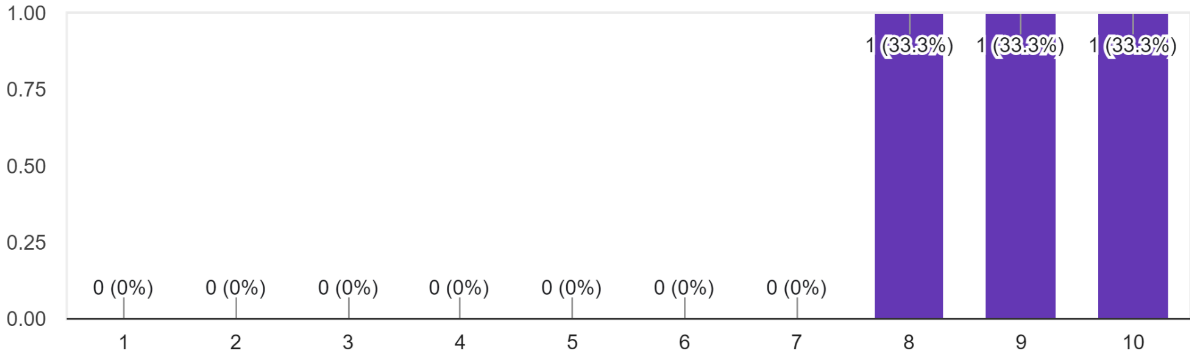
The group's page layout confused me.

3 responses



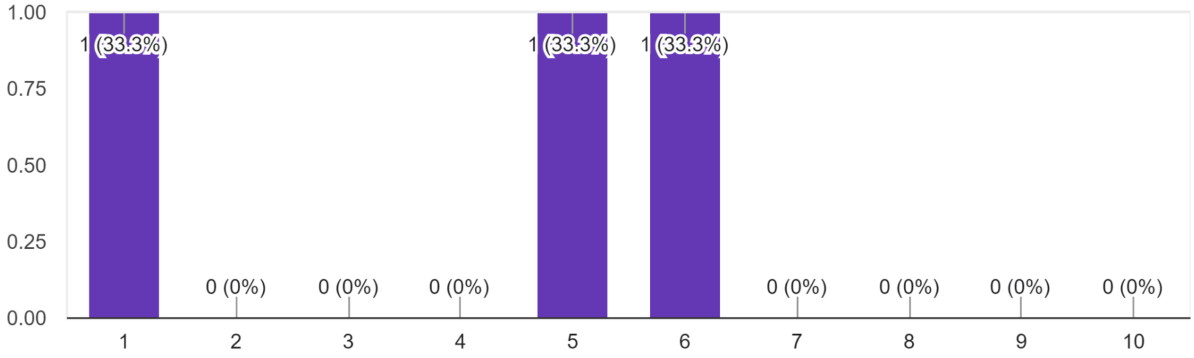
I was able to intuitively navigate the dashboard to create a group.

3 responses



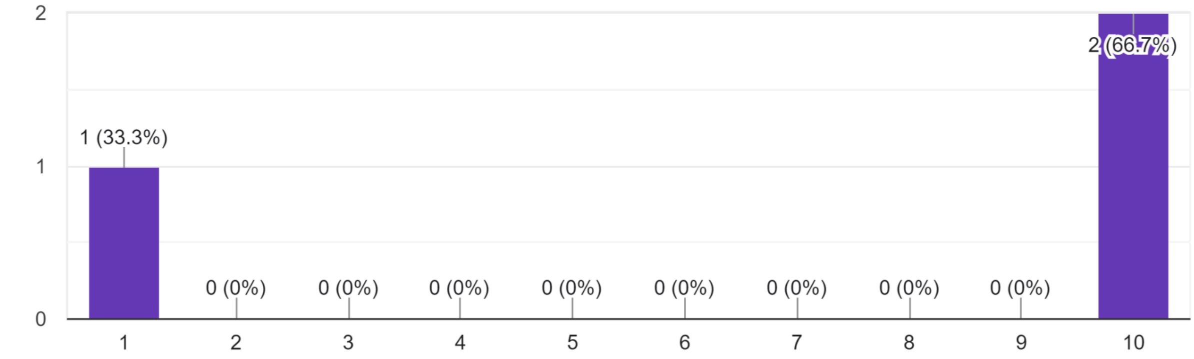
I struggled to find the awards page.

3 responses



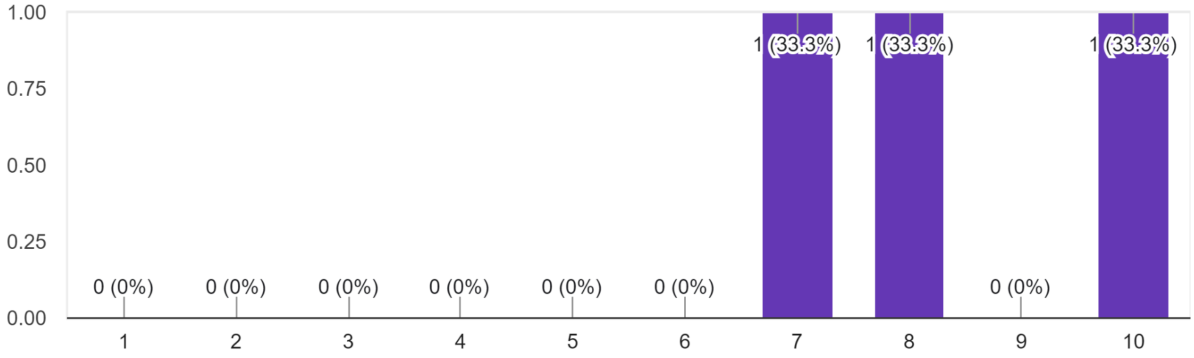
The layout of the awards page was consistent with the other pages.

3 responses



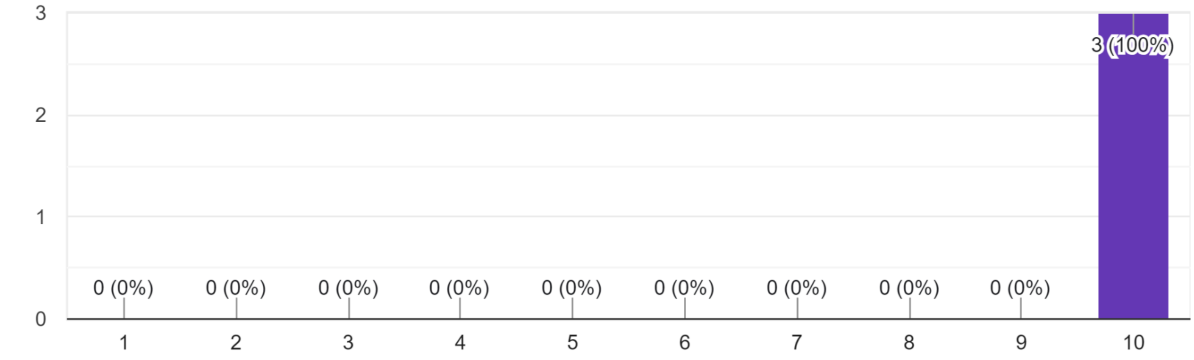
I was able to intuitively navigate the dashboard to create an award.

3 responses



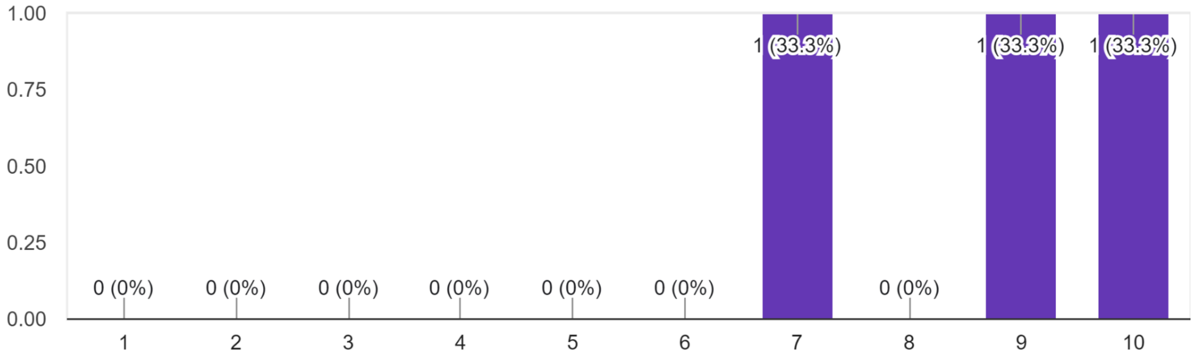
The event page’s layout was consistent with the rest of the application.

3 responses



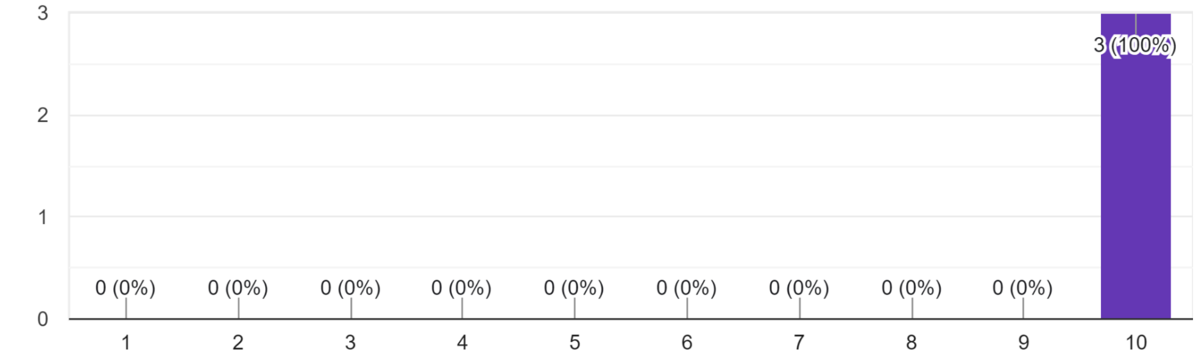
The event details page layout contained all the information I wanted it to.

3 responses



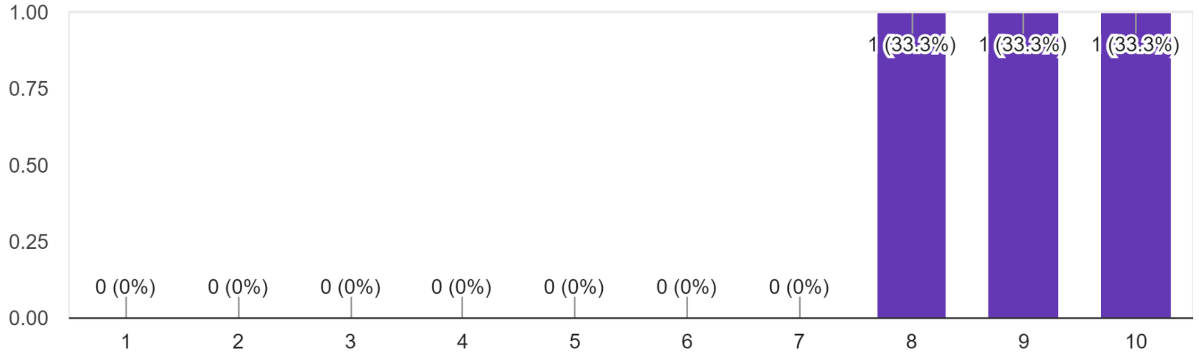
I was able to intuitively navigate the dashboard to create an event.

3 responses



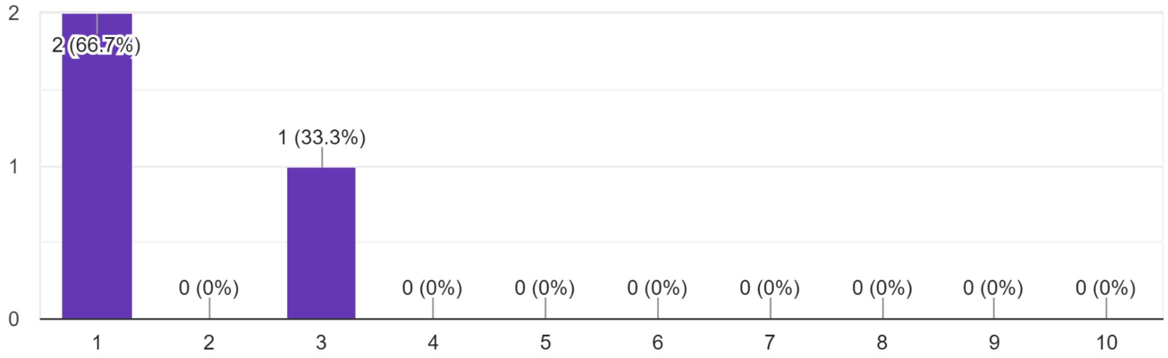
The layout of the members page made intuitive sense.

3 responses



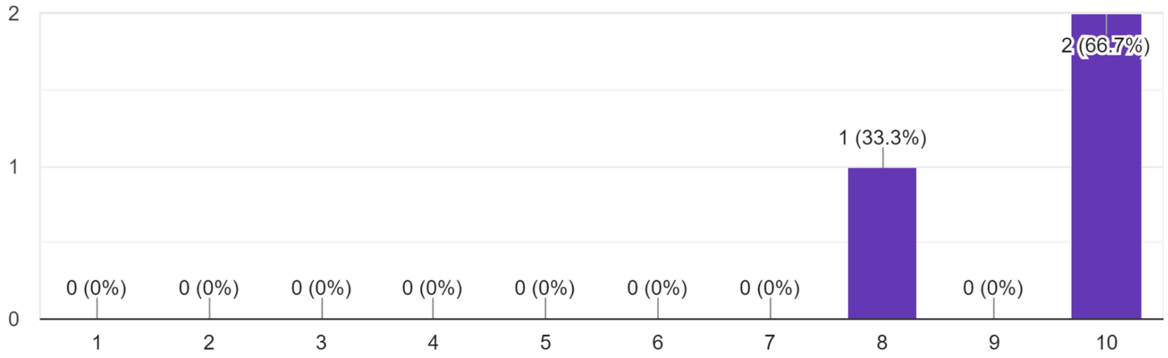
Using the add button in the members page was confusing.

3 responses



I was able to intuitively navigate the dashboard to add a member.

3 responses



QA Test Plan

QA Test 1 (Password Security)

Objective:
Create an account with a variety of passwords, valid and invalid. Ensure that password validation during registration is following modern security standards.

Plan:
The horizontal prototype already supports these features, the user will start from the website's registration page. The three tests will be to try a password that's too short, a password that is long enough, but doesn't have special characters, and a valid/secure password.
http://ec2   compute-1.amazonaws.com:3000/register

Number	Description	Test Input	Expected Output	Pass/Fail
1	Short Password	"Test"	Password must have 8 characters	Pass
2	Weak Password	"Password123"	Password must contain special characters	Pass
3	Strong Password	"Password@123"	No errors	Pass

QA Test 2 (Member Addition)

Objective:
Add three members to your organization using their Member IDs. Ensure that member addition is following good usability standards and error handling.

Plan:
The horizontal prototype already supports these features, the user will start from the website's dashboard after logging in and creating an organization. The three tests will be to add a member that doesn't exist, a member that does exist, and a member that is already in the organization..
http://ec2   compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Invalid Member	1234	Member does not exist.	Pass
2	Valid Member	2017	Member added to list.	Pass
3	Existing Member	9141	Member is already in the organization.	Fail

QA Test 3 (Search Bar)

Objective:
Search for three members from the members page. Ensure the search bar is following along with our expected load standards and requirements.

Plan:
The horizontal prototype already supports this feature, the user will start from a populated members page. The three tests will be to search for an existing member, searching for a member that doesn't exist, and a blank search bar.
http://ec2 compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Search	“John Doe”	Member list with one item	Fail
2	Invalid Search	“TestUser”	Empty member list	Fail
3	Empty Search	“ “	Unfiltered member list	Fail

QA Test 4 (Error Logging)

Objective:

Check if your actions are being logged in the console. Ensure the browser console is populating with appropriate messages in-line with our fault tolerance standards and requirements.

Plan:

The horizontal prototype already supports this feature, the user will start from an empty members page. The three tests will be to try to add a member that is valid, a member that does not exist, and the user themself. The browser's console will be monitored during the test to check for results.

http://ec2 compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Member	2017	"Member added successfully"	Pass
2	Invalid Member	1234	"Member not found"	Pass
3	Add Self	9141	"Member already in organization"	Fail

QA Test 5 (Concurrent Users)

Objective:

Create an account and organization while the application is under load. Ensure the application can support many concurrent users without significant degradation in performance in-line with our performance standards and requirements.

Plan:

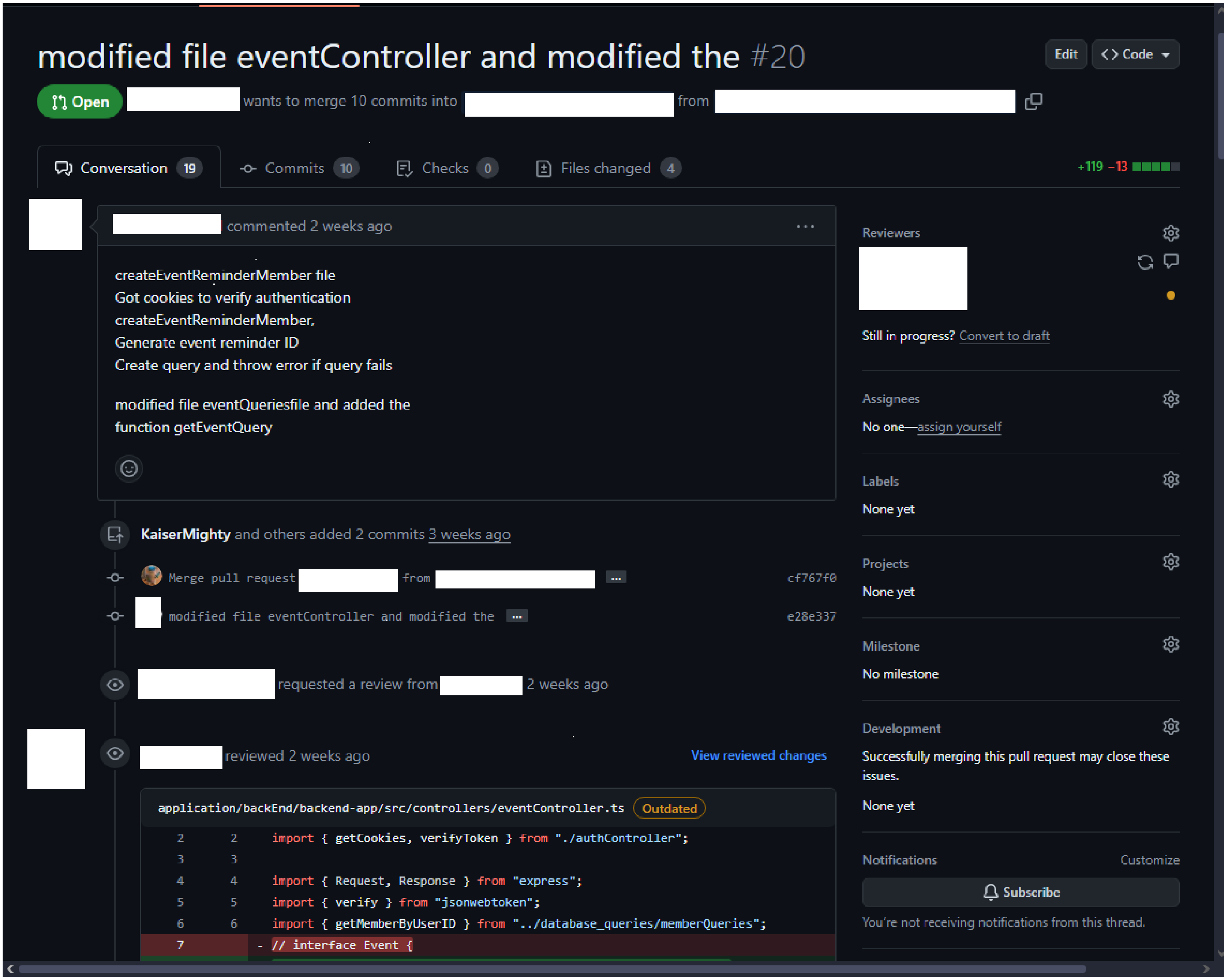
The horizontal prototype already supports the features we will be stress testing. We will be using Postman to simulate network traffic. The user will start from the homepage. The user will try registering, logging in, and creating an organization while the database and backend are under stress.

http://ec2   compute-1.amazonaws.com:3000/

Number	Description	Test Input	Expected Output	Pass/Fail
1	Registration	New Account	Minimal Delay	Pass
2	Login	Existing Account	Minimal Delay	Pass
3	Create Organization	New Organization	Minimal Delay	Pass

Code Review

Internal Code Review



modified file eventController and modified the #20

Open [redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted] [redacted] commented 2 weeks ago

createEventReminderMember file
Got cookies to verify authentication
createEventReminderMember,
Generate event reminder ID
Create query and throw error if query fails

modified file eventQueriesfile and added the
function getEventQuery

KaiserMighty and others added 2 commits 3 weeks ago

```
Merge pull request [redacted] from [redacted] …                cf767f0
[redacted] modified file eventController and modified the …    e28e337
```

[redacted] requested a review from [redacted] 2 weeks ago

[redacted] [redacted] reviewed 2 weeks ago

application/backEnd/backend-app/src/controllers/eventController.ts Outdated

```
import { getCookies, verifyToken } from "./authController";

import { Request, Response } from "express";
import { verify } from "jsonwebtoken";
import { getMemberByUserID } from "../database_queries/memberQueries";
- // interface Event {
```

Reviewers

[redacted]

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

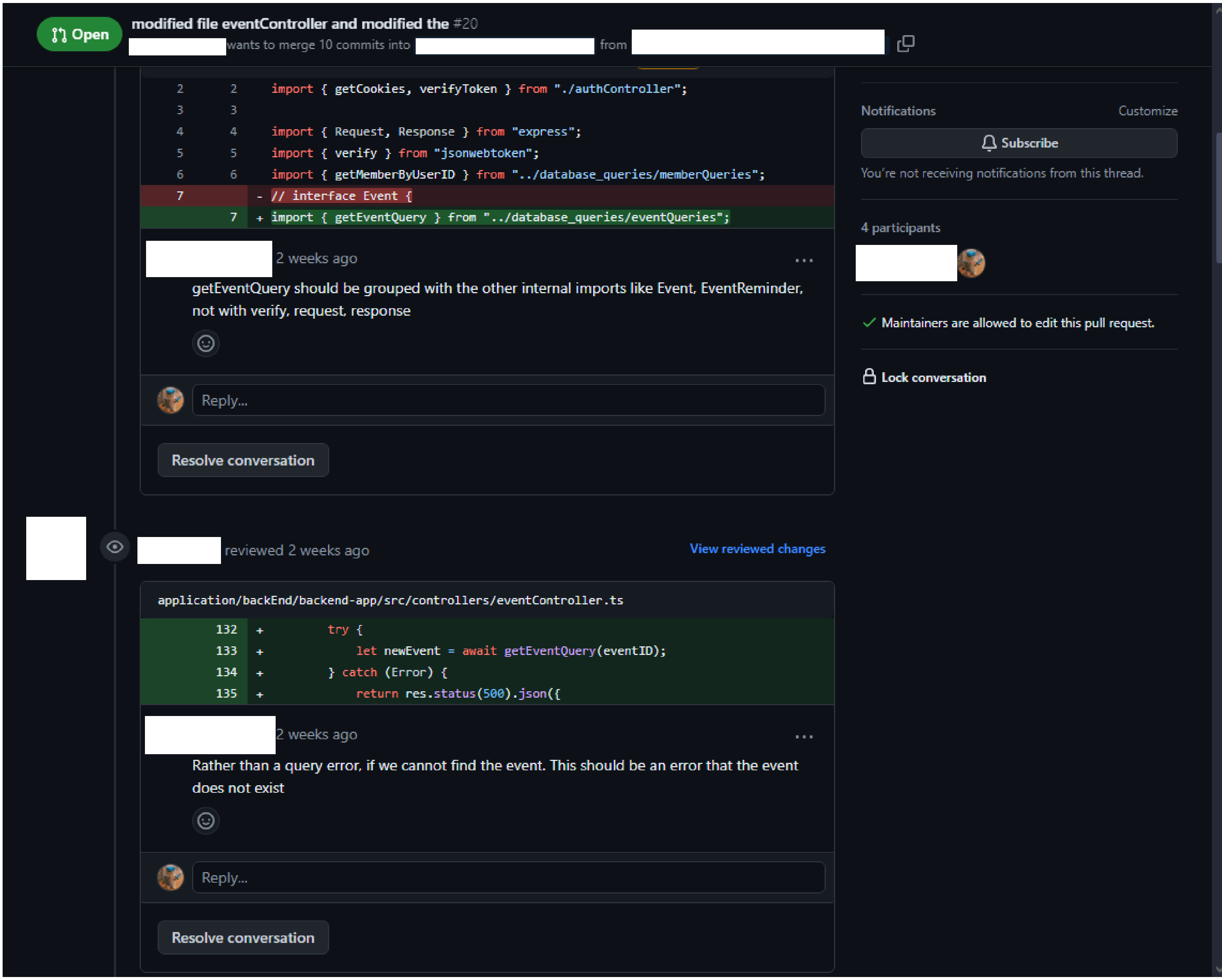
Development

Successfully merging this pull request may close these issues.

None yet

Notifications

You're not receiving notifications from this thread.



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

```
    import { getCookies, verifyToken } from "./authController";

    import { Request, Response } from "express";
    import { verify } from "jsonwebtoken";
    import { getMemberByUserID } from "../database_queries/memberQueries";
  - // interface Event {
  + import { getEventQuery } from "../database_queries/eventQueries";
```

[redacted] 2 weeks ago

getEventQuery should be grouped with the other internal imports like Event, EventReminder, not with verify, request, response

[redacted]

[redacted] reviewed 2 weeks ago

application/backEnd/backend-app/src/controllers/eventController.ts

```
  +         try {
  +             let newEvent = await getEventQuery(eventID);
  +         } catch (Error) {
  +             return res.status(500).json({
```

[redacted] 2 weeks ago

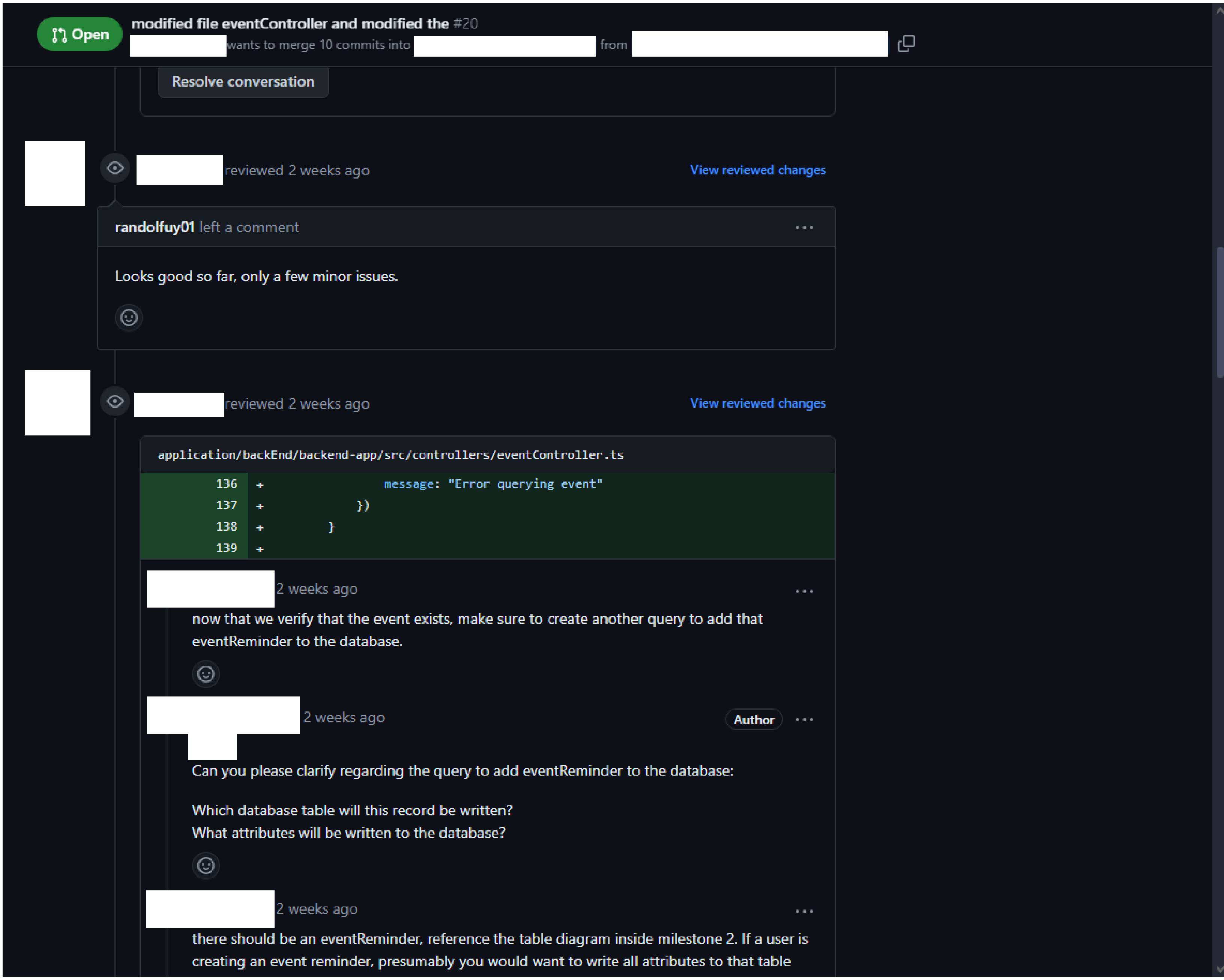
Rather than a query error, if we cannot find the event. This should be an error that the event does not exist

You're not receiving notifications from this thread.

4 participants

[redacted]

✓ Maintainers are allowed to edit this pull request.



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted] [redacted] reviewed 2 weeks ago

randolfuy01 left a comment

Looks good so far, only a few minor issues.

[redacted] [redacted] reviewed 2 weeks ago

application/backEnd/backend-app/src/controllers/eventController.ts

```
+               message: "Error querying event"
+           })
+       }
+
```

[redacted] 2 weeks ago

now that we verify that the event exists, make sure to create another query to add that eventReminder to the database.

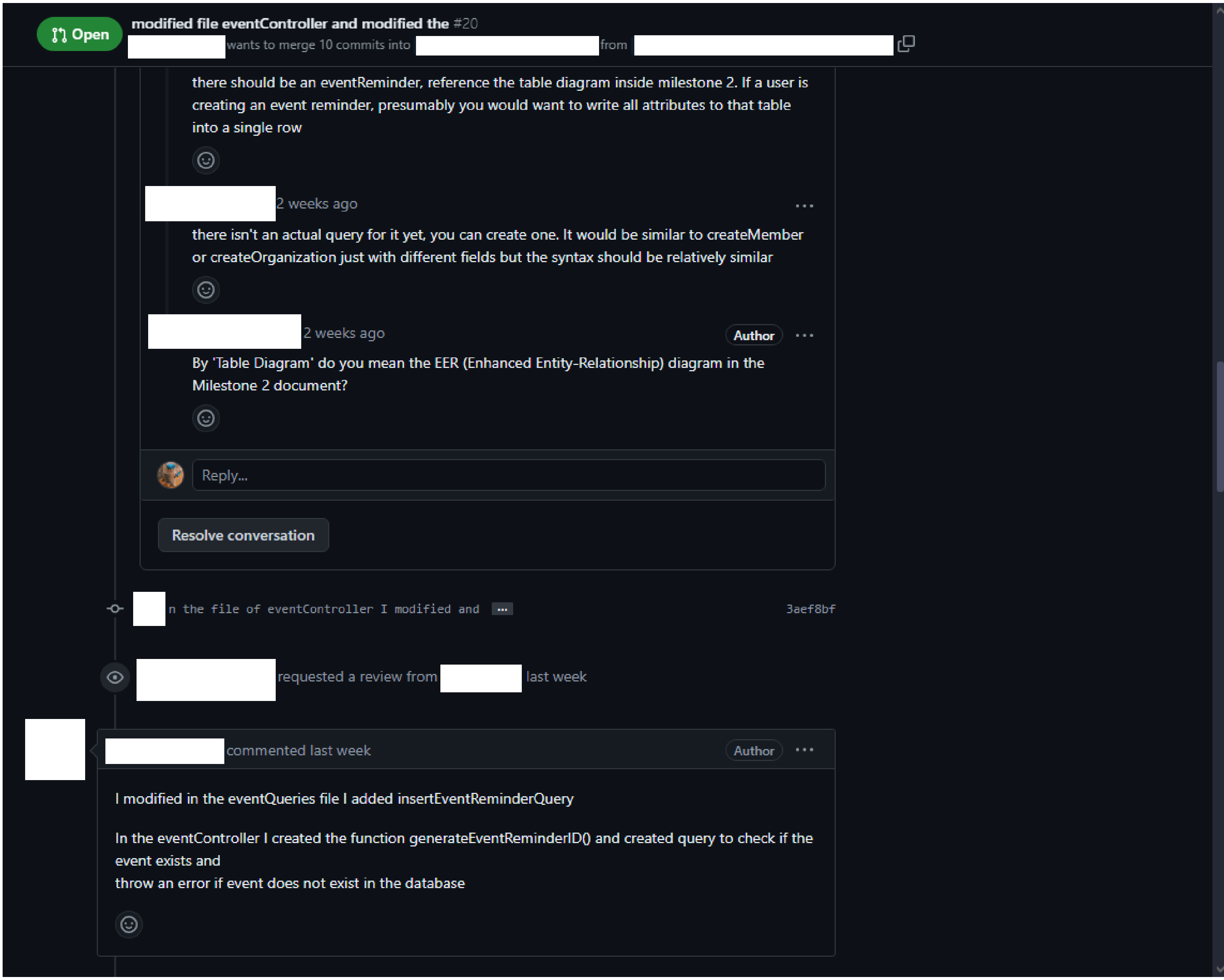
[redacted] [redacted] 2 weeks ago

Can you please clarify regarding the query to add eventReminder to the database:

Which database table will this record be written?
What attributes will be written to the database?

[redacted] 2 weeks ago

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table into a single row

[redacted] 2 weeks ago

there isn't an actual query for it yet, you can create one. It would be similar to createMember or createOrganization just with different fields but the syntax should be relatively similar

[redacted] 2 weeks ago

By 'Table Diagram' do you mean the EER (Enhanced Entity-Relationship) diagram in the Milestone 2 document?

[redacted] n the file of eventController I modified and ... 3aef8bf

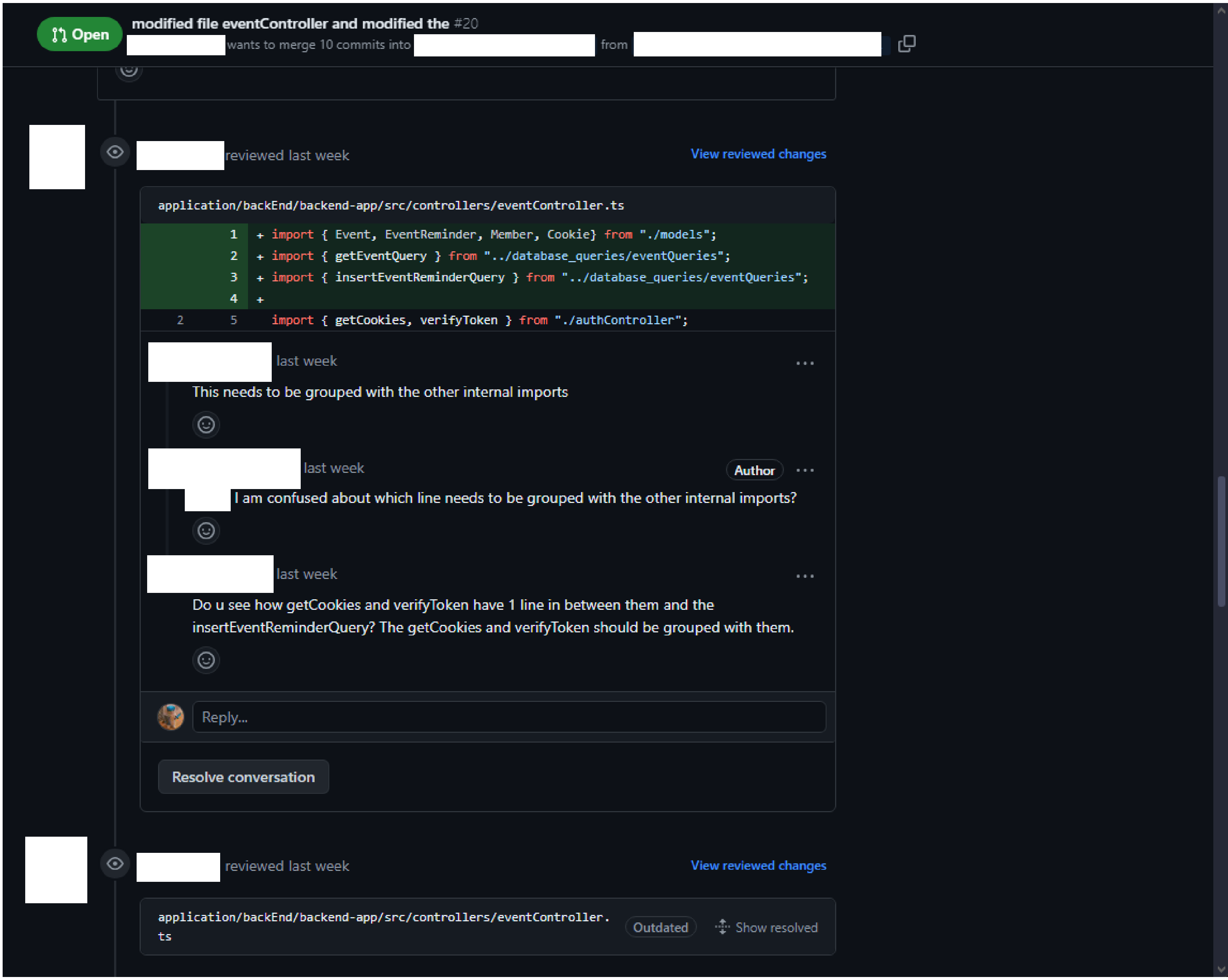
[redacted] requested a review from [redacted] last week

[redacted]

[redacted] commented last week

I modified in the eventQueries file I added insertEventReminderQuery

In the eventController I created the function generateEventReminderID() and created query to check if the event exists and
throw an error if event does not exist in the database



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted]

[redacted] reviewed last week

application/backEnd/backend-app/src/controllers/eventController.ts

```
+ import { Event, EventReminder, Member, Cookie} from "./models";
+ import { getEventQuery } from "../database_queries/eventQueries";
+ import { insertEventReminderQuery } from "../database_queries/eventQueries";
+
  import { getCookies, verifyToken } from "./authController";
```

[redacted] last week

This needs to be grouped with the other internal imports

[redacted] last week Author

[redacted] I am confused about which line needs to be grouped with the other internal imports?

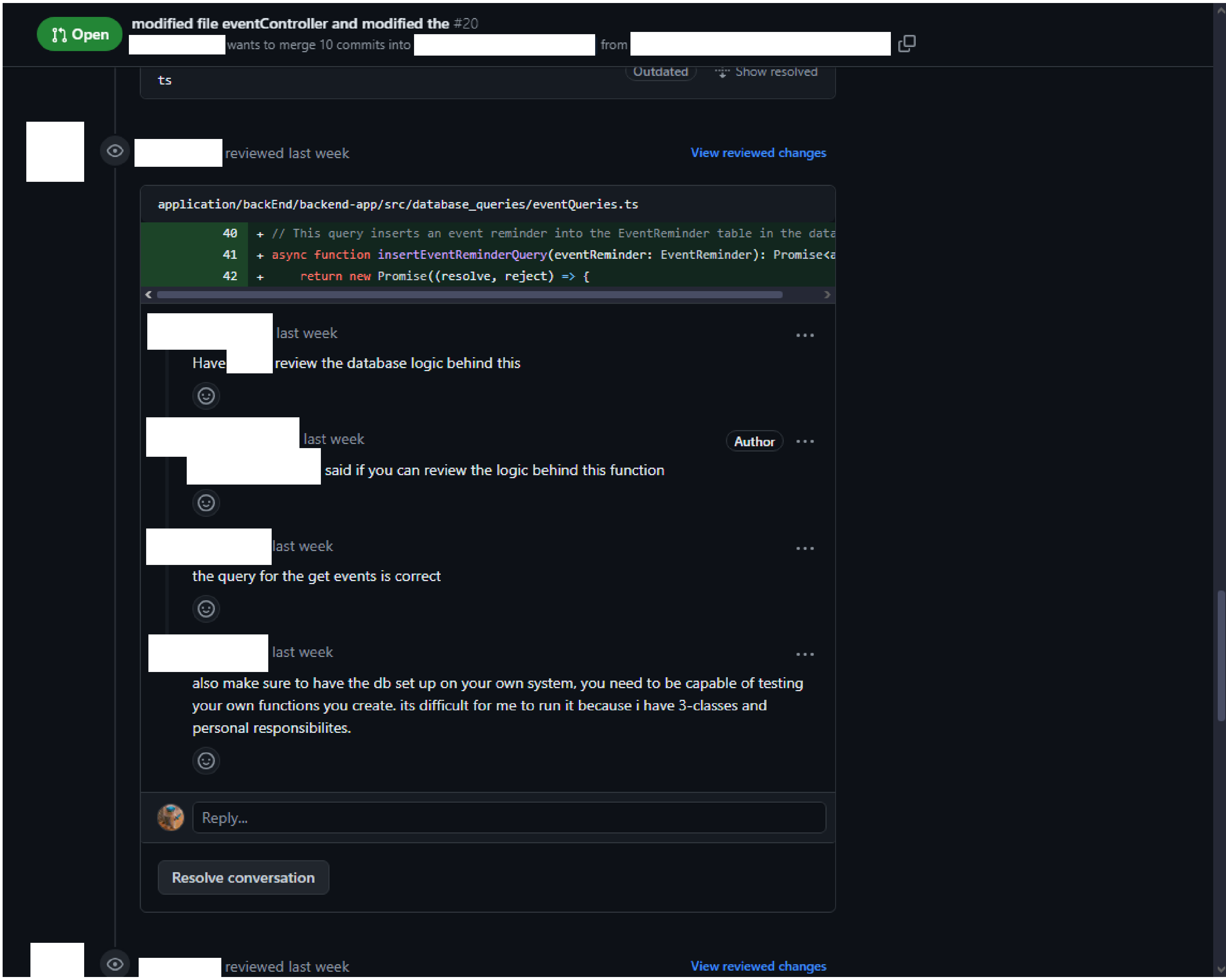
[redacted] last week

Do u see how getCookies and verifyToken have 1 line in between them and the insertEventReminderQuery? The getCookies and verifyToken should be grouped with them.

[redacted]

[redacted] reviewed last week

application/backEnd/backend-app/src/controllers/eventController.ts



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

```
ts
```

[redacted]

[redacted] reviewed last week

```
application/backEnd/backend-app/src/database_queries/eventQueries.ts
+ // This query inserts an event reminder into the EventReminder table in the data
+ async function insertEventReminderQuery(eventReminder: EventReminder): Promise<a
+     return new Promise((resolve, reject) => {
```

[redacted] last week

Have [redacted] review the database logic behind this

[redacted] last week

[redacted] said if you can review the logic behind this function

[redacted] last week

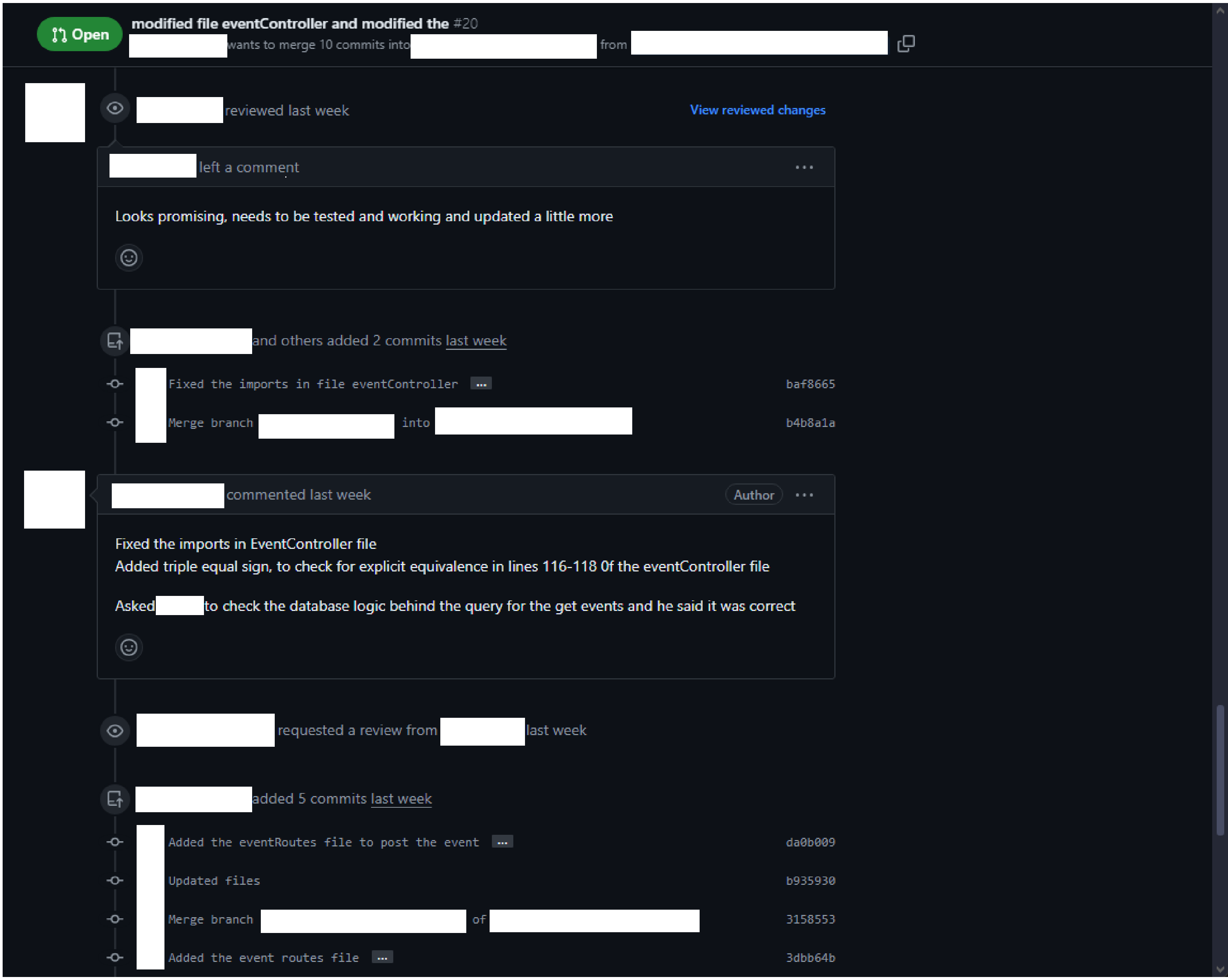
the query for the get events is correct

[redacted] last week

also make sure to have the db set up on your own system, you need to be capable of testing your own functions you create. its difficult for me to run it because i have 3-classes and personal responsibilites.

[redacted]

[redacted] reviewed last week



Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted] [redacted] reviewed last week View reviewed changes

[redacted] left a comment ···

Looks promising, needs to be tested and working and updated a little more

[redacted] and others added 2 commits last week

[redacted]

```
Fixed the imports in file eventController ...                baf8665
Merge branch [redacted] into [redacted]                      b4b8a1a
```

[redacted] [redacted] commented last week Author ···

Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file

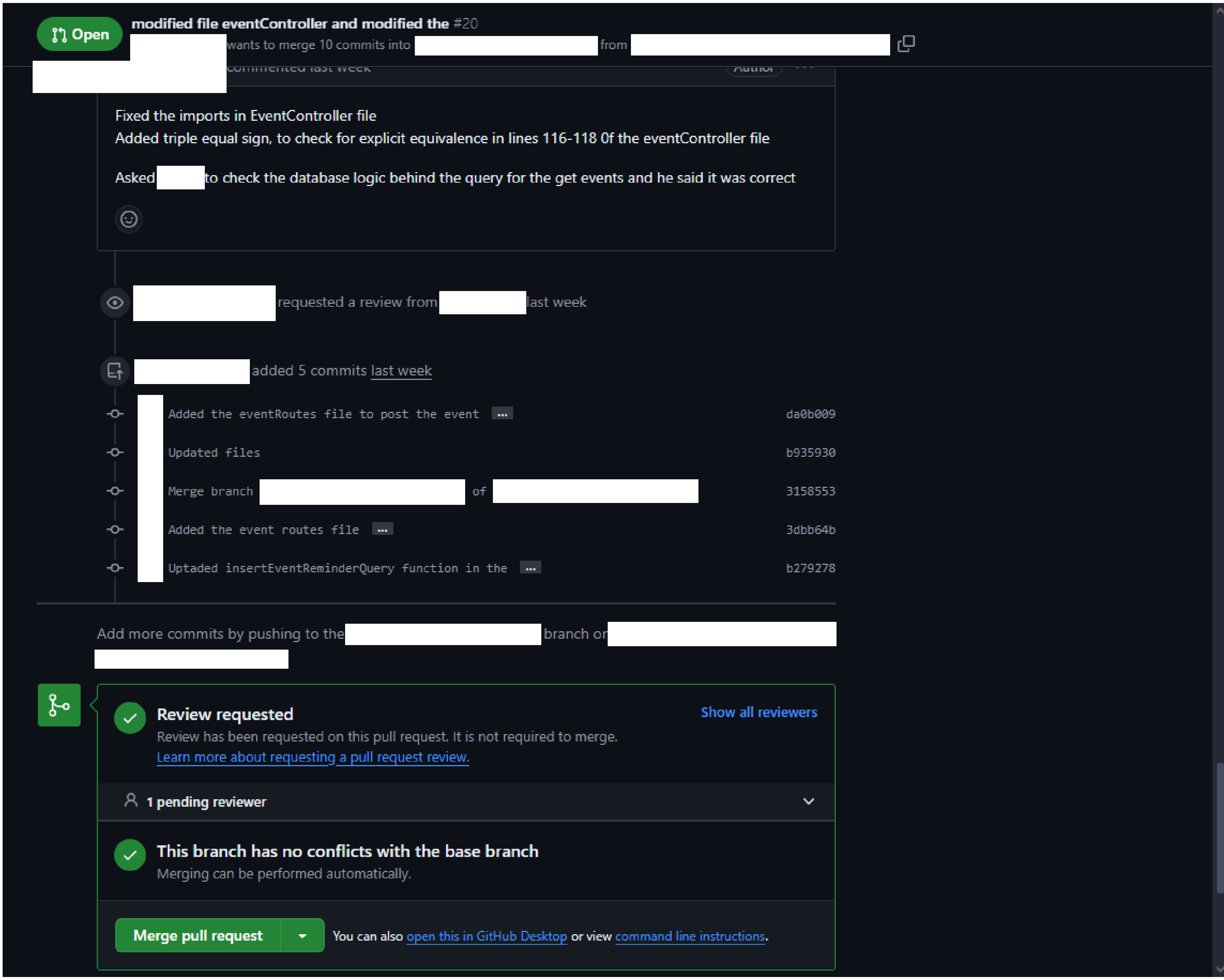
Asked [redacted] to check the database logic behind the query for the get events and he said it was correct

[redacted] requested a review from [redacted] last week

[redacted] added 5 commits last week

[redacted]

```
Added the eventRoutes file to post the event ...             da0b009
Updated files                                                b935930
Merge branch [redacted] of [redacted]                        3158553
Added the event routes file ...                              3dbb64b
```



modified file eventController and modified the #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted] commented last week

Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file

Asked [redacted] to check the database logic behind the query for the get events and he said it was correct

[redacted] requested a review from [redacted] last week

[redacted] added 5 commits last week

```
[redacted]
Added the eventRoutes file to post the event  …                         da0b009
Updated files                                                           b935930
Merge branch [redacted] of [redacted]                                   3158553
Added the event routes file  …                                          3dbb64b
Uptaded insertEventReminderQuery function in the  …                     b279278
```

Add more commits by pushing to the [redacted] branch on [redacted]

Review requested

Review has been requested on this pull request. It is not required to merge.
Learn more about requesting a pull request review.

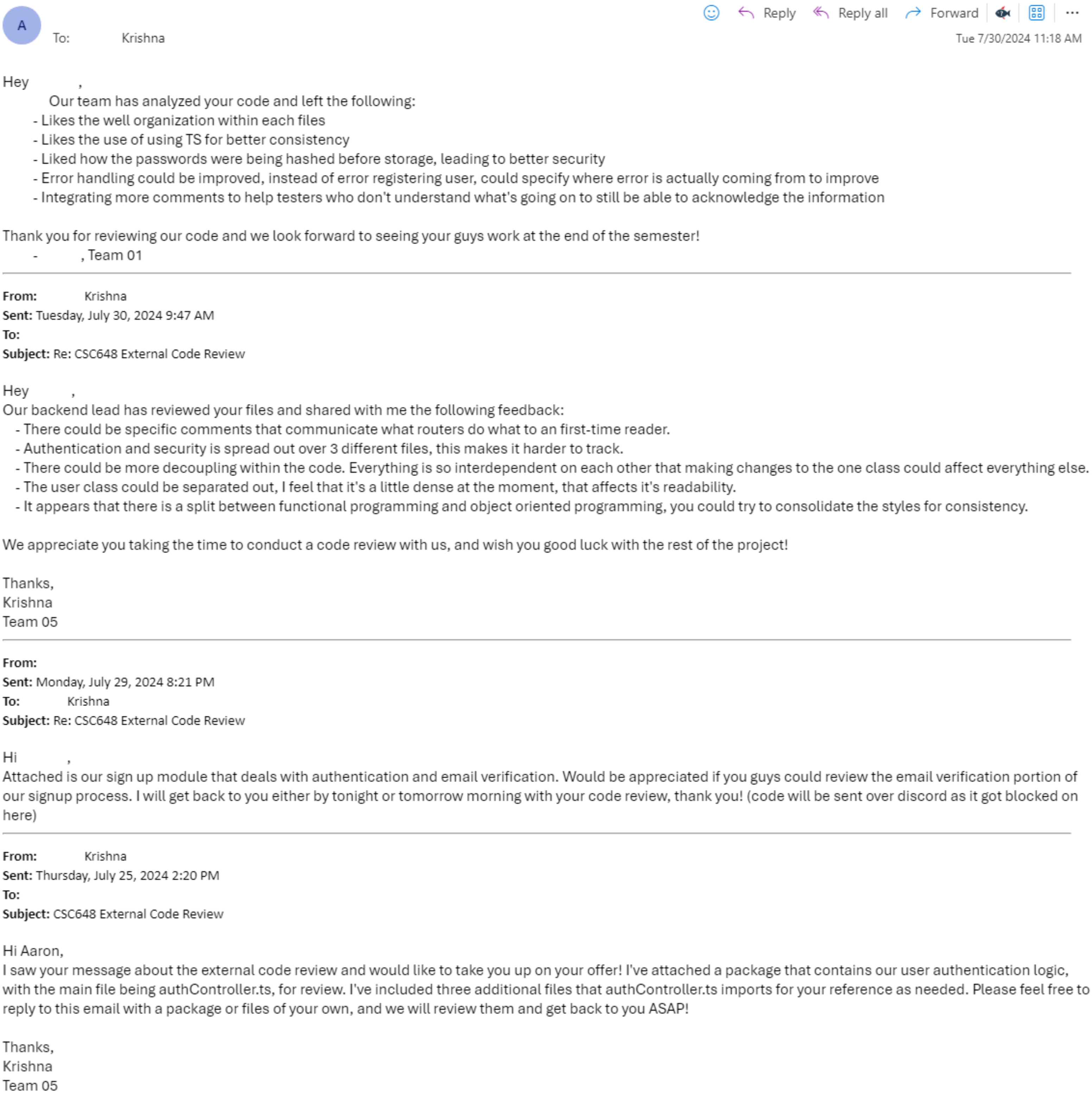
1 pending reviewer

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

External Code Review



To: Krishna

Tue 7/30/2024 11:18 AM

Hey ,

Our team has analyzed your code and left the following:

- Likes the well organization within each files
- Likes the use of using TS for better consistency
- Liked how the passwords were being hashed before storage, leading to better security
- Error handling could be improved, instead of error registering user, could specify where error is actually coming from to improve
- Integrating more comments to help testers who don't understand what's going on to still be able to acknowledge the information

Thank you for reviewing our code and we look forward to seeing your guys work at the end of the semester!

- , Team 01

From: Krishna
Sent: Tuesday, July 30, 2024 9:47 AM
To:
Subject: Re: CSC648 External Code Review

Hey ,
Our backend lead has reviewed your files and shared with me the following feedback:

- There could be specific comments that communicate what routers do what to an first-time reader.
- Authentication and security is spread out over 3 different files, this makes it harder to track.
- There could be more decoupling within the code. Everything is so interdependent on each other that making changes to the one class could affect everything else.
- The user class could be separated out, I feel that it's a little dense at the moment, that affects it's readability.
- It appears that there is a split between functional programming and object oriented programming, you could try to consolidate the styles for consistency.

We appreciate you taking the time to conduct a code review with us, and wish you good luck with the rest of the project!

Thanks,
Krishna
Team 05

From:
Sent: Monday, July 29, 2024 8:21 PM
To: Krishna
Subject: Re: CSC648 External Code Review

Hi ,
Attached is our sign up module that deals with authentication and email verification. Would be appreciated if you guys could review the email verification portion of our signup process. I will get back to you either by tonight or tomorrow morning with your code review, thank you! (code will be sent over discord as it got blocked on here)

From: Krishna
Sent: Thursday, July 25, 2024 2:20 PM
To:
Subject: CSC648 External Code Review

Hi Aaron,
I saw your message about the external code review and would like to take you up on your offer! I've attached a package that contains our user authentication logic, with the main file being authController.ts, for review. I've included three additional files that authController.ts imports for your reference as needed. Please feel free to reply to this email with a package or files of your own, and we will review them and get back to you ASAP!

Thanks,
Krishna
Team 05

Security Self-Check

In-line with modern standards, our application does not store unsecured plaintext passwords, but instead stored a secure hashed password to keep users' accounts safe.

When the backend receives the password, our controller calls a hashing function.

```
// HASH AND VERIFY HASHING PASSWORD
const hashedPassword:string = await hashPassword(registerUser.password);
const verify = verifyPassword(registerUser.password, hashedPassword);
    if (!verify) {
        return res.status(400).json({
            error: 'Password Hashing unsuccessful'
        });
    }
```

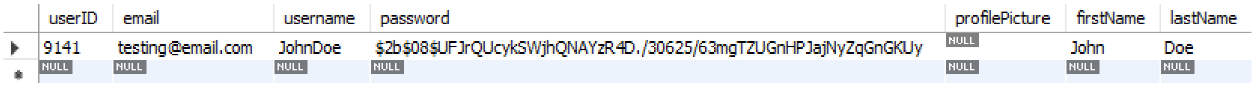
The hashing function uses bcrypt, and goes as follows:

```
import bycript from 'bcrypt';

// HASHES PASSWORD
async function hashPassword(password: string): Promise<string> {
    return await bycript.hash(password,8);
}

// VERIFIES PASSWORD
async function verifyPassword(password: string, hashPassword: string):
Promise<boolean> {
    return await bycript.compare(password, hashPassword);
}
```

This hashed password is what is passed in the MySQL query and stored in the DB:



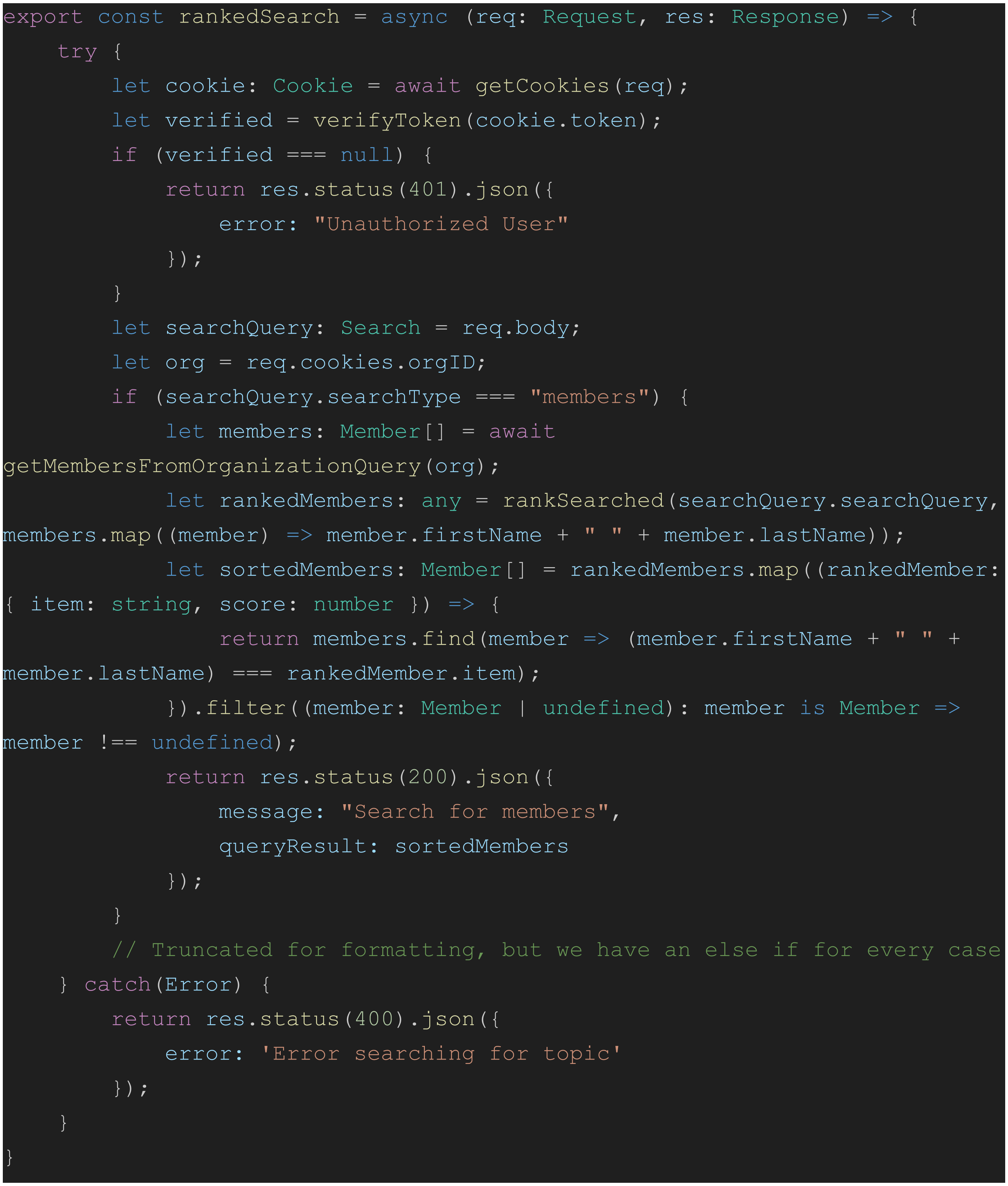
	userID	email	username	password	profilePicture	firstName	lastName
▶	9141	testing@email.com	JohnDoe	$2b$08$UFJrQUcykSWjhQNAYzR4D./30625/63mgTZUGnHPJajNyZqGnGKUy	NULL	John	Doe
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Our search bar input data are validated to make sure we are searching the right tables in the database:

Our autocomplete function is as follows:

```
export const autoCompleteSearch = async (req: Request, res: Response) => {
    try {
        try {
            let token = req.cookies.token;
            let decoded = jwt.verify(token, JWT_SECRET_KEY);
            if (decoded === null) {
                throw new Error('Unauthorized User');
            }
        } catch (Error) {
            res.status(401).json({
                error: 'Unauthorized User'
            });
        }
        let searchQuery: string = req.body.searchQuery;
        // Query words from the database
        let members: Member[] = await
getMembersFromOrganizationQuery(req.cookies.orgID);
        let queryWords: string[] = members.map((members) =>
members.firstName + " " + members.lastName);
        const completer = new Trie();
        for (const word of queryWords) {
            completer.insert(word);
        }
        const prefix: string = searchQuery;
        const result: string[] = completer.search(prefix);
        res.status(200).json({
            queryResults: result
        });
    } catch(Error) {
        res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Out ranked search function is as follows:



```
export const rankedSearch = async (req: Request, res: Response) => {
    try {
        let cookie: Cookie = await getCookies(req);
        let verified = verifyToken(cookie.token);
        if (verified === null) {
            return res.status(401).json({
                error: "Unauthorized User"
            });
        }
        let searchQuery: Search = req.body;
        let org = req.cookies.orgID;
        if (searchQuery.searchType === "members") {
            let members: Member[] = await
getMembersFromOrganizationQuery(org);
            let rankedMembers: any = rankSearched(searchQuery.searchQuery,
members.map((member) => member.firstName + " " + member.lastName));
            let sortedMembers: Member[] = rankedMembers.map((rankedMember:
{ item: string, score: number }) => {
                return members.find(member => (member.firstName + " " +
member.lastName) === rankedMember.item);
            }).filter((member: Member | undefined): member is Member =>
member !== undefined);
            return res.status(200).json({
                message: "Search for members",
                queryResult: sortedMembers
            });
        }
        // Truncated for formatting, but we have an else if for every case
    } catch(Error) {
        return res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Non-Functional Specs Self-Check

Performance

- **DONE -** The application shall respond quickly to user prompts with minimal latency.
- **DONE -** The application shall be able to host/handle up to 1,000 accounts without any performance problems.
- **DONE -** The application shall support many active-concurrent users without significant degradation in performance.

Security

- **DONE -** The application shall only store hashed passwords.
- **ON TRACK -** The application database shall be backed up regularly.

Coding Standard

- **DONE -** The application shall follow consistent coding and naming conventions throughout.
- **DONE -** The application shall gracefully handle errors without detrimentally affecting the rest of the user experience.
- **ON TRACK -** The application shall be well documented for easy understandability for other developers and admins alike.

Media Storage

- **ISSUE -** The application shall restrict profile pictures to 500x500 pixels and 5 megabytes or less.
    - Images are not part of our priority 1 requirements, and as such, are not included in our prototype.

Privacy

- **DONE -** The application shall hide organizations, preventing users not in the organization from viewing it.
- **DONE -** Data shall be anonymized and pseudonymized wherever possible to minimize damage in the event of a leak.

- **DONE -** Only the minimum amount of data required shall be stored in order to minimize the risk of security breaches.

Scalability

- **DONE -** The application shall be designed to scale efficiently to handle an increasing number of users and data without performance degradation.
- **DONE -** The database shall scale down if the workload is low to remain cost effective and optimize resource utilization.

Reliability

- **DONE -** The application shall ensure high availability, with a 99.9% uptime, and incorporate robust error-handling mechanism to maintain stability.

Usability

- **DONE -** The application shall provide an user friendly interface, ensuring that users can easily navigate and use all features without extensive training.

Compliance

- **DONE -** The application shall comply with relevant industry standards and regulations, such as for data protection and privacy.

Expected Load

- **DONE -** The application shall respond to user queries within 500 milliseconds for requests.
- **DONE -** The application shall handle a minimum of 1000 transactions per second during peak hours.
- **DONE -** The application shall horizontally scale to support up to 10,000 concurrent users without degradation in response time.

Fault Tolerance

- **DONE -** The application shall recover from a server failure within 15 minutes without loss of data or interruption of service.
- **DONE -** Critical components of the application shall have a backup system that can take over within 5 minutes of a primary system failure.
- **DONE -** The application shall log all errors and provide automatic recovery for non-critical errors within 60 seconds.

Database High Level Specs

- **DONE -** The database shall use appropriate indexes to optimize query performance, especially for frequently accessed data.
- **DONE -** The database should be capable of storing up to 1 terabyte of data efficiently, with automatic data compression for storage optimization.
- **DONE -** All sensitive data stored in the database should be encrypted.

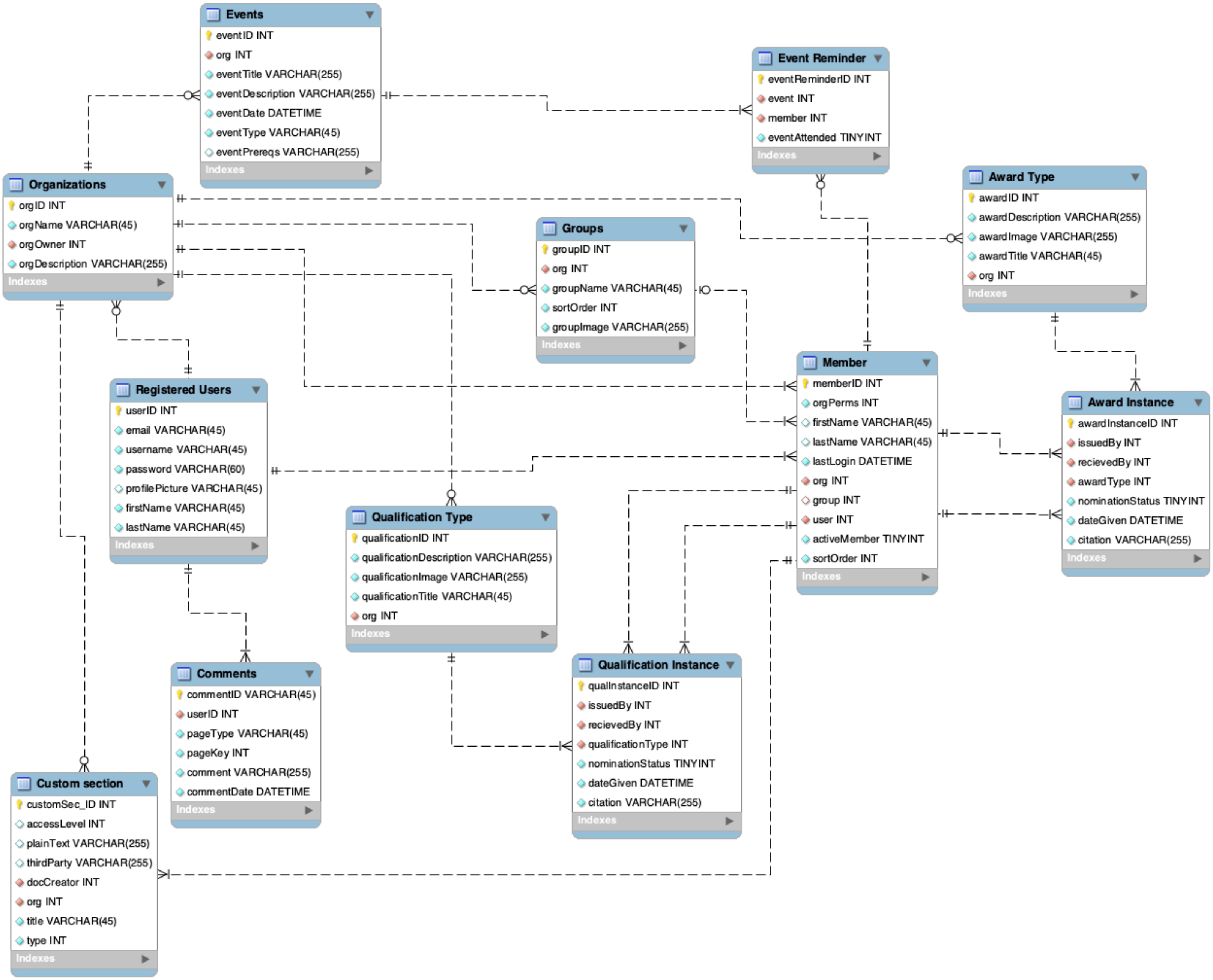
Browser and OS Support

- **DONE -** The application shall support Chrome Stable 126 (126.0.6478.153).
- **DONE -** The application shall support Windows 10 22H2 (19045.4651).
- **DONE -** The application shall support Windows 11 23H2 (22631.3737).
- **DONE -** The application shall support macOS Sonoma (14.5).
- **DONE -** The application shall support Amazon Linux (2023.4.20240611.0).

Additional Comments

Changes to the DBMS

Adding a Title (VARCHAR 45) and Type (INT) field to the Custom Section entity in order to differentiate between Projects/Tasks and Highlights.



Team Contributions

- **Krishna**
    - Revised Product Summary Section of Documentation.
    - Usability Test Plan Section of Documentation.
    - QA Test Plan Section of Documentation.
    - Security Self-Check Section of Documentation.
    - Non-Functional Requirements Self-Check Section of Documentation.
    - Group page frontend of Beta Prototype.
    - Recognitions page frontend of Beta Prototype.

-
    - Prototype revisions as per horizontal prototype feedback.
    - General restyling of all pages.
    - Oversaw other front-end developers progress on static pages.
    - Refined static pages.
    - Tasks/Projects page frontend of Beta Prototype.
    - Events page frontend of Beta Prototype.
    - Highlights page frontend of Beta Prototype.

-
    - Leading efforts on Backend of the Prototype.
    - Revised search controller of Beta Prototype.
    - Created recognition controller of Beta Prototype.
    - Created custom sections controller of Beta Prototype.
    - Revised password hashing functions.
    - Added various routing components for the backend.
    - Miscellaneous bug fixing on the backend component.
    - Completed Internal Code Review.
    - Complete External Code Review.

-
    - Revised custom sections queries of Beta Prototype.
    - Revised custom sections controller of Beta Prototype.
    - Adjusted DBMS files (EER and .sql files).
    - Revised group queries of Beta Prototype.
    - Miscellaneous bug fixing on the backend component.

-
    - Revised Case Studies page of Beta Prototype.
    - Revised Reviews page of Beta Prototype.
    - Revised Updates page of Beta Prototype.
    - Revised Server Status page of Beta Prototype.
    - Revised Features page of Beta Prototype.
    - Revised Pricing page of Beta Prototype.
    - Revised About page of Beta Prototype.
-
    - Revised Contact Us page of Beta Prototype.
    - Revised Chat Support page of Beta Prototype.
    - Revised Report a Bug page of Beta Prototype.
    - Revised Careers page of Beta Prototype.
    - Revised Culture page of Beta Prototype.
    - Revised Blog page of Beta Prototype
    - Revised Help Center page of Beta Prototype
-
    - Product Summary Description Component of Documentation.
    - Product Summary Superior Features Component of Documentation.
    - Product Summary Competitors Component of Documentation.
-
    - Product Summary Committed Functions Component of Documentation.
    - Contributed to Internal Code Review.
    - MIscellaneous contributions to backend API.