Design document: Pass The Ball

By: Simeon Markov

Institution Name: Fontys UAS

Course/Class: ICT/EN08

Date: 2025-10-09

Introduction

The purpose of this document is to set a clear design plan, illustrating incorporated layout principles. It includes the wireframing process, explaining each wireframe and the user flow as well as the design of the database (shown in EER diagram) explaining fields and relations between tables.

Design Guide (UI/UX Principles)

This section presents the initial design plan (incorporating layout principles, accessibility, etc.)

User-Centric & Action-Oriented: The primary call-to-action (CTA) should always be "Find an Activity" or "Create an Event/Post." The design should guide users towards these goals.

Visual Hierarchy for Scannability: Using clear typography, contrasting colors, and ample white space to make scanning easy.

Personalization from the Start: The home page should display relevant, personalized feeds, joined groups and followings.

Consistent & Intuitive Navigation: Standard upper sidebar navigation (on mobile).

Design for Trust & Safety: Using real names and photos for profiles where possible. Including a clear rating/review system for users and events.

Accessibility-First: Sufficient color contrast, alt-text for all images.

Wireframing

The wireframing was done on Figma, because the software has proven to be robust, agile and one of the most used softwares in the industry.

Login system

The login system is the entry-point of the social network platform. The user is navigated to the login/registration page for authentication and afterwards he/she could access the platform's content.

