1. **Project Management Tool**:
   * Knowledge: Basic understanding of web development and databases.
   * Tools: HTML/CSS/JavaScript for frontend, a backend language like Python (Django or Flask), Ruby (Rails), or JavaScript (Node.js), SQL or NoSQL databases.
2. **Inventory Management System**:
   * Knowledge: Web development, databases.
   * Tools: HTML/CSS/JavaScript for frontend, a backend language, SQL or NoSQL databases.
3. **E-commerce Website**:
   * Knowledge: Web development, databases, and security.
   * Tools: HTML/CSS/JavaScript for frontend, a backend language, SQL or NoSQL databases, payment processing APIs.
4. **Cryptocurrency Tracker**:
   * Knowledge: Web development, APIs, basic understanding of financial markets.
   * Tools: Any programming language (Python/JavaScript are commonly used), data visualization libraries.
5. **Real-time Traffic Monitoring System**:
   * Knowledge: Web development, APIs, maps, and GPS data.
   * Tools: Any programming language, data visualization libraries.
6. **Health Monitoring System**:
   * Knowledge: Mobile development, databases.
   * Tools: A mobile development platform (iOS/Swift or Android/Java or Kotlin), SQL or NoSQL databases.
7. **Cloud-based File Storage System**:
   * Knowledge: Cloud storage services, file handling, security.
   * Tools: A backend language, cloud storage services like AWS S3 or Google Cloud Storage.
8. **Smart Parking System**:
   * Knowledge: APIs, maps, GPS data, possibly IoT programming, payment processing.
   * Tools: Any programming language, IoT tools for hardware interaction, payment processing APIs.
9. **AI Chatbot**:
   * Knowledge: Machine learning, natural language processing.
   * Tools: Python, NLP tools like NLTK or SpaCy, machine learning libraries like TensorFlow or PyTorch, chatbot frameworks like Dialogflow or Rasa.
10. **Privacy-Focused Browser Extension**:
    * Knowledge: Web development, web security and privacy.
    * Tools: HTML/CSS/JavaScript, browser extension APIs.
11. **Voice Controlled Home Automation System**:
    * Knowledge: IoT programming, voice recognition technology, home automation technology.
    * Tools: IoT programming tools, voice recognition APIs, home automation technologies like ZigBee, Z-Wave, or SmartThings.
12. **Sentiment Analysis Tool**:
    * Knowledge: Machine learning, natural language processing.
    * Tools: Python, NLP tools, machine learning libraries.
13. **Image Recognition System**:
    * Knowledge: Machine learning, computer vision.
    * Tools: Python, machine learning libraries like TensorFlow or PyTorch, computer vision libraries like OpenCV.
14. **Automated Resume Screening System**:
    * Knowledge: Machine learning, natural language processing, potentially document parsing.
    * Tools: Python, NLP tools, machine learning libraries.
15. **E-Learning Platform**:
    * Knowledge: Web development, databases, potentially video processing and storage.
    * Tools: HTML/CSS/JavaScript for frontend, a backend language, SQL or NoSQL databases, possibly video processing tools.

**Project Management Tool**:

* Knowledge: Basic understanding of web development and databases.
* Tools: HTML/CSS/JavaScript for frontend, a backend language like Python (Django or Flask), Ruby (Rails), or JavaScript (Node.js), SQL or NoSQL databases.

1. **Requirements Gathering and Analysis (1-2 weeks):** Understand and document what your project management tool needs to do. What features will it have? Who will use it and how? Define clear, measurable goals for the project.
2. **Design (1-2 weeks):** Plan your application's architecture and create wireframes or mockups of the user interface. Decide on the technologies you will use. You might also design your database schema at this stage.
3. **Development - Frontend (2-4 weeks):** Implement your designs in code, starting with the user interface. At this stage, you'll be writing HTML, CSS, and JavaScript to create the pages of your application.
4. **Development - Backend (3-5 weeks):** After that, develop the server-side part of your application. This includes setting up your server, defining routes, and implementing the logic for each route. You'll also set up your database and write code to interact with it.
5. **Integration and Testing (2-3 weeks):** Once your frontend and backend are both developed, you'll need to integrate them together and test the entire system. This includes unit tests, integration tests, and functional tests. Be prepared to find bugs and have to go back to your code to fix them.
6. **Deployment (1 week):** Finally, you'll deploy your application to a server so that it's accessible on the internet. This involves setting up a hosting service, uploading your code, and configuring the server.
7. **Maintenance (Ongoing):** After deployment, you'll enter the maintenance phase, where you'll need to monitor the application, fix any bugs that come up, and possibly add new features or improvements.

**Target Users:**

1. **Project Managers:** These are the main users who will be creating projects, assigning tasks, tracking progress, and managing resources.
2. **Team Members:** They need to update the status of their tasks, log hours, and communicate with others.
3. **Stakeholders:** They might not directly interact with the tool but would need access to reports and overviews of the project status.

