TaskFlow AI - Project Requirements Document (PRD)

1. Executive Summary

Project Name: TaskFlow AI
Version: 1.0
Date: December 7, 2024
Project Duration: 3 Days
Team Size: 1 Developer

1.1 Project Overview

TaskFlow AI is a real-time collaborative task management platform powered by artificial intelligence. It enables teams to create, organize, and track tasks with intelligent automation for priority prediction, time estimation, and task enhancement. The platform provides real-time synchronization across all team members and actionable analytics to improve productivity.

1.2 Problem Statement

Current task management tools lack intelligent assistance and often require manual refresh to see team updates. Teams waste time on administrative overhead like setting priorities, estimating time, and maintaining task descriptions. There's a need for a smart, real-time collaborative platform that helps teams work more efficiently.

1.3 Project Goals

- Build a full-stack application showcasing modern web technologies
- Implement real-time collaboration features
- Integrate AI-powered task management assistance
- Demonstrate microservices architecture
- Create a production-ready, deployable application

---

2. Technical Stack

2.1 Frontend

- **Framework:** Next.js 14+ (App Router)
- **UI Library:** React 18+
- **Language:** TypeScript
- **Styling:** Tailwind CSS

- **UI Components:** shadcn/ui or Material-UI
- **State Management:** React Context API + React Query
- **Real-time:** Socket.io-client
- **Drag & Drop:** react-beautiful-dnd
- **Charts:** Recharts
- **Form Handling:** React Hook Form + Zod validation

2.2 Backend - Main API

- **Framework:** NestJS 10+
- **Language:** TypeScript
- **Authentication:** Passport.js + JWT
- **ORM:** TypeORM
- **WebSocket:** @nestjs/websockets + Socket.io
- **Validation:** class-validator
- **API Documentation:** Swagger/OpenAPI

2.3 Backend - AI Service

- **Framework:** FastAPI
- **Language:** Python 3.11+
- **AI/ML:** OpenAI API / Scikit-learn
- **Validation:** Pydantic
- **API Documentation:** Auto-generated Swagger

2.4 Databases

- **Primary Database:** PostgreSQL 15+ (relational data)
- **Secondary Database:** MongoDB 6+ (logs, analytics)
- **Caching:** Redis (optional, for sessions)

