AI-Powered Task Management System (PERN Stack)

This project is a **PERN (PostgreSQL, Express.js, React.js, Node.js)** application that integrates AI to intelligently prioritize tasks, predict deadlines, and allocate resources. It leverages AI models to analyze user habits and task dependencies, suggesting optimal workflows while identifying potential bottlenecks.

---

1. System Architecture

- **Frontend (React)**: Interactive UI for task creation, visualization, prioritization, and analytics dashboards.
- **Backend (Node.js + Express)**: RESTful API with routes for authentication, task CRUD operations, AI analysis requests.
- **Database (PostgreSQL)**: Stores user data, tasks, resources, logs, and AI insights.
- **AI Layer (Python/Node.js AI service or integrated libraries)**: Performs intelligent prioritization, predictions, and recommendations.
- **Optional Message Queue (e.g., Redis/Kafka)**: For async task analysis & notifications.

---

2. PostgreSQL Schema Design

Tables:

users

```sql
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

tasks

```sql
CREATE TABLE tasks (
    task_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    priority INT DEFAULT 0, -- AI adjusted
    deadline TIMESTAMP,
    estimated_hours NUMERIC,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
```

resources

```
CREATE TABLE resources (
    resource_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    type VARCHAR(50),
    capacity NUMERIC,
    allocated_hours NUMERIC DEFAULT 0
);
```

user_activity (for AI training)

```
CREATE TABLE user_activity (
    activity_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    task_id INT REFERENCES tasks(task_id),
    action VARCHAR(50), -- created, completed, delayed
    timestamp TIMESTAMP DEFAULT NOW()
);
```

habit_embeddings (AI insights storage)

```
CREATE EXTENSION IF NOT EXISTS vector;

CREATE TABLE habit_embeddings (
    embedding_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    vector_embedding VECTOR(128),
    last_updated TIMESTAMP DEFAULT NOW()
);
```

workload_summary (materialized view)

```
CREATE MATERIALIZED VIEW workload_summary AS
SELECT user_id,
       COUNT(*) FILTER (WHERE status='pending') AS pending_tasks,
       SUM(estimated_hours) AS total_estimated_hours,
       MIN(deadline) AS next_deadline
FROM tasks
WHERE deadline >= NOW()
GROUP BY user_id;
```

---

3. Backend (Express.js)

Key Routes

- `POST /auth/register` → User registration
- `POST /auth/login` → JWT-based authentication
- `GET /tasks` → Fetch all tasks (with AI-prioritized sorting)
- `POST /tasks` → Create a new task
- `PUT /tasks/:id` → Update task details
- `DELETE /tasks/:id` → Delete a task
- `GET /ai/prioritize` → AI-based task prioritization
- `GET /ai/deadline-predict` → AI-based deadline predictions
- `GET /ai/resource-allocate` → AI-based resource allocation suggestions

Middleware

- JWT Authentication
- Error handling
- Request validation (Joi/Zod)

---

4. AI Component

- **Task Prioritization Algorithm:** Uses weighted scoring model with:
- Deadline urgency
- Estimated effort
- Historical completion patterns (from `user_activity`)

- User preferences (via embeddings)

- **Deadline Prediction:** Regression/ML model trained on past task completion data.

- **Resource Allocation:** Linear programming or heuristic optimization assigning resources based on available capacity.

---

5. Frontend (React.js)

Pages/Components:

1. **Dashboard**: Overview of tasks, workload summary, and AI suggestions.
2. **Task List**: Interactive list with filters (priority, deadline, status).
3. **Task Form**: Create/update tasks.
4. **Resource Manager**: Track and allocate resources.
5. **Analytics**: Charts for productivity trends, delays, and AI insights.

UI Features:

- Drag-and-drop task prioritization (AI suggestions appear as highlights)

- Gantt chart/Calendar view of deadlines
- Notifications for predicted delays

Example React Component (Task Card)

```
import { Card } from "@/components/ui/card";

export default function TaskCard({ task }) {
  return (
    <Card className="p-4 shadow-md rounded-2xl">
      <h3 className="text-lg font-semibold">{task.title}</h3>
      <p className="text-sm text-gray-600">{task.description}</p>
      <div className="flex justify-between mt-2">
        <span className="text-xs text-gray-500">Deadline: {new
Date(task.deadline).toLocaleDateString()}</span>
        <span className="text-xs font-medium">Priority: {task.priority}</
span>
      </div>
    </Card>
  );
}
```

---

6. Deployment

- **Docker Compose Setup:**

```
version: "3.9"
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
      POSTGRES_DB: ai_task_db
    ports:
      - "5432:5432"
    volumes:
      - ./db_data:/var/lib/postgresql/data

  backend:
    build: ./backend
    environment:
      DATABASE_URL: postgres://admin:password@db:5432/ai_task_db
    ports:
      - "5000:5000"
    depends_on:
      - db
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
    depends_on:
      - backend
```

---

7. Testing & Security

- **Testing**: Jest for backend unit tests, React Testing Library for frontend, Cypress for end-to-end.
- **Security**:
- JWT authentication
- Password hashing (bcrypt)
- Input validation (Joi/Zod)
- HTTPS with TLS
- Role-based access control (admin/user)

---

8. Extensions

- Multi-user collaboration (shared tasks)
- AI chatbot assistant for task queries
- Integration with Google Calendar / Slack
- Mobile app (React Native)

✅This blueprint gives you **frontend, backend, database, AI logic, and deployment setup** for the AI-Powered Task Management System using PERN.