TASK MANAGER APPLICATION

Complete Project Documentation

Final Project - Pemrograman Berbasis Kerangka Kerja (PBKK)

Disusun oleh:

Nama : Felda Ega Fadhila
NRP : 5025231199
Mata Kuliah : Pemrograman Berbasis Kerangka Kerja
Semester : Ganjil 2025/2026

DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2025

Daftar Isi

1. Pendahuluan

1.1. Latar Belakang

Dalam era digital yang serba cepat, manajemen tugas dan produktivitas menjadi aspek krusial dalam kehidupan sehari-hari, baik untuk keperluan pribadi maupun profesional. Banyak orang menghadapi kesulitan dalam mengorganisir tugas-tugas mereka, mengingat deadline, dan melacak progres pekerjaan. Hal ini mendorong kebutuhan akan aplikasi manajemen tugas yang efektif dan mudah digunakan.

Task Manager Application dikembangkan sebagai solusi komprehensif untuk mengatasi masalah tersebut. Aplikasi ini memungkinkan pengguna untuk membuat, mengelola, dan melacak tugas-tugas mereka dengan fitur-fitur canggih seperti kategorisasi, prioritisasi, notifikasi email, dan kemampuan berbagi tugas secara publik.

1.2. Tujuan Proyek

Proyek ini bertujuan untuk:

1. **Mengembangkan aplikasi web full-stack** yang menerapkan arsitektur modern dengan pemisahan yang jelas antara frontend dan backend.
2. **Mengimplementasikan fitur CRUD lengkap** untuk manajemen tugas dengan validasi dan otorisasi yang proper.
3. **Menerapkan sistem autentikasi dan otorisasi** menggunakan JWT (JSON Web Token) untuk keamanan aplikasi.
4. **Mengintegrasikan fitur notifikasi email** untuk reminder tugas dan ringkasan harian.
5. **Menyediakan antarmuka pengguna yang responsif** dan user-friendly menggunakan React dan Tailwind CSS.

1.3. Ruang Lingkup

Aplikasi Task Manager mencakup fitur-fitur berikut:

- **User Management**: Registrasi, login, dan manajemen profil pengguna
- **Task Management**: Operasi CRUD untuk tugas dengan berbagai atribut
- **Category Management**: Pengorganisasian tugas berdasarkan kategori
- **File Attachment**: Upload dan manajemen file lampiran untuk tugas
- **Email Notifications**: Reminder otomatis dan ringkasan harian via email
- **Public Tasks**: Kemampuan untuk membagikan tugas secara publik

2. Technology Stack

2.1. Backend Technologies

2.1.1. NestJS Framework

NestJS adalah framework Node.js yang progressive untuk membangun aplikasi server-side yang efisien dan scalable. Framework ini menggunakan TypeScript secara default dan menggabungkan elemen-elemen dari OOP (Object Oriented Programming), FP (Functional Programming), dan FRP (Functional Reactive Programming).

Alasan pemilihan NestJS:
- **Modular Architecture**: Struktur kode yang terorganisir dengan baik menggunakan modules
- **Dependency Injection**: Built-in DI container untuk manajemen dependencies
- **TypeScript Support**: Type safety yang kuat untuk mengurangi runtime errors
- **Decorators**: Syntax yang clean dan ekspresif
- **Extensive Ecosystem**: Banyak library dan tools yang tersedia

2.1.2. TypeORM

TypeORM adalah Object-Relational Mapping (ORM) library yang mendukung TypeScript dan JavaScript. Digunakan untuk interaksi dengan database SQLite dalam proyek ini.

Fitur TypeORM yang digunakan:
- Entity decorators untuk definisi model
- Repository pattern untuk operasi database
- Automatic migration dan synchronization
- Support untuk relations (OneToMany, ManyToOne)

2.1.3. SQLite Database

SQLite dipilih sebagai database untuk development karena:
- Tidak memerlukan server terpisah
- File-based storage yang portable
- Cocok untuk development dan testing
- Mudah di-setup dan maintain

2.1.4. Dependencies Backend

Package	Version	Fungsi
`@nestjs/core`	`^11.0.1`	Core framework NestJS
`@nestjs/jwt`	`^11.0.1`	JWT authentication
`@nestjs/passport`	`^11.0.5`	Passport.js integration
`@nestjs/typeorm`	`^11.0.0`	TypeORM integration
`@nestjs/schedule`	`^6.0.1`	Task scheduling (cron jobs)
`bcrypt`	`^6.0.0`	Password hashing
`class-validator`	`^0.14.3`	DTO validation

`nodemailer`	`^7.0.11`	Email sending
`multer`	`^2.0.2`	File upload handling
`sqlite3`	`^5.1.7`	SQLite database driver

2.2. Frontend Technologies

2.2.1. React 19

React adalah library JavaScript untuk membangun user interface. Versi 19 yang digunakan membawa banyak peningkatan performa dan fitur baru.

