# **📄 Document 1: Project Scaffolding and Code Snippets**

## **1. Project Structure**

This is the recommended file and folder structure for clean separation and architecture.

Plaintext

project-manager/  
├── fastapi-backend/  
│ ├── app/  
│ │ ├── main.py # FastAPI entry point  
│ │ ├── database.py # DB session, base class, engine  
│ │ ├── models.py # SQLAlchemy ORM Models  
│ │ ├── schemas.py # Pydantic Schemas  
│ │ ├── crud.py # Core logic for DB operations  
│ │ ├── auth.py # JWT token generation/validation  
│ │ ├── dependencies.py # Role-based access control (RBAC)  
│ │ └── routes/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── users.py # Admin-only user endpoints  
│ │ ├── stories.py # Core story CRUD  
│ │ ├── comments.py # Comments CRUD  
│ │ └── ai\_estimation.py# Placeholder AI endpoint  
│ ├── seed.py # Initial data population script  
│ ├── requirements.txt # Python dependencies  
│ ├── .env.example # Environment variables placeholder  
│ └── README.md # Backend run instructions  
└── react-frontend/  
 ├── public/  
 ├── src/  
 │ ├── components/  
 │ │ ├── kanban/  
 │ │ │ ├── KanbanBoard.jsx  
 │ │ │ └── StoryCard.jsx  
 │ │ ├── forms/  
 │ │ │ └── StoryForm.jsx  
 │ │ ├── layout/  
 │ │ │ └── Header.jsx  
 │ │ └── Dashboard.jsx  
 │ ├── context/  
 │ │ └── AuthContext.jsx # For JWT, user data, and roles  
 │ ├── services/  
 │ │ ├── api.js # Axios instance for API communication  
 │ │ └── aiService.js # Dedicated AI calls  
 │ ├── pages/  
 │ │ ├── Login.jsx  
 │ │ ├── Project.jsx # Main Kanban view  
 │ │ └── DashboardPage.jsx  
 │ ├── App.jsx  
 │ └── main.jsx  
 ├── package.json  
 ├── tailwind.config.js  
 └── README.md # Frontend run instructions

## **2. FastAPI Backend Starter Code**

### **fastapi-backend/requirements.txt**

Plaintext

fastapi  
uvicorn[standard]  
sqlalchemy  
pydantic  
python-jose[cryptography] # For JWT  
passlib[bcrypt] # For password hashing  
python-dotenv # For .env file loading

### **fastapi-backend/app/database.py (SQLite Setup)**

Python

# fastapi-backend/app/database.py  
from sqlalchemy import create\_engine  
from sqlalchemy.orm import sessionmaker  
from sqlalchemy.ext.declarative import declarative\_base  
  
# SQLite DB file  
SQLALCHEMY\_DATABASE\_URL = "sqlite:///./project\_manager.db"  
  
engine = create\_engine(  
 SQLALCHEMY\_DATABASE\_URL, connect\_args={"check\_same\_thread": False}  
)  
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
  
Base = declarative\_base()  
  
# Dependency for getting a database session  
def get\_db():  
 db = SessionLocal()  
 try:  
 yield db  
 finally:  
 db.close()

### **fastapi-backend/app/dependencies.py (RBAC)**

Python

# fastapi-backend/app/dependencies.py  
from fastapi import Depends, HTTPException, status  
from .auth import get\_current\_user  
from .models import User  
  
def get\_admin\_user(current\_user: User = Depends(get\_current\_user)):  
 if current\_user.role != "Admin":  
 raise HTTPException(  
 status\_code=status.HTTP\_403\_FORBIDDEN,  
 detail="Requires Admin role permission."  
 )  
 return current\_user  
  
def get\_teamlead\_or\_admin\_user(current\_user: User = Depends(get\_current\_user)):  
 if current\_user.role not in ["Admin", "Team Lead"]:  
 raise HTTPException(  
 status\_code=status.HTTP\_403\_FORBIDDEN,  
 detail="Requires Team Lead or Admin role."  
 )  
 return current\_user

### **fastapi-backend/app/routes/ai\_estimation.py (AI Placeholder)**

Python

# fastapi-backend/app/routes/ai\_estimation.py  
from fastapi import APIRouter  
from pydantic import BaseModel  
import random  
  
router = APIRouter()  
  
class StoryInput(BaseModel):  
 title: str  
 description: str  
  
class AiEstimate(BaseModel):  
 estimated\_pointer: float  
 priority\_suggestion: str  
  
@router.post("/estimate", response\_model=AiEstimate)  
def get\_ai\_estimate(story: StoryInput):  
 """  
 Placeholder endpoint for AI estimation logic.  
 """  
 # Simple logic based on description length for demo  
 base\_pointer = 1  
 if len(story.description) > 100:  
 base\_pointer = 5  
 elif len(story.description) > 50:  
 base\_pointer = 3  
  
 # Add a random element for variability  
 random\_factor = random.choice([0, 0.5, 1])  
  
 # Simple priority logic  
 if "critical" in story.title.lower() or "bug" in story.description.lower():  
 priority = "High"  
 else:  
 priority = random.choice(["Medium", "Low"])  
  
 return AiEstimate(  
 estimated\_pointer=base\_pointer + random\_factor,  
 priority\_suggestion=priority  
 )

