DIVISIBLE

Team Members:

Jace Simons, Noah Bays, Benjamin Booker, Lucca Giannini, Owen Villas and Logan Haase

Project Description:

Divisible is a group payment application, where you can keep track of group expenses with venmo-like functionality. You can add friends, and create groups, which are all displayed on your homepage when you login to Divisible. Each group and individual friend can be easily accessed, where you can see specific payment requests. Then, using the name of the transaction, the user can authenticate their information before making a payment. Simplify splitting your tab with Divisible.

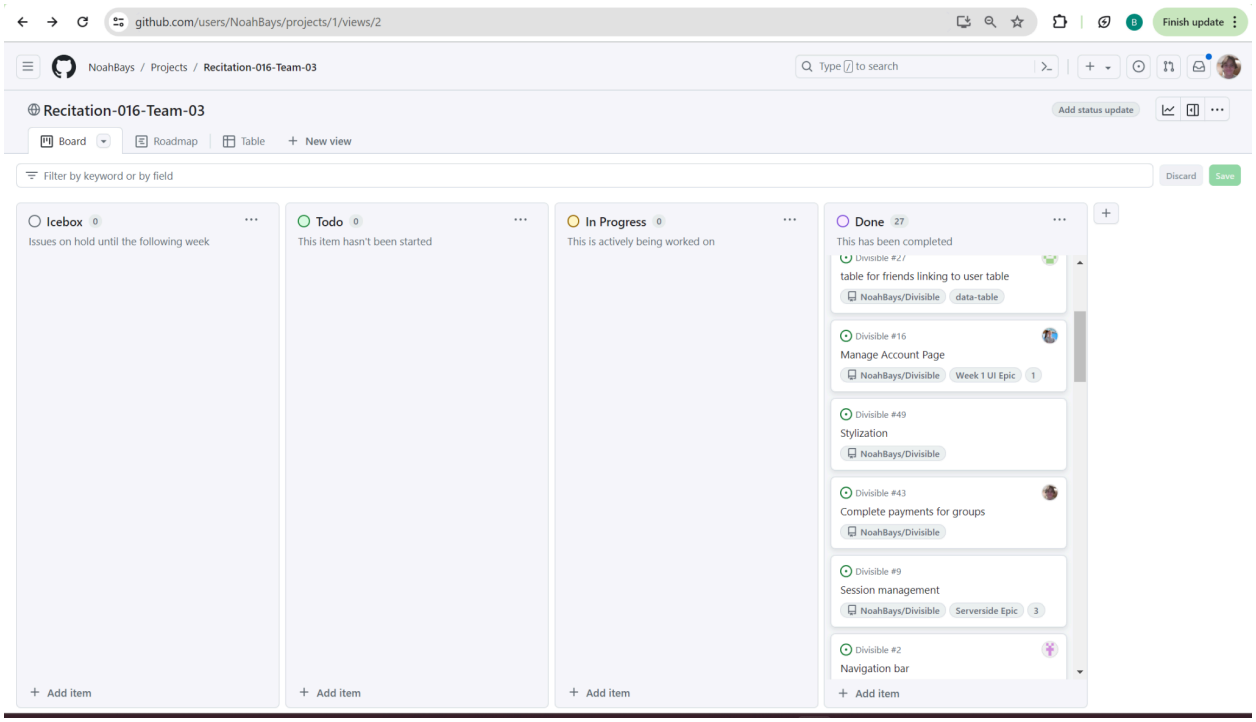
Video Demo

https://drive.google.com/file/d/1vlQhU4wfMVSAeAU-GUBbrOd-qC4G4dPb/view?usp=sharing (CU ONLY)