2.5 DevOps & Deployment

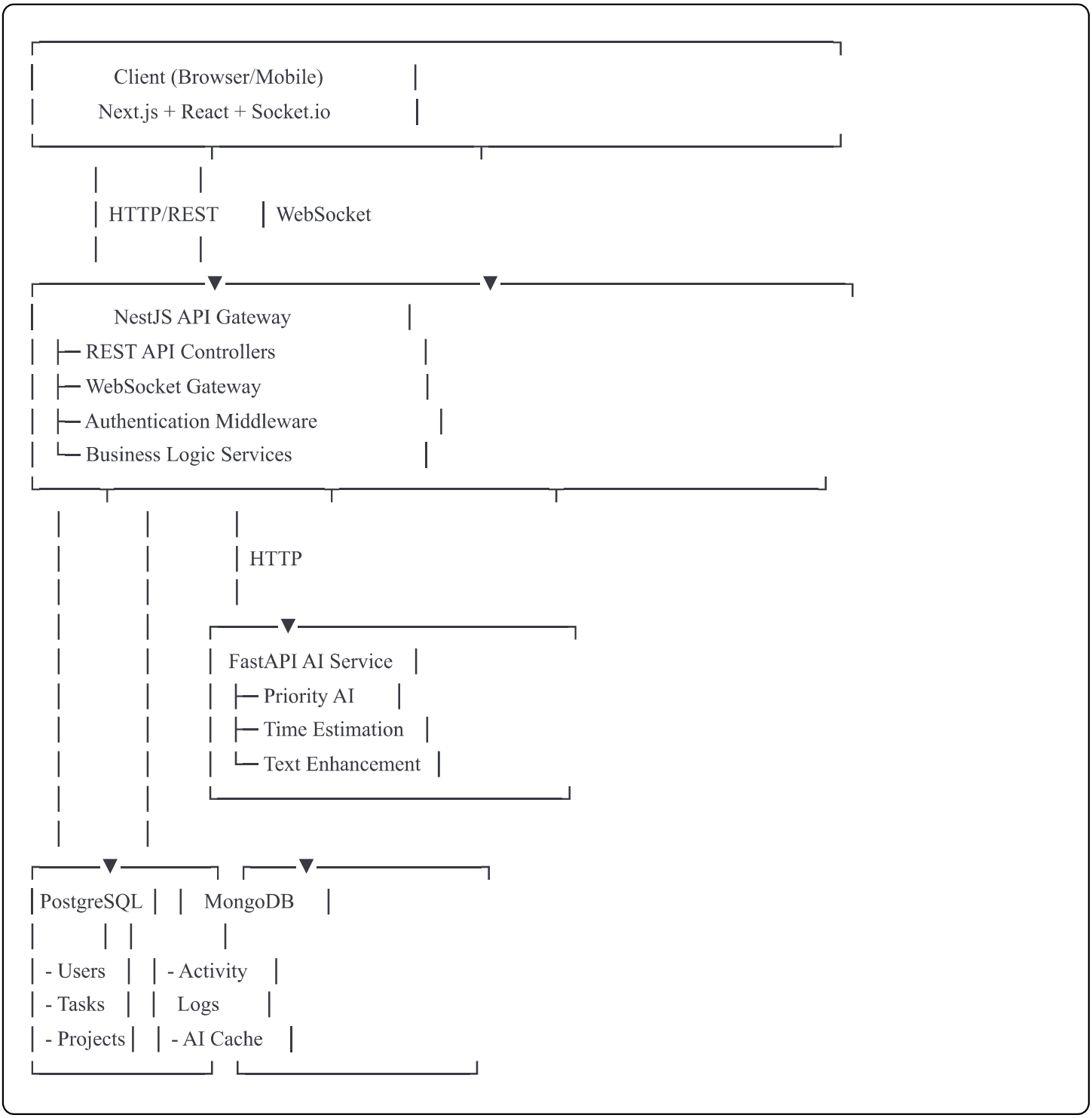
- **Containerization:** Docker + Docker Compose
- **Frontend Hosting:** Vercel

- **Backend Hosting:** Railway / Render / Heroku
- **Version Control:** Git + GitHub

---

3. System Architecture

3.1 High-Level Architecture



3.2 Communication Patterns

- **Client ↔ NestJS:** REST API (HTTP) + WebSocket (real-time)
- **NestJS ↔ FastAPI:** REST API (HTTP)
- **NestJS ↔ PostgreSQL:** TypeORM (ORM)
- **NestJS ↔ MongoDB:** Mongoose (ODM)

---

4. Functional Requirements

4.1 User Management

4.1.1 User Registration

- **ID:** FR-UM-001
- **Description:** Users can create an account with email and password
- **Acceptance Criteria:**
  - Email validation (proper format)
  - Password strength validation (min 8 chars, 1 uppercase, 1 number)
  - Password hashing using bcrypt
  - Duplicate email prevention
  - Email verification (optional for MVP)
- **Priority:** High
- **Estimated Time:** 2 hours

4.1.2 User Login

- **ID:** FR-UM-002
- **Description:** Users can login with credentials
- **Acceptance Criteria:**
  - Email and password authentication
  - JWT token generation (access + refresh)
  - Token stored in HTTP-only cookies
  - Invalid credentials show appropriate error

- **Priority:** High
- **Estimated Time:** 2 hours

4.1.3 User Profile

- **ID:** FR-UM-003
- **Description:** Users can view and update their profile
- **Acceptance Criteria:**
  - View name, email, avatar
  - Update name and avatar
  - Cannot change email (for MVP)
- **Priority:** Medium
- **Estimated Time:** 1 hour

4.2 Workspace Management

4.2.1 Create Workspace

- **ID:** FR-WS-001
- **Description:** Users can create a workspace/project
- **Acceptance Criteria:**
  - Provide name and description
  - Creator becomes owner
  - Workspace is created in database
  - User is redirected to workspace
- **Priority:** High
- **Estimated Time:** 1 hour

4.2.2 List Workspaces

- **ID:** FR-WS-002
- **Description:** Users can view all their workspaces
- **Acceptance Criteria:**
  - Show all workspaces user is member of