**Features:**

1. **User Registration and Authentication:** Users should be able to register, log in, and manage their accounts. You might also want different levels of access for different types of users (like administrators, project managers, and team members).
2. **Project Creation and Management:** Project managers should be able to create new projects, set their parameters (like start date, end date, budget), and close them when completed.
3. **Task Management:** Tasks can be created, assigned to team members, and tracked through their lifecycle. Tasks should have properties like title, description, assignee, due date, and status.
4. **Time Tracking:** Team members should be able to log the hours they've spent on each task.
5. **Collaboration Tools:** These could include comments on tasks, file attachments, and even a chat feature.
6. **Notifications:** Users should be notified about relevant updates, like new tasks assigned to them or changes in task status.
7. **Reporting and Analytics:** The system should provide overviews of project progress, budget usage, task status, etc. These could be shown in the form of dashboards and could also be exported as reports for stakeholders.

**1. Design Phase:**

Design pages for:

* User registration and login
* Logo on top.
* Fields for username and password.
* Buttons for "Log In" and "Register".
* Links for "Forgot Password" and "Register" (on the Login page) or "Log In" (on the Register page).
* Project creation and management

1. Navigation bar on top with links to different pages.
2. Main area of the page divided into two sections:
   * List of existing projects on the left, each with options to view, edit, or delete.
   * Form on the right for creating a new project, with fields for the project name, description, start date, end date, and budget.

* Task creation, assignment, and tracking

1. Navigation bar on top.
2. Dropdown to select a project.
3. Main area of the page divided into two sections:
   * List of existing tasks on the left, each with options to view, edit, or delete.
   * Form on the right for creating a new task, with fields for the task name, description, assignee, due date, and status.

* A dashboard or report page showing project progress

1. Navigation bar on top.
2. Dropdown to select a project.
3. A series of charts or graphs showing project progress, like a Gantt chart, a pie chart of task status, and a bar chart of budget usage.

* **Frontend Development (Angular)**
  + **Step 1.1:** Initialize the Angular application
    - Using Angular CLI, you can initialize your Angular project. You have already done this as seen in your provided directory structure.
  + **Step 1.2:** Create the folder structure for your project
    - **components** (holds all the components of the app)
    - **services** (holds all the services needed like authentication service, project management service etc.)
    - **models** (holds the model classes/interfaces for your entities like User, Project, Task etc.)
    - **helpers** (for any helper classes/functions)
    - **interceptors** (for HTTP interceptors)
    - **assets** (for static files like images, icons, etc.)
  + **Step 1.3:** Install necessary Angular libraries
    - Angular Material for UI components
    - Angular Forms for form handling
    - RxJS for reactive programming
  + **Step 1.4:** Develop your components as per your requirements and following the Angular best practices. For instance, the registration page can be a component, the project dashboard can be another component, etc.
* **Backend Development**
  + **Step 2.1:** Decide on the technology for your backend. This could be Node.js (Express.js), Python (Django, Flask), Java (Spring Boot), etc.
  + **Step 2.2:** Set up your backend application and connect it with a database. Depending on your technology choice, this could mean different things. For instance, if you're using Node.js, you might want to use MongoDB as your database and mongoose as your ORM.
  + **Step 2.3:** Design your database schema based on your requirements. You will need tables/collections for Users, Projects, Tasks, etc.
  + **Step 2.4:** Develop your APIs following RESTful best practices. You will need APIs for user registration/login, project creation, task management, etc.
  + **Step 2.5:** Implement security measures like hashing passwords, using JWT for authentication, preventing SQL injection, etc.
* **Integration and Testing**
  + **Step 3.1:** Integrate your frontend and backend. Make HTTP calls from your Angular application to your backend APIs.
  + **Step 3.2:** Write unit tests for your Angular components and services using Jasmine and Karma.
  + **Step 3.3:** Write unit tests for your backend APIs. The tool for this will depend on your backend technology. For instance, if you're using Node.js, you could use Mocha and Chai for your backend testing.
  + **Step 3.4:** Conduct integration tests to ensure that the entire system is working together properly.
  + **Step 3.5:** Test your application on different browsers and screen sizes to ensure compatibility.
* **Deployment**
  + **Step 4.1:** Choose a hosting platform for your frontend and backend. Some options are Heroku, AWS, Azure, Google Cloud, Netlify, Vercel, etc.
  + **Step 4.2:** Create a production build of your Angular application using the **ng build --prod** command.
  + **Step 4.3:** Deploy your backend application and your database on your chosen platform.
  + **Step 4.4:** Deploy your Angular application on your chosen platform.
  + **Step 4.5:** Ensure that your Angular application can communicate with your backend application. You might need to set up CORS on your backend to allow this.
  + **Step 4.6:** Test your application thoroughly to ensure everything works as expected in the production environment.
* For folder and file naming, it is recommended to use kebab-case (lowercase letters with hyphens between words, like **my-folder-name**). You can name your folders based on the type of files they will hold (like **components**, **services**, **models**, etc.) and your files based on the feature/module they correspond to (like **user-registration.component.ts**, **project.service.ts**, **task.model.ts**, etc.).
* **Frontend Development (Angular)**
  + Initialize the Angular application



* + Create the folder structure for your project



* + Install necessary Angular libraries

**Angular Material** (UI components)

* + Run **ng add @angular/material** in the terminal.