## **3. React Frontend Starter Code**

### **react-frontend/src/context/AuthContext.jsx (State Setup)**

JavaScript

// react-frontend/src/context/AuthContext.jsx  
import React, { createContext, useState, useEffect, useContext } from 'react';  
import api from '../services/api';  
  
const AuthContext = createContext();  
  
export const AuthProvider = ({ children }) => {  
 const [user, setUser] = useState(null); // Contains { id, email, name, role }  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 // Check for token on mount and fetch user details  
 const token = localStorage.getItem('token');  
 if (token) {  
 api.defaults.headers.common['Authorization'] = `Bearer ${token}`;  
 fetchUser(token);  
 } else {  
 setLoading(false);  
 }  
 }, []);  
  
 const fetchUser = async (token) => {  
 try {  
 // Endpoint to get current user details from JWT payload  
 const response = await api.get('/users/me');   
 setUser(response.data);  
 } catch (error) {  
 console.error('Failed to fetch user:', error);  
 localStorage.removeItem('token');  
 delete api.defaults.headers.common['Authorization'];  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 const login = async (email, password) => {  
 const response = await api.post('/auth/token', { email, password });  
 const token = response.data.access\_token;  
  
 localStorage.setItem('token', token);  
 api.defaults.headers.common['Authorization'] = `Bearer ${token}`;  
 await fetchUser(token);  
 };  
  
 const logout = () => {  
 localStorage.removeItem('token');  
 delete api.defaults.headers.common['Authorization'];  
 setUser(null);  
 };  
  
 // Helper function for Role-Based UI  
 const hasRole = (requiredRole) => user && user.role === requiredRole;  
 const isAdmin = () => hasRole('Admin');  
 const isTeamLead = () => hasRole('Team Lead');  
  
 return (  
 <AuthContext.Provider value={{ user, loading, login, logout, isAdmin, isTeamLead }}>  
 {children}  
 </AuthContext.Provider>  
 );  
};  
  
export const useAuth = () => useContext(AuthContext);

### **react-frontend/src/components/kanban/KanbanBoard.jsx (Drag-and-Drop)**

JavaScript

// react-frontend/src/components/kanban/KanbanBoard.jsx  
import React, { useState } from 'react';  
import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';  
import StoryCard from './StoryCard'; // Assume this component exists  
  
const initialData = {  
 // Mock data structure  
 'To Do': [{ id: '1', title: 'Setup Backend', priority: 'High' }, { id: '2', title: 'Design Layout', priority: 'Medium' }],  
 'In Progress': [{ id: '3', title: 'Implement Auth', priority: 'High' }],  
 'Completed': [{ id: '4', title: 'Init Project', priority: 'Low' }],  
};  
  
const KanbanBoard = () => {  
 const [columns, setColumns] = useState(initialData);  
  
 const onDragEnd = (result) => {  
 const { source, destination, draggableId } = result;  
  
 if (!destination) return; // Dropped outside of a list  
 if (source.droppableId === destination.droppableId && source.index === destination.index) return; // Same position  
  
 const startColumn = columns[source.droppableId];  
 const endColumn = columns[destination.droppableId];  
  
 // ... (rest of the drag-and-drop logic as provided previously)  
 // \*\*\* API CALL HERE: Update the story status on the backend \*\*\*  
 };  
  
 const columnNames = ['To Do', 'In Progress', 'Blocked', 'Validation', 'Completed'];  
  
 return (  
 <DragDropContext onDragEnd={onDragEnd}>  
 <div className="flex space-x-4 p-4 overflow-x-auto bg-gray-50 dark:bg-gray-900">  
 {columnNames.map((columnName) => (  
 <Droppable droppableId={columnName} key={columnName}>  
 {(provided) => (  
 <div  
 {...provided.droppableProps}  
 ref={provided.innerRef}  
 className="flex-shrink-0 w-80 bg-white dark:bg-gray-800 p-3 rounded-lg shadow-md"  
 >  
 <h3 className="font-semibold text-lg mb-4 border-b pb-2 dark:text-white">{columnName} ({columns[columnName]?.length || 0})</h3>  
 {columns[columnName]?.map((story, index) => (  
 <Draggable key={story.id} draggableId={story.id} index={index}>  
 {(provided) => (  
 <div  
 ref={provided.innerRef}  
 {...provided.draggableProps}  
 {...provided.dragHandleProps}  
 className="mb-3"  
 >  
 <StoryCard story={story} />  
 </div>  
 )}  
 </Draggable>  
 ))}  
 {provided.placeholder}  
 </div>  
 )}  
 </Droppable>  
 ))}  
 </div>  
 </DragDropContext>  
 );  
};  
  
export default KanbanBoard;

# **📄 Document 2: Local Run Instructions**

## **Local Run Instructions (README.md Summary)**

This guide provides the steps to set up and run both the FastAPI backend and the React frontend locally.

### **1. Prerequisites**

* **Python 3.10+** (or newer)
* **Node.js (LTS)** and **npm** (or yarn/pnpm)

### **2. Backend Setup (FastAPI)**

1. **Navigate** to the fastapi-backend/ directory.
2. **Create Virtual Environment** and activate it:

Bash

python -m venv venv  
source venv/bin/activate # On Windows: venv\Scripts\activate

1. **Install Dependencies:**

Bash

pip install -r requirements.txt

1. **Seed the Database:** This creates project\_manager.db and populates the initial data (including **Admin User:** [**admin@project.com**](mailto:admin@project.com) **/ password**).

Bash

python seed.py

1. **Run the Server:**

Bash

uvicorn app.main:app --reload

* 1. **Access Backend:** <http://127.0.0.1:8000>
  2. **Access Swagger Docs:** <http://127.0.0.1:8000/docs>

### **3. Frontend Setup (React)**

1. **Navigate** to the react-frontend/ directory.
2. **Install Dependencies:**

Bash

npm install

1. **Run the Client:**

Bash

npm run dev

* 1. **Access Frontend:** <http://localhost:5173> (or the port shown in your terminal).