Fitur React yang diimplementasikan:

- Functional Components dengan Hooks
- React Router untuk navigasi
- Context API untuk state management
- Custom hooks untuk reusable logic

2.2.2. Vite

Vite adalah build tool generasi berikutnya yang menyediakan:

- Hot Module Replacement (HMR) yang sangat cepat
- Build yang optimal dengan Rollup
- Support native untuk ES modules
- Plugin ecosystem yang kaya

2.2.3. Tailwind CSS

Tailwind CSS adalah utility-first CSS framework yang memungkinkan:

- Rapid UI development dengan utility classes
- Konsisten design system
- Responsive design yang mudah
- Optimized production builds dengan PurgeCSS

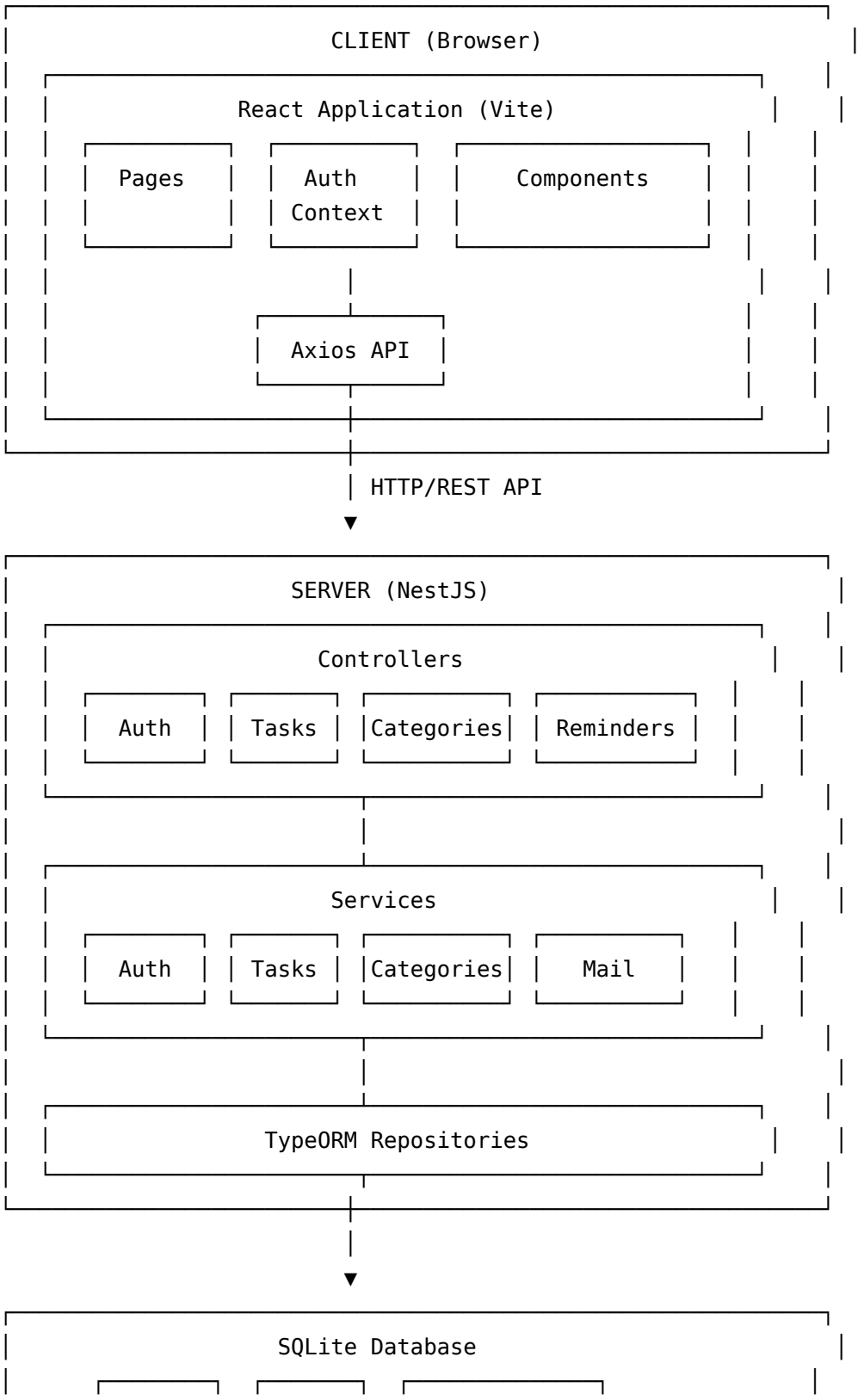
2.2.4. Dependencies Frontend

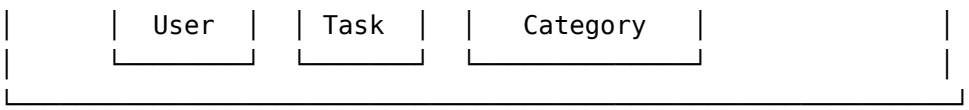
Package	Version	Fungsi
`react`	`^19.2.0`	UI Library
`react-dom`	`^19.2.0`	DOM rendering
`react-router-dom`	`^7.9.6`	Client-side routing
`axios`	`^1.13.2`	HTTP client
`tailwindcss`	`^4.1.17`	CSS framework
`vite`	`^7.2.5`	Build tool

3. Arsitektur Sistem

3.1. Overview Arsitektur

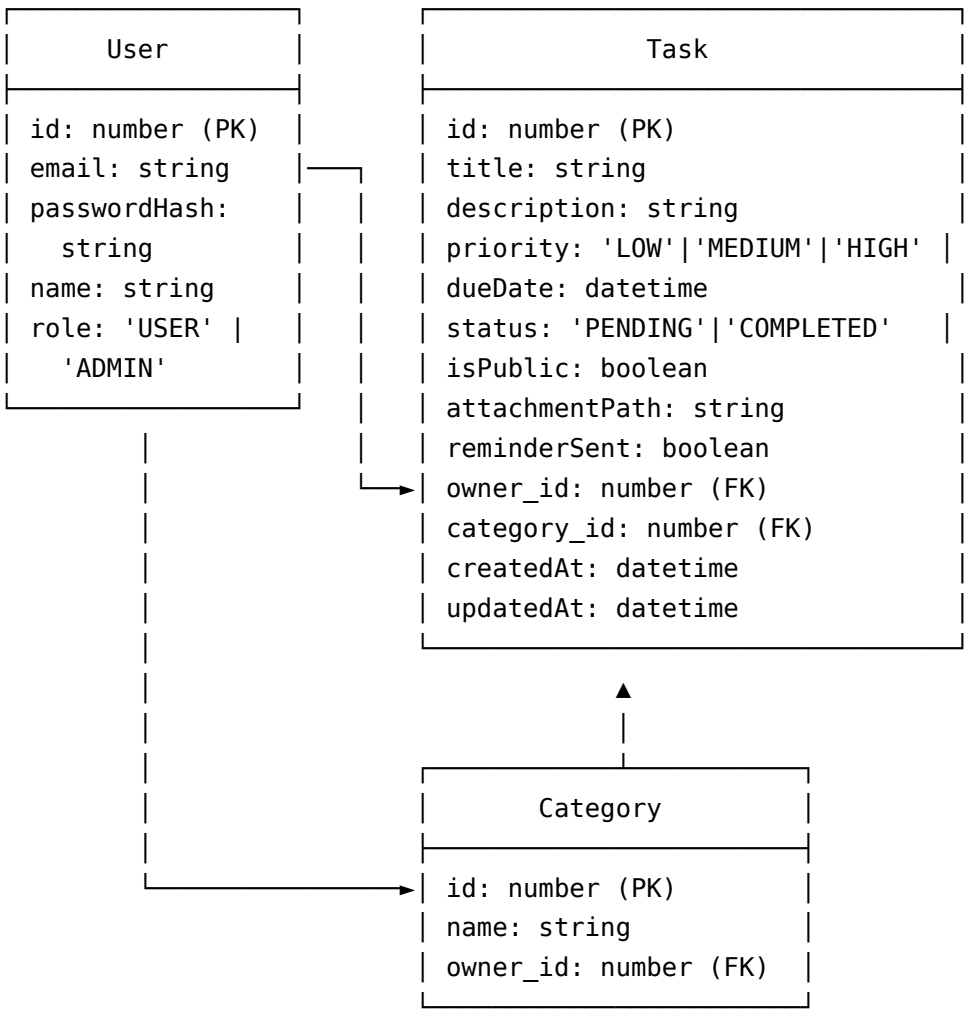
Aplikasi Task Manager menggunakan arsitektur **Client-Server** dengan pemisahan yang jelas antara frontend dan backend:





3.2. Database Schema

3.2.1. Entity Relationship Diagram



3.2.2. Relasi antar Entity

1. **User - Task** (One-to-Many)
   - Satu user dapat memiliki banyak tasks
   - Setiap task dimiliki oleh satu user
2. **User - Category** (One-to-Many)
   - Satu user dapat membuat banyak categories
   - Setiap category dimiliki oleh satu user
3. **Category - Task** (One-to-Many)
   - Satu category dapat memiliki banyak tasks
   - Setiap task dapat memiliki satu category (opsional)

3.3. Module Structure

3.3.1. Backend Modules

```
src/
├── main.ts                 # Application entry point
├── app.module.ts           # Root module
├── auth/                   # Authentication module
│   ├── auth.module.ts
│   ├── auth.controller.ts
│   ├── auth.service.ts
│   ├── jwt-auth.guard.ts
│   ├── dto/
│   │   ├── login.dto.ts
│   │   └── register.dto.ts
│   └── strategies/
│       └── jwt.strategy.ts
├── users/                  # Users module
│   ├── users.module.ts
│   ├── users.controller.ts
│   ├── users.service.ts
│   └── user.entity.ts
├── tasks/                  # Tasks module
│   ├── tasks.module.ts
│   ├── tasks.controller.ts
│   ├── tasks.service.ts
│   ├── task.entity.ts
│   └── dto/
│       ├── create-task.dto.ts
│       └── update-task.dto.ts
├── categories/             # Categories module
│   ├── categories.module.ts
│   ├── categories.controller.ts
│   ├── categories.service.ts
│   └── category.entity.ts
├── mail/                   # Email service module
│   ├── mail.module.ts
│   └── mail.service.ts
└── reminders/              # Reminder scheduling module
    ├── reminders.module.ts
    ├── reminders.controller.ts
    └── reminders.service.ts
```

3.3.2. Frontend Structure

```
src/
├── main.jsx                # Application entry point
├── App.jsx                 # Root component with routes
├── App.css                 # Global styles
```

```
├── index.css               # Tailwind imports
├── api.js                  # Axios configuration
├── auth/
│   └── AuthContext.jsx     # Authentication context
└── pages/
    ├── LoginPage.jsx       # Login page
    ├── RegisterPage.jsx    # Registration page
    ├── DashboardLayout.jsx # Main layout with sidebar
    ├── MyTasksPage.jsx     # Task listing page
    ├── NewTaskPage.jsx     # Create task page
    ├── TaskDetailPage.jsx  # Task detail view
    ├── CategoriesPage.jsx  # Category management
    ├── UsersListPage.jsx   # User listing
    └── PublicTasksPage.jsx # Public tasks view
```

4. API Documentation

4.1. Base URL

`http://localhost:3000`

4.2. Authentication

API menggunakan **JWT Bearer Token** untuk autentikasi. Token dikirim melalui header:

```
Authorization: Bearer <token>
```

4.3. Auth Endpoints

4.3.1. Register User

POST `/auth/register`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123",
  "name": "John Doe"
}
```

Response (201 Created):

```
{
  "id": 1,
  "email": "user@example.com",
  "name": "John Doe",
  "role": "USER"
}
```

4.3.2. Login

POST `/auth/login`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Response (200 OK):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

4.4. Users Endpoints

4.4.1. Get Current User

GET `/users/me` 🔒

Response:

```
{
  "id": 1,
  "email": "user@example.com",
  "name": "John Doe",
  "role": "USER"
}
```

4.4.2. Get All Users

GET `/users` 🔒

Response:

```
[
  {
    "id": 1,
    "email": "user@example.com",
    "name": "John Doe",
    "role": "USER"
  }
]
```

4.4.3. Search Users

GET `/users/search?q=john` 🔒

4.4.4. Get User by ID

GET `/users/:id` 🔒

4.5. Tasks Endpoints

4.5.1. Create Task

POST `/tasks` 🔒

Request Body:

```
{
  "title": "Complete documentation",
  "description": "Write project documentation",
  "priority": "HIGH",
  "dueDate": "2025-11-30T00:00:00.000Z",
  "categoryId": 1,
  "isPublic": false
}
```

Response (201 Created):

