PERENCANAAN DAN PERANCANGAN WEBSITE BOARDLY

Kelompok 2

Aditia Naik Brahmana	30123043
Ahmad Fadhel Hafizhuddin	30123067
Dimas Faiz Ahmad	30123337
Nike Nova Yani	31123380
Reitza Pia Aryantine	31123391

Identifikasi Permasalahan

Kelompok mahasiswa sering mengalami kendala dalam membagi tugas dan memantau progres kerja ketika mengerjakan proyek bersama. Koordinasi yang hanya mengandalkan chat dan spreadsheet sering kali menimbulkan kebingungan, tugas yang tumpang tindih, hingga keterlambatan penyelesaian proyek. Hal ini juga berdampak pada rendahnya kolaborasi antar anggota.

Tujuan Situs

Tujuan utama dari pembuatan situs ini adalah untuk:

- Menyediakan **platform manajemen tugas berbasis visual** (metode Kanban) yang memudahkan kolaborasi antar anggota tim.
- Membantu pengguna dalam **merencanakan, membagi, dan memantau progres tugas** dengan cara yang efisien dan real-time.
- Meningkatkan **transparansi kerja tim**, memperjelas pembagian peran, dan mengurangi tumpang tindih tugas.
- Menyediakan sistem **pengarsipan tugas** untuk manajemen kerja jangka panjang.

Mengapa Meluncurkannya?

Aplikasi ini diluncurkan karena banyak pengguna—terutama mahasiswa dan tim kecil—membutuhkan **alat bantu kerja tim yang terorganisir, mudah digunakan, dan dapat**

diakses secara online tanpa harus berpindah antar aplikasi. Dengan satu platform terpadu, kolaborasi dapat dilakukan lebih efektif dan efisien.

Jenis Situs

Situs ini tergolong ke dalam:

- **Aplikasi Web** (Web Application).
- Fokus pada **alat bantu produktivitas dan kolaborasi tim.**
- Bukan situs informasi atau promosi, tetapi **aplikasi fungsional.**

Perangkat Lunak & Tools yang Digunakan

Teknologi dan alat yang digunakan untuk membangun situs ini meliputi:

- **Frontend:** React.js.
- **Backend:** Node.js + Express.js.
- **Database:** MongoDB.
- **Authentication:** JSON Web Token (JWT).
- **State Management:** Redux atau Context API.
- **Drag and Drop Library:** react-beautiful-dnd.
- **Deployment:** Vercel / Render (Frontend), Railway / MongoDB Atlas / Heroku (Backend).
- **Version Control:** Git + GitHub.

Pengguna Sistem

Dalam aplikasi Boardly, terdapat tiga jenis pengguna utama dengan hak akses berbeda, yaitu:

1. **Admin:** Mengelola sistem secara global (pengguna, pelaporan, dll).
2. **Pemilik Board (Owner):** Membuat board proyek dan mengelola struktur board.
3. **Anggota Tim (Member):** Berkontribusi pada board yang dimasuki.

1. Data yang Dibutuhkan oleh Sistem

a. Fungsi Umum untuk Semua Pengguna

i. Login dan Logout.
ii. Mengelola profil pribadi (edit nama, avatar, dll).
iii. Melihat daftar board yang diikuti.

iv. Notifikasi dan histori aktivitas.

b. Fungsi Khusus Pemilik Board (Owner)

i. Membuat board baru.
ii. Mengundang anggota ke dalam board.
iii. Menghapus board atau mengarsipkan board.
iv. Membuat list status (To Do, In Progress, Done, dll).
v. Menyusun ulang urutan list dengan drag & drop.
vi. Membuat, mengedit, dan menghapus kartu tugas (card).
vii. Menambahkan checklist, deadline, label, komentar, dan lampiran ke card.
viii. Mengarsipkan dan me-*restore* card atau list.
ix. Melihat log aktivitas dalam board.

c. Fungsi Anggota Tim (Member)

i. Mengakses board yang diundang.
ii. Membuat dan mengedit card (jika diberikan akses).
iii. Menambahkan komentar dan checklist ke dalam card.
iv. Menyelesaikan checklist.
v. Mengedit deskripsi, deadline, dan label card (dengan izin).
vi. Mengarsipkan card yang sudah selesai (jika diberi akses).

d. Fungsi Admin

i. Mengelola seluruh pengguna (CRUD user).
ii. Melihat dan mengatur data board secara global.
iii. Menghapus pengguna yang melanggar.
iv. Melihat statistik penggunaan aplikasi.

e. Fungsi Pendukung

i. Sistem Arsip: mengarsipkan dan meng-*restore* card atau list.
ii. Sistem Drag and Drop: untuk memindahkan list dan card.
iii. Filter dan Pencarian: berdasarkan label, deadline, kata kunci.
iv. Label berwarna untuk memudahkan identifikasi tugas.
v. Checklist/Subtask untuk rincian tugas.

f. Pengembangan

Komponen	Detail
OS	Windows 10/11, macOS, atau Linux
Editor	Visual Studio Code
Database Tool	MongoDB Compass
Desain UI	Figma atau Diagrams.net (untuk wireframe & diagram)

Version Control	Git + GitHub
Server Lokal	Node.js + npm, MongoDB (lokal atau Atlas)
Testing Tool	Postman (untuk API testing)
Browser	Google Chrome / Firefox

Perancangan

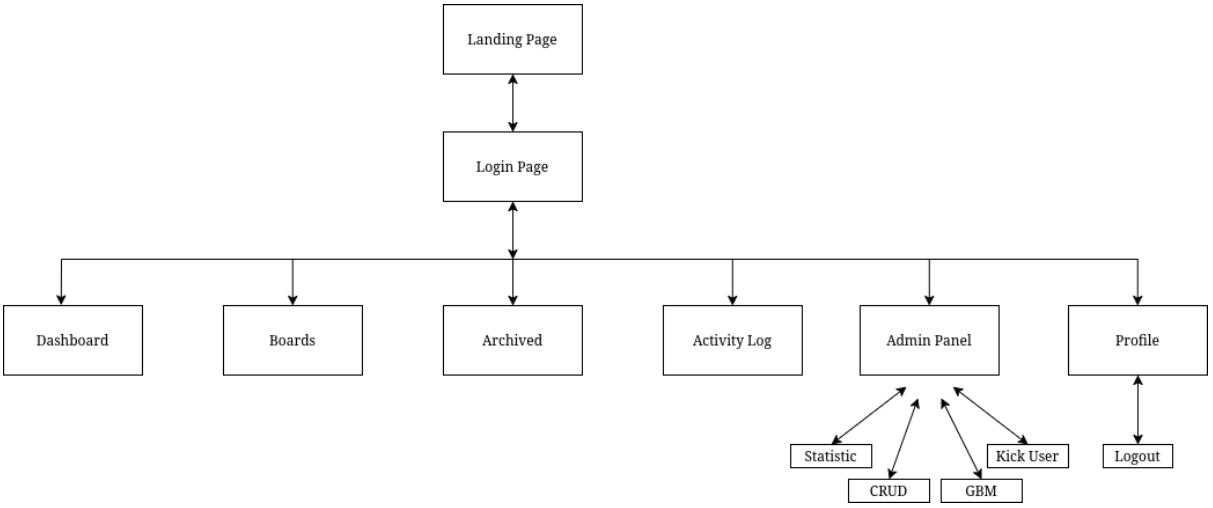
Rencana Layout

[Logo] | Dashboard | Boards | Archived | Activity Log | Profile ▼

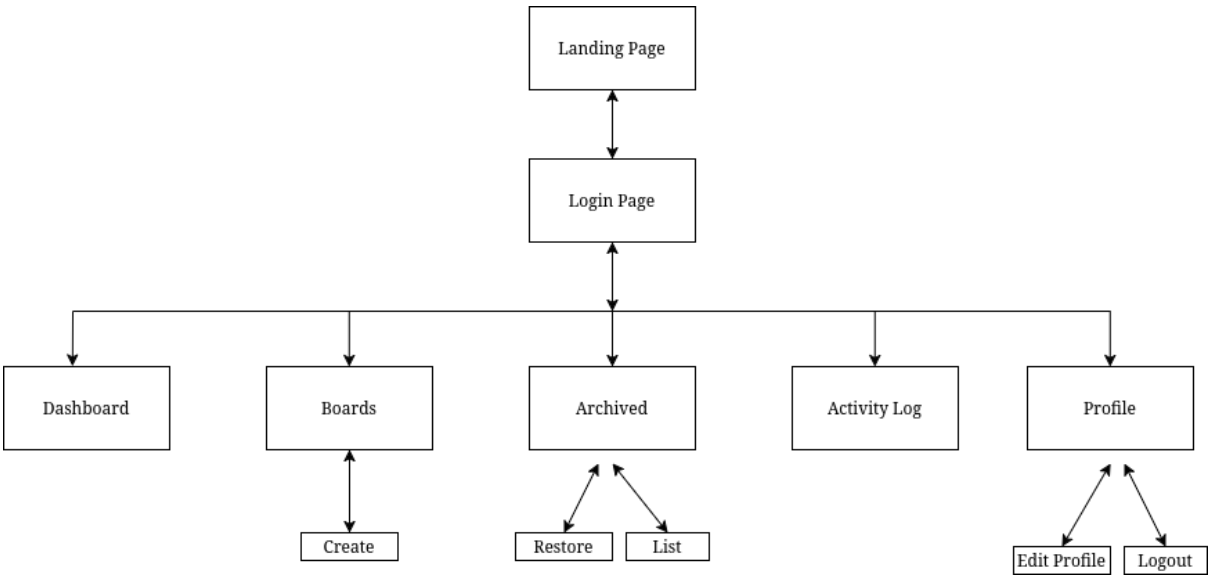
Menu	Akses Untuk	Fungsi
Dashboard	Semua peran	Halaman utama, ringkasan board, aktivitas terkini
Boards	Owner & Member	Lihat daftar board yang dibuat/diikuti, tombol “Create Board”
Archived	Owner & Member	Lihat kartu atau list yang diarsipkan, opsi restore
Activity Log	Owner (dalam board)	Melihat log aktivitas pengguna dalam board
Profile ▼	Semua peran	Akses ke edit profil, logout, preferensi
Admin Panel	Hanya Admin	(Tampil jika role = admin) Akses kelola user & board global

Struktur Navigasi

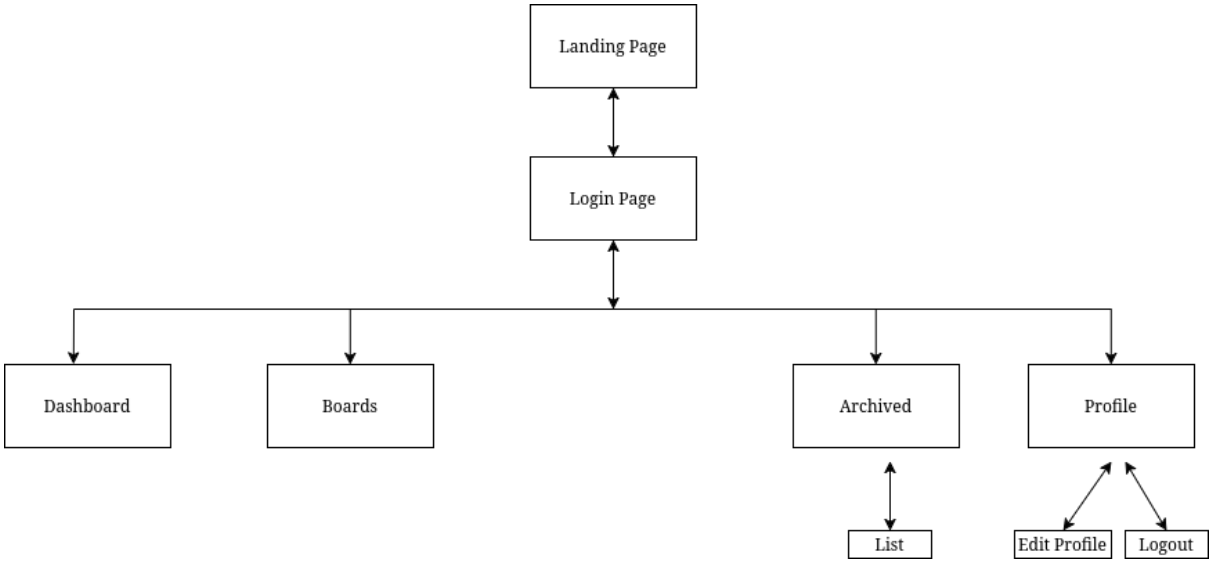
1. Admin



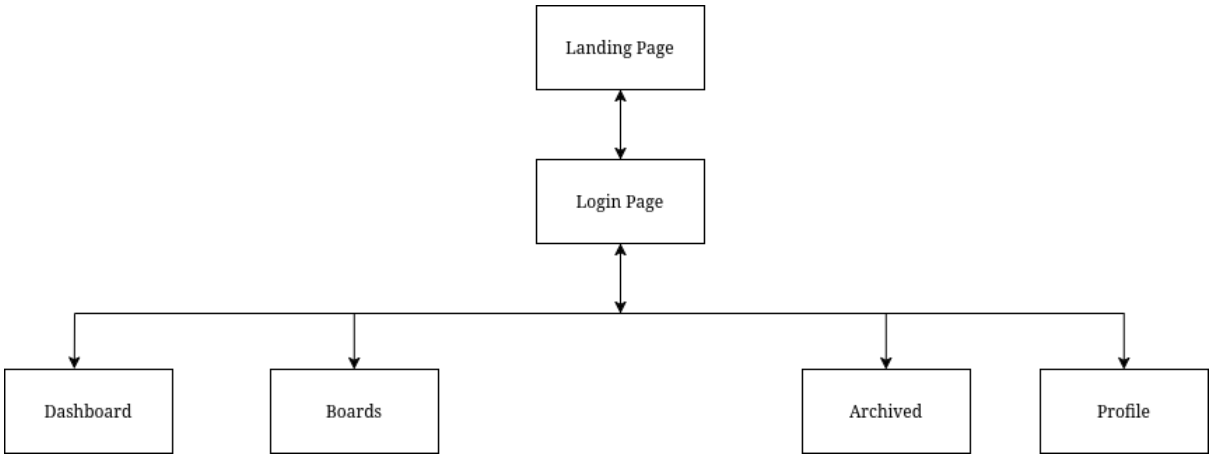
2. Owner



3. Member



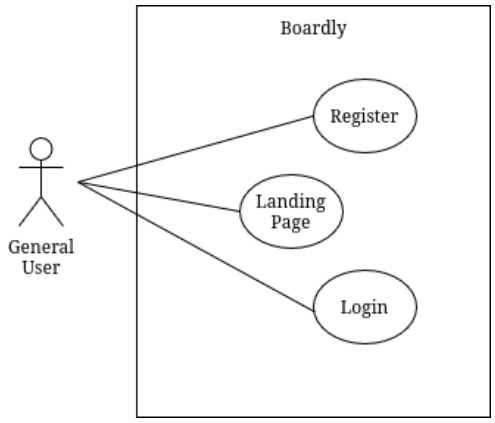
4. Pengguna Umum (Tidak Login)



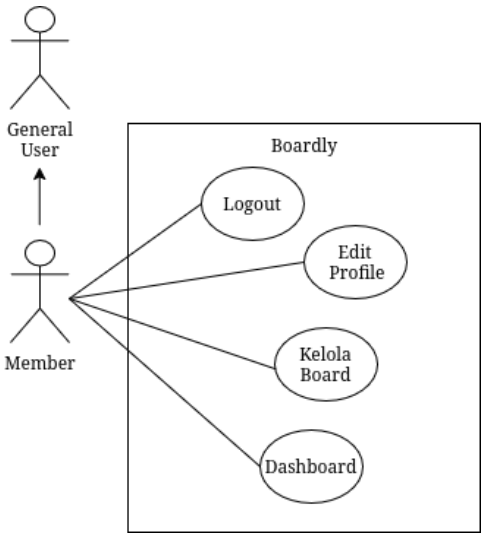
Diagram

1. Use-Case Diagram

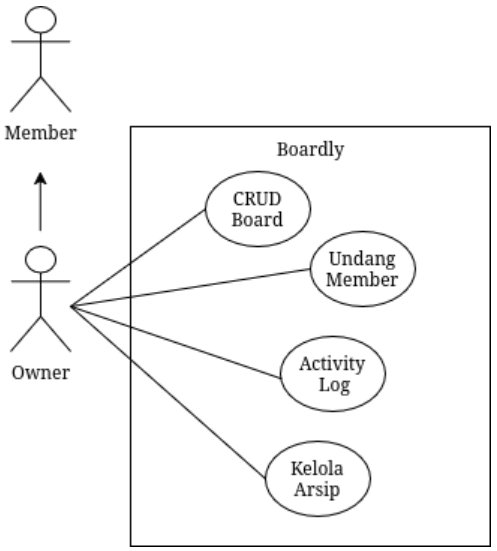
a. Pengguna Umum (Tidak Login)



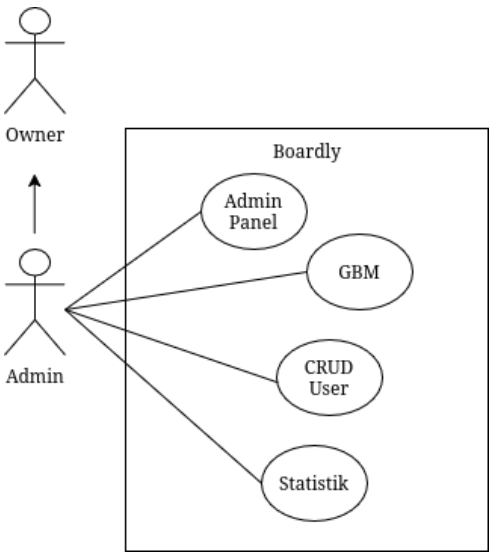
b. Member



c. Owner

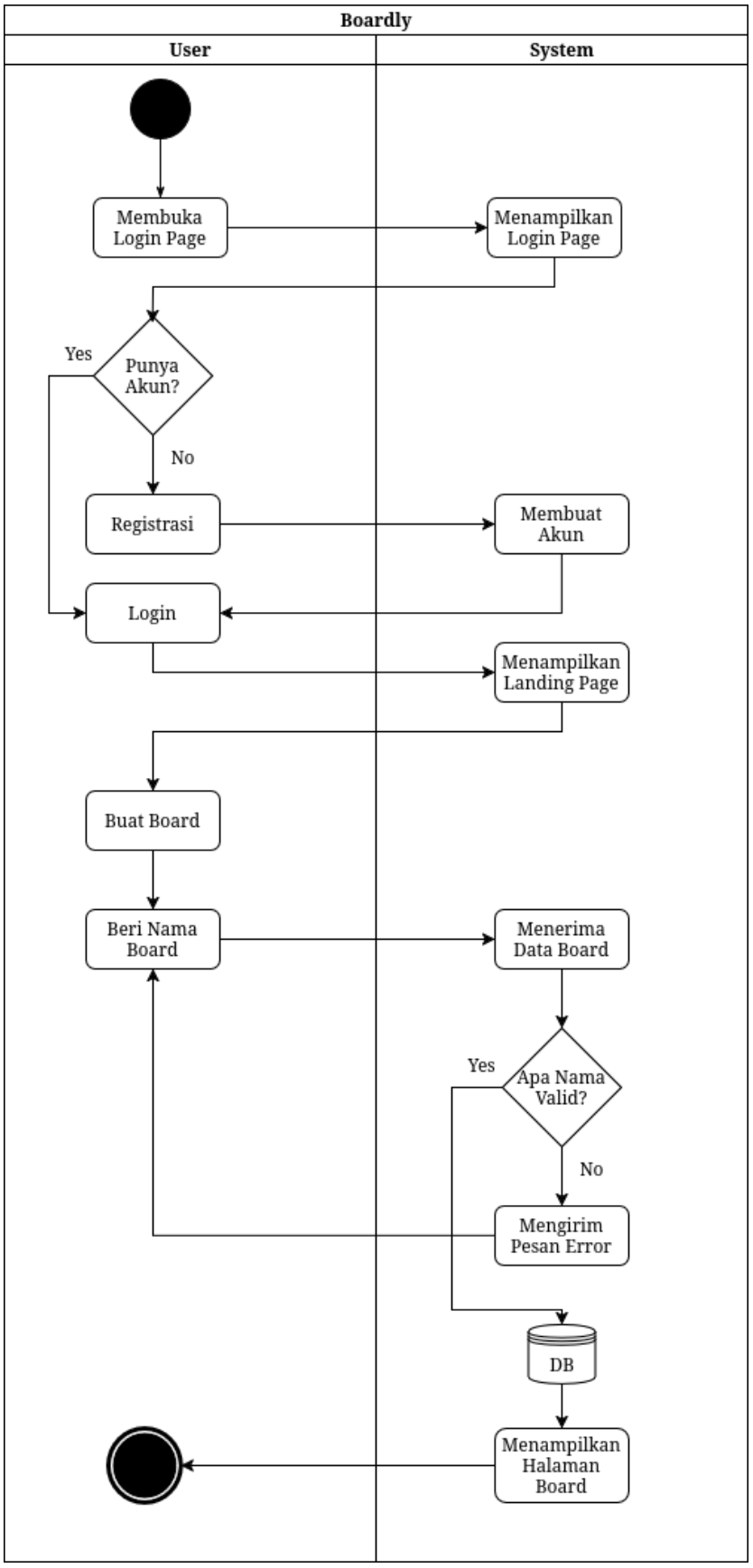


d. Admin

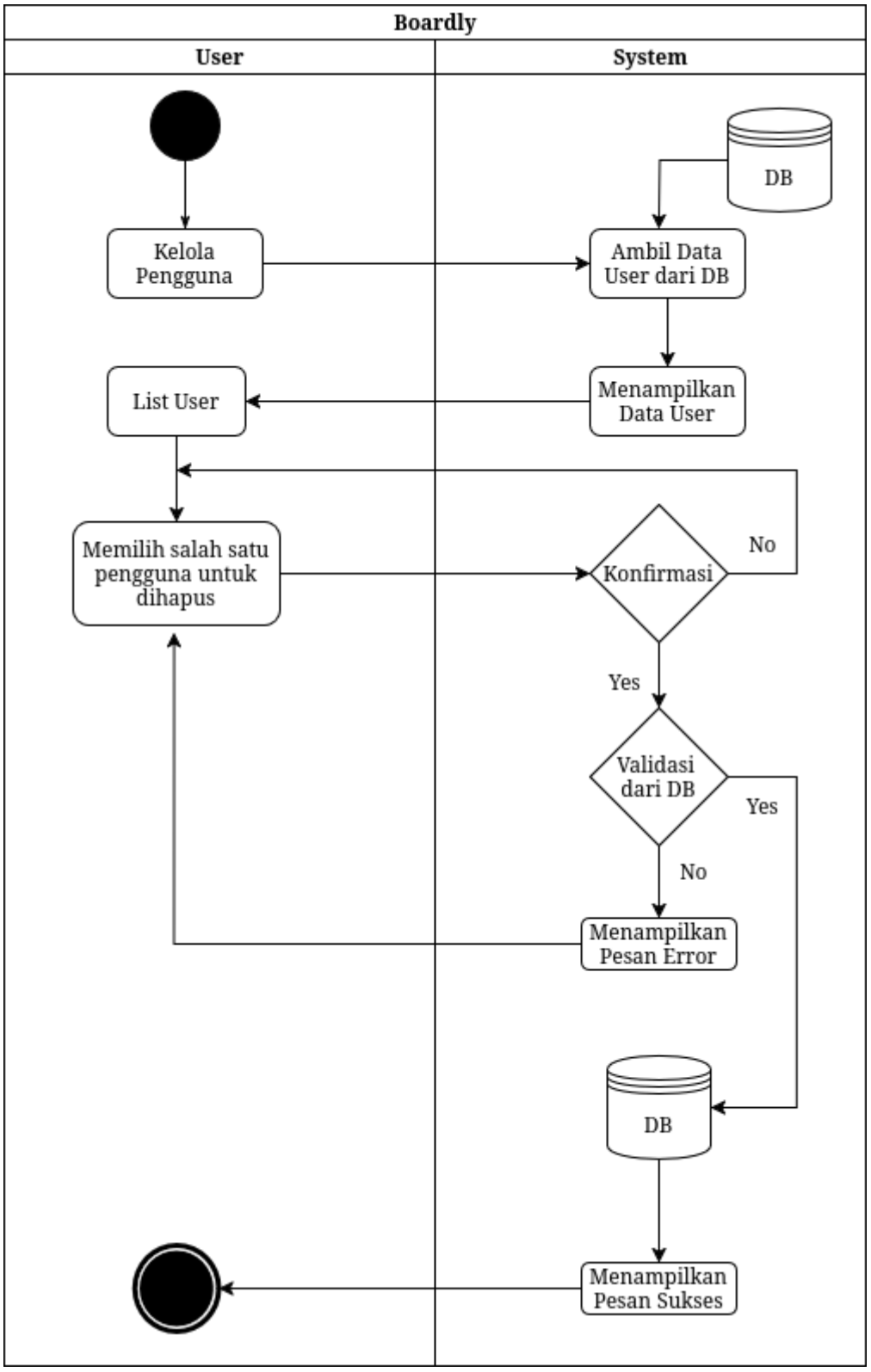


2. Activity Diagram

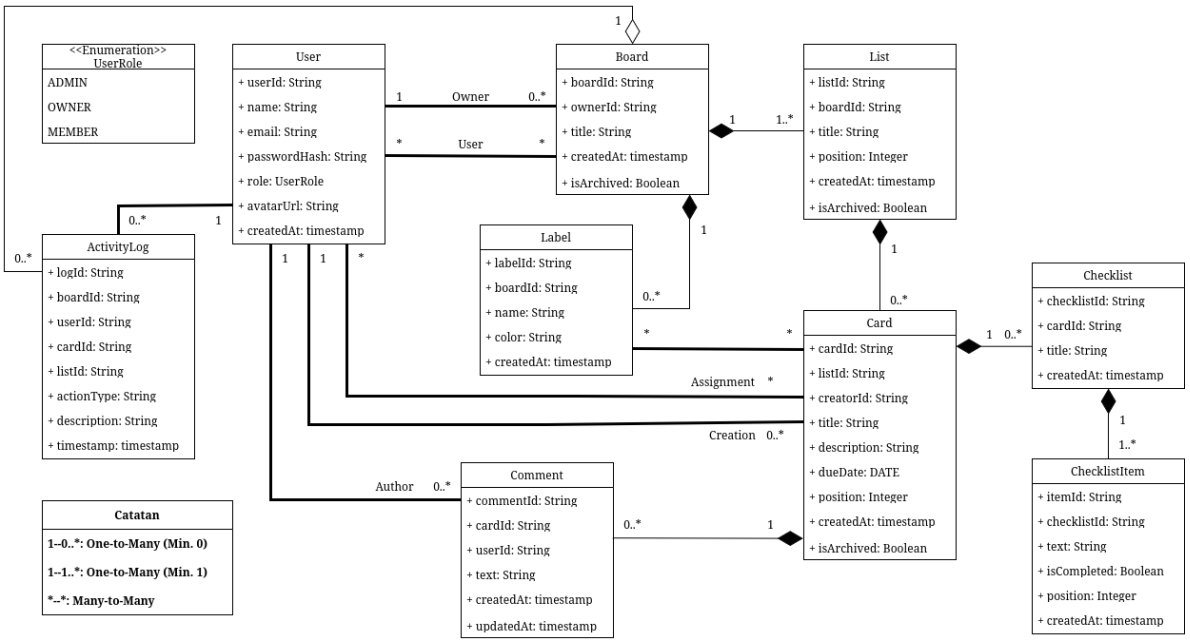
a. Pengguna Login dan Membuat Board



b. Interaksi Admin saat Menghapus Pengguna



3. Class Diagram



Pembuatan Website

Pada pembuatan website, kami akan memecah menjadi 4 bagian, pertama Fondasi dan Persiapan Pengembangan, Pengembangan Frontend, Pengembangan Backend, dan Integrasi dan Aspek Lanjutan.

Bagian I: Fondasi dan Persiapan Pengembangan

1. Pengantar Arsitektur Website
   Arsitektur aplikasi web modern mendasari bagaimana komponen-komponen aplikasi berinteraksi untuk menyajikan fungsionalitas kepada pengguna.
   a. **Konsep Client-server:**
      Ini adalah model arsitektur dasar di mana aplikasi dibagi menjadi dua bagian utama: *Client* (klien) dan *Server*.

      - **Client:** Biasanya adalah browser web di perangkat pengguna (laptop, HP). Bertugas untuk menampilkan antarmuka pengguna (UI) dan mengirimkan permintaan (request) ke server. Untuk proyek Boardly, ini akan ditangani oleh React.js.
      - **Server:** Adalah komputer (atau kumpulan komputer) yang menyimpan data, menjalankan logika bisnis aplikasi, dan merespons permintaan dari client. Untuk Boardly, ini akan ditangani oleh Node.js dan Express.js.

Interaksi: Client meminta data atau layanan, Server memproses permintaan tersebut dan mengirimkan kembali respons (seringkali berupa data JSON atau halaman HTML).

b. Single Page Application (SPA) vs Multi-Page Application (MPA):

Ini adalah dua pendekatan utama dalam membangun antarmuka pengguna web.

- **SPA (Single Page Application):** Seluruh aplikasi dimuat dalam satu halaman HTML awal. Navigasi antar "halaman" atau tampilan dilakukan secara dinamis di sisi client (browser) dengan JavaScript (React akan melakukan ini) tanpa perlu memuat ulang seluruh halaman dari server. Ini memberikan pengalaman pengguna yang lebih cepat dan mulus seperti aplikasi desktop. Boardly akan menjadi SPA.
- **MPA (Multi-Page Application):** Setiap kali pengguna berpindah halaman atau berinteraksi secara signifikan, browser akan meminta halaman HTML baru sepenuhnya dari server. Ini adalah pendekatan tradisional.

Konteks Boardly: Sebagai SPA, Boardly akan terasa lebih responsif karena hanya data yang relevan yang akan di-request dari server setelah pemuatan awal.

c. **Arsitektur Berbasis API (API-Driven Development):**

Dalam arsitektur ini, backend (server) diekspos sebagai sekumpulan API (Application Programming Interface) yang bisa diakses oleh berbagai jenis client (web, mobile, dll.). Frontend (client) berkomunikasi dengan backend hanya melalui API ini, biasanya menggunakan protokol HTTP dengan format data JSON.

Keuntungan: Memisahkan logika frontend dan backend secara jelas (decoupling), memungkinkan pengembangan paralel, dan memudahkan jika di masa depan ingin membuat versi mobile untuk Boardly tanpa mengubah banyak di backend.

Boardly: Backend Node.js/Express.js akan menyediakan RESTful API yang akan dikonsumsi oleh frontend React.js.

d. **Peran Teknologi Pilihan (React, Node.js/Express, MongoDB dalam konteks proyek Boardly):**

- **React.js (Frontend):** Library JavaScript untuk membangun antarmuka pengguna (UI) yang interaktif dan reusable berbasis komponen. Cocok untuk SPA seperti Boardly karena efisien dalam memanipulasi DOM dan mengelola state UI.
- **Node.js (Backend Runtime):** Lingkungan eksekusi JavaScript di sisi server. Memungkinkan penggunaan JavaScript untuk seluruh stack (full-stack JavaScript), yang bisa menyederhanakan pengembangan.
- **Express.js (Backend Framework):** Framework minimalis dan fleksibel yang berjalan di atas Node.js. Memudahkan pembuatan API, routing, dan pengelolaan middleware untuk backend Boardly.
- **MongoDB (Database):** Database NoSQL berbasis dokumen. Fleksibel dalam skema data, cocok untuk pengembangan iteratif dan data yang mungkin memiliki struktur beragam atau berkembang seperti pada kartu-kartu Kanban dengan berbagai atributnya.

2. Persiapan Lingkungan Pengembangan (Development Environment Setup)

Lingkungan pengembangan adalah seperangkat alat dan konfigurasi di komputer Anda yang memungkinkan Anda untuk membangun dan menjalankan aplikasi.

- **a. Instalasi Tools Esensial:**
  Node.js dan NPM/Yarn:
  - **Node.js:** Diperlukan untuk menjalankan JavaScript di luar browser (untuk backend) dan juga sebagai basis untuk banyak tools pengembangan frontend.
  - **NPM (Node Package Manager) / Yarn:** Manajer paket yang datang bersama Node.js (NPM) atau alternatif populer (Yarn). Digunakan untuk menginstal dan mengelola library atau paket pihak ketiga yang dibutuhkan proyek (misal: Express, React, Mongoose).
  - **Git & Konfigurasi Dasar:**
    - **Git:** Sistem kontrol versi terdistribusi untuk melacak perubahan pada kode, berkolaborasi dengan tim, dan kembali ke versi sebelumnya jika ada masalah. Wajib diinstal.
    - **Konfigurasi Dasar:** Mengatur nama pengguna dan email Anda di Git.
  - **Text Editor/IDE (misal: Visual Studio Code) beserta Ekstensi Pendukung:**
    - **VS Code:** Editor kode populer yang ringan, gratis, dan memiliki banyak fitur serta ekstensi yang mendukung pengembangan JavaScript, React, Node.js, dll.

        - **Ekstensi Pendukung:** Contoh: Prettier (untuk formatting kode), ESLint (untuk_ linting_ kode), Debugger for Chrome/Node, dll.
- **b. Pengenalan dan Penggunaan Dasar Version Control dengan Git & GitHub:**
    - **Penjelasan Ringkas:**
        - **Version Control (Git):** Sistem yang merekam perubahan pada file dari waktu ke waktu sehingga Anda dapat melihat histori, membandingkan versi, atau mengembalikan ke versi tertentu. Penting untuk pengembangan individu maupun tim.
        - **GitHub (atau GitLab/Bitbucket):** Platform hosting repositori Git berbasis web. Memfasilitasi kolaborasi, backup kode, dan manajemen proyek.
    - **Workflow Dasar (Teori Singkat):**
        - **clone:** Mengunduh repositori dari GitHub ke komputer lokal.
        - **add:** Menambahkan perubahan file ke area "staging" (siap untuk di-commit).
        - **commit:** Menyimpan "snapshot" perubahan yang ada di area staging ke histori repositori lokal dengan pesan deskriptif.
        - **push:** Mengirim commit dari repositori lokal ke repositori remote (misal: GitHub).
        - **pull:** Mengambil perubahan dari repositori remote dan menggabungkannya ke repositori lokal.
        - **branch:** Membuat "cabang" terpisah dari kode utama untuk mengerjakan fitur baru atau perbaikan tanpa mengganggu kode utama.
        - **merge:** Menggabungkan perubahan dari satu branch ke branch lain.

Bagian II: Pengembangan Frontend

Berikut adalah poin-poin teori penting yang perlu dipahami sebelum atau saat memulai pengembangan frontend dengan React.js untuk proyek Boardly:

1. **Dasar-Dasar React.js**
    - **Pengenalan React (Library UI):** React adalah library JavaScript yang deklaratif, efisien, dan fleksibel untuk membangun antarmuka pengguna (UI) yang kompleks. Fokus utamanya adalah pada pembuatan komponen UI yang reusable.
    - **JSX (JavaScript XML):** Ekstensi sintaks untuk JavaScript yang memungkinkan penulisan struktur mirip HTML langsung di dalam kode JavaScript. Ini mempermudah visualisasi dan pembuatan elemen UI di React.
    - **Komponen (Components):** Blok bangunan dasar dalam React. Setiap bagian UI (misalnya, tombol, kartu tugas, list, board, halaman) bisa dianggap sebagai

komponen. Komponen bisa menerima data (disebut props) dan mengelola data internalnya sendiri (disebut state). Kita akan fokus pada *Functional Components* dengan *Hooks*.

- **Props (Properties):** Cara komponen menerima data dari komponen induknya (parent). Props bersifat *read-only* (tidak bisa diubah oleh komponen penerima).
- **State:** Data internal yang dikelola oleh sebuah komponen. Ketika state berubah, React akan secara otomatis me-render ulang komponen tersebut untuk menampilkan perubahan. Hook useState digunakan untuk mengelola state dalam functional components.
- **Lifecycle Komponen (Hooks Ekuivalen):** Dalam functional components, *lifecycle methods* tradisional digantikan oleh Hooks. Hook useEffect adalah yang paling umum digunakan untuk menangani efek samping (side effects) seperti pemanggilan API, manipulasi DOM langsung, atau berlangganan event, yang sebelumnya ditangani oleh componentDidMount, componentDidUpdate, dan componentWillUnmount.

2. **Manajemen State (State Management)**
   - **Pentingnya Manajemen State:** Seiring aplikasi tumbuh kompleks, mengelola state yang tersebar di banyak komponen menjadi sulit. Manajemen state terpusat membantu menyederhanakan aliran data dan membuat state lebih prediktabel.
   - **Pilihan Manajemen State (Sesuai PDF Anda: Context API vs. Redux):**
     - **Context API:** Fitur bawaan React untuk membagikan state secara global antar komponen tanpa harus melewati props secara manual melalui setiap level hierarki komponen (prop drilling). Cocok untuk state yang tidak terlalu sering berubah atau untuk aplikasi skala menengah.
     - **Redux:** Library manajemen state pihak ketiga yang lebih powerful dan terstruktur dengan konsep seperti *store*, *actions*, *reducers*, dan *middleware*. Cocok untuk aplikasi skala besar dengan state yang kompleks dan sering berubah, serta membutuhkan fitur debugging yang canggih.
   - **Implementasi Dasar:** Memahami bagaimana mendefinisikan state, cara mengubahnya, dan cara komponen "mendengarkan" perubahan state tersebut.
3. **Routing di Sisi Klien (Client-Side Routing)**
   - **Konsep:** Dalam SPA (Single Page Application), perpindahan antar "halaman" ditangani di sisi klien (browser) tanpa meminta halaman baru dari server. Ini menciptakan ilusi navigasi multi-halaman yang cepat.

    - **Menggunakan React Router DOM:** Library paling populer untuk routing di aplikasi React. Memungkinkan definisi rute (URL path) yang akan me-render komponen tertentu.
    - **Navigasi antar Halaman/View:** Cara membuat link navigasi dan cara melakukan navigasi secara programatik (misalnya, setelah login berhasil).
4. **Interaksi dengan API Backend**
    - **HTTP Client:** Frontend perlu cara untuk mengirim HTTP request (GET, POST, PUT, DELETE) ke API backend dan menerima respons.
    - **Menggunakan `Workspace` API atau `axios`:**
        - `Workspace` API: Bawaan browser modern untuk melakukan request HTTP.
        - `axios`: Library pihak ketiga yang populer, menawarkan fitur lebih seperti pembatalan request, interceptor, dan penanganan error yang lebih mudah.
    - **Menampilkan Data, Mengirim Data (Form Handling):** Bagaimana mengambil data dari API dan menampilkannya di komponen, serta bagaimana mengumpulkan data dari form pengguna dan mengirimkannya ke API.
5. **Implementasi Fitur Spesifik (Teori Dasar)**
    - **Drag and Drop (`react-beautiful-dnd`):** Memahami konsep dasar bagaimana library drag and drop bekerja di React, seperti `Draggable`, `Droppable`, dan `DragDropContext`. Bagaimana state dikelola saat item dipindahkan.
    - **Manajemen Form dan Validasi:** Strategi untuk menangani input form, menyimpan nilainya dalam state, dan melakukan validasi baik di sisi klien (untuk UX yang lebih baik) maupun mengandalkan validasi sisi server.
6. **Styling dan UI/UX**
    - **Pilihan Styling:** Berbagai cara untuk menerapkan style pada komponen React:
        - CSS Biasa: Menggunakan file `.css` global atau per komponen.
        - CSS Modules: File CSS yang di-scope secara lokal ke komponen tertentu untuk menghindari konflik nama class.
        - Styled Components / Emotion: CSS-in-JS libraries yang memungkinkan penulisan CSS langsung di dalam file JavaScript komponen.
        - Framework UI (jika digunakan, misal: Material UI, Ant Design, Tailwind CSS): Menyediakan komponen UI siap pakai dan sistem styling.
    - **Prinsip Desain Responsif:** Teori dasar bagaimana membuat UI yang beradaptasi dengan baik di berbagai ukuran layar (desktop, tablet, mobile) menggunakan teknik seperti media queries, flexbox, atau grid.

Struktur Folder Proyek Frontend (React)

Struktur folder yang baik sangat penting untuk kemudahan pengelolaan, skalabilitas, dan kolaborasi dalam tim. Berikut adalah salah contoh struktur folder yang direkomendasikan

untuk proyek React seperti Boardly, yang mencoba menggabungkan pengelompokan berdasarkan jenis dan fitur:

```
src/
|
|-- App.jsx                    # Komponen utama aplikasi, routing utama
|-- index.jsx                  # Titik masuk aplikasi React
|
|-- assets/                    # File statis
|   |-- images/
|   |-- styles/                # CSS global, variabel tema, dll.
|       |-- global.css
|
|-- components/                # Komponen UI yang reusable
|   |-- common/                # Komponen sangat umum (Button, Modal, InputField)
|   |   |-- Button.jsx
|   |   |-- Modal.jsx
|   |-- layout/                # Komponen untuk tata letak (Navbar, Sidebar, Footer)
|   |   |-- Navbar.jsx
|   |-- board/                 # Komponen spesifik untuk fitur board (BoardCard, TaskCard, ListColumn)
|   |   |-- BoardCard.jsx
|   |   |-- TaskCard.jsx
|   |-- auth/                  # Komponen spesifik untuk fitur autentikasi (LoginForm)
|       |-- LoginForm.jsx
|
|-- pages/                     # Komponen yang merepresentasikan satu halaman/view penuh
|   |-- LoginPage.jsx
|   |-- RegisterPage.jsx
|   |-- DashboardPage.jsx
|   |-- BoardDetailPage.jsx    # Halaman detail satu board spesifik
|   |-- UserProfilePage.jsx
|   |-- AdminPage.jsx          # Halaman untuk admin mengelola user/board
|   |-- NotFoundPage.jsx
|
|-- services/                  # Fungsi untuk melakukan panggilan API ke backend
|   |-- authService.js         # Panggilan API terkait login, register
|   |-- boardService.js        # Panggilan API terkait data board, list, card
|   |-- userService.js         # Panggilan API terkait data pengguna (profil, admin)
|   |-- apiClient.js           # (Opsional) Konfigurasi instance axios/fetch global
|
|-- contexts/                  # Untuk React Context API (jika digunakan)
|   |-- AuthContext.js         # Mengelola state autentikasi pengguna
|   |-- ThemeContext.js        # (Opsional) Jika ada fitur ganti tema
|
|-- hooks/                     # Custom Hooks (jika ada logika yang ingin dipakai ulang)
|   |-- useAuth.js             # (Opsional) Hook untuk logika autentikasi
|
|-- utils/                     # Fungsi utilitas helper dan konstanta
    |-- helpers.js             # Fungsi bantu umum (misal: format tanggal)
    |-- constants.js           # Konstanta aplikasi (misal: URL API base)
```