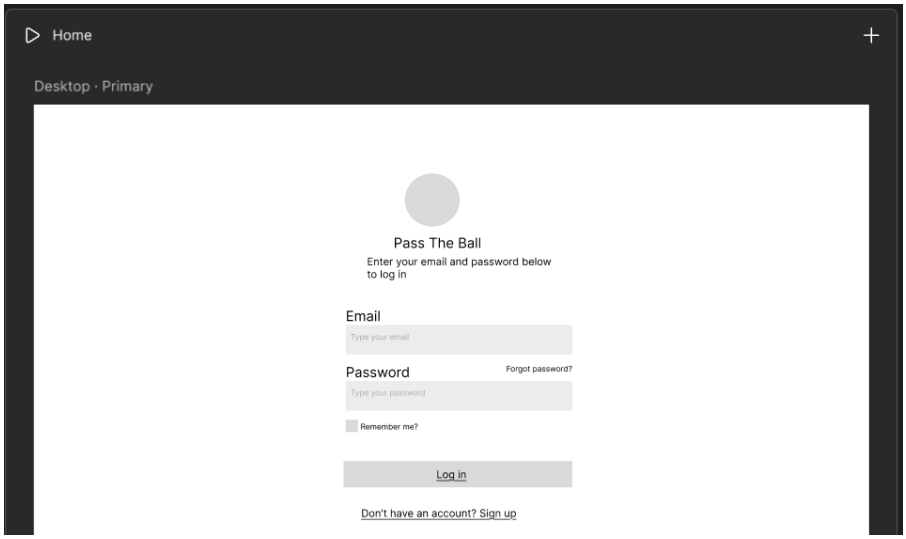

Figure 1: Login page

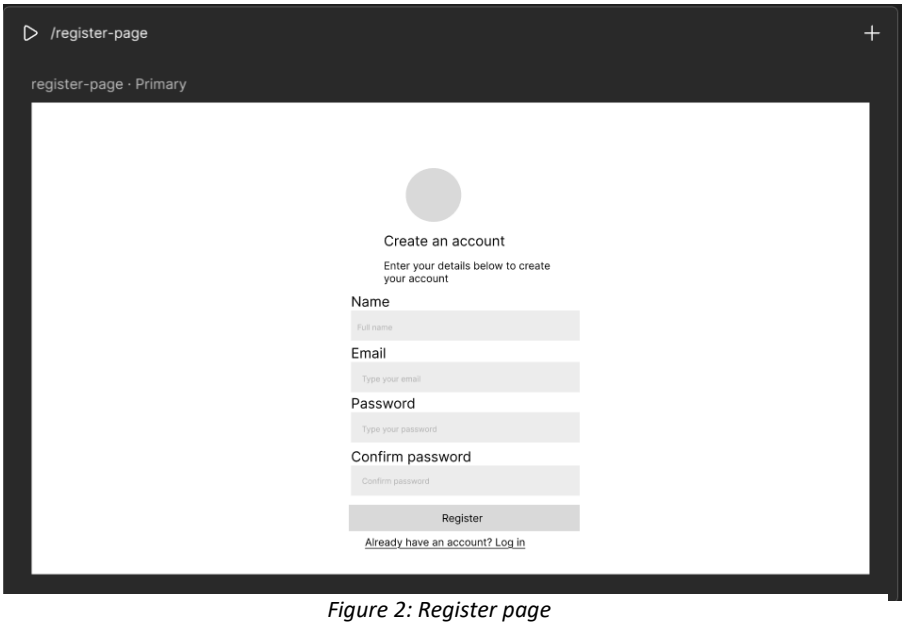

Figure 2: Register page

Index page

This page contains the main content (posts, groups, friends). It is the first page the users are going to see when logged in.

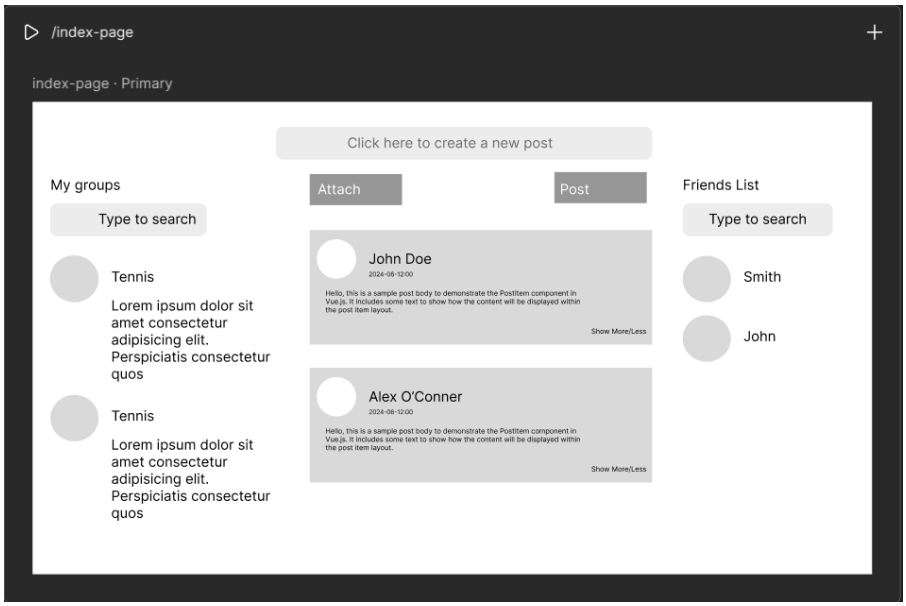

Figure 3: Index page

<u>Breakdown:</u> The page consists of several containers: main one (responsible for displaying the posts/tweets content) and two sidebars showing user's groups and friends. The authorized user could post new tweets, upload images, videos, documents (post/attach buttons).

Settings (Profile edit)

This page contains the settings menu (emails, passwords, 2fa…).