```
{
  "id": 1,
  "title": "Complete documentation",
```

```
  "description": "Write project documentation",
  "priority": "HIGH",
  "dueDate": "2025-11-30T00:00:00.000Z",
  "status": "PENDING",
  "isPublic": false,
  "attachmentPath": null,
  "reminderSent": false,
  "createdAt": "2025-11-29T10:00:00.000Z",
  "updatedAt": "2025-11-29T10:00:00.000Z"
}
```

4.5.2. Get All User Tasks

GET `/tasks` 🔒

Response:

```
{
  "items": [...],
  "total": 10,
  "page": 1,
  "limit": 10
}
```

4.5.3. Get Task by ID

GET `/tasks/:id` 🔒

4.5.4. Update Task

PATCH `/tasks/:id` 🔒

Request Body (partial update):

```
{
  "title": "Updated title",
  "status": "COMPLETED"
}
```

4.5.5. Toggle Task Status

PATCH `/tasks/:id/toggle` 🔒

Toggles between PENDING and COMPLETED status.

4.5.6. Delete Task

DELETE `/tasks/:id` 🔒

4.5.7. Upload Attachment

POST `/tasks/:id/attachment` 🔒

Request: `multipart/form-data` dengan field `file`

Constraints:

- Max file size: 5MB
- Supported formats: PDF, DOC, DOCX, XLS, XLSX, PPT, PPTX, TXT, ZIP, RAR, JPG, JPEG, PNG, GIF, MP4, MP3

4.5.8. Get Public Tasks by User

GET `/tasks/public/:userId`

No authentication required.

4.6. Categories Endpoints

4.6.1. Create Category

POST `/categories` 🔒

Request Body:

```
{
  "name": "Work"
}
```

4.6.2. Get All Categories

GET `/categories` 🔒

4.6.3. Get Category by ID

GET `/categories/:id` 🔒

4.6.4. Update Category

PATCH `/categories/:id` 🔒

4.6.5. Delete Category

DELETE `/categories/:id` 🔒

4.7. Reminders Endpoints

4.7.1. Test Due Soon Reminder

POST `/reminders/test/due-soon`

Manually triggers the due-soon email reminder.

4.7.2. Test Daily Summary

POST `/reminders/test/daily-summary`

Manually triggers the daily summary email.

4.7.3. Check Email Configuration

GET `/reminders/test/config`

Response:

```
{
  "mailUser": "use***",
```

```
  "mailPass": "SET (hidden)"
}
```

5. Implementasi Fitur

5.1. User Authentication

5.1.1. Registrasi dan Login

Sistem autentikasi menggunakan JWT (JSON Web Token) dengan flow sebagai berikut:

1. **Registrasi**:
   - User mengisi form dengan email, password, dan nama
   - Password di-hash menggunakan bcrypt sebelum disimpan
   - User record dibuat di database
2. **Login**:
   - User memasukkan email dan password
   - System memverifikasi kredensial
   - Jika valid, JWT token di-generate dan dikembalikan
   - Token disimpan di localStorage browser
3. **Protected Routes**:
   - Setiap request ke endpoint yang dilindungi harus menyertakan token
   - JwtAuthGuard memverifikasi token dan extract user info
   - Jika token invalid/expired, request ditolak dengan 401

5.1.2. JWT Token Structure

```
{
  "sub": 1,           // User ID
  "email": "user@example.com",
  "role": "USER",
  "iat": 1732873200,  // Issued At
  "exp": 1732876800   // Expiration (1 hour)
}
```

5.2. Task Management

5.2.1. Create Task

Form pembuatan task mencakup:

- **Title** (required): Judul task
- **Description** (optional): Deskripsi detail
- **Priority**: LOW, MEDIUM, atau HIGH dengan visual indicator
- **Due Date** (optional): Tanggal deadline
- **Category** (optional): Kategori untuk organisasi
- **Is Public**: Toggle untuk membuat task terlihat publik
- **Attachment** (optional): File lampiran

5.2.2. Task List dengan Filter

Halaman My Tasks menampilkan:

- **Stats Cards**: Total tasks, completed, pending, high priority
- **Filter Options**: Search by title, filter by priority, filter by category
- **Task Cards**: Menampilkan informasi task dengan visual priority indicator
- **Quick Actions**: View, Done/Undo, Edit, Delete

5.2.3. Task Detail Page

Halaman detail task menampilkan informasi lengkap:

- Header dengan warna sesuai priority
- Description dengan formatting
- Grid informasi: Priority, Due Date, Category, Status
- Attachment preview (image atau download link)
- Timestamps (created, updated)
- Action buttons: Toggle status, Edit, Delete

5.3. Category Management

Fitur kategori memungkinkan:

- Membuat kategori baru
- Melihat daftar kategori dengan jumlah task
- Edit nama kategori
- Hapus kategori (dengan konfirmasi)

5.4. Email Notifications

5.4.1. Due Soon Reminder

Cron job berjalan setiap jam untuk:

1. Mencari task dengan status PENDING
2. Filter task yang due dalam 24 jam ke depan
3. Check apakah reminder sudah pernah dikirim (reminderSent flag)
4. Kirim email reminder dengan detail task
5. Update reminderSent = true untuk menghindari spam

Email content:

```
Subject: ⏰ Reminder: Task "Task Title" is due soon!

Hi User Name,

This is a reminder that your task is due soon:

📋 Task: Task Title
📝 Description: Task description
🎯 Priority: HIGH
📅 Due Date: Sabtu, 30 November 2025, 00:00

Please make sure to complete it on time!
```

```
Best regards,
Task Manager App
```

5.4.2. Daily Summary

Cron job berjalan setiap hari jam 08:00 untuk:

1. Mengambil semua user
2. Untuk setiap user, ambil pending tasks
3. Kategorikan tasks: Overdue, Due Today, Upcoming, No Due Date
4. Kirim email summary jika ada pending tasks

Email content:

```
Subject: 📋 Daily Task Summary - X pending tasks

Hi User Name,

Here is your daily task summary:

📊 Total Pending Tasks: X

🔴 OVERDUE TASKS (X):
   • Task 1 [HIGH] - Due: 28/11/2025

🟡 DUE TODAY (X):
   • Task 2 [MEDIUM]

🟢 UPCOMING TASKS (X):
   • Task 3 [LOW] - Due: 01/12/2025

Have a productive day!
Best regards,
Task Manager App
```

5.5. File Upload

5.5.1. Upload Process

1. User memilih file melalui drag-drop area atau click
2. Frontend memvalidasi ukuran (max 5MB) dan tipe file
3. Setelah task dibuat, file di-upload sebagai attachment
4. Backend menyimpan file dengan unique filename di folder `/uploads`
5. Path file disimpan di database (attachmentPath)

5.5.2. File Preview

- **Image files**: Ditampilkan sebagai preview dengan kemampuan zoom
- **Document files**: Ditampilkan dengan icon sesuai tipe dan link download

5.6. Public Tasks

User dapat membuat task menjadi publik dengan mengaktifkan toggle “Make this task public”. Task publik dapat dilihat oleh user lain melalui halaman “Public Tasks” dengan memilih user tertentu.

6. Keamanan

6.1. Authentication Security

6.1.1. Password Hashing

Password user di-hash menggunakan **bcrypt** dengan cost factor 10 sebelum disimpan ke database:

```
const passwordHash = await bcrypt.hash(password, 10);
```

Saat login, password yang diinput dibandingkan dengan hash:

```
const isValid = await bcrypt.compare(password, user.passwordHash);
```

6.1.2. JWT Security

- Token memiliki expiration time (1 jam)
- Secret key disimpan di environment variable
- Token diverifikasi di setiap request ke protected endpoint

6.2. Authorization

6.2.1. Resource Ownership

Setiap operasi pada task dan category divalidasi untuk memastikan user hanya bisa mengakses resource miliknya:

```
if (task.owner.id !== userId) {
  throw new ForbiddenException();
}
```

6.2.2. Role-Based Access

User memiliki role: USER atau ADMIN. Saat ini digunakan untuk potensial future features seperti admin dashboard.

6.3. Input Validation

6.3.1. DTO Validation

Menggunakan `class-validator` untuk validasi input:

```
export class CreateTaskDto {
  @IsNotEmpty()
  @IsString()
  title: string;

  @IsOptional()
  @IsString()
  description?: string;

  @IsEnum(['LOW', 'MEDIUM', 'HIGH'])
  priority: string;
}
```

6.3.2. File Upload Validation

- Maximum file size: 5MB
- Allowed file types: documents, images, archives, media
- Unique filename generation untuk mencegah overwrite

6.4. CORS Configuration

Backend dikonfigurasi untuk hanya menerima request dari frontend origin:

