System Design Documentation

1. Arsitektur design

HeyWebsite adalah aplikasi berbasis web yang memungkinkan pengguna untuk membuat **space/grup**, mengirim pesan, membuat dan mengelola catatan (note), serta mengatur undangan anggota.

1.1 Teknologi yang digunakan

Frontend	Backend	Database	Deployment
1. Framework: Next.js 2. Styling: TailwindCSS + shadcn/ui 3. Rich text editor: BlockNote.	1. Database ORM: Prisma ORM. 2. Hash password: bcryptjs. 3. Token management: jsonwebtoken.	1. PostgreSQL (Supabase)	1. Vercel (frontend + backend) 2. Supabase (database + storage)

1.2 Alasan pemilihan teknologi

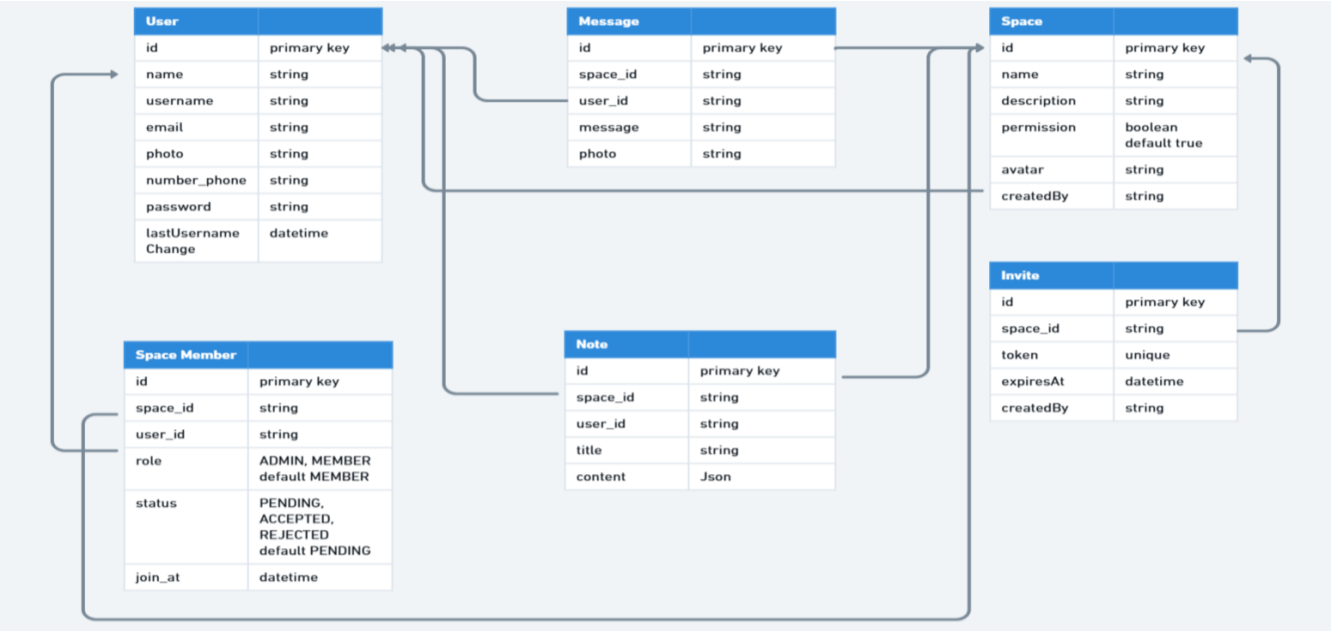
a. Prisma → mempermudah query & maintain schema dengan migration.
b. Supabase → database Postgres & storage langsung tersedia.
c. TailwindCSS → styling cepat dan mudah.
d. BlockNote → mendukung text editing seperti notion.
e. Cookies + JWT → karena digunakan untuk skala kecil.

1.3 Authentication Flow

a. User melakukan login dengan username, email, dan password.
b. Server memverifikasi apakah ada username dan email di database.
c. Server memverifikasi password (bcrypt compare).
d. Jika valid, server membuat JWT token dan mengirimkannya ke client via HTTPOnly cookies.
e. Logout = server menghapus cookies.

2. Database schema dan relationship

Di aplikasi ini, terdapat beberapa tabel yang dibuat berdasarkan requirement yang ditentukan, dan ada pula tabel yang dibuat untuk penambahan. Tabel -tabel tersebut antara lain :



a. **User**: menyimpan informasi akun pengguna.
b. **Space**: grup yang dibuat oleh user, user yang membuat otomatis menjadi seorang admin dan masuk ke dalam space member.
c. **SpaceMember**: relasi banyak-ke-banyak antara User dan Space dengan role & status.
d. **Message**: pesan yang dikirim oleh pengguna dalam space.
e. **Note**: catatan terkait space.
f. **Invite**: token undangan untuk join space.

