**Project: Invoice Generator (MERN Stack)**

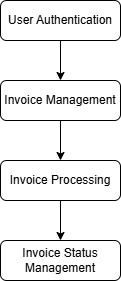
**1. Overview**

The Invoice Generator is a web-based application designed to create, manage, and track invoices efficiently. This project will leverage the MERN stack (MongoDB, Express.js, React.js, and Node.js) to provide a seamless experience for generating and managing invoices.

**2. High-Level Flow**

1. User logs into the system (if authentication is enabled).
2. User creates a new invoice by entering details.
3. The system stores the invoice data in MongoDB.
4. User can view, edit, or delete invoices.
5. The system generates a PDF version of the invoice.
6. (Optional) Invoice is sent via email.
7. User can track the status of invoices (e.g., Paid, Pending, Overdue).

**3. Workflow**



1. **User Authentication** 
   * Users register or log in to the system.
   * JWT tokens are issued for authentication.
2. **Invoice Management**
   * Users create invoices by entering customer details, items, tax, and total amount.
   * The system assigns a unique invoice number.
   * Users can view a list of all invoices.
   * Users can update or delete invoices.
3. **Invoice Processing**
   * The system calculates the total amount including taxes.
   * A PDF is generated and stored.
   * (Optional) The invoice is sent via email.
4. **Invoice Status Management**
   * Users update invoice statuses (e.g., Paid, Pending, Overdue).
   * The system maintains a history of status changes.

**4. Work Entity State Transition Life Cycles**

**Invoice State Transitions:**

1. **Draft** → The invoice is created but not yet finalized.
2. **Pending** → The invoice is sent but not yet paid.
3. **Paid** → The invoice has been paid by the customer.
4. **Overdue** → The payment due date has passed.
5. **Cancelled** → The invoice is voided by the user.

These state transitions ensure that invoices move through a structured lifecycle from creation to finalization and tracking.

**5. Technology Stack**

* **Frontend:** React.js (with Tailwind CSS/Bootstrap for UI styling)
* **Backend:** Node.js with Express.js
* **Database:** MongoDB with Mongoose ORM
* **Authentication:** JWT (optional)
* **PDF Generation:** PDFKit or html-pdf
* **State Management:** React Context API or Redux (optional)

**6. System Architecture**

* **Frontend:** React.js handles user interactions, invoice creation forms, and data visualization.
* **Backend:** Express.js serves API endpoints for invoice CRUD operations.
* **Database:** MongoDB stores invoice details securely.
* **Third-party Integrations:** PDFKit for PDF generation, Email API for sending invoices.

**7. Functional Requirements**

**7.1 User Roles**

* **Admin:** Can manage all invoices and users.
* **User:** Can create and manage their invoices.

**7.2 Features**

* **Invoice Management:**
  + Create, update, delete, and view invoices.
  + Assign unique invoice numbers.
  + Include customer details, items, tax, and total amount.
* **PDF Generation:**
  + Convert invoices into downloadable PDFs.
* **Authentication (Optional):**
  + User registration and login using JWT.
* **Email Notifications (Future Scope):**
  + Send invoices to customers via email.

**8. API Endpoints (Backend)**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/invoices | Create a new invoice |
| GET | /api/invoices | Get all invoices |
| GET | /api/invoices/:id | Get invoice by ID |
| PUT | /api/invoices/:id | Update an invoice |
| DELETE | /api/invoices/:id | Delete an invoice |

**9. Database Schema (MongoDB)**

{

"invoiceNumber": "INV-1001",

"customerName": "John Doe",

"items": [

{ "item": "Laptop", "quantity": 1, "price": 1000 }

],

"tax": 10,

"totalAmount": 1100,

"status": "Pending",

"createdAt": "2025-02-12T00:00:00.000Z"

}

**10. Frontend Components (React)**

* **InvoiceForm.js**: Form to create/edit invoices.
* **InvoiceList.js**: Displays a list of invoices.
* **InvoiceDetail.js**: View detailed invoice info.
* **PDFGenerator.js**: Generates PDF invoices.

**11. Implementation Plan**

**Phase 1: Backend Setup**

* Set up Node.js and Express.js.
* Define API endpoints and connect to MongoDB.
* Implement CRUD operations for invoices.

**Phase 2: Frontend Development**

* Set up React project and UI components.
* Connect to backend API using Axios.
* Implement invoice creation and display.

**Phase 3: Additional Features**

* Implement PDF generation.
* Add authentication.
* Deploy the application.

**12. Deployment Strategy**

* **Backend:** Deploy to AWS, Heroku, or Render.
* **Frontend:** Deploy using Vercel or Netlify.
* **Database:** Host MongoDB on MongoDB Atlas.

**13. Conclusion**

This PoC provides a structured plan for building the Invoice Generator using the MERN stack. The project can be expanded with additional features like email notifications, role-based access, and payment integration for enhanced functionality.