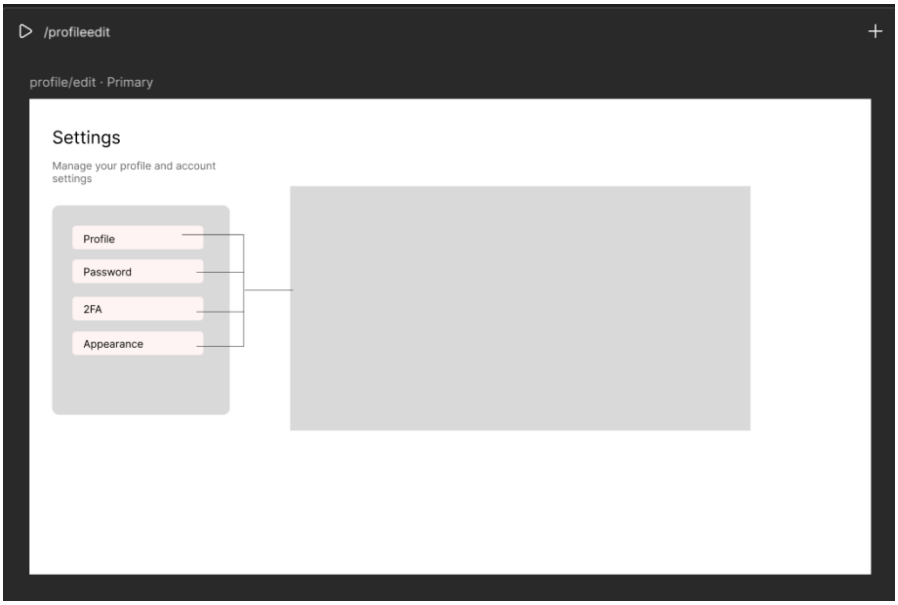

Figure 4: Setting page

Breakdown: The page consists of reusable components for the different sections of the setting menu which will appear on the main container each according to its section.

Profile component

Contains personal information.

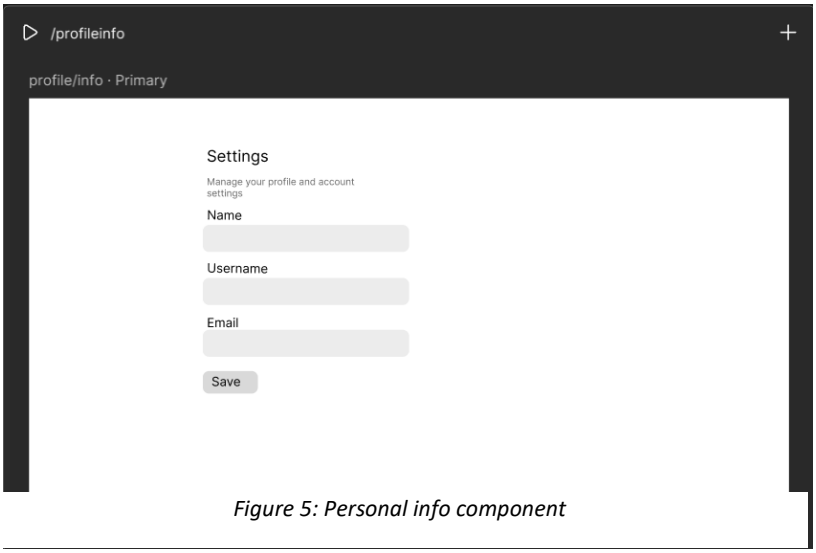

Figure 5: Personal info component

Breakdown: Users can change their personal information by updating the related fields.

Password edit component

Contains the password of the user.

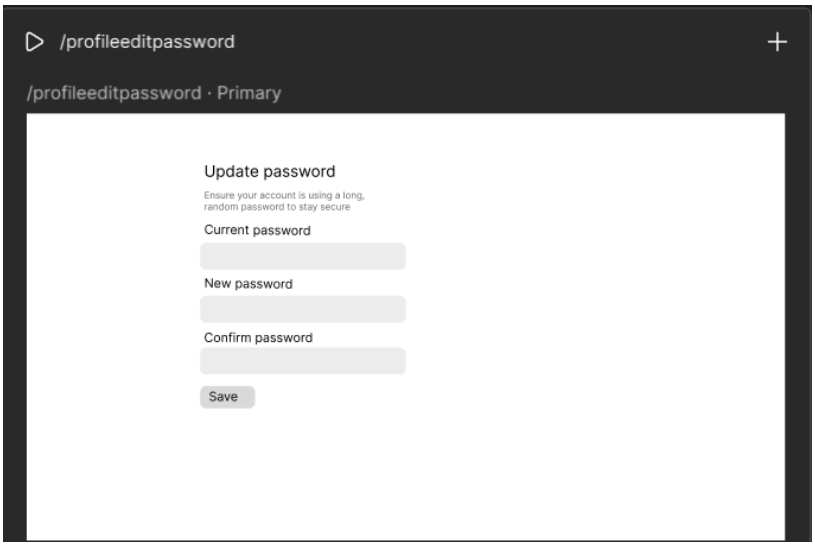

Figure 6: Password component

Breakdown: Users can change their password after providing their previous one and confirm their new password by updating the related fields.