```
app.enableCors({
  origin: 'http://localhost:5173',
  credentials: true,
});
```

7. Panduan Instalasi

7.1. Prerequisites

Sebelum menjalankan aplikasi, pastikan sudah terinstall:

- **Node.js** v18+ (direkomendasikan v20 LTS)
- **pnpm** package manager
- **Git** untuk version control

7.2. Clone Repository

```
git clone https://github.com/nafkhanzam-classrooms/
  x02-final-project-feldaega17.git
cd x02-final-project-feldaega17
```

7.3. Backend Setup

```
# Masuk ke direktori backend
cd backend

# Install dependencies
pnpm install

# Buat file .env
cp .env.example .env

# Edit .env dengan konfigurasi yang sesuai
```

7.3.1. Environment Variables

Buat file `.env` di folder backend:

```
# Mail Configuration
MAIL_USER=your-email@gmail.com
MAIL_PASS=your-app-password

# JWT Configuration
JWT_SECRET=your-secret-key
```

7.3.2. Gmail App Password

Untuk menggunakan Gmail sebagai email sender:

1. Aktifkan 2-Factor Authentication di akun Google
2. Buka: https://myaccount.google.com/apppasswords
3. Generate App Password untuk “Mail”
4. Gunakan 16-karakter password tersebut di MAIL_PASS

7.3.3. Run Backend

```
# Development mode
pnpm run start:dev
```

```
# Production mode
pnpm run build
pnpm run start:prod
```

Backend akan berjalan di `http://localhost:3000`

7.4. Frontend Setup

```
# Masuk ke direktori frontend
cd frontend

# Install dependencies
pnpm install

# Run development server
pnpm run dev
```

Frontend akan berjalan di `http://localhost:5173`

7.5. Build for Production

7.5.1. Backend Build

```
cd backend
pnpm run build
# Output di folder dist/
```

7.5.2. Frontend Build

```
cd frontend
pnpm run build
# Output di folder dist/
pnpm run preview  # Preview production build
```

8. Testing

8.1. Backend Unit Tests

Project menggunakan **Jest** sebagai testing framework.

8.1.1. Run Tests

```
cd backend

# Run all tests
pnpm test

# Run tests with coverage
pnpm run test:cov

# Run tests in watch mode
pnpm run test:watch
```

8.1.2. Test Structure

Setiap module memiliki file spec untuk unit testing:

```
src/
├── auth/
│   ├── auth.controller.spec.ts
│   └── auth.service.spec.ts
├── tasks/
│   ├── tasks.controller.spec.ts
│   └── tasks.service.spec.ts
├── categories/
│   ├── categories.controller.spec.ts
│   └── categories.service.spec.ts
└── ...
```

8.1.3. Test Coverage

Test coverage report tersedia di folder `coverage/`:

```
pnpm run test:cov
open coverage/lcov-report/index.html
```

8.2. API Testing

API dapat ditest menggunakan tools seperti:

- **Postman**: Import collection dari dokumentasi
- **curl**: Command line testing
- **Thunder Client**: VS Code extension

8.2.1. Example curl Commands

```
# Register
curl -X POST http://localhost:3000/auth/register \
  -H "Content-Type: application/json" \
```

```
  -d '{"email":"test@test.com","password":"123456","name":"Test"}'

# Login
curl -X POST http://localhost:3000/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@test.com","password":"123456"}'

# Get Tasks (dengan token)
curl http://localhost:3000/tasks \
  -H "Authorization: Bearer <token>"
```

9. User Interface Screenshots

Berikut adalah deskripsi tampilan aplikasi:

9.1. Login Page

Halaman login menampilkan form dengan:

- Input email
- Input password
- Tombol Login
- Link ke halaman Register
- Design modern dengan gradient background

9.2. Register Page

Halaman registrasi menampilkan form dengan:

- Input nama
- Input email
- Input password
- Tombol Register
- Link ke halaman Login

9.3. Dashboard / My Tasks

Halaman utama menampilkan:

- **Sidebar Navigation**: Logo, menu items, logout button
- **Stats Cards**: Total tasks, completed, pending, high priority dengan warna berbeda
- **Filter Section**: Search bar, priority dropdown, category dropdown
- **Task Grid**: Card-based task display dengan priority indicator

9.4. Create Task Page

Form pembuatan task dengan:

- Header gradient dengan icon
- Input title dan description
- Priority selection dengan button group (Low/Medium/High)
- Date picker dan category dropdown
- Public toggle switch
- File upload area dengan drag-drop support
- Submit button dengan loading state

9.5. Task Detail Page

Halaman detail menampilkan:

- Header dengan warna sesuai priority
- Status badge (Completed/Pending)
- Description section
- Info grid: Priority, Due Date, Category, Status

- Attachment viewer/downloader
- Timestamps
- Action buttons: Toggle, Edit, Delete

9.6. Categories Page

Halaman manajemen kategori dengan:

- Add category form
- Category list dengan task count
- Edit dan delete buttons per category

9.7. Users Page

Halaman daftar user dengan:

- User cards menampilkan nama dan email
- Link untuk melihat public tasks setiap user

10. Kesimpulan

10.1. Ringkasan

Proyek **Task Manager Application** telah berhasil dikembangkan dengan fitur-fitur berikut:

1. ✅ **User Authentication** - Register, login dengan JWT
2. ✅ **Task CRUD** - Create, read, update, delete tasks
3. ✅ **Task Attributes** - Priority, due date, status, description
4. ✅ **Category Management** - Organisasi task dengan kategori
5. ✅ **File Attachment** - Upload dokumen dan gambar
6. ✅ **Email Notifications** - Reminder H-1 dan daily summary
7. ✅ **Public Tasks** - Berbagi task secara publik
8. ✅ **Responsive UI** - Tampilan yang baik di berbagai device

10.2. Teknologi yang Diterapkan

- **Backend**: NestJS, TypeORM, SQLite, JWT, Nodemailer
- **Frontend**: React 19, Vite, Tailwind CSS, React Router
- **Tools**: Git, VS Code, Postman

10.3. Pembelajaran

Melalui proyek ini, telah dipelajari:

1. Pengembangan aplikasi full-stack dengan arsitektur modern
2. Implementasi autentikasi dan otorisasi dengan JWT
3. Pengelolaan state dan routing di React
4. Desain database relasional dengan TypeORM
5. Implementasi REST API dengan NestJS
6. Integrasi email service dengan Nodemailer
7. Task scheduling dengan cron jobs
8. File upload handling

10.4. Pengembangan Selanjutnya

Fitur yang dapat dikembangkan di masa depan:

- **Real-time Notifications** menggunakan WebSocket
- **Task Collaboration** untuk berbagi task dengan user lain
- **Task Comments** untuk diskusi dalam task
- **Recurring Tasks** untuk task yang berulang
- **Mobile Application** menggunakan React Native
- **Advanced Analytics** dashboard untuk produktivitas
- **Integration** dengan calendar apps (Google Calendar, etc.)

11. Referensi

11.1. Dokumentasi Official

1. **NestJS Documentation**
   https://docs.nestjs.com/
2. **React Documentation**
   https://react.dev/
3. **TypeORM Documentation**
   https://typeorm.io/
4. **Tailwind CSS Documentation**
   https://tailwindcss.com/docs
5. **Vite Documentation**
   https://vitejs.dev/guide/

11.2. Resources

1. **JWT.io** - JSON Web Token Introduction
   https://jwt.io/introduction
2. **Nodemailer** - Email sending for Node.js
   https://nodemailer.com/
3. **bcrypt** - Password hashing library
   https://www.npmjs.com/package/bcrypt

12. Lampiran

12.1. Struktur Folder Lengkap

```
x02-final-project-feldaega17/
├── README.md
├── package.json
├── profile.txt
├── docs/
│   └── main.typ
├── backend/
│   ├── package.json
│   ├── tsconfig.json
│   ├── nest-cli.json
│   ├── .env
│   ├── database.sqlite
│   ├── uploads/
│   ├── coverage/
│   ├── dist/
│   ├── src/
│   │   ├── main.ts
│   │   ├── app.module.ts
│   │   ├── app.controller.ts
│   │   ├── app.service.ts
│   │   ├── auth/
│   │   ├── users/
│   │   ├── tasks/
│   │   ├── categories/
│   │   ├── mail/
│   │   └── reminders/
│   └── test/
└── frontend/
    ├── package.json
    ├── vite.config.js
    ├── index.html
    ├── public/
    └── src/
        ├── main.jsx
        ├── App.jsx
        ├── api.js
        ├── auth/
        └── pages/
```

12.2. Informasi Penulis

Nama	Felda Ega Fadhila
NRP	5025231199
Email	feldaega17@gmail.com

Program Studi Teknik Informatika
Institusi Institut Teknologi Sepuluh Nopember

Dokumentasi ini dibuat sebagai bagian dari Final Project
Mata Kuliah Pemrograman Berbasis Kerangka Kerja (PBKK)
Semester Ganjil 2025/2026