    - Display task count per workspace
    - Show member count
    - Sort by recent activity
- **Priority:** High
- **Estimated Time:** 1 hour

4.2.3 Invite Members

- **ID:** FR-WS-003
- **Description:** Workspace owners can invite team members
- **Acceptance Criteria:**
    - Enter user email to invite
    - User receives invitation (email or in-app notification)
    - Invited user can accept/reject
    - Members shown in workspace settings
- **Priority:** Medium
- **Estimated Time:** 2 hours

4.2.4 Manage Permissions

- **ID:** FR-WS-004
- **Description:** Owners can assign roles to members
- **Acceptance Criteria:**
    - Roles: Owner, Admin, Member
    - Only owners can change roles
    - Owners can remove members
- **Priority:** Medium
- **Estimated Time:** 1 hour

4.3 Task Management

4.3.1 Create Task

- **ID:** FR-TM-001

- **Description:** Users can create a new task
- **Acceptance Criteria:**
  - Required fields: title, workspace
  - Optional fields: description, assignee, due date, priority, tags
  - AI automatically suggests priority
  - Task saved to database
  - Real-time broadcast to all workspace members
- **Priority:** High
- **Estimated Time:** 2 hours

4.3.2 View Tasks

- **ID:** FR-TM-002
- **Description:** Users can view tasks in a board layout
- **Acceptance Criteria:**
  - Tasks organized by status columns (Todo, In Progress, Review, Done)
  - Show task title, assignee, priority indicator
  - Click to open task details
  - Filter by assignee, priority, tags
  - Search by title/description
- **Priority:** High
- **Estimated Time:** 3 hours

4.3.3 Update Task

- **ID:** FR-TM-003
- **Description:** Users can update task details
- **Acceptance Criteria:**
  - Edit any field (title, description, assignee, etc.)
  - Changes saved to database
  - Real-time broadcast to all members

    - Activity log recorded

- **Priority:** High
- **Estimated Time:** 1 hour

4.3.4 Drag & Drop Tasks

- **ID:** FR-TM-004
- **Description:** Users can drag tasks between status columns
- **Acceptance Criteria:**
    - Smooth drag and drop animation
    - Task status updates on drop
    - Position preserved within column
    - Real-time sync across all users
- **Priority:** High
- **Estimated Time:** 2 hours

4.3.5 Delete Task

- **ID:** FR-TM-005
- **Description:** Users can delete tasks
- **Acceptance Criteria:**
    - Confirmation dialog before deletion
    - Soft delete (mark as deleted, keep in DB)
    - Real-time broadcast to all members
- **Priority:** Medium
- **Estimated Time:** 0.5 hours

4.3.6 Task Comments

- **ID:** FR-TM-006
- **Description:** Users can comment on tasks
- **Acceptance Criteria:**
    - Add comment with text content

    - Comments shown chronologically
    - Show author and timestamp
    - Real-time updates when others comment
- **Priority:** Medium
- **Estimated Time:** 2 hours

4.4 AI Features

4.4.1 Priority Prediction

- **ID:** FR-AI-001
- **Description:** AI predicts task priority based on content
- **Acceptance Criteria:**
    - Analyze task title and description
    - Return priority (Low, Medium, High, Urgent)
    - Show confidence score
    - User can override AI suggestion
- **Priority:** High
- **Estimated Time:** 2 hours

4.4.2 Time Estimation

- **ID:** FR-AI-002
- **Description:** AI estimates time required for task
- **Acceptance Criteria:**
    - Analyze task complexity
    - Return estimated hours
    - Show confidence score
    - User can override estimate
- **Priority:** Medium
- **Estimated Time:** 2 hours

4.4.3 Description Enhancement

- **ID:** FR-AI-003
- **Description:** AI improves task descriptions
- **Acceptance Criteria:**
  - User clicks "Enhance with AI" button
  - AI expands brief description into structured format
  - Includes: Issue, Steps to Reproduce, Expected vs Actual
  - User can accept or reject enhancement
- **Priority:** Medium
- **Estimated Time:** 2 hours