2FA Component

Lets users set up two-factor authentication.

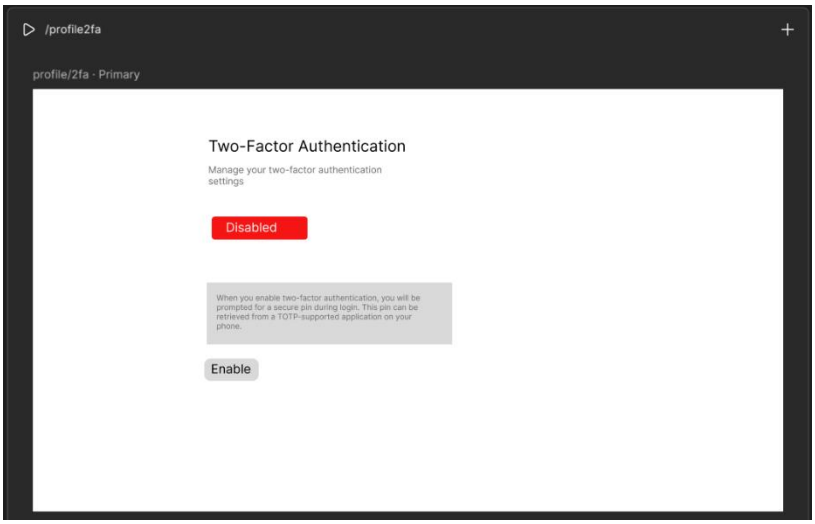

Figure 7: 2FA component

Breakdown: Users are prompted to enable/disable their extra layer of protection. The two-factor authentication state (whether enabled or disabled) is displayed to the user.

Appearance component

Lets users set up their preferred appearance of the platform.

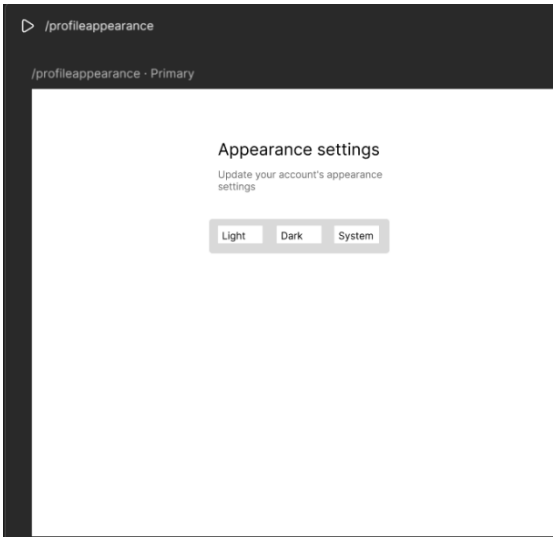

Figure 8: Appearance settings

<u>Breakdown:</u> In a dialog with options for Light, Dark, System mode, the user has the choice to align the platform's theme to his preferred one.

Database design

Workbench (RDBMS) was used for generating the EER diagram. It was chosen because of its compatibility with MySQL (MariaDB) with the project's framework and its easy navigation and intuitive GUI.

