Nama : Tamam Fajar Briliansyah
NRP : 5025231142

Dokumentasi FP PBKK

I. Backend

1. Skema Database

A. User:

Fields:

- id	: BigInt (Primary Key, Auto-increment)
- name	: String
- email	: String (Unique)
- password	: String (Hashed with bcryptjs)
- is_verified	: Boolean (Default: false)
- created_at	: DateTime (Default: now)
- updated_at	: DateTime (Auto-update)

Relations:

- categories	: Category[] (One-to-Many)
- tasks	: Task[] (One-to-Many)
- refreshToken	: RefreshToken[] (One-to-Many)
- emailTokens	: EmailVerificationToken[] (One-to-Many)
- emailLogs	: EmailLog[] (One-to-Many)

B. EmailVerificationToken:

Purpose : Menyimpan OTP code untuk verifikasi email

Fields:

- id	: BigInt (Primary Key, Auto-increment)
- user_id	: BigInt (Foreign Key -> User.id)
- otp_code	: String (6-digit code)
- expires_at	: DateTime (10 minutes expiry)
- used	: Boolean (Default: false)
- created_at	: DateTime (Default: now)

Relations:

- user	: User (Many-to-One)

C. RefreshToken:

Purpose: Menyimpan refresh tokens untuk JWT authentication

Fields:

- id	: BigInt (Primary Key, Auto-increment)
- user_id	: BigInt (Foreign Key -> User.id)
- token	: String
- expires_at	: DateTime

- created_at	: DateTime (Default: now)

Relations:

- user	: User (Many-to-One)

D. Category:

Purpose	: Kategorisasi tasks

Fields:

- id	: BigInt (Primary Key, Auto-increment)
- user_id	: BigInt (Foreign Key -> User.id)
- name	: String
- color	: String (Default: "#6b7280", Hex color code)
- icon	: String (Default: "folder", Icon name)
- created_at	: DateTime (Default: now)
- updated_at	: DateTime (Auto-update)

Relations:

- user	: User (Many-to-One)
- tasks	: Task[] (One-to-Many)

Note:
Setiap user baru otomatis mendapat category "Home" setelah verifikasi OTP

E. Task:

Purpose	: Task/Todo items dengan berbagai properties

Fields:

- id	: BigInt (Primary Key, Auto-increment)
- user_id	: BigInt (Foreign Key -> User.id)
- category_id	: BigInt? (Foreign Key -> Category.id, Nullable)
- title	: String
- description	: String? (Nullable)
- priority	: Priority (Enum: low, medium, high)
- due_date	: DateTime? (Nullable)
- reminder_at	: DateTime? (Nullable, waktu kirim reminder email)
- reminder_sent	: Boolean (Default: false)
- status	: Status (Enum: pending, complete, Default: pending)
- visibility	: Visibility (Enum: private, public, Default: private)
- created_at	: DateTime (Default: now)
- updated_at	: DateTime (Auto-update)

Relations:

- user	: User (Many-to-One)
- category	: Category? (Many-to-One, Nullable)
- files	: TaskFile[] (One-to-Many)
- emailLogs	: EmailLog[] (One-to-Many)

Note:
- Public tasks dapat dilihat oleh user lain
- Private tasks hanya dilihat oleh owner
- reminder_at digunakan oleh scheduler untuk kirim email reminder

F. TaskFile:
Purpose : File attachments untuk tasks (upload files)

Fields:
- id : BigInt (Primary Key, Auto-increment)
- task_id : BigInt (Foreign Key -> Task.id)
- file_path : String (Path to uploaded file)
- original_name : String (Original filename)
- mime_type : String (File MIME type)
- size : BigInt (File size in bytes)
- uploaded_at : DateTime (Default: now)

Relations:
- task : Task (Many-to-One)

G. EmailLog:
Purpose : Logging semua email yang dikirim oleh system

Fields:
- id : BigInt (Primary Key, Auto-increment)
- user_id : BigInt (Foreign Key -> User.id)
- task_id : BigInt? (Foreign Key -> Task.id, Nullable)
- type : EmailType (Enum)
- sent_at : DateTime (Default: now)

