LANTERN

Presented by Aiden Gonzalez, Shibi Balamurugan, Tosin Ogunjobi

The Lantern Team

Shibi Balamurugan

Senior
Computer Engineering

Backend and Frontend

Aiden Gonzalez

Senior
Computer Engineering

API and Backend

Tosin Ogunjobi

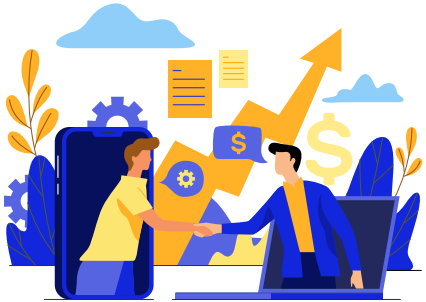
Senior
Computer Engineering

Frontend and UI

What is Lantern?

Overview

- A free, open source accounting and financial planning web application designed for small businesses
- **Why ?** According to US Bank, **82%** of small businesses fail due to poor cash flow management and organization
- Lantern aims to give a clear insight of financials for businesses owners with provided features and analysis
- In the future, we hope for small businesses to utilize our service and collaborate with other developers to further advance the application



Project Goal

Create an interactive platform to allow users to view their financials through dashboards and reports and performs in real-time.

S.M.A.R.T. Requirements

Plaid Connectivity: Utilizes Plaid, reliable and secure fintech platform, to allow for seamless integration into user's financial life

100% → 100%

Insights: Widgets, tools, and dashboard components to give user instant insights into their organization's financial well-being

90% → 100%

Reporting: Accurate financial reporting using user's current financial data

10% → 100%

Dashboards: Intuitive, customizable dashboards with accurate at-a-glance statistics

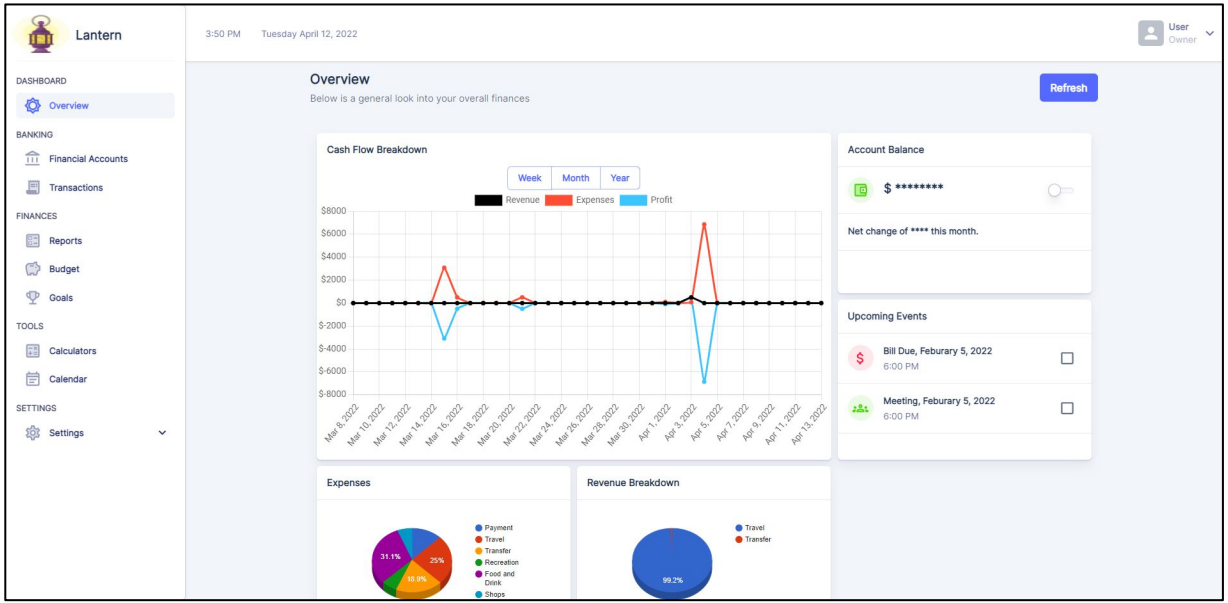
50% → 100%

Performance: Responsive interface and robust backend allow for seamless interaction and a highly scalable number of concurrent users

