Gello

Jasmitha Gourabathini, Wren Gurung, Payton Hsu, Joanne Kim, Bradley Wistehuff, Ronald Gospel Jangam

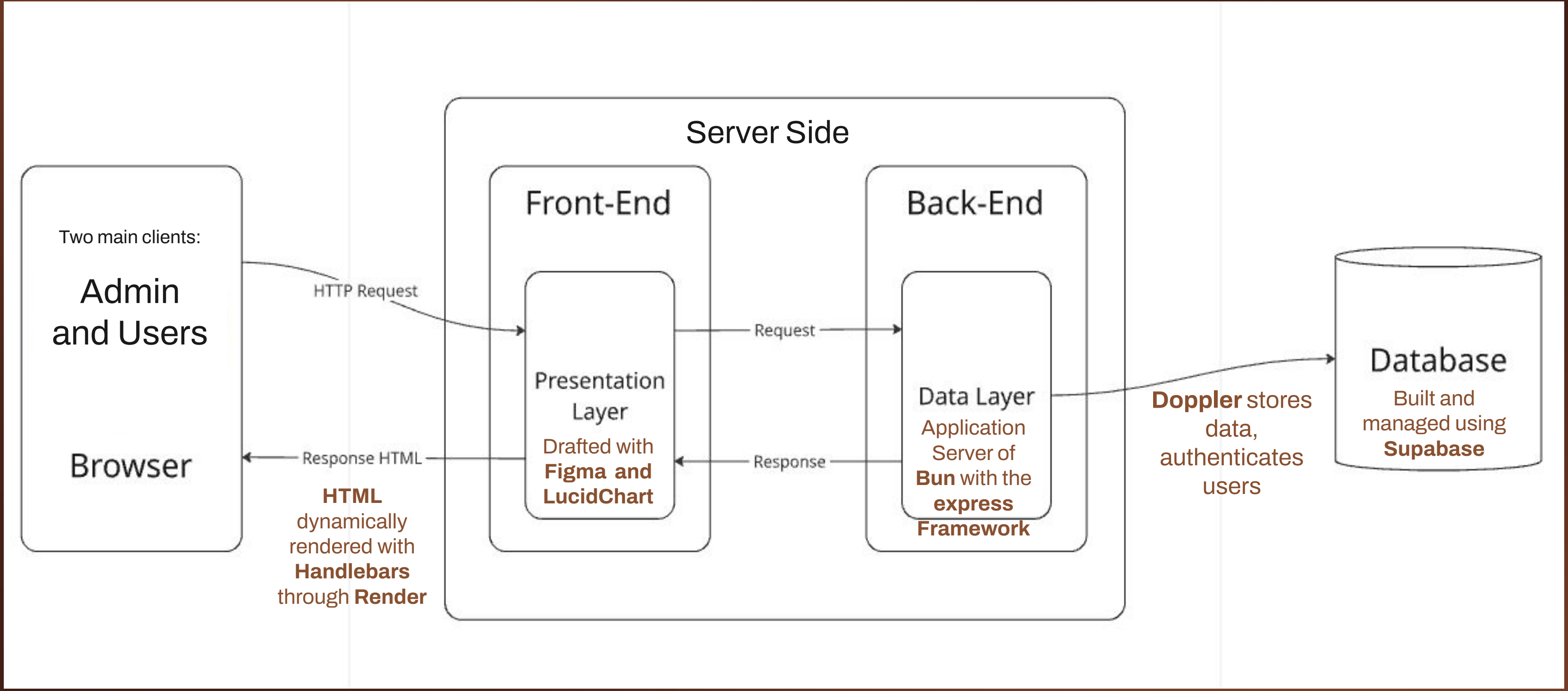
What is Gello?

- ★ Focus on productivity and team organization
- ★ Admin roles to organize teams and assign tasks/goals
- ★ Team member roles to complete tasks and earn points
- ★ Security and authentication: access is limited to specific teams in case of use of sensitive data.

What makes Gello Unique?

- ★ Gamifies productivity by using a point system
- ★ Team members are awarded points based on task difficulty and priority
- ★ Introduce friendly competition and motivation into the workplace

Architecture Diagram


Server Side
Two main clients:
Admin and Users
Browser
HTTP Request
Response HTML
HTML dynamically rendered with Handlebars through Render
Front-End
Presentation Layer
Drafted with Figma and LucidChart
Request
Response
Back-End
Data Layer
Application Server of Bun with the express Framework
Doppler stores data, authenticates users
Database
Built and managed using Supabase

Future Scope/Enhancements

Introduce a Timer to Earn Points

User a timer API for users to be able to track how much time they spend on a project and earn points.

Summary Statistics

A page for admins and teams to see how their completed tasks over time and how many points they have accumulated.

Co-admin positions

Add a more in depth role system, that way more than one person can have some perks of admin abilities. That way more than one person can do admin tasks to avoid overexertion.

Implement Teams API

Implement a place for people to be able to be able to have online meetings on the app in a more corporate workplace, and so people can earn points for attending meetings.

Challenges

Incorporation

Actually trying to implement all of the features was difficult. We had to give up a few features, such as forgoing using the Trello API.

Backend and Frontend Integration

Originally we had a lot of problems trying to finalize decisions regarding features. Therefore, it led to issues when integrating the backend with the frontend.

Stack Changes

Originally, we were going to use different tools for our frontend and backend. However, we had to change this at the start of the project, which meant we had to spend a portion of the time migrating to using other tools.

Communication/Timing

It was difficult to plan meeting times with six people, as well as communicate clearly when everyone had a different idea of what was going on.

Tools

Project Tracker: GitHub Projects	It was fairly easy to use, but it was a bit of a learning curve	★☆☆☆☆
IDE: VSCode	Most of us familiar with using VSCode, and we liked the connectivity with Github	★☆☆☆☆
UI tools: Handlebars, Figma	Figma created an easy way to create a wireframe for the application. Handlebars provided a templating system which was nice for organization.	★☆☆☆☆
Application Server: Bun	Javascript runtime with same functionality with node.js in lab, so it was not very hard to use	★☆☆☆☆
Deployment Server: Render	Render was easy and straightforward to use according to the lab instructions.	★☆☆☆☆
VCS Repository: GitHub	It was a lot harder to use for a larger scale project.	★★☆☆☆

Version Control: Git	Fairly easy to use and most of us had experience with Git. We just had to make sure we could collaborate effectively.	☆★★★★
Doppler	Secrets could have just been added into the env file, and caused some issues	☆☆☆☆★
Database: Supabase	We had lot of errors with connection.	☆☆☆★★
Testing Tools: Bun, Playwright	The testing format was a bit on the confusing side, and it was also a learning curve	☆☆★★★
Framework: Express	We could use the skills that we learned in lab. Other ways to do a POST and GET request would be a lot more complicated	☆★★★★
Auto-Documenter: Gello Release Bot, Dependabot, Semantic Versioning	Not necessary for a four weeks project.	☆☆☆☆★

Live Demo of Gello

Q & A