Introduction

Since my early high school years, developing websites for friends and family has been my way of generating my pocket income. Throughout the years, I've run out of family and friends to make sites for and since then, I've expanded to building sites for clients I have never met. As the clients became unfamiliar and increased in number, keeping track of each project's requirements, deadlines, finances, documents and agreements became increasingly difficult. The sheer number of interrelated and cross-referenced data about a project made it impossible to use sheets without formulas that induced circular dependencies.

To resolve the issue and build myself a platform to ease this problem, I met a software engineer friend of mine to ask about how he handled the situation about his freelance work. He suggested that I build a platform where my clients could sign in, enter details about their projects which would then allow me to assign subscriptions, add related documents, agreements and invoices.

I needed to automate financial transactions for which he offered I use Stripe for. That would automate my invoice processes and subscriptions, taking away some of the difficulty of implementing that. I finally mentioned that some of my clients had problems communicating with me because some clients I had had lots of employees who talked to me. Together we decided on a multi-tenant dashboard and shared data including a messaging system to keep track of all requests.

See Appendix A for a detailed discussion transcript.

Rationale

To simplify development and reduce the hosting costs on my side, I will be using PHP which offers flexibility for deployment even on very low-resource servers. Due to the high number of relationships, Laravel framework will be utilized which provides an easy-to-implement ORM with powerful but simple relationship features. This will allow me to develop models that are easily interlinked and provide an interface that allows the user to find relevant information very easily by navigating through relationships

The system will mostly be made up of CRUD pages so to accelerate the development, the library Filament PHP will be used which will serve as a boilerplate for the CRUD pages and the dashboard interface. Custom interface components will be created to facilitate some case specific fields. If any new type of data is required, it will be simple to implement.

For managing payments and financial transactions, Stripe will be used. Stripe will offer invoice generation and automated subscription billing. If there is a need to cancel a failed subscription, it will tell the app directly, removing the complexity of payment retries.

Success Criteria

- App allows for authentication through a SSO provider into a tenant that represents a company
- Users can request projects, make updates and the admins can provide quotes
- Admins can add platform suggestions to projects
- Users and admins can add requirements to projects and cost and time for each requirement can be adjusted
- Admins can assign users with subscriptions which contain information about the service details and features offered by a subscription
- Users can make project and subscription payments through the dashboard and the invoices are generated automatically and the project / subscription statuses are updated respectively.
- Signatures are collected from each payment and are available for the admins to see.
- Agreements can be requested by admins and can be signed by users.
- Documents can be requested by admins and can be uploaded by users.
- Users may create support cases and everyone in the tenant may add messages to a support case as well as the admins who can reply.
- Admins can see activity logs of changes made to records.
- Users can add payment methods and these saved payment methods can be used to make payments.