Relations:
- user : User (Many-to-One)
- task : Task? (Many-to-One, Nullable)

ENUM EmailType:
- otp_verification
- deadline_reminder
- task_completed
- task_assigned

2. Struktur Folder Backend

```
server/
├── prisma/
│   ├── schema.prisma                    # Prisma schema definition
│
├── src/
```

```
│   ├── main.ts                          # Application entry point
│   ├── app.module.ts                    # Root module
│   ├── app.controller.ts                # Root controller
│   ├── app.service.ts                   # Root service
│   │
│   ├── auth/                            # Authentication module
│   │   ├── auth.module.ts
│   │   ├── auth.controller.ts           # Login, Register, OTP endpoints
│   │   ├── auth.service.ts              # Auth business logic
│   │   ├── jwt.strategy.ts              # Passport JWT strategy
│   │   ├── jwt-auth.guard.ts            # JWT guard
│   │   └── dto.ts                       # DTOs (RegisterDto, dll)
│   │
│   ├── users/                           # Users module
│   │   ├── users.module.ts
│   │   ├── users.controller.ts          # User profile endpoints
│   │   └── users.service.ts             # User business logic
│   │
│   ├── categories/                      # Categories module
│   │   ├── categories.module.ts
│   │   ├── categories.controller.ts     # Category CRUD endpoints
│   │   ├── categories.service.ts        # Category business logic
│   │   └── dto/
│   │       └── create-category.dto.ts
│   │
│   ├── tasks/                           # Tasks module
│   │   ├── tasks.module.ts
│   │   ├── tasks.controller.ts          # Task CRUD
│   │   ├── tasks.service.ts             # Task business logic
│   │   └── dto/
│   │       └── create-task.dto.ts
│   │
│   ├── mail/                            # Email service module
│   │   ├── mail.module.ts
│   │   └── mail.service.ts              # Nodemailer email sending logic
│   │
│   ├── scheduler/                       # Task reminder scheduler
│   │   ├── scheduler.module.ts
│   │   └── reminder.scheduler.ts        # Cron-like task reminder
│   │
│   └── prisma/                          # Prisma module
│       ├── prisma.module.ts
│       └── prisma.service.ts            # Prisma client service
│
├── uploads/                             # File upload storage
├── package.json                         # Dependencies & scripts
├── tsconfig.json                        # TypeScript configuration
└── nest-cli.json                        # NestJS CLI configuration
```

3. Features & Modules

   A. MODUL AUTENTIKASI (src/auth/)

   Fitur:
   Pendaftaran OTP email, verifikasi OTP, login JWT, hashing password bcrypt, auto-create kategori "Home"

   Endpoints:
   POST /auth/register, POST /auth/login, POST /auth/verify-otp, POST /auth/resend-otp

   Logika:
   OTP 6 digit (100000-999999), kadaluarsa 10 menit, JWT payload {sub: user.id, email}, hanya verified user bisa login

   B. MODUL PENGGUNA (src/users/)

   Fitur:
   Ambil profil pengguna saat ini

   Endpoint:
   GET /users/me (JWT protected)

   C. MODUL KATEGORI (src/categories/)

   Fitur:
   CRUD kategori, case-insensitive duplicate checking, default kategori "Home"

   Endpoints:
   GET /categories, POST /categories, DELETE /categories/:id

   Property:
   name (string), color (hex, default "#6b7280"), icon (folder, phone, wifi, tag, home, cart, heart, star, mail, calendar, briefcase, book)

   Logika:
   Tidak boleh duplikat, kategori dengan tasks tidak bisa dihapus, trim whitespace

   D. MODUL TUGAS (src/tasks/)

   Fitur:
   CRUD tasks, upload/delete file, mark complete, filter (Today/Upcoming/Completed), search, public/private, priority (low/medium/high), overdue detection, task reminder

