Software Design and Engineering

Lab Document

High Level Purpose Statement:	The goal for this lab is to create a fully functional SaaS boilerplate. The boilerplate should be able to process billing through Stripe, Send email receipts with Resend, handle personal info with Supabase, and most importantly use NextJS for server side rendering (less work than Express).
Experimental Design:	**NextJS, Supabase, Stripe, Resend, ShadCN, NextAuth, Cursor Code Editor (Built upon VSCode)** ● I will first use Vercel's very simple hello world boilerplate so that the app is correctly setup ● I will then create the SaaS landing page with a design similar to popular ones available, and I will include Stripe plans on this page ● From the landing page, I will set up an authentication flow so that Supabase is used to store authentication and verify users. ● Within the authentication flow if a user is wanting to access the SaaS, authentication will be handled with NextAuth. NextAuth will grant the verified user a Session which will hold usable data when needed later. ● Once the authentication flow is in place, the last thing really needed for the templated landing page is the plan functionality with Stripe, a user should be checked for authentication before accepting a plan, if so, they are redirected to the billing page where they can then access the Stripe plan. ● For this to be a technical SaaS boilerplate, I need to then make the dashboard panel where the user can access the products, billing, settings, and more. ● The main focus of the dashboard is to establish a functioning billing plan with Stripe, then Resend will be used to offer receipts to their Stripe subscription.
Resources Available:	https://github.com/ixartz/Next-js-Boilerplate https://shipfa.st/ Primary sources for boilerplate examples. Tech stack inspired from ShipFast.
Time Estimate:	The project took about 30-40 hours to create the boilerplate, but Cursor was very helpful in reducing the time it would have to create most pages/functionality by hand.
Experiment Notes:	● NextJS makes it very easy to use API's

	• Supabase has verification for emails (didn't know this previously) • Supabase was VERY easy to learn because of our PostgreSQL lab. • Because Next.js uses static site generation it was hard to figure out where to wrap session providers from NextAuth, so that a user's plan is updated on the page after checkout.
Results:	The project now properly runs using NPM and Next.js to provide a fully functioning SaaS boilerplate.
Consequences for the Future:	Rethink where static site generation should be used other than wrapping a component with the session. If I use static site generation with a wrapped session, the site takes longer to load. On the other hand if I only wrap the component, the site loads quickly, but the component will load while the site is active. All in all, think about how loading times could effect UX.