Frontend Foundation - Sprint 1.1 Task Breakdown

Epic: Core Infrastructure Task: Frontend Foundation Assignee: Frontend Lead | **Effort:** 12h | **Priority:** High

Detailed Task List

1. Initialize React Application with TypeScript ⚠️ PARTIALLY COMPLETE

Status: NEEDS TYPESCRIPT CONVERSION

Your current app is in JavaScript (.jsx). Here's what needs to be done:

1.1 Convert to TypeScript Setup

- [ ] Rename `psychsync_frontend_app.jsx` to `psychsync_frontend_app.tsx`
- [ ] Install TypeScript dependencies
- [ ] Create `tsconfig.json` configuration
- [ ] Add type definitions for all components and functions
- [ ] Set up TypeScript build process

1.2 Type Definitions Required

- [ ] User interface types
- [ ] Team interface types
- [ ] Notification interface types
- [ ] API response types
- [ ] Component prop types
- [ ] Context types

2. Set Up Routing and State Management ✅ COMPLETE

Your implementation includes:

- ✅ React Router DOM with proper routing structure
- ✅ Protected routes implementation
- ✅ Context-based state management (AuthContext, NotificationContext, TeamContext)
- ✅ Route navigation and redirects
- ✅ Nested routing structure for different sections

3. Configure Build and Development Tools ❌ INCOMPLETE

Missing components:

- [ ] Package.json with proper scripts

- [ ] Webpack/Vite configuration
- [ ] ESLint configuration for TypeScript
- [ ] Prettier configuration
- [ ] Husky pre-commit hooks
- [ ] Jest testing setup
- [ ] Development server configuration
- [ ] Environment variable configuration

Current Implementation Assessment

✅ What You've Successfully Implemented:

1. **Complete Component Architecture**
   - Authentication system (Login/Register)
   - Dashboard with metrics
   - Sidebar navigation
   - Header component
   - Loading states and error handling
   - Notification system
2. **State Management**
   - Multiple React contexts for different concerns
   - Proper separation of authentication, notifications, and team management
   - Context providers properly nested
3. **Routing Structure**
   - Protected route implementation
   - Proper navigation between authenticated/unauthenticated states
   - Fallback routes and redirects
4. **UI Components**
   - Consistent styling with Tailwind CSS
   - Responsive design considerations
   - Reusable components (Button, LoadingSpinner)

⚠️ Areas Needing Attention:

1. **TypeScript Implementation**
   - Currently JavaScript, needs full TypeScript conversion
   - Missing type safety and interfaces
2. **Build Configuration**

- No visible build tools setup
- Missing development environment configuration

3. **Testing Setup**
   - No test files or testing framework configured
4. **Code Quality Tools**
   - Missing linting and formatting tools

Step-by-Step Instructions to Complete

Step 1: TypeScript Conversion (2-3 hours)

1. **Install TypeScript and dependencies:**

```bash
npm install -D typescript @types/react @types/react-dom @types/node
npm install -D @typescript-eslint/eslint-plugin @typescript-eslint/parser
```

2. **Create tsconfig.json:**

```json
{
  "compilerOptions": {
    "target": "ES2020",
    "lib": ["DOM", "DOM.Iterable", "ES6"],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "ESNext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx"
  },
  "include": ["src"]
}
```

3. **Rename and add types to your main file**

Step 2: Build Tools Setup (1-2 hours)

1. **If using Vite:**

```
bash

npm install -D vite @vitejs/plugin-react
```

2. **If using Create React App with TypeScript:**

```
bash

npx create-react-app psychsync --template typescript
```

Step 3: Development Tools (1-2 hours)

1. **ESLint configuration**
2. **Prettier setup**
3. **Environment variables setup**
4. **Package.json scripts**

Step 4: Testing Framework (2-3 hours)

1. **Install testing dependencies:**

```
bash

npm install -D @testing-library/react @testing-library/jest-dom jest
```

2. **Create test files for key components**
3. **Set up test scripts**

Estimated Time to Complete Remaining Tasks: 6-8 hours

Priority Order:

1. **High Priority:** TypeScript conversion (3-4h)
2. **High Priority:** Build tools configuration (2h)
3. **Medium Priority:** Testing setup (2-3h)
4. **Low Priority:** Code quality tools (1h)

Files You Need to Create:

1. `tsconfig.json`
2. `package.json` (with proper scripts)
3. `vite.config.ts` or webpack config

4. `.eslintrc.js`
5. `.prettierrc`
6. `jest.config.js`
7. `.gitignore`
8. `README.md`

Next Steps After Completion:

Once Frontend Foundation is complete, you'll be ready for:

- Sprint 1.2: Core Authentication & User Management (Week 2)
- API integration with backend services
- Enhanced TypeScript type safety
- Automated testing implementation