Selain itu, para tabel juga berelasi satu sama lain. Berikut penjelasan hubungan antara tabel:

a. **User ↔ Space (One-to-Many, createdBy).** Seorang pengguna dapat membuat banyak space/group namun, sebuah space hanya memiliki 1 creator/pembuatnya. Kemudian id pengguna disimpan pada attribute createdBy.
b. **User ↔ SpaceMember ↔ Space (Many-to-Many).** Seorang user dapat bergabung ke banyak space, dan sebuah space dapat memiliki banyak anggota. SpaceMember menyimpan informasi tambahan seperti role (ADMIN atau MEMBER) ketika seorang pengguna membuat group atau space, otomatis akan masuk ke SpaceMember dan rolenya adalah ADMIN, sedangkan lainnya akan menjadi seorang MEMBER.
c. **User ↔ Message ↔ Space (One-to-Many).** Seorang user bisa mengirim banyak pesan ke dalam suatu space, dan sebuah space akan menampung banyak pesan dari berbagai user.
d. **User ↔ Note ↔ Space (One-to-Many).** Setiap user yang memiliki role ADMIN dapat membuat banyak note di dalam suatu space, sementara sebuah space bisa memiliki kumpulan note dari beberapa user.
e. **Space ↔ Invite (One-to-Many).** Satu space bisa memiliki banyak undangan, masing-masing berisi token unik yang memungkinkan user lain bergabung.

3. API design

Pada website ini ada banyak sekali api yang digunakan untuk menghubungkan antara route satu dengan yang lain, sehingga data tampak dinamis. **Base URL : https://heywebsite-six.vercel.app/**

a. Auth

1. Sign up
   a. POST /api/sign-up
   b. Req: { username, name, email, number_phone, password }
   c. Res: 201 Created | { id, username, name, email, number_phone, password }

2. Sign in
   a. POST /api/sign-in
   b. Req: { username, email, password }
   c. Res: 200 OK | { message: "Login Success" } + Set-Cookie(token)

3. Sign out (HARUS MELALUI AUTH)
   a. POST /api/sign-out
   b. Res: 200 OK | { message: "Logged out" }

(HARUS MELALUI AUTH)

b. Profile

1. Lihat profile
   a. GET /api/me/read (auth required)
   b. Res: { id, username, email, name, number_phone, totalSpaces }

2. Update profile
   a. PUT /api/me/update (auth required)
   b. Req: { username, name, email, photo? }

c. Res: { id, username, name, email, number_phone, photo }

c. Space Management

1. **Buat space**
   a. POST /api/space/create
   b. Req { name, description?, permission?, avatar? }
   c. Res { id, name, description, permission, avatar, createdBy, SpaceMember[] }
2. **Lihat semua space milik user yang telah login**
   a. GET /api/space/read
   b. Res [{ id, name, description, creator, SpaceMember[], lastMessage }]
3. **Detail dari space by id**
   a. GET /api/space/:id/detail/read
   b. Res { id, name, description, Note[], SpaceMember[], Invite[], creator, role }
4. **Update space by id**
   a. PATCH /api/space/:id/detail/update
   b. Req { name?, description?, permission?, avatar? }
   c. Res { id, name, description, permission, avatar }
5. **Validasi member**
   a. PATCH /api/space/member/:id/update
   b. Res { Status: “ACCEPTED” }

d. Invite Management

1. **Buat token invite**
   a. POST /api/space/:id/invite/create
   b. Res { inviteUrl }
2. **Baca token invite**
   a. GET /api/invite/:token/read
   b. Req params :token (string)
   c. Res { message, redirect }

e. Message Management

1. **Buat pesan**
   a. POST /api/space/:id/message/create
   b. Req { message, photo? }
   c. Res { id, spaceId, userId, message, photo, createdAt }
2. **Baca pesan**
   a. GET /api/space/:id/message/read
   b. Res { messages: [...], currentUserId }

f. Note Management

1. **Buat note**
   a. POST /api/space/:id/message/create
   b. Req { title, content? }
   c. Res { id, spaceId, userId, title, content, createdAt }
2. **Lihat semua note berdasarkan id space**
   a. GET /api/space/:id/note/read
   b. Res { notes: [...], currentUserId }
3. **Buat note**
   a. GET /api/space/:id/note/:noteId/read
   b. Res { id, title, content, user, space, role }
4. **Lihat semua note berdasarkan id space**
   a. PUT /api/space/:id/note/:noteId/update
   b. Req { title?, content? }
   c. Res { id, title, content, updatedAt }