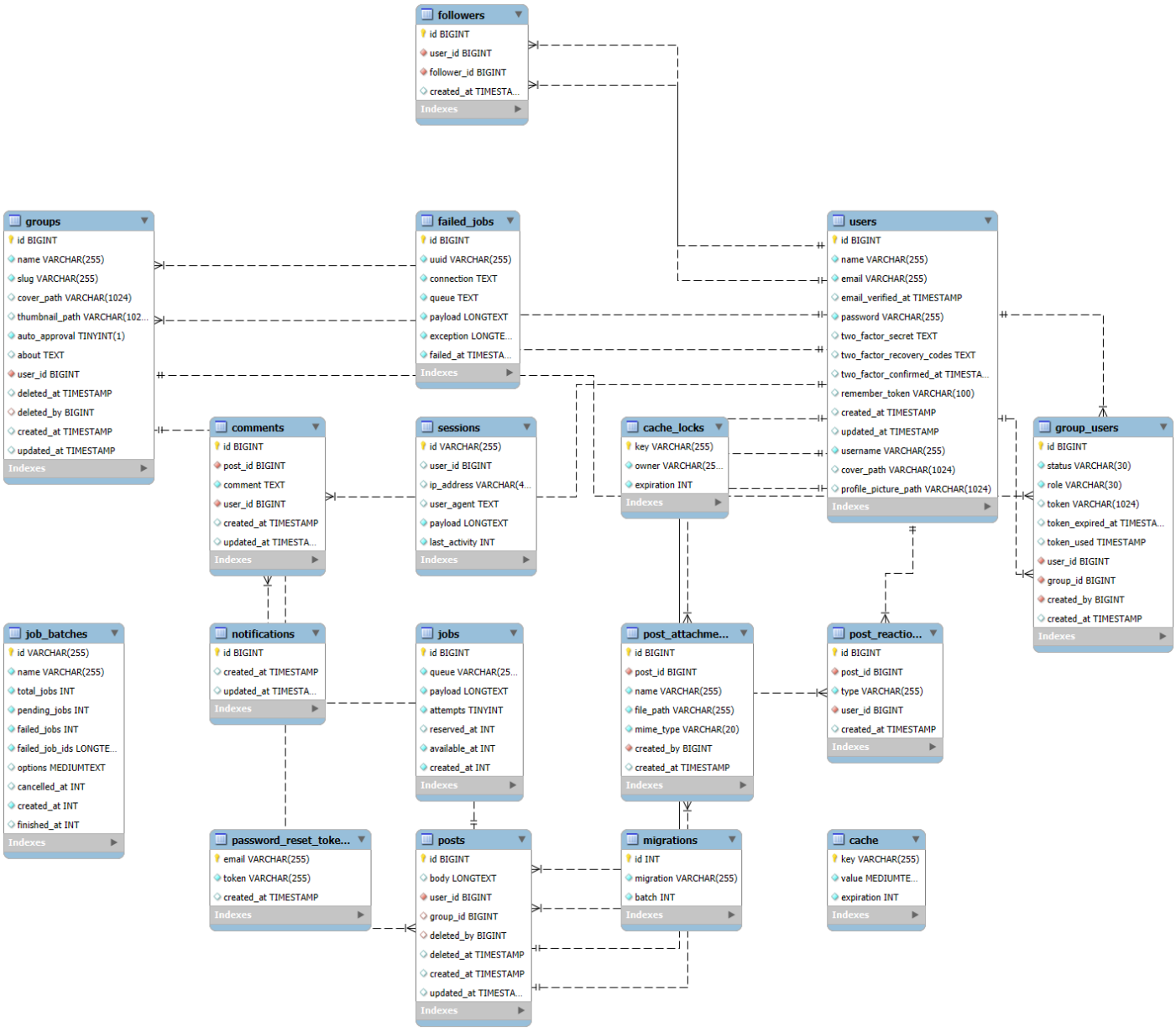

Figure 9: Database diagram

Overview

Core Entities:

- ***users:*** *This central table stores user information, including their name, email, password, and profile details. It serves as the primary entity that interacts with most other tables.*
- ***posts****: This table contains user-generated posts, including the post content (body), the author (user_id), and potentially the group it belongs to (group_id).*
- ***groups:*** *This table defines user groups with details like name, description (about), and cover images.*

- ***comments:*** *This table stores comments made on posts, linking a comment to a specific post (post_id) and the user who wrote it (user_id).*

User Interaction and Relationships:

- ***followers:*** *Manages the follower relationships between users, with user_id representing the user being followed and follower_id representing the user who is following.*
- ***group_users:*** *This is a pivot table that connects users to groups, defining a user's membership and their role within a group.*
- ***post_reactions:*** *This table stores user reactions (e.g., likes) to posts, linking a user to a post they reacted to.*
- ***post_attachments:*** *This table stores files or media attached to posts.*

System and Utility Tables:

- ***jobs, failed_jobs, and job_batches:*** *These tables represent background job processing system used for managing asynchronous operations.*
- ***notifications:*** *This table stores notifications.*
- ***sessions:*** *This table manages user login sessions.*
- ***password_reset_tokens:*** *This table securely stores tokens for password reset requests.*
- ***cache and cache_locks:*** *These tables are used for caching data to improve application performance.*
- ***migrations:*** *This table tracks database schema changes over time (history changes)**.***

Key Relationships:

- *A user can create multiple posts, comments, and groups (many-to-one).*
- *A user can belong to multiple groups, and a group can have multiple users (many-to-many).*
- *A post can have multiple comments and reactions (one-to-many).*
- *Users can follow other users (many-to-one/one-to-many).*

Tables description

'users':

Column	*Type*	*Constraint*	*Description*
Id	*BIGINT*	*PRIMARY KEY*	*Unique identifier for each user.*
name	*VARCHAR*	*NOT NULL*	*Stored user's name.*
email	*VARCHAR*	*UNIQUE, NOT NULL*	*Stores user's email.*

password	*VARCHAR*	*UNIQUE, NOT NULL*	*Stores user's password.*
two_factor_secret	*TEXT*	*UNIQUE*	*Stores 2fa secret key.*
two_factor_recovery_codes	*TEXT*	*UNIQUE*	*Stores user's recovery codes for 2fa authentication.*
two_factor_confirmed_at	*TIMESTAMP*		*Stores user's 2fa date of creation/activation.*
remember_token	*VARCHAR*	*NOT NULL*	*Used for storing session data about user's login credentials.*
created_at	*TIMESTAMP*	*NOT NULL*	*Stores data about when the user signed up.*
updated_at	*TIMESTAMP*	*NOT NULL*	*Stores data about when the user made updates on his/her profile last.*
username	*VARCAHAR*	*UNIQUE, NOT NULL*	*Auto generated firstly, then user have chance to change it; used as slug.*
cover_path	*VARCHAR*	*NULLABLE*	*Stores the URL for user's profile cover image.*
profile_picture_path	*VARCHAR*	*NULLABLE*	*Stores the URL for user's profile image.*

Table 1: Users table

'followers":

Column	Type	Constraint	Description

Id	BIGINT	PRIMARY KEY	Unique identifier for each user.
user_id	BIGINT	FOREIGN KEY	Foreign key for user's following.
follower_id	BIGINT	FOREIGN KEY	Foreign key for user's followers.
created_at	TIMESTAMP	NOT NULL	Stores data about when the user became a follower/have following.

Table 2: Followers table

'posts':

Column	Type	Constraint	Description
Id	BIGINT	PRIMARY KEY	Unique identifier for each post.
body	LONGTEXT	NULLABLE	Stores the post content description.
user_id	BIGINT	FOREIGN KEY	Foreign key for user table (post->user).
group_id	BIGINT	FOREIGN KEY	Foreign key pointing to particular user.
deleted_by	BIGINT	FOREIHN KEY	Foreign key defining the user who deleted the post.
created_at	TIMESTAMP	NOT NULL	Stores data about when a post is created.
updated_at	TIMESTAMP	NOT NULL	Stores data about when a post is updated.

Table 3: Posts table

'comments':

Column	Type	Constraint	Description
Id	BIGINT	PRIMARY KEY	Unique identifier for each comment.
post_id	BIGINT	FOREIGN KEY	Foreign key for each comment made on a post(comment->post).
user_id	BIGINT	FOREIGN KEY	Foreign key for the user who made the comment.
created_at	TIMESTAMP	NOT NULL	Stores data about when a comment was made.
updated_at	TIMESTAMP	NOT NULL	Stores data about when a comment is updated.