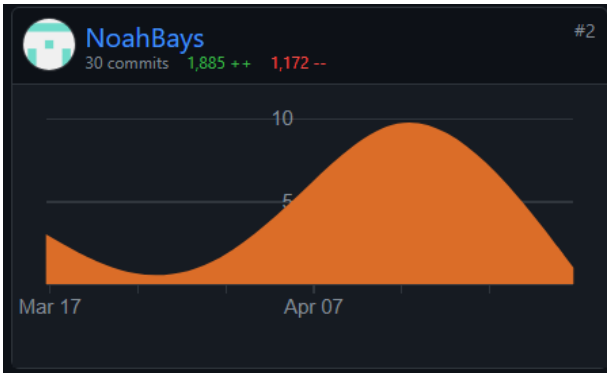
Github Repository and Project Board

Divisible: Automated expense sharing for groups (github.com)

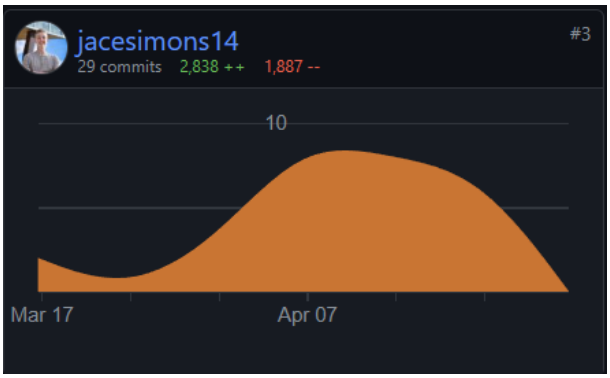
Board · Recitation-016-Team-03 (github.com)



Individual Contributions:

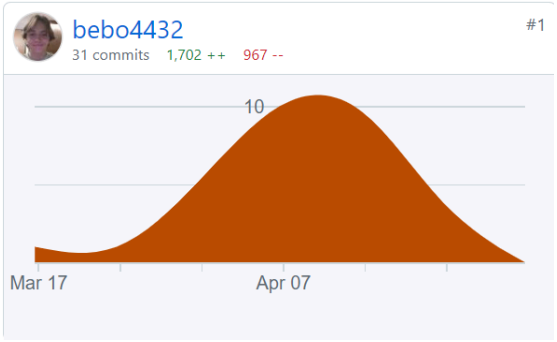


Noah Bays: I created the home page html and API, detailing all of the user's info on the home page such as friends, groups, and transaction history. Additionally, I created all elements regarding the request system, including the HTML and API for the notifications and requests for individual payment. I also created most of the insert SQL.

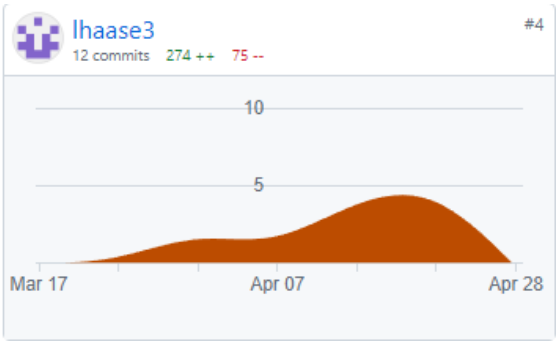


Jace Simons: I set up the initial index.js with basic register and login

routes, as well as some other basic endpoints, like page rendering. I then rewrote the register and login endpoints as we were having trouble with them and their error handling. I also created the manage account page, and wrote all the register endpoint tests for Lab 11. I also refactored user id out of create.sql and its relations. I then redid every page in the application with css and html, and most of all, handlebars, to match the design style that I came up with in week 2.