4.5 Real-Time Collaboration

4.5.1 Live Task Updates

- **ID:** FR-RT-001
- **Description:** Changes made by any user appear instantly for all
- **Acceptance Criteria:**
  - Task creation appears immediately
  - Task updates reflect in real-time
  - Task deletion removes from all views
  - No page refresh required
- **Priority:** High
- **Estimated Time:** 3 hours

4.5.2 Online Presence

- **ID:** FR-RT-002
- **Description:** Show which team members are currently online
- **Acceptance Criteria:**
  - Green indicator for online users
  - List of active members in workspace
  - Updates when users join/leave

- **Priority:** Low
- **Estimated Time:** 1 hour

4.5.3 Live Comments

- **ID:** FR-RT-003
- **Description:** Comments appear in real-time
- **Acceptance Criteria:**
  - New comments show instantly
  - Typing indicators (optional)
  - Smooth animation for new comments
- **Priority:** Medium
- **Estimated Time:** 1 hour

4.6 Analytics & Reporting

4.6.1 Workspace Dashboard

- **ID:** FR-AN-001
- **Description:** Overview of workspace metrics
- **Acceptance Criteria:**
  - Total tasks, completed tasks, completion rate
  - Tasks by status (pie chart)
  - Tasks by priority (bar chart)
  - Recent activity timeline
- **Priority:** Medium
- **Estimated Time:** 3 hours

4.6.2 Productivity Trends

- **ID:** FR-AN-002
- **Description:** Track productivity over time
- **Acceptance Criteria:**
  - Line chart showing tasks completed per day/week

- Compare current period to previous
- Filter by date range

- **Priority:** Medium
- **Estimated Time:** 2 hours

4.6.3 Team Performance

- **ID:** FR-AN-003
- **Description:** Individual member statistics
- **Acceptance Criteria:**
  - Tasks assigned vs completed per member
  - Average completion time
  - Leaderboard (optional)
- **Priority:** Low
- **Estimated Time:** 2 hours

4.6.4 Activity Log

- **ID:** FR-AN-004
- **Description:** Audit trail of all workspace activities
- **Acceptance Criteria:**
  - Log all actions (create, update, delete, comment)
  - Show user, action, timestamp
  - Filterable by action type
  - Stored in MongoDB
- **Priority:** Medium
- **Estimated Time:** 1 hour

---

5. Non-Functional Requirements

5.1 Performance

- **NFR-PERF-001:** Page load time < 2 seconds

- **NFR-PERF-002:** API response time < 500ms (95th percentile)
- **NFR-PERF-003:** Real-time updates latency < 200ms
- **NFR-PERF-004:** Support 100 concurrent users (MVP target)

5.2 Security

- **NFR-SEC-001:** All passwords must be hashed using bcrypt
- **NFR-SEC-002:** JWT tokens expire after 15 minutes (access) and 7 days (refresh)
- **NFR-SEC-003:** HTTPS only in production
- **NFR-SEC-004:** API rate limiting: 100 requests/minute per user
- **NFR-SEC-005:** SQL injection prevention via ORM
- **NFR-SEC-006:** XSS prevention via input sanitization
- **NFR-SEC-007:** CORS properly configured

5.3 Reliability

- **NFR-REL-001:** 99% uptime (target)
- **NFR-REL-002:** Database backups daily
- **NFR-REL-003:** Error logging with stack traces
- **NFR-REL-004:** Graceful WebSocket reconnection

5.4 Scalability

- **NFR-SCAL-001:** Horizontal scaling capability
- **NFR-SCAL-002:** Database connection pooling
- **NFR-SCAL-003:** Stateless API design
- **NFR-SCAL-004:** CDN for static assets

5.5 Usability

- **NFR-USE-001:** Mobile responsive (min width: 320px)
- **NFR-USE-002:** Keyboard navigation support
- **NFR-USE-003:** Loading states for all async operations
- **NFR-USE-004:** Error messages are user-friendly
- **NFR-USE-005:** Accessible (WCAG 2.1 AA compliance - basic)

5.6 Maintainability

- **NFR-MAINT-001:** Code coverage > 70% (target)
- **NFR-MAINT-002:** TypeScript strict mode enabled
- **NFR-MAINT-003:** ESLint + Prettier configured
- **NFR-MAINT-004:** API documentation via Swagger
- **NFR-MAINT-005:** README with setup instructions

