Bestimator

PART I -- MockUp/Prototype

https://www.bestimator.ca/dashboard/overview

PART II -- Technology Requirements (as a Word document or a pdf) (max. 4 pages)

A) **Technological Requirements**

- Create a tabular (i.e. table of rows and columns) representation of the technology (for e.g., Database, Programming languages, Framework, Hardware, etc.) that is being planned by your team to build the application/s.
- State pros and cons why each technology is best fit for your application.

Category	**Tool/Framework**	**Pros**	**Cons**
Frontend Framework	Next.js	SSR, SSG, and API routes built-in; optimized for performance; Vercel-native support	Learning curve for SSR/SSG if coming from CSR frameworks
Language	TypeScript	Type safety, better tooling, error prevention, and maintainable code	Adds complexity for beginners unfamiliar with type systems
Styling	Tailwind CSS	Utility-first, fast development, responsive out of the box	Requires time to learn class utilities; large HTML files with many classes
	CSS Modules	Scoped, conflict-free CSS; integrates well with Next.js	Less efficient for large-scale applications compared to utility-based frameworks like Tailwind

Component Library	ShadCN UI	Flexible, Tailwind-based components with customizable design	Requires integration effort compared to some pre-packaged libraries
State Management	Context API	Built-in, simple, and lightweight	Not suitable for complex state logic or deep component trees
Animations	Framer Motion (now Motion)	Intuitive API, powerful animations, SSR-friendly	Adds to bundle size, not ideal for very simple animations
Forms	React Hook Form	Lightweight, integrates well with schema validation libraries like Yup/Zod	Requires schema knowledge for advanced validations
Data Fetching	React Query	Advanced query handling, supports REST/GraphQL, handles complex caching scenarios	Steeper learning curve for basic data fetching
Backend Framework	Next.js API Routes	Built-in serverless functions, no need for an external backend	Limited to serverless use cases; might require external services for complex backends
Authentication	NextAuth.js	Out-of-the-box support for OAuth, email, JWT	Limited flexibility for custom authentication flows
Database	PostgreSQL	Relational, ACID-compliant, widely supported, and scalable	Requires database management knowledge

ORM	Prisma	Type-safe, easy migrations, integrates with TypeScript	Limited support for some advanced SQL features
Hosting	Vercel	Fast deployments, serverless, great DX, environment variable management	Limited execution time for serverless functions
Testing	Jest	Fast and reliable unit testing	Steep learning curve for beginners
	Cypress	User-friendly, robust end-to-end testing; great debugging tools	Can be slower compared to Playwright for certain large-scale tests
Version Control	GitHub	Best-in-class version control, collaboration tools, and CI/CD integration	May require Git expertise for complex workflows
Code Formatting	Prettier	Enforces consistent code style	Can override some user preferences
	ESLint	Identifies code issues and enforces coding standards	Requires configuration for larger projects
Package Manager	pnpm	Faster and more efficient than npm or Yarn	Less mainstream, so smaller community compared to npm
Monitoring	Sentry	Comprehensive error tracking, supports backend and frontend	Can be expensive for large-scale applications
Environment Mgmt	Vercel Env Variables	Secure, easy to configure	Limited flexibility compared to some alternatives

Media Storage	UploadThing	Serverless file and media uploads, integrates seamlessly with Next.js	Early-stage tool, fewer advanced features compared to Cloudinary
Location	Google Maps API	Accurate location services	Potential costs, only works with internet connectivity

B) Learning Plan

- Create a tabular (i.e. table) representation of the technical skills required for the development of this application.
-
- State for each team member the Responsibility and existing skill level (%).
- State the Learning Plan for each team member (for e.g., start date, end date, resource/s, etc.).

Skill levels written beside the check mark in percentage.

	Layout/Design	Code Logic	Database Management	Testing/Quality Assurance
Simon		✓(75%)	✓(50%)	✓(50%)
Onat	✓	✓		✓
Pablo		✓	✓	✓
Benn	✓(80%)	✓(75%)		✓(50%)

Simon: Back-End
Onat: Front-End
Pablo: Back-End
Benn: Front-End

Start Date: Sept, 2024

End Date: March, 2025

Resources: Mockup, Google Docs, Discord