Endpoints:
GET /tasks/today, /tasks/upcoming, /tasks/completed, /tasks/sidebar-summary, /tasks/category/:id, /tasks/search

POST /tasks, POST /tasks/:id/files

DELETE /tasks/:id/files/:fileId

PATCH /tasks/:id, PATCH /tasks/:id/complete

Filter:
TODAY: due_date <= hari ini, status != complete, include own & public tasks

UPCOMING: due_date 7 hari ke depan, grouped by date

COMPLETED: status = complete, own tasks only, grouped by completion date

E. **MODUL EMAIL (src/mail/)**

Fitur:
OTP verification email, task reminder email, HTML templates, retry mechanism

SMTP:
Host (env), Port 587 (STARTTLS), Auth (env)

F. **MODUL SCHEDULER (src/scheduler/)**

Tujuan:
Kirim email reminder untuk tasks dengan reminder_at yang sudah lewat

Fitur:
Cek setiap 60 detik, query reminder_at <= now AND reminder_sent = false AND status = pending, send email & update reminder_sent = true, log ke EmailLog

Lifecycle:
onModuleInit (start), onModuleDestroy (cleanup)

4. API Endpoints Documentation

A. **AUTH endpoints:**
POST /auth/register Body: {name, email, password}
Response: {status, email} Error: 409 Email sudah terdaftar

POST /auth/login Body: {email, password}
Response: {access_token}
Error: 401 Invalid email/password atau email belum verified

POST /auth/verify-otp
Body: {email, otp}
Response: {status}
Side effect: is_verified=true, OTP dimark as used, "Home" category dibuat
Error: 404 User not found, 400 Invalid/expired OTP

POST /auth/resend-otp
Body: {email}
Response: {status, email}
Error: 404 User not found, 400 Email sudah verified

B. **USER endpoints:**
GET /users/me (JWT protected)
Response: {id, name, email, created_at}
Error: 401 Invalid/missing JWT

C. **CATEGORY endpoints**
GET /categories (JWT protected)
Response: [{id, name, color, icon, created_at}, ...]

POST /categories (JWT protected)
Body: {name, color, icon}
Response: {id, name, color, icon, created_at}
Error: 409 Category sudah exists (case-insensitive)

DELETE /categories/:id (JWT protected)
Response: {message}
Error: 404 Not found, 409 Cannot delete category dengan tasks

D. **TASK endpoints**
GET /tasks/today (JWT protected)
Response: {today: [...], overdue: [...]} Termasuk own tasks + public tasks dari users lain

GET /tasks/upcoming (JWT protected)
Response: {overdue: [...], grouped: {date: [...]}} Grouped by date (YYYY-MM-DD)

GET /tasks/completed (JWT protected)
Response: {grouped: {date: [...]}} Own tasks only, grouped by completion date (desc)

GET /tasks/sidebar-summary (JWT protected)
Response: {todayCount, upcomingCount, categories: [{categoryId, categoryName, taskCount}, ...]}

GET /tasks/category/:categoryId (JWT protected)
Response: [{id, title, category, ...}, ...]

GET /tasks/search?q=keyword (JWT protected)
Response: [{id, title, description, ...}, ...]

POST /tasks (JWT protected) Body: {title, description?, priority?, due_date?, reminder_at?, category_id?, visibility?}
Response: {id, title, description, priority, status, due_date, reminder_at, reminder_sent, visibility, created_at, updated_at}
Notes: title=required, priority default="low", visibility default="private"

POST /tasks/:id/files (JWT protected, multipart/form-data)
Body: files (max 10)
Response: {message, files: [{id, file_path, original_name, mime_type, size}, ...]}
Error: 404 Task not found, 403 Not task owner, 400 No files

DELETE /tasks/:id/files/:fileId (JWT protected)
Response: {message}
Error: 404 Not found, 403 Not task owner

PATCH /tasks/:id (JWT protected)
Body: {title?, description?, priority?, due_date?, reminder_at?, category_id?, visibility?}
Response: {id, title, ...} Error: 404 Not found, 403 Not task owner

PATCH /tasks/:id/complete (JWT protected)
Response: {id, status, ...}
Error: 404 Not found, 403 Not task owner

5. Authentication & Authorization

   A. JWT authentication

   Strategy: Passport JWT (passport-jwt) Guard: JwtAuthGuard

   Payload: {sub (user ID), email, iat, exp (24h)} Algorithm: HS256 Secret: JWT_SECRET (env var)

   Protected Routes: Semua kecuali POST /auth/register, POST /auth/login, POST /auth/verify-otp, POST /auth/resend-otp

   Header: Authorization: Bearer <JWT_TOKEN>

   Validasi: Extract dari header → verify signature & expiry → attach ke req.user

   B. Password Hashing:

Library: bcryptjs Algorithm: bcrypt Rounds: 10

Register: bcrypt.hash(password, 10)

Login: bcrypt.compare(inputPassword, hashedPassword)

C. **Authorization:**

Task Ownership: Only owner bisa update/delete task, upload/delete files. Public tasks bisa dilihat siapa saja, private tasks hanya visible untuk owner.

Category Ownership: Only owner bisa delete category. Categories isolated per user.

6. Email System:

A. **EMAIL SERVICE:**
Transport: Nodemailer
Protocol: SMTP with STARTTLS
Config: Host (SMTP_HOST env), Port 587 (SMTP_PORT env), Secure=false, Auth dengan SMTP_USER & SMTP_PASS (env)

B. **EMAIL TEMPLATES:**
OTP Verification: Subject "Your TodoList Email Verification Code", header "Email Verification", OTP code dalam chip besar (background hitam, text putih), expiry notice 10 minutes, design minimalist modern.

Task Reminder: Subject "Task Reminder: [Task Title]", greeting dengan user name, task details card (title, description, due date formatted, priority badge color-coded, category name), CTA button, footer, design professional card-based.

C. **EMAIL LOGGING:**
Semua email logged ke EmailLog table: user_id, task_id, type (enum: otp_verification, deadline_reminder, task_completed, task_assigned), sent_at

7. File Upload System

A. **File Storage**
Location: uploads/ directory (server root) Access: Static files served at /uploads/ prefix Config (main.ts): app.useStaticAssets(uploadsPath, {prefix: '/uploads/'})

B. **Upload Mecahnism**
Library: Multer (@nestjs/platform-express)
Interceptor: FilesInterceptor('files', 10) Max files: 10 per request
File naming: [timestamp]-[original-filename],

contoh: 1733654321000-document.pdf
Metadata stored: file_path (relative path), original_name, mime_type, size (bytes)

C. File Operations
Upload: Validate task ownership → save ke uploads/ → create TaskFile records → return metadata
Delete: Validate task ownership → delete physical file → delete TaskFile record
Supported: Semua file types supported Size limit: Configurable via Multer (default: no limit)

D. File Security
Authorization: Only task owner bisa upload/delete files. Files accessible via public URL (no auth required for viewing).
Validation: Task harus exist, user harus task owner, files array tidak boleh kosong

8. Scheduler System (Task Reminders)

A. Reminder Scheduler
File: src/scheduler/reminder.scheduler.ts
Class: ReminderSchedulerService (implements OnModuleInit)
Schedule: Every 60 detik, runs immediately on module init, continuous sampai module destroyed

B. Reminder Logic
Query: reminder_at <= NOW() AND reminder_sent = false AND status = 'pending'
Process: Fetch task dengan user & category → send reminder email via MailService → if success: update reminder_sent = true & create EmailLog entry (type: deadline_reminder) → log success/failure
Error handling: Individual task errors logged tapi tidak stop batch, email send failures logged, database errors logged

C. Email Content
Reminder email includes: User name (greeting), task title, task description, due date (formatted), priority level, category name
Timing: reminder_at harus sebelum due_date (contoh: 1 jam atau 1 hari sebelum)
Example flow: Task created dengan due_date 2025-12-10 10:00:00 & reminder_at 2025-12-10 09:00:00 → at 09:00:00 scheduler detect reminder_at <= now → send email to task owner → set reminder_sent = true → task tidak akan trigger lagi

II. Frontend

1. Project Structure

```
client/
├── public/                    # Static assets
```

```
│
├── src/
│   ├── main.tsx                    # Application entry point
│   ├── App.tsx                     # Root component dengan routing
│   ├── index.css                   # Global styles (2168 lines)
│   ├── vite-env.d.ts               # Vite type definitions
│   │
│   ├── api.ts                      # API client & endpoints (306 lines)
│   │
│   ├── assets/                     # Static assets
│   │   ├── bell-klink.ts           # Audio notification sound
│   │   ├── DataScanning.json       # Lottie animation (auth)
│   │   └── Loveisblind.json        # Lottie animation (empty states)
│   │
│   └── components/                 # React components
│       ├── AuthLayout.tsx          # Login & Register page
│       ├── OtpPage.tsx             # OTP verification page
│       ├── TodayPage.tsx           # Today's tasks view
│       ├── UpcomingPage.tsx        # Upcoming tasks view
│       ├── CompletedPage.tsx       # Completed tasks view
│       │
│       └── layout/
│           └── MainLayout.tsx      # Main app layout dengan sidebar
│
├── index.html                      # HTML template
├── package.json                    # Dependencies & scripts
├── tsconfig.json                   # TypeScript configuration
├── tsconfig.app.json               # App-specific TS config
├── tsconfig.node.json              # Node-specific TS config
├── vite.config.ts                  # Vite configuration
├── eslint.config.js                # ESLint configuration
```

2. Features & Components

A. Authentication Features

Register dengan name, email, password → OTP 6-digit dikirim ke email → verify OTP → account activated & "Home" category auto-created → login dengan email & password → receive JWT → token stored in localStorage → auto auth header untuk semua API calls.

Features: User registration, email OTP verification, login JWT, OTP resend, auto token storage, protected routes, auto-redirect based auth status, toast notifications

B. Task Management Features

Today View: Show tasks due today & overdue tasks (include own tasks + public tasks dari users lain), real-time filtering, task completion toggle, quick task creation, file attachments, priority indicators,

category badges, owner info untuk public tasks, empty state dengan Lottie animation.
Upcoming View: Show tasks next 7 days grouped by date, include overdue section, calendar-based visualization, task details preview, same management features as Today.
Completed View: Show all completed tasks grouped by completion date (descending), own tasks only, task details dengan completion timestamp, archive visualization.

C. **Category Management**
Create custom categories, choose color (hex picker), choose icon (folder, phone, wifi, tag, home, cart, heart, star, mail, calendar, briefcase, book), view/filter tasks by category, delete categories (if no tasks), category count in sidebar, default "Home" category untuk new users.

D. **Search & File**
Search: Real-time search di task title & description, case-insensitive, search results page, highlight terms, keyboard shortcut.
File: Max 10 files per upload, drag & drop upload, file preview (image preview, PDF page navigation, document icon), file download, file deletion, file metadata, upload progress, file type icons.

E. **UI/UX Featurse**
Responsive sidebar navigation, dynamic sidebar summary (counts), modal dialogs (task creation/editing), toast notifications, loading states, empty states dengan Lottie animations, smooth animations & transitions, priority & category color coding, interactive hover states, keyboard navigation, bell sound notification pada task completion, professional color scheme, glassmorphism effects.

3. Routing & Navigation

A. **Routing**
Router: React Router DOM v7.9.6 Type: BrowserRouter
Public Routes: / (redirect ke /signin), /signin (AuthLayout mode signin), /signup (AuthLayout mode signup), /otp (OtpPage)
Protected Routes (MainLayout dengan sidebar & header): /today (today's & overdue tasks), /upcoming (next 7 days), /completed (completed tasks)
Fallback: * (redirect ke /signin)

B. **Navigation Guards**
Auth check: Token presence di localStorage, if no token → redirect /signin, if token exists → allow protected routes.
Auto-redirect: Logged in user accessing /signin → redirect /today. Unauthenticated user accessing protected route → redirect /signin.
Navigation: useNavigate() hook (programmatic), <Link> component (declarative), location.state (pass data between routes)

4. State Management

   A. Local Component State
   Task lists: today (TaskDTO[]), overdue (TaskDTO[]), upcoming (grouped by date), completed (grouped by completion date).
   UI state: loading (boolean), showModal (boolean), selectedTask (TaskDTO | null), searchQuery (string), activeCategory (number | null).
   Form state: formData {title, description, priority, etc.}, errors {field: string}, isSubmitting (boolean).

   B. Shared State
   MainLayout provides shared data ke child routes: currentUser (CurrentUser), categories (CategoryDTO[]), sidebarSummary (SidebarSummaryResponse). Passed via context atau re-fetching di setiap component.

   C. Persistent State
   localStorage: access_token (JWT token, set on login, removed on logout), OTP email (untuk OTP page).

   D. State Sync
   Data refetching: After create/update/delete operations, useEffect dependencies untuk auto-refresh, manual refresh buttons.

5. Authentication Flow

   A. Registration Flow
   Component: AuthLayout (mode signup). User fills form (name, email, password) → frontend validate input → POST /auth/register {name, email, password} → backend create user (is_verified=false), generate 6-digit OTP, send ke email → frontend receive success → navigate /otp dengan email in state.

   B. OTP Verification Flow
   Component: OtpPage. User receive email dengan OTP code → enter OTP (6 digits) → POST /auth/verify-otp {email, otpCode} → backend validate OTP (match, not expired <10 min, not already used) → if valid: set is_verified=true, create "Home" category, mark OTP as used → frontend receive success → show success toast → navigate /signin.
   Resend OTP: User click "Resend OTP" → POST /auth/resend-otp → old OTP marked as used, new OTP generated & sent.

   C. Login Flow
   Component: AuthLayout (mode signin). User enter credentials (email, password) → POST /auth/login {email, password} → backend validate email exists, password correct, account verified → if valid: generate

JWT access token → return {access_token} → frontend store token di localStorage → navigate /today.

D. Auth Req and Logout

All API requests include Authorization header dengan Bearer token dari localStorage. Jika 401 Unauthorized received → clear localStorage → redirect /signin.

Logout: Click logout button → clear localStorage (remove access_token) → navigate /signin.

6. File Upload & Preview

A. File Upload System

Methods: Drag & drop atau file input button. Max 10 files per upload.

Endpoint: POST /tasks/:taskId/files

Flow: Select/drop files → validate count → create FormData → POST multipart/form-data → update file list → show toast.

B. File Preview System

Image: Thumbnail di task card, click buka full-size modal.

PDF: pdfjs-dist v5.4.449, page navigation, page indicator, zoom & download buttons.

Document: Non-previewable files show icon, filename, size, download button.

File icons: PDF=red, DOC=blue, XLS=green, ZIP=yellow, generic=gray.

C. File Management

File card: icon/thumbnail, filename, size (KB/MB), action buttons (preview, download, delete).

Delete: Only owner, confirmation dialog, DELETE /tasks/:taskId/files/:fileId, update UI.

Download: Direct link, URL format: /uploads/filename

7. Animations & Interactions

A. Lottie Animations

Library: lottie-react v2.4.1

DataScanning.json: Used di AuthLayout (login/register background), tech/scanning animation, loop true.

Loveisblind.json: Used di empty states (TodayPage, etc.), no tasks found, cute/friendly illustration, loop true.

Implementation: Import Lottie dari lottie-react, pass animationData prop, set loop & style.

B. Sound Efect

Bell klink sound: File src/assets/bell-klink.ts, trigger on task completion, audio feedback untuk user action.

Implementation: Create Audio object dari /bell-klink.mp3, set volume

0.5, call play() dengan error handling. Trigger saat task marked complete.