---

6. Database Schema

6.1 PostgreSQL Schema

sql

```sql
-- USERS
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    avatar_url VARCHAR(500),
    role VARCHAR(20) DEFAULT 'member',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- WORKSPACES
CREATE TABLE workspaces (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    owner_id UUID REFERENCES users(id) ON DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- WORKSPACE_MEMBERS
CREATE TABLE workspace_members (
    workspace_id UUID REFERENCES workspaces(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(20) DEFAULT 'member', -- owner, admin, member
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (workspace_id, user_id)
);

-- TASKS
CREATE TABLE tasks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workspace_id UUID REFERENCES workspaces(id) ON DELETE CASCADE,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    status VARCHAR(20) DEFAULT 'todo', -- todo, in_progress, review, done
    priority VARCHAR(20) DEFAULT 'medium', -- low, medium, high, urgent
    assigned_to UUID REFERENCES users(id) ON DELETE SET NULL,
    created_by UUID REFERENCES users(id) ON DELETE SET NULL,
    due_date TIMESTAMP,
    estimated_hours DECIMAL(5,2),
```

```
    ai_priority_score DECIMAL(3,2), -- 0.00 to 1.00
    position INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- COMMENTS
CREATE TABLE comments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    task_id UUID REFERENCES tasks(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- TAGS
CREATE TABLE tags (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workspace_id UUID REFERENCES workspaces(id) ON DELETE CASCADE,
    name VARCHAR(50) NOT NULL,
    color VARCHAR(7) DEFAULT '#3B82F6', -- hex color
    UNIQUE(workspace_id, name)
);

-- TASK_TAGS
CREATE TABLE task_tags (
    task_id UUID REFERENCES tasks(id) ON DELETE CASCADE,
    tag_id UUID REFERENCES tags(id) ON DELETE CASCADE,
    PRIMARY KEY (task_id, tag_id)
);

-- INDEXES
CREATE INDEX idx_tasks_workspace ON tasks(workspace_id);
CREATE INDEX idx_tasks_assigned ON tasks(assigned_to);
CREATE INDEX idx_tasks_status ON tasks(status);
CREATE INDEX idx_comments_task ON comments(task_id);
CREATE INDEX idx_workspace_members_user ON workspace_members(user_id);
```

6.2 MongoDB Collections

javascript

```
// Collection: activity_logs
{
  _id: ObjectId,
  workspace_id: String,
  user_id: String,
  user_name: String,
  action: String, // 'task_created', 'task_updated', 'task_deleted', 'comment_added'
  entity_type: String, // 'task', 'comment', 'workspace'
  entity_id: String,
  metadata: {
    field_changed: String,
    old_value: Mixed,
    new_value: Mixed,
    task_title: String,
    // ... flexible fields
  },
  timestamp: ISODate
}

// Collection: ai_suggestions
{
  _id: ObjectId,
  task_id: String,
  suggestion_type: String, // 'priority', 'time_estimate', 'description'
  input: {
    title: String,
    description: String
  },
  output: {
    suggested_priority: String,
    confidence: Number,
    reasoning: String,
    estimated_hours: Number,
    enhanced_description: String
  },
  model_version: String,
  created_at: ISODate
}

// Collection: analytics_cache
{
  _id: ObjectId,
  workspace_id: String,
```

```
  metric_type: String, // 'completion_rate', 'productivity_trend', 'member_stats'
  period: String, // 'daily', 'weekly', 'monthly'
  date: ISODate,
  data: Mixed, // flexible JSON data
  calculated_at: ISODate
}
```

---

7. API Specifications

7.1 Authentication APIs

POST /auth/register

```
json

Request:
{
  "email": "user@example.com",
  "password": "SecurePass123",
  "name": "John Doe"
}

Response: 201 Created
{
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "name": "John Doe"
  },
  "access_token": "jwt_token",
  "refresh_token": "jwt_token"
}
```

POST /auth/login

```
json
```

```
Request:
{
  "email": "user@example.com",
  "password": "SecurePass123"
}

Response: 200 OK
{
  "user": {...},
  "access_token": "jwt_token",
  "refresh_token": "jwt_token"
}
```

