QuickTix

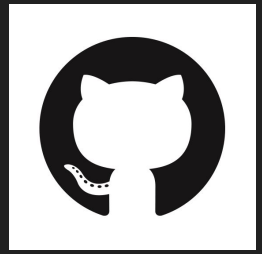
By:
Matthew Kachensky, Louis Marfone, Diego Marrero Zilenziger, Liam McChesney, Daniel Medvedev, and Cheng Zhou

Our Project:

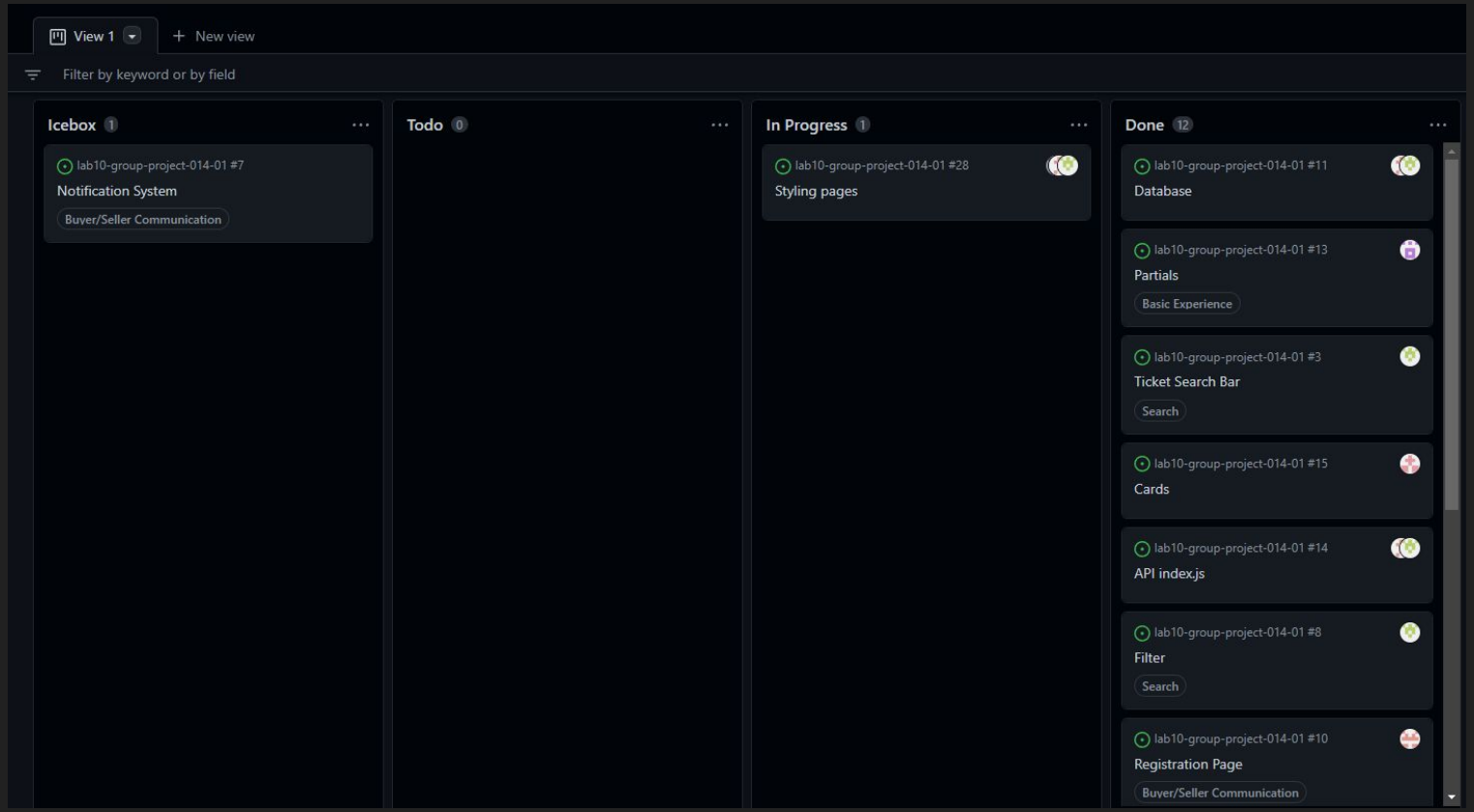
QuickTix is a platform which allows CU Students to resell their tickets for events they cannot attend. This ticket hub allows registered users to view sellers who are offering tickets, purchase tickets of interest, as well as sell their own tickets. We have also added a review system which will ensure that scammers are flagged, and both parties stay safe.