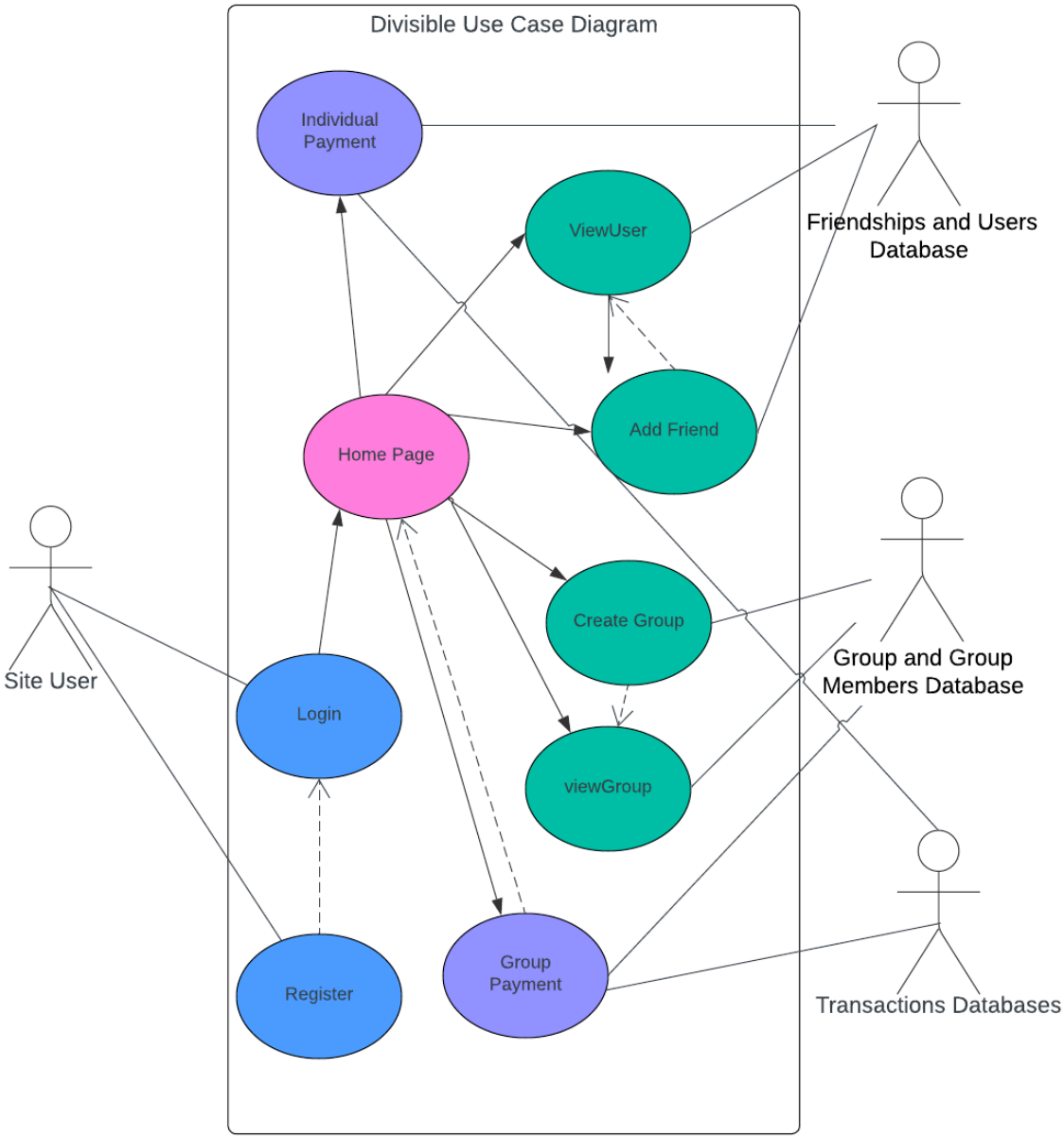
Benjamin Booker: I created the createGroup page and all of the functionality, including several tweaks to the group and group members SQL databases. I also used jQuery event listeners for several buttons and wrote corresponding index.js API routes, including one to pass information to the viewGroup page Owen was developing. I also created the viewUser page, and added some additional data to the user database such as profile picture and email. I also did Lab 13.



Logan Haase: I created the basic handlebar pages for the login and register pages. Also, I created the addFriends page and the functionality that went with it. Lastly, I worked with the insert SQL a little to create fake accounts so we could experiment more with our site.

Owen Villas: I worked on developing the database for the most part. Made sure every table had the right variables to capture all of the information we wanted to. I also worked on the routing and display of the individual group pages, with a lot of help from Noah's home routes and Ben's addUser routes.