50% → 100%

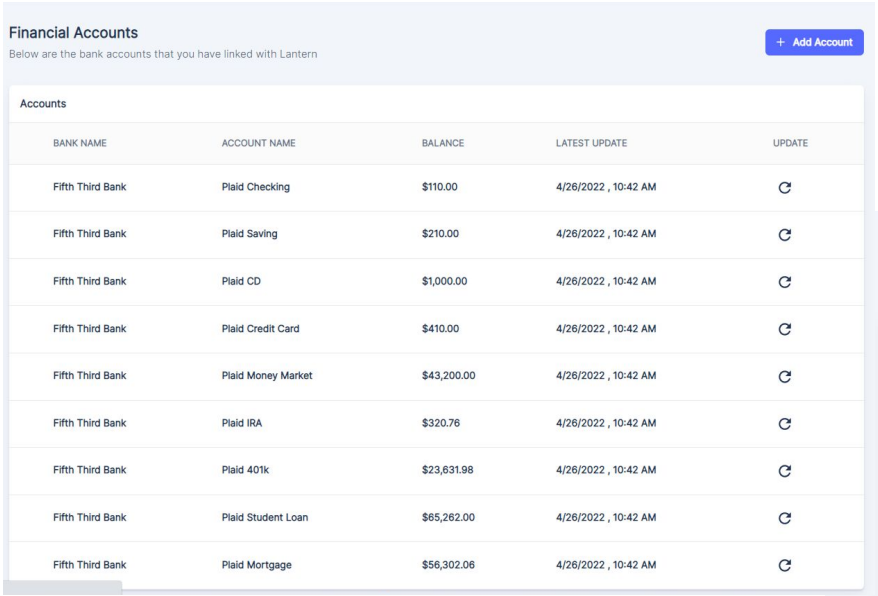
Features

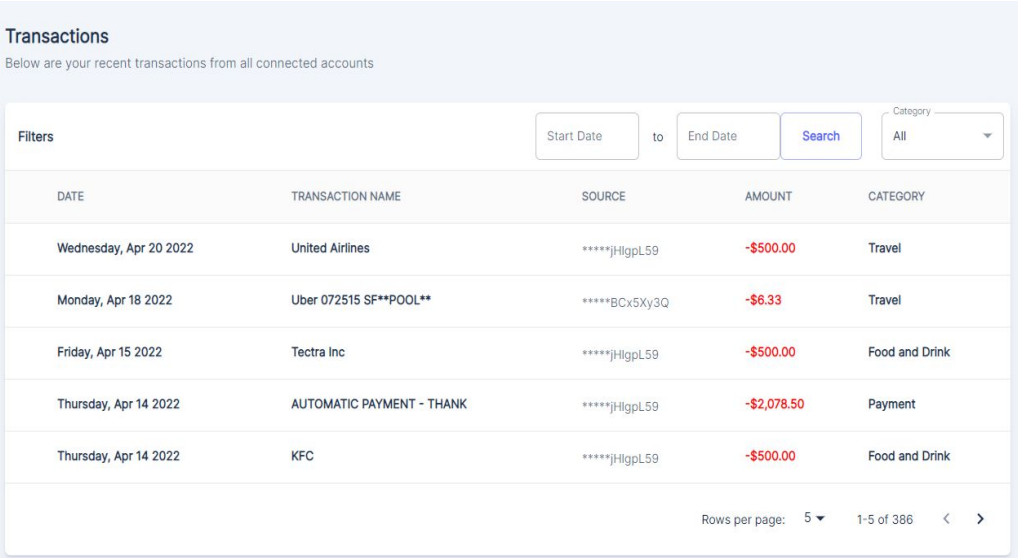
Dashboard -



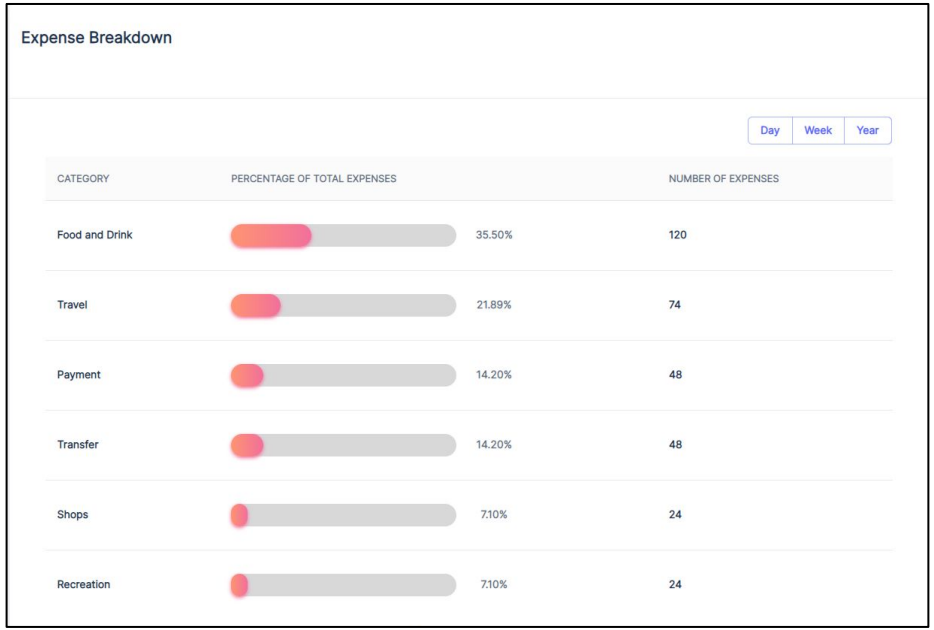
- Total Account Balance
- Widgets -
  - Expenses Breakdown
  - Revenue Breakdown
  - Cash Flow Breakdown
  - Upcoming Events
- Draggable Components

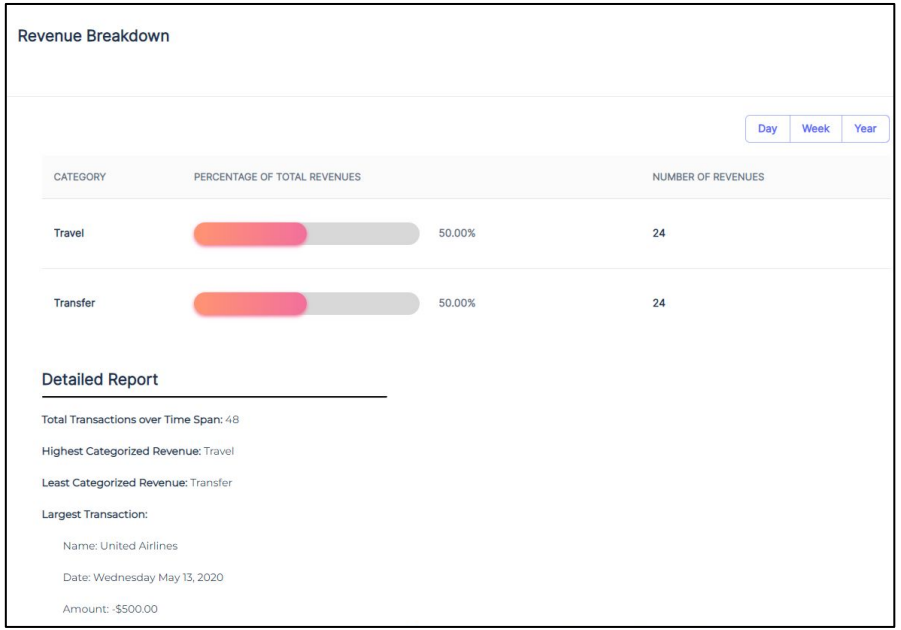
Financial Accounts and Transactions -





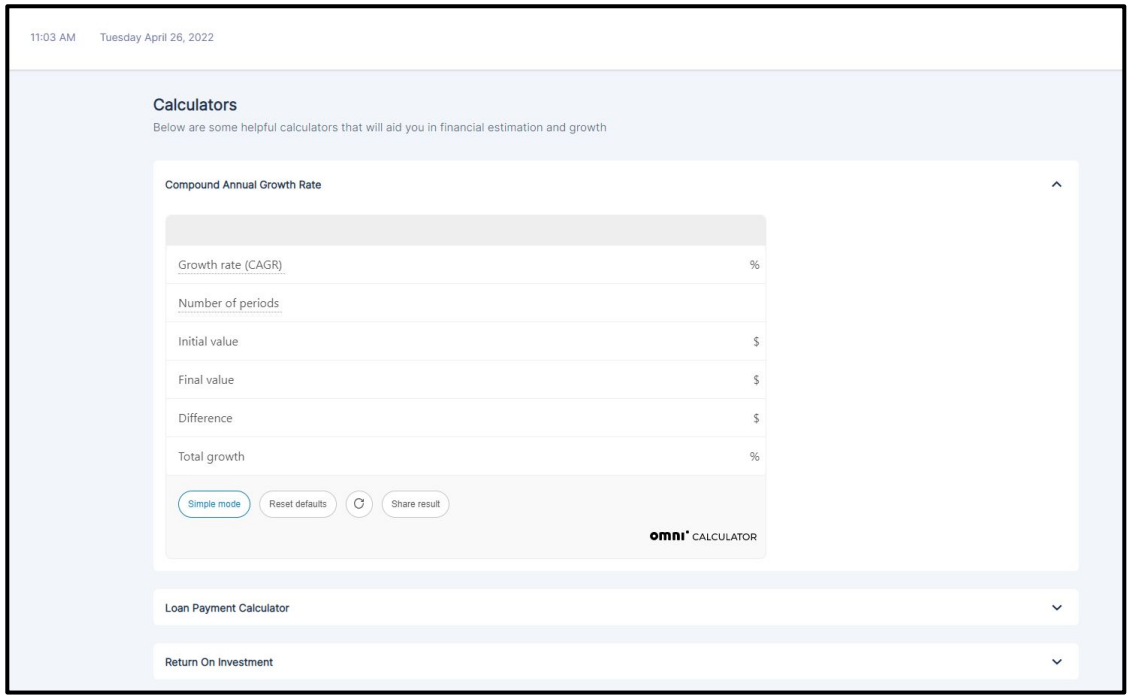
Financial Reports -





Tools -

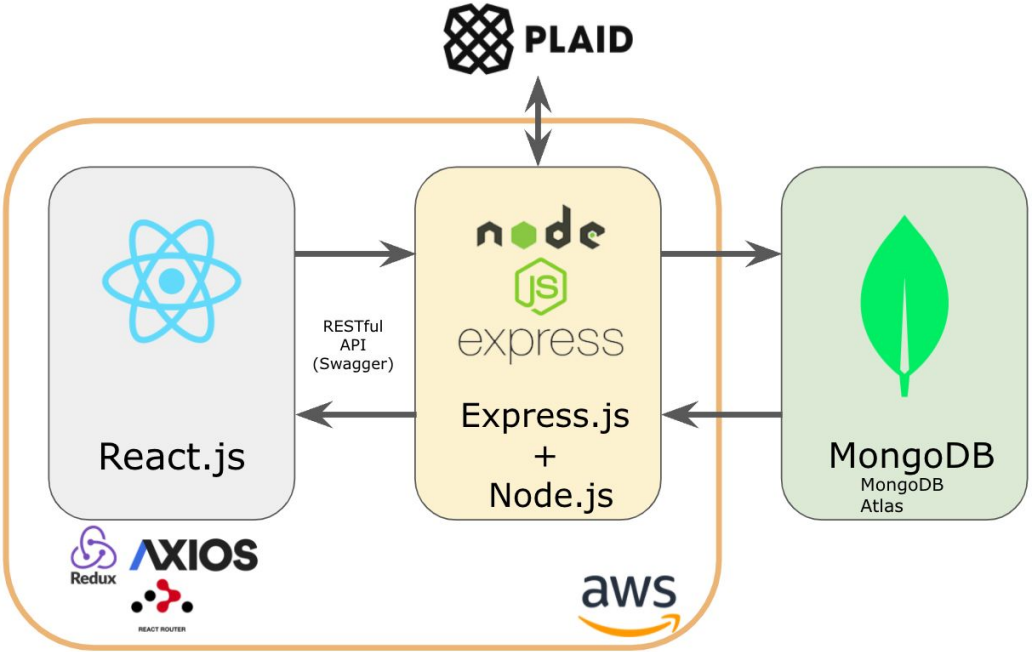
Calculators



Calendar

Software Architecture

MERN Stack



- MERN
  - Frontend - React
  - Backend - Express + Node
  - Database - MongoDB
- MongoDB Atlas - Managed cloud NoSQL storage
- AWS hosting + EC2
- Plaid - Access financial data
- 100% JavaScript + TypeScript

Open Source Development

CI/CD

- **“Build” system:**
  - Node Package Manager (npm) used for managing dependencies for both the frontend and backend applications
  - Test and build commands used to automate common but important tasks
- **Continuous Integration:**
  - Used GitHub Actions workflows to automatically measure and enforce S.M.A.R.T. goals
- **Continuous Deployment:**
  - One-click deployment pipeline leveraging AWS CodeDeploy and PM2 to deploy and manage a hosted Lantern instance

Product Vision

Future of Lantern

- Beta Release v1.0.0
- Making issues that outside contributors can work on
  - Detailed user profiles
  - Business accounts with multiple users
  - More robust accounting features
  - Custom data import (outside of Plaid)
- Fully deploy product onto AWS infrastructure
- Alternate methods of product delivery
- Upgrading Plaid Credentials for user data

Skills Learned

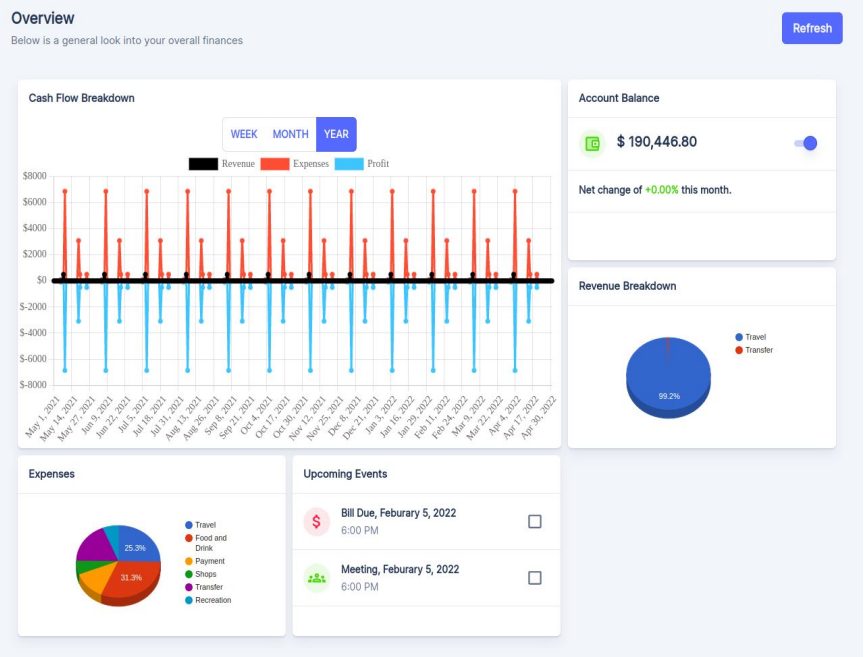
- Technical
  - MERN Stack Experience (Mongo - Express - React.js - Node)
  - Designing and Writing Unit Tests and performing integration/end-to-end tests
  - Git/Github (50 PRs)
  - CI/CD

- Non-Technical
  - Market/Competitive Analysis
  - Long-term Project Management (94 completed issues)
  - Communicating Feature Requirements

Technical Challenges and Solutions

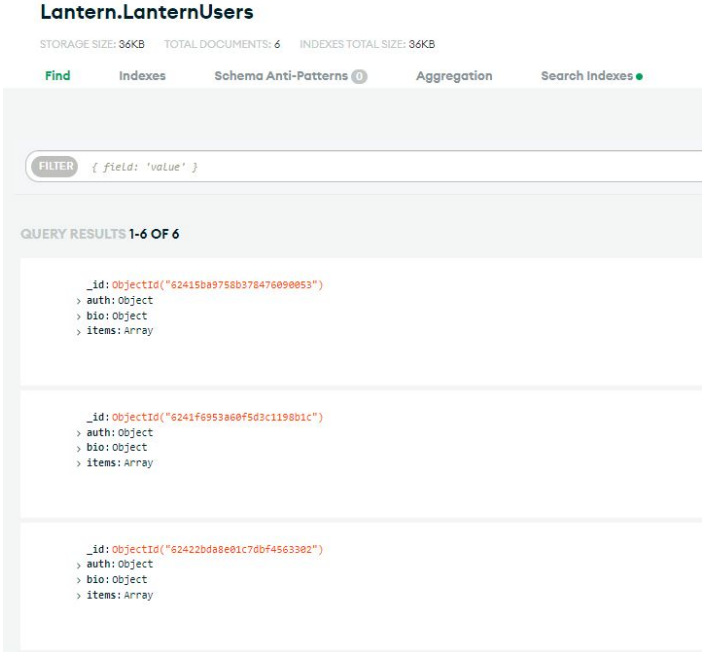
Performance Issues

- Fetching a user's transactions was very slow, and was performed every time the user navigated to a different page or refreshed the current page
- Created a solution to pull in transactions in smaller batches, so that the UI can begin render useful information after just the first batch
- Redux saved transaction data to be shared with other components
- Automatic refresh periodically, manual refresh button available if desired



Database Testing

- Unit testing with MongoDB required us to create and teardown MongoDB collections
- Used temporary in-memory database for testing
  - Allowed us to make our tests much cheaper (less time and less Atlas cost)
  - Also could avoid any potential network issues trying to connect to Atlas
- For more transparency, we added a log of the in-memory database to be saved to the local disk for post-test inspection and analysis, as-needed

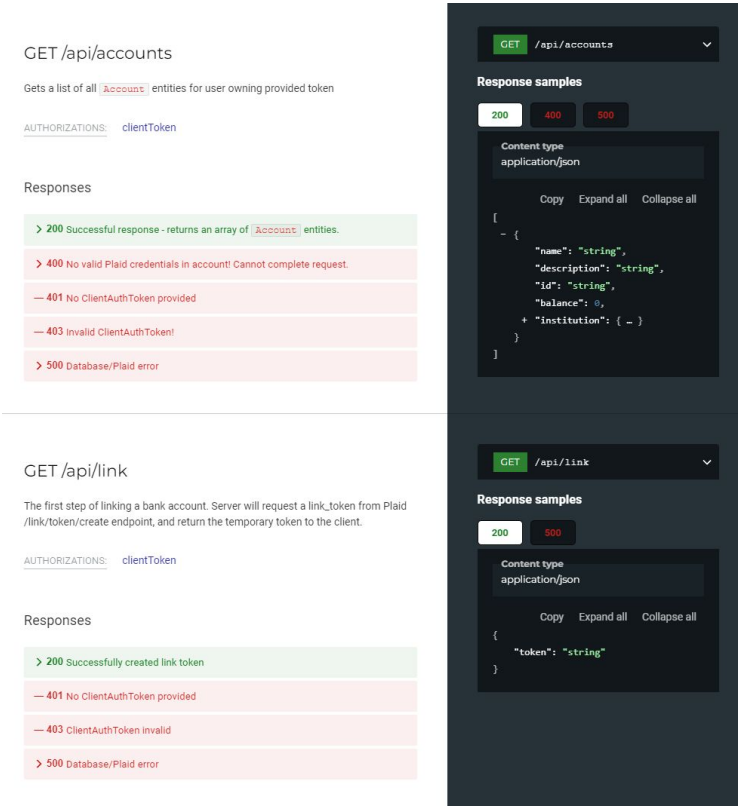


Standardizing Client-Server Interaction

- Working on frontend and backend in parallel
  - Needed a clearly defined API specification
  - As long as the interface was clearly defined, both sides could implement and test new features independently

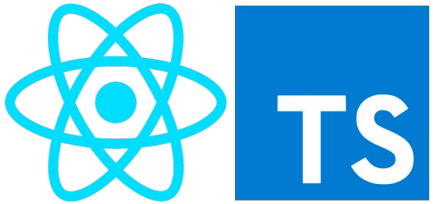
- Used open-source API design tool “apicurio”
  - Easy graphical interface to define OpenAPI specification
  - Automatically generate documentation, code snippets, and tests

- Hosted site enabled collaboration, so that the spec could be an evolving document throughout development



Converting Ideas to Code

- After brainstorming new ideas, we had to find ways to implement them using Typescript and React
- Front-End Development very different from ECE undergraduate courses
- “What Should We Do Next?”
- Using online resources and asking questions was a very productive to come up with solutions



DEMONSTRATION

Questions ?

THANK YOU FOR LISTENING