Project Structure Documentation

Monorepo Structure for React-Vite and FastAPI

```
techprep-ai/
├── packages/
│   ├── shared/                    # Shared code between frontend and backend
│   │   ├── types/                 # TypeScript type definitions
│   │   │   ├── interview.types.ts
│   │   │   ├── user.types.ts
│   │   │   └── resume.types.ts
│   │   └── constants/
│   │       └── index.ts          # Shared constants
│   │
│   ├── frontend/                  # React-Vite application
│   │   ├── src/
│   │   │   ├── components/       # Atomic design components
│   │   │   │   ├── atoms/
│   │   │   │   │   ├── Button.tsx
│   │   │   │   │   ├── Input.tsx
│   │   │   │   │   └── Typography.tsx
│   │   │   │   │
│   │   │   │   ├── molecules/
│   │   │   │   │   ├── SearchBar.tsx
│   │   │   │   │   └── QuestionCard.tsx
│   │   │   │   │
│   │   │   │   └── organisms/
│   │   │   │       ├── InterviewChat.tsx
│   │   │   │       └── ResumeUploader.tsx
│   │   │   │
│   │   │   ├── pages/           # Page components
│   │   │   │   ├── Dashboard.tsx
│   │   │   │   ├── Interview.tsx
│   │   │   │   └── Profile.tsx
│   │   │   │
│   │   │   ├── hooks/           # Custom React hooks
│   │   │   │   ├── useInterview.ts
│   │   │   │   └── useResume.ts
│   │   │   │
│   │   │   ├── api/            # API integration
│   │   │   │   └── client.ts   # Axios/fetch configuration
│   │   │   │
│   │   │   ├── lib/            # Utility functions
│   │   │   │   └── helpers.ts
│   │   │   │
│   │   │   ├── styles/         # Global styles and themes
│   │   │   │   └── global.css
│   │   │   │
│   │   │   ├── types/         # Frontend-specific types
│   │   │   ├── main.tsx       # Entry point
```

```
│   │   │   ├── App.tsx        # Root component
│   │   │   └── vite-env.d.ts  # Vite type declarations
│   │   │
│   │   ├── index.html
│   │   ├── vite.config.ts
│   │   ├── tsconfig.json
│   │   └── package.json
│   │
│   └── backend/                  # FastAPI application
│       ├── app/
│       │   ├── api/              # API routes
│       │   │   ├── endpoints/
│       │   │   │   ├── interviews.py
│       │   │   │   ├── users.py
│       │   │   │   └── resumes.py
│       │   │   └── deps.py       # Dependencies and utilities
│       │   │
│       │   ├── core/             # Core configurations
│       │   │   ├── config.py     # Environment settings
│       │   │   └── security.py   # Auth configurations
│       │   │
│       │   ├── models/           # SQLAlchemy models
│       │   │   ├── user.py
│       │   │   └── interview.py
│       │   │
│       │   ├── schemas/          # Pydantic schemas
│       │   │   ├── user.py
│       │   │   └── interview.py
│       │   │
│       │   └── services/         # Business logic
│       │       ├── interview.py
│       │       └── resume.py
│       │
│       ├── tests/                # Backend tests
│       ├── alembic/              # Database migrations
│       ├── requirements.txt
│       └── main.py               # FastAPI entry point
│
├── package.json                   # Root package.json
└── turbo.json                    # Turborepo configuration


This structure is specifically optimized for React-Vite and FastAPI because:
Frontend (React-Vite):

Uses .tsx extension for all React components to ensure proper TypeScript support
Follows Vite's recommended project structure with src/ as the main source
directory
Implements atomic design while maintaining Vite's conventions for static assets
and configuration
Includes Vite-specific configuration files (vite.config.ts) and type declarations

FastAPI Backend:
```

```
Follows FastAPI's recommended project structure with app/ as the main package
Separates concerns into models, schemas, and services
Includes Alembic for database migrations
Maintains Python package structure with requirements.txt

Shared Types:
The shared types package helps maintain consistency between frontend and backend
by:

Defining interfaces that match FastAPI's Pydantic models
Providing type safety for API responses
Ensuring consistent data structures across the application

Would you like me to explain any specific part of this structure in more detail?
Or should we move on to documenting the technical stack choices in tech-stack.md?
```