**Angular Forms** (For form handling)

* + Angular Forms comes bundled with Angular. Just ensure to import the ReactiveFormsModule or FormsModule into your application module or feature module.

**Angular Flex Layout** (For responsive UI)

* + Run **npm install @angular/flex-layout @angular/cdk** in the terminal.

**ngx-toastr** (For notifications)

* + Run **npm install ngx-toastr** in the terminal.

**ng2-charts** (For analytics dashboard)

* + Run **npm install --save ng2-charts** in the terminal.

**Angular JWT** (For authentication)

* + Run **npm install @auth0/angular-jwt** in the terminal.

* + Develop your components
* **Develop Login/Register Component**
  + Generate component: Run **ng generate component components/registration** in terminal
  + Implement user registration and login functionality
  + Interact with AuthenticationService
* **Develop Project Creation Component**
  + Generate component: Run **ng generate component components/project-creation** in terminal
  + Implement functionality to create new projects
  + Interact with ProjectService (to be created)
* **Develop Task Management Component**
  + Generate component: Run **ng generate component components/task-management** in terminal
  + Implement functionality to create, assign, and track tasks
  + Interact with TaskService (to be created)
* **Develop Time Tracking Component**
  + Generate component: Run **ng generate component components/time-tracking** in terminal
  + Implement functionality to log hours
  + Interact with TimeLogService (to be created)
* **Develop Collaboration Tools Component**
  + Generate component: Run **ng generate component components/collaboration-tools** in terminal
  + Implement features for:
    - comments, file attachments, and chat
    - notification component
    - creation component
    - reporting and analytics component
    - task management component
    - time tracking component
  + Interact with relevant services
* **Develop Notification Component**
  + Generate component: Run **ng generate component components/notification** in terminal
  + Implement functionality to display updates
  + Interact with relevant services
* **Develop Reporting and Analytics Component**
  + Generate component: Run **ng generate component components/reporting-analytics** in terminal
  + Implement functionality to display project progress, budget usage, task status, etc.
  + Interact with ReportingService (to be created)

**Create Services**

* Create AuthenticationService: Run **ng generate service services/authentication** in terminal
  + Implement methods for user registration, login, and logout
    - **Register:** Accepts user credentials (such as username, password, and other relevant information), sends a POST request to your backend to create a new user, and handles any errors or success messages.
    - **Login:** Accepts user credentials, sends a POST request to your backend to authenticate the user, saves any returned tokens or user information for later requests, and handles errors or success messages.
    - **Logout:** Clears any saved user tokens or information and possibly sends a request to the backend to invalidate the user's session.
    - **Check Authentication:** Checks if the user's tokens or session is still valid, usually by sending a GET request to a protected route in your backend and checking if the response is successful.
  + Implement methods for handling user tokens
* Create ProjectService: Run **ng generate service services/project** in terminal
  + Implement methods for creating projects, fetching project details, updating projects
* Create TaskService: Run **ng generate service services/task** in terminal
  + Implement methods for creating tasks, fetching task details, updating tasks, and changing task status
* Create TimeLogService: Run **ng generate service services/time-log** in terminal
  + Implement methods for logging hours, fetching time logs, updating time logs
* Create CollaborationService: Run **ng generate service services/collaboration** in terminal
  + Implement methods for handling comments, file attachments, and chat features
* Create NotificationService: Run **ng generate service services/notification** in terminal
  + Implement methods for displaying updates and notifications
* Create ReportingService: Run **ng generate service services/reporting** in terminal
  + Implement methods for fetching and creating reports, and analytics data

* **Backend Development**
  + Decide on the technology for your backend
  + Set up your backend application and connect it with a database
  + Design your database schema
  + Develop your APIs
  + Implement security measures
* **Integration and Testing**
  + Integrate your frontend and backend
  + Write unit tests for your Angular components and services
  + Write unit tests for your backend APIs
  + Conduct integration tests
  + Test your application on different browsers and screen sizes
* **Deployment**
  + Choose a hosting platform for your frontend and backend
  + Create a production build of your Angular application
  + Deploy your backend application and your database
  + Deploy your Angular application
  + Ensure that your Angular application can communicate with your backend application
  + Test your application in the production environment