POST /auth/refresh

```
json

Request:
{
  "refresh_token": "jwt_token"
}

Response: 200 OK
{
  "access_token": "new_jwt_token"
}
```

POST /auth/logout

```
json

Response: 200 OK
{
  "message": "Logged out successfully"
}
```

7.2 Workspace APIs

POST /workspaces

```
json
```

Request:

```
{
  "name": "My Project",
  "description": "Project description"
}
```

Response: 201 Created

```
{
  "id": "uuid",
  "name": "My Project",
  "description": "Project description",
  "owner_id": "uuid",
  "created_at": "2024-12-07T10:00:00Z"
}
```

GET /workspaces

json

Response: 200 OK

```
{
  "workspaces": [
    {
      "id": "uuid",
      "name": "My Project",
      "description": "...",
      "role": "owner",
      "task_count": 15,
      "member_count": 3
    }
  ]
}
```

GET /workspaces/:id

json

Response: 200 OK

```
{
  "id": "uuid",
  "name": "My Project",
  "description": "...",
  "owner": {...},
  "members": [...],
  "stats": {
    "total_tasks": 20,
    "completed_tasks": 8
  }
}
```

7.3 Task APIs

POST /tasks

json

Request:

```
{
  "workspace_id": "uuid",
  "title": "Fix login bug",
  "description": "Users can't login on mobile",
  "assigned_to": "uuid",
  "due_date": "2024-12-10T23:59:59Z",
  "priority": "high"
}
```

Response: 201 Created

```
{
  "id": "uuid",
  "workspace_id": "uuid",
  "title": "Fix login bug",
  "description": "...",
  "status": "todo",
  "priority": "high",
  "ai_priority_score": 0.87,
  "estimated_hours": 4.5,
  "assigned_to": {...},
  "created_by": {...},
  "created_at": "2024-12-07T10:00:00Z"
}
```

GET /tasks?workspace_id=uuid

json

Response: 200 OK

```
{
  "tasks": [
    {
      "id": "uuid",
      "title": "Fix login bug",
      "status": "todo",
      "priority": "high",
      "assigned_to": {...},
      "due_date": "2024-12-10T23:59:59Z"
    }
  ]
}
```

PATCH /tasks/:id

json

Request:

```
{
  "status": "in_progress",
  "priority": "urgent"
}
```

Response: 200 OK

```
{
  "id": "uuid",
  "status": "in_progress",
  "priority": "urgent",
  "updated_at": "2024-12-07T11:00:00Z"
}
```

DELETE /tasks/:id

json

Response: 200 OK

```
{
  "message": "Task deleted successfully"
}
```

7.4 AI Service APIs (FastAPI)

POST /ai/predict-priority

json

Request:

```
{
  "title": "Fix critical login bug",
  "description": "Users cannot access the application"
}
```

Response: 200 OK

```
{
  "priority": "urgent",
  "confidence": 0.92,
  "reasoning": "Keywords 'critical' and 'cannot access' indicate high urgency"
}
```

POST /ai/estimate-time

json

Request:

```
{
  "title": "Create user profile page",
  "description": "Build a profile page with avatar upload and bio editing"
}
```

Response: 200 OK

```
{
  "estimated_hours": 8,
  "confidence": 0.75,
  "breakdown": {
    "frontend": 4,
    "backend": 3,
    "testing": 1
  }
}
```

POST /ai/enhance-description

json

Request:

```
{
  "title": "Fix bug",
  "description": "Button not working"
}
```

Response: 200 OK

```
{
  "enhanced_description": "**Issue:** The submit button on the login form is not responding to user clicks...\n\n**Steps to R
}
```

7.5 WebSocket Events

Client → Server

javascript

```
// Join workspace
emit('join_workspace', { workspaceId: 'uuid' })

// Leave workspace
emit('leave_workspace', { workspaceId: 'uuid' })

// Update task (optional, can use REST API)
emit('task_updated', { taskId: 'uuid', changes: {...} })
```

Server → Client

javascript

```
// User joined
on('user_joined', { userId: 'uuid', userName: 'John' })

// User left
on('user_left', { userId: 'uuid' })

// Task created
on('task_created', { task: {...} })

// Task updated
on('task_updated', { taskId: 'uuid', changes: {...} })

// Task deleted
on('task_deleted', { taskId: 'uuid' })

// Comment added
on('comment_added', { comment: {...} })
```