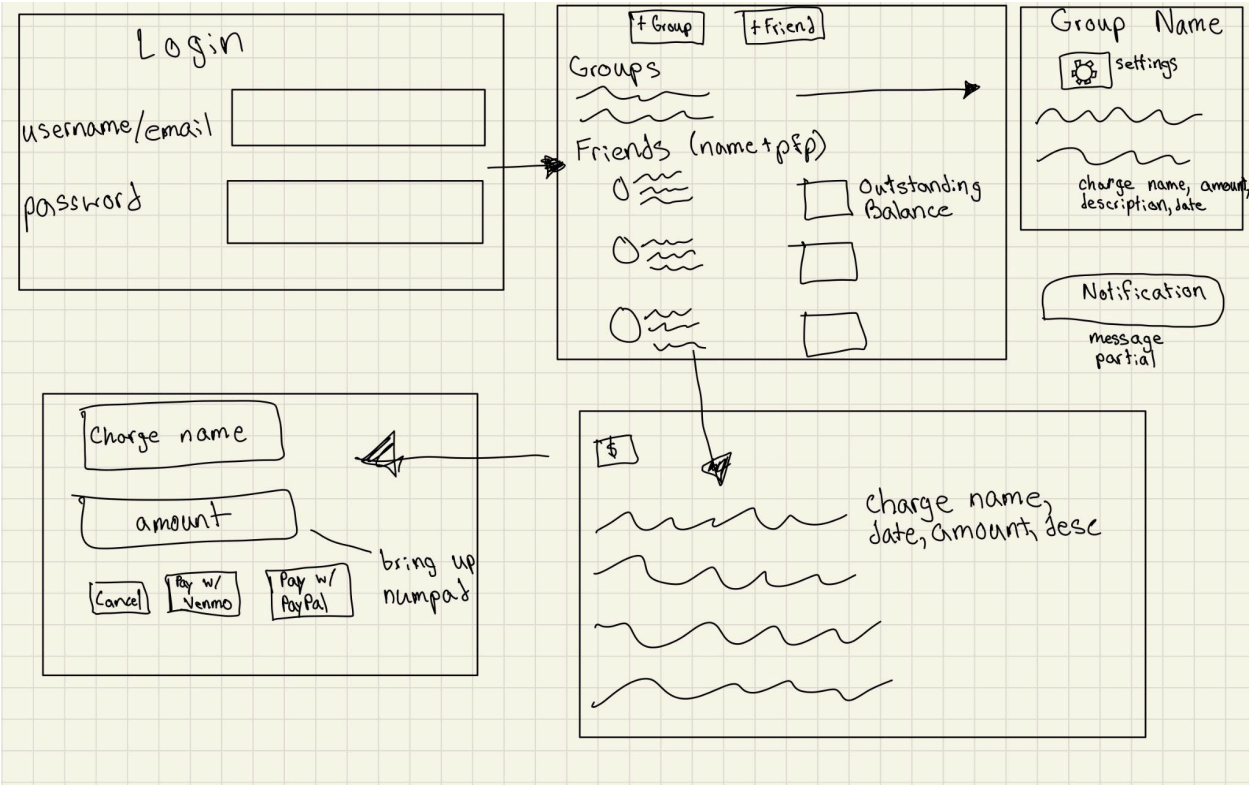
Lucca Giannini: Worked on finding a payment API initially before deciding to use fake money. Planned data storage for our payment methods. Helped debug login endpoints for lab 11 tests. Helped plan our website functionalities and development timeline. Developed all of the individual as well as group payment and request systems using endpoints and transaction (SQL) tables. Finally, created payment windows and integrated functionality on each respective page.

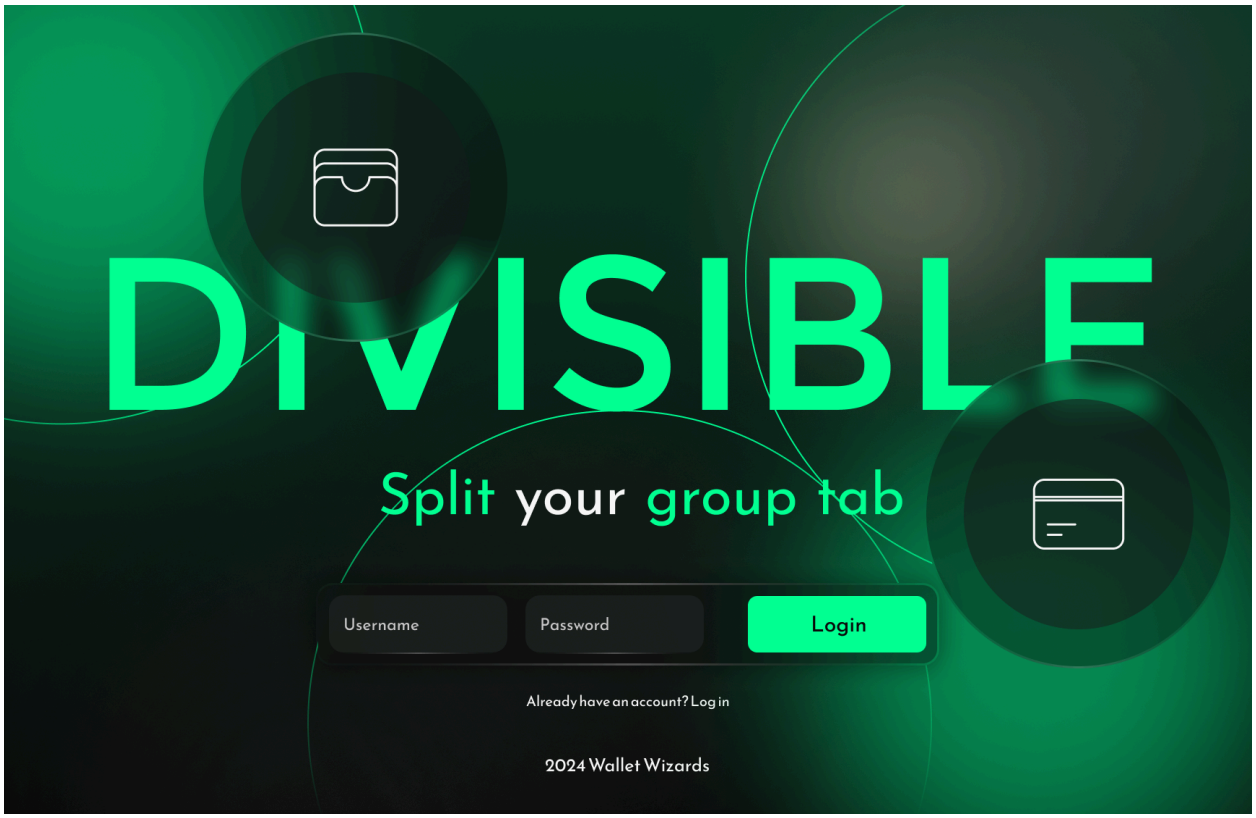


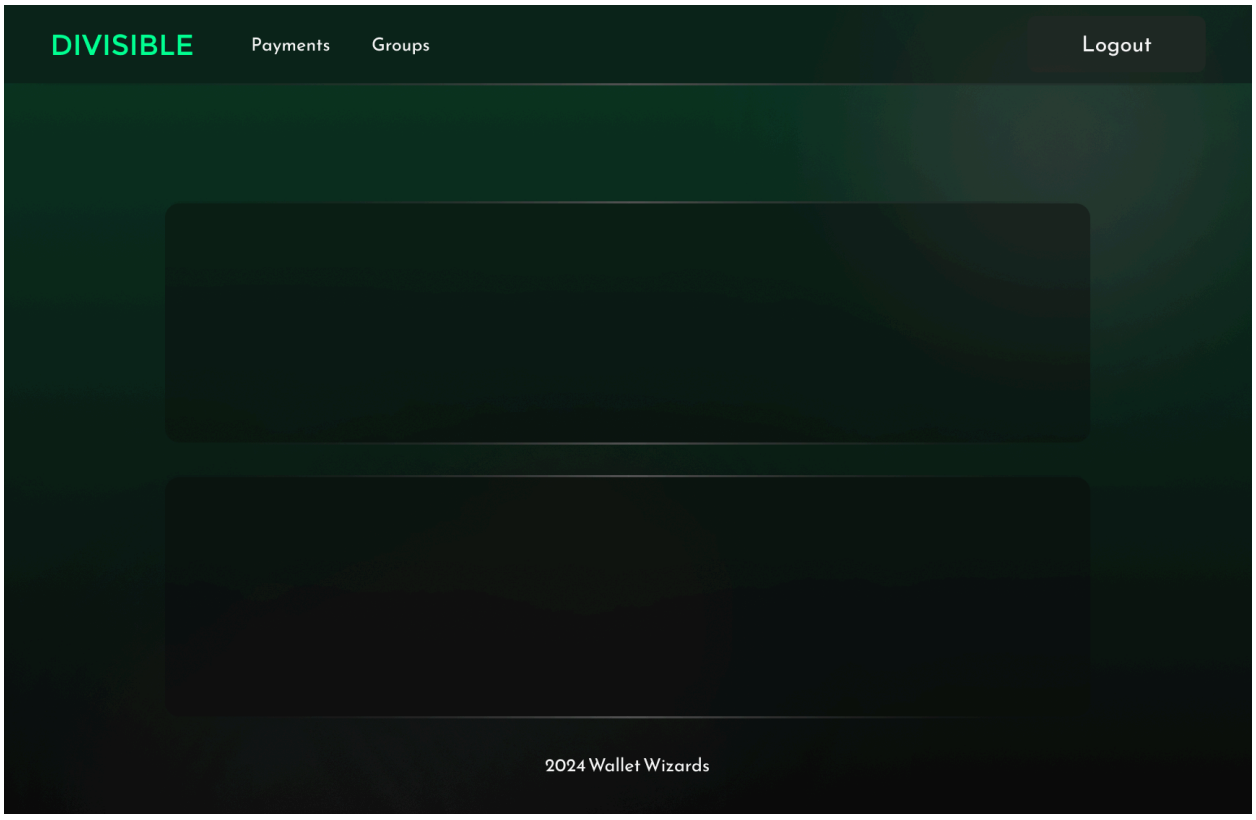
Wireframes

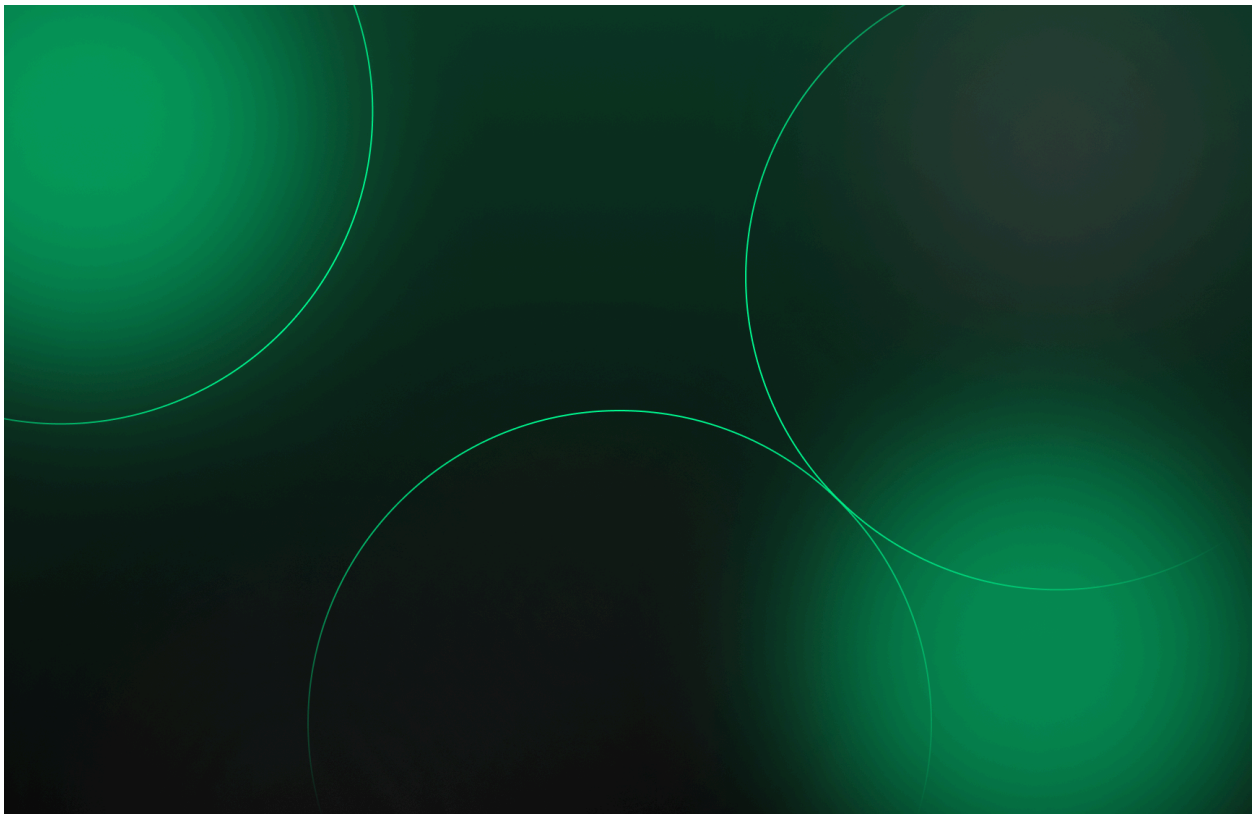
In week one, as a part of Lab 11 we drew out a wireframe that helped us establish the functionality of Divisible and begin working on our code. In week 2 we spent a little bit of time establishing a more distinct design style, with the distinct green and black color scheme. We then took it to Figma,

where we made some admittedly simple mockups for the login and register pages, as well as a generic setup for the content pages. You'll see that when it came time to implement in our actual application, we were able to replicate the mockups almost perfectly with CSS and handlebars, something we are very proud of. We also used Figma to create this simple page background, that we were able to use as a static png background in the application. We also made one additional background, which is the same thing but without the foreground green rings to use as the background for the content pages.




DIVISIBLE
Split your group tab
Username
Password
Login
Already have an account? Log in
2024 Wallet Wizards


DIVISIBLE
Payments
Groups
Logout
2024 Wallet Wizards



Testing:

1. Add Friends Test: The user should be able to add friends if providing valid friend credentials

- Test results: The user was able to add friends without issues. They selected the username, added them, and they were now visible as friends on the home page.
- Observations: The user intuitively added a friend without issues.

2. Group Navigation Test: Navigating to a specific group should present a page that contains all of the users in the group and the overall balance, as well as the most recent transactions on the group wallet

- Test results: The user was automatically navigated to the group page after group creation, and after attempting to do a group payment returned by clicking on the group name on the home page.
- Observations: Redirecting to the group page after creation is good. Slight delay in attempting group payment as they thought it would not be in the nav bar. Creating a group payment/request within each group page may be a good idea.

3. Transaction Window Test: Making a transaction (charging someone or paying them) should provide a payment window for that specific user/group. Once submitted, the funds assigned to the sender and the recipient should be updated accordingly.

- Test results: Navigated to the payment group via the nav bar. Entered all the info correctly, but got confused after submitting a group payment. Then tried requesting a negative amount, throwing an error message. Next tried individual payment, entered all info correctly including the payback checkbox, then was confused after the home page displayed an outstanding balance.
- Observations: Group payment and individual payments work as intended. Correctly throwing errors and changing balances. However, functions for users are not clear. May need more information on what they do on payment pages.

4. Login Test: The user should be able to log in when providing the correct credentials
- Test results: The user typed in their newly registered account username and password into their respective forms and logged into the home page successfully.
- Observations: The user intuitively logged in without issues.

Deployment Directions

Copy the github repository and navigate to the ProjectSourceCode folder. Make sure you have npm and docker dependencies installed, and run docker compose up -d to initialize the PostgreSQL database. More detailed directions can be found in the Readme in the github repository, linked above.