Our Tools:

- Project tracker: GitHub project board - 3.5
- VCS repository: GitHub - 4.5
- Database: - PostgresSQL - 4.5
- IDE: VSCode - 4.5
- UI Tools: HTML, EJS - 3.5
- Application server: NodeJS - 4.5
- Deployment environment: CUServer - 1
- Documentations - 4
- Testing: Postman - 2.5
- Methodology: Agile - 4



Our project storyboard:



Documentations

Description

Async functions can contain zero or more `await` expressions. Await expressions make promise-returning functions behave as though they're synchronous by suspending execution until the returned promise is fulfilled or rejected. The resolved value of the promise is treated as the return value of the await expression. Use of `async` and `await` enables the use of ordinary `try` / `catch` blocks around asynchronous code.

UPDATE

UPDATE — update rows of a table

Synopsis

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
UPDATE [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = [ ROW ] ( { expression | DEFAULT } [, ...] ) |
          ( column_name [, ...] ) = ( sub-SELECT )
        } [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```


```
<div class="card" style="width: 18rem;">
  <div class="card-header">
    Featured
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Vestibulum at eros</li>
  </ul>
</div>
```

Architecture Diagram

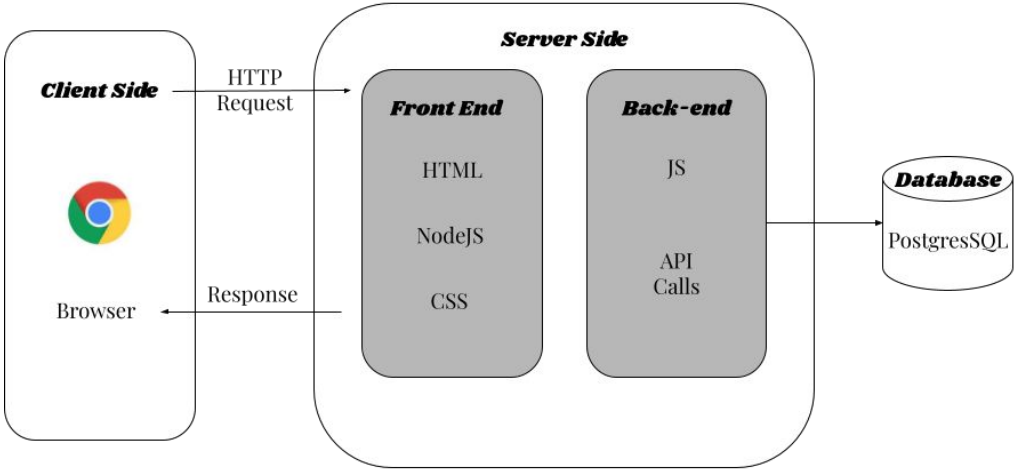
Buyer:

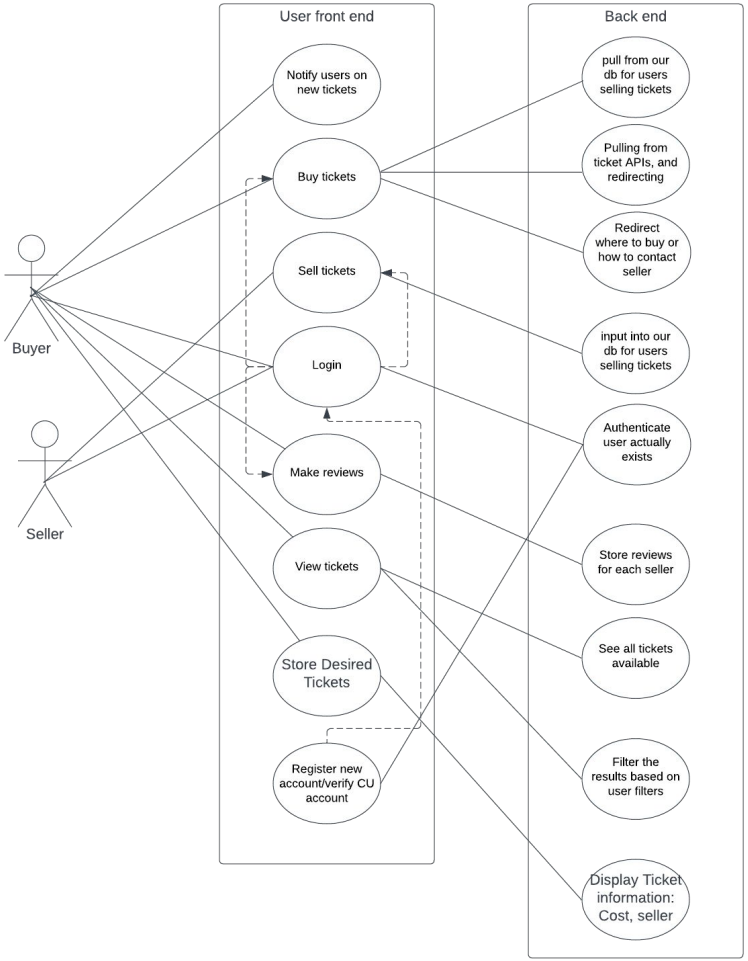
-Buy tickets, make reviews, view tickets

-Login, and store desired tickets

Seller:

- Sell tickets, and login.

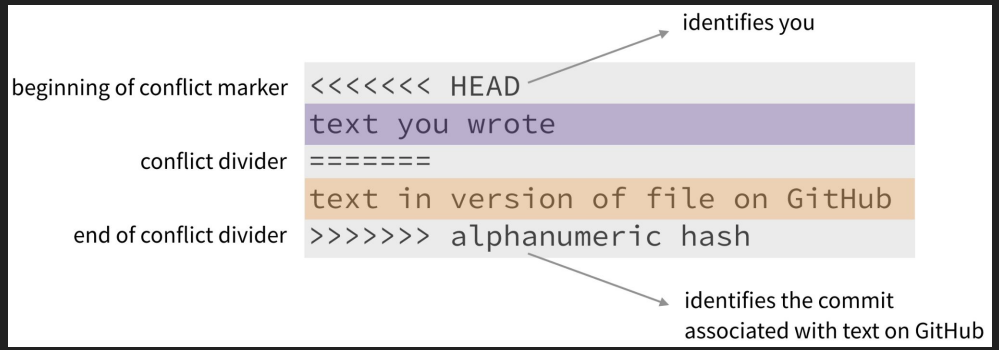




Challenges:

- Merge conflicts
- Unified ideology
- Team coordination
- API call
- Assigned work
- Logo

```
//Login page
//Loads the login page when the page is attempted to be accessed
app.get("/login", (req, res) => {
    if(req.session.user){
        res.redirect('/home');
    } else {
        res.render('pages/login', {
            logged_in: req.session.user
        });
    }
});
```



Future Enhancements

- TicketMaster API
- Administrator account
- Chat functionality between buyers and sellers
- Notification systems we never finished

Demo

- Demo Website: