Tech Stack Recommendation

To maximize speed and quality for a solo developer building a polished SaaS, we recommend a modern, full-stack JavaScript/TypeScript solution with these components:

- **Frontend: Next.js (React)** with TypeScript. Next.js provides an SEO-friendly, responsive framework with server-side rendering support. For styling, use **Tailwind CSS** (utility-first, highly customizable) with **Shadcn/UI** (Radix-based React components) to accelerate UI development. Add **Framer Motion** for smooth animations and micro-interactions. This combination (React + Tailwind + Radix/Shadcn + Framer Motion) is used by leading SaaS templates. It ensures a **responsive, animated, "3D" feel** with minimal boilerplate. Use premium fonts (via Google Fonts or Adobe Fonts). Components like Tabs, Dropdowns, and Modals can be sourced from Radix/Shadcn for accessibility.

- **Backend: Node.js** with **Express** (or a lightweight framework). Node.js is a top choice for fast MVP development and lets you use JavaScript/TypeScript end-to-end. Alternatively, use Next.js API routes to avoid a separate server. The backend handles API routing, business logic, and integration (Twilio, PDF). Use **TypeORM** or **Prisma** for DB access with PostgreSQL. These libraries speed up development and ensure type safety.

- **Database: PostgreSQL** – a powerful open-source SQL database ideal for structured data like payments and attendance. It supports transactions (important for payments). For faster MVP, consider **Supabase** as a hosted Postgres platform with built-in auth and APIs. Supabase ("Firebase for Postgres") can handle user management and real-time features with minimal setup.

- **Hosting:** Deploy frontend on **Vercel** or **Netlify** (easy Next.js hosting). Run the backend on a managed Node platform (Heroku/AWS Elastic Beanstalk) or serverless (AWS Lambda, Google Cloud Run). Use managed Postgres (AWS RDS or Supabase). For CI/CD, use GitHub Actions. Ensure HTTPS via Let's Encrypt or the host's certificates.

- **SMS/WhatsApp Integration:** Use **Twilio** or **Vonage** APIs to send SMS and WhatsApp messages. These provide Node SDKs and webhooks. Integrate with a reusable service class. Track delivery receipts via the API callback.

- **PDF Generation:** Use a Node library like **PDFKit** or **Puppeteer**. For example, render an HTML invoice template and generate PDF via Puppeteer. This allows rich styling (charts, logos) in the receipt. Alternatively, a service like **PDFMonkey** can generate PDFs via simple API calls.

- **Analytics:** Instrument the frontend with Google Analytics (GA4) or **Mixpanel** to track user flows. Use backend logging (e.g. Datadog, LogRocket) for error monitoring. For open-source, consider **PostHog** for self-hosted analytics.

- **Developer Productivity:** Use TypeScript everywhere for reliability (node, React). Leverage component libraries (Shadcn, Tailwind) to cut CSS time. Use Docker for local dev parity. JavaScript/TypeScript, React, PostgreSQL and Docker are all modern tools with huge communities and support. This ensures you can find help and plugins easily.

- **Security Libraries:** Use packages like **bcrypt** or **argon2** for hashing, **helmet.js** for Express security headers, and **csurf** for CSRF protection. For auth, consider **Passport.js** or **next-auth** (with OAuth providers if needed).