---

8. User Interface Specifications

8.1 Color Scheme

css

Primary: #3B82F6 (Blue)
Secondary: #8B5CF6 (Purple)
Success: #10B981 (Green)
Warning: #F59E0B (Orange)
Error: #EF4444 (Red)
Background: #F9FAFB (Light Gray)
Dark Background: #1F2937 (Dark Gray)
Text: #111827 (Almost Black)

8.2 Typography

- **Headings:** Inter, sans-serif (Bold)
- **Body:** Inter, sans-serif (Regular)
- **Code:** Fira Code, monospace

8.3 Key Pages

8.3.1 Login Page

- Clean, centered form
- Email and password fields
- "Remember me" checkbox
- "Forgot password?" link
- "Sign up" link
- Social login buttons (optional)

8.3.2 Dashboard

- Top navigation bar (logo, search, notifications, profile)
- Sidebar (workspace list, settings)
- Main content area:
  - "My Workspaces" grid
  - Quick stats cards
  - Recent activity feed

8.3.3 Task Board

- Kanban-style layout

- Four columns: Todo, In Progress, Review, Done
- Drag and drop between columns
- "Add Task" button in each column
- Filters (assignee, priority, tags)
- Search bar

8.3.4 Task Detail Modal

- Large modal overlay
- Task title (editable inline)
- Status dropdown
- Priority dropdown
- Assignee selector
- Due date picker
- Description editor (markdown support)
- "Enhance with AI" button
- Tags
- Comments section
- Activity history

8.3.5 Analytics Dashboard

- Time period selector (7 days, 30 days, 90 days)
- Grid of charts:
  - Tasks completion rate (pie chart)
  - Productivity trend (line chart)
  - Tasks by priority (bar chart)
  - Team performance (table)
- Activity timeline

---

9. Development Phases

Phase 1: Foundation (Day 1 - 8-10 hours)

Milestone 1.1: Project Setup (2 hours)

- Initialize Next.js project
- Initialize NestJS project
- Initialize FastAPI project
- Setup Docker Compose (PostgreSQL, MongoDB)
- Configure environment variables
- Setup ESLint, Prettier
- Git repository initialization

Milestone 1.2: Authentication System (3 hours)

- User registration endpoint
- User login endpoint
- JWT token generation
- Password hashing with bcrypt
- Protected route middleware
- Login/Register UI pages

Milestone 1.3: Workspace CRUD (2 hours)

- Create workspace endpoint
- List workspaces endpoint
- Get workspace details endpoint
- Database schema implementation
- Basic workspace UI

Milestone 1.4: Task CRUD (3 hours)

- Create task endpoint
- List tasks endpoint
- Update task endpoint

- Delete task endpoint
- Task list UI (simple table/list view)

Deliverables: ✅ Working authentication ✅ Basic workspace management ✅ Basic task CRUD ✅ Simple UI with navigation

---

Phase 2: Real-time & AI (Day 2 - 8-10 hours)

Milestone 2.1: WebSocket Setup (2 hours)

- Configure NestJS WebSocket Gateway
- Implement Socket.io server
- Client-side Socket.io integration
- Join/leave room functionality
- Connection/disconnection handling

Milestone 2.2: Real-time Task Updates (2 hours)

- Broadcast task creation
- Broadcast task updates
- Broadcast task deletion
- Update React UI on WebSocket events
- Handle optimistic updates

Milestone 2.3: FastAPI AI Service (3 hours)

- Setup FastAPI project structure
- Implement priority prediction endpoint
- Implement time estimation endpoint
- Implement description enhancement endpoint
- Connect NestJS to FastAPI

Milestone 2.4: Activity Logging (1 hour)

- Setup MongoDB connection
- Create activity log model

- Log all task actions
- Store AI suggestions

Milestone 2.5: Task Board UI (2 hours)

- Kanban board layout
- Drag and drop implementation
- Task card component
- Real-time position updates

Deliverables: ✅ Real-time collaboration working ✅ AI features integrated ✅ Kanban board with drag & drop ✅ Activity logging system

---

Phase