Table 4: Commenst table

‘groups’:

Column	Type	Constraint	Description
Id	BIGINT	PRIMARY KEY	Unique identifier for each group created.
name	VARCHAR	NOT NULL	Stores the name of a group.
slug	VARCHAR	NOT NULL	Used as slug for the URL.
thumbnail_path	VARCHAR	NULLABLE	Stores the URL path of the group’s thumbnail.
auto_approval	TINYINT	DEFAULT=TRUE	Used as a flag for auto approval.
deleted_at	TIMESTAMP	NULLABLE	Stores the time when deletion is performed.
about	TEXT	NULLABLE	Stores the information about a group.
cover_path	VARCAHAR	NULLABLE	Stores the URL for group’s cover.
user_id	BIGINT	FOREIGN KEY	Foreign key for a user joined in a group.
created_at	TIMESTAMP	NOT NULL	Stores data about when a group is created.
updated_at	TIMESTAMP	NOT NULL	Stores data about when a group is updated.

Table 5: Groups table

‘notifications’:

Column	Type	Constraint	Description
Id	BIGINT	PRIMARY KEY	Unique identifier for each notification.
created_at	TIMESTAMP	NOT NULL	Stores data about when a notification is created.
updated_at	TIMESTAMP	NOT NULL	Stores data about when a notification is updated.

Table 6: Notifications table.

‘migrations’:

Column	Type	Constraint	Description
Id	BIGINT	PRIMARY KEY	Unique identifier for each created migration.
migration	VARCHAR	NOT NULL, UNIQUE	Migration’s file name identifier.

batch	INT	NOT NULL, UNIQUE	The batch number for the migration, grouping related migrations.

Table 7: Migrations table

'cache':

Column	Type	Description	Constraints
key	VARCHAR(255)	The unique key for the cached item.	Primary Key
value	MEDIUMTEXT	The serialized value of the cached item.	Not Null
expiration	INT	Unix timestamp indicating when the cached item will expire.	Not Null

Table 8: Cache table

'failed_jobs':

Column	Type	Description	Constraints
id	BIGINT	Primary key, unique identifier for the failed job entry.	Primary Key
uuid	VARCHAR(255)	A universally unique identifier for the job.	Not Null, Unique
connection	TEXT	The name of the queue connection used for the job.	Not Null
queue	TEXT	The name of the queue the job was dispatched to.	Not Null
payload	LONGTEXT	The serialized payload of the job, containing all necessary data for execution.	Not Null
exception	LONGTEXT	The full exception trace that occurred when the job failed.	Not Null
failed_at	TIMESTAMP	Timestamp indicating when the job failed.	Not Null

Table 9: Failed jobs table

‘cache_locks’:

Column	Type	Description	Constraints
key	VARCHAR(255)	The unique key identifying the cache lock.	Primary Key
owner	VARCHAR(25.)	Identifier of the process or entity that currently holds the lock.	Not Null
expiration	INT	Unix timestamp indicating when the lock will expire.	Not Null

Table 10: Cache locks table

‘sessions’:

Column	Type	Description	Constraints
id	VARCHAR(255)	Primary key, unique identifier for the session.	Primary Key
user_id	BIGINT	Foreign key, references the ID of the logged-in user (can be null for guests).	Foreign Key (users.id), Nullable
ip_address	VARCHAR(4...)	The IP address from which the session originated.	Nullable
user_agent	TEXT	The user agent string of the client browser.	Nullable
payload	LONGTEXT	Serialized session data.	Not Null
last_activity	INT	Unix timestamp of the last activity within the session.	Not Null

Table 11: Sessions table

‘group_users’:

Column	Type	Description	Constraints
id	BIGINT	Primary key, unique identifier for the group-user relationship.	Primary Key
status	VARCHAR(30)	The user's status within the group (e.g., 'pending', 'active', 'banned').	Not Null
role	VARCHAR(30)	The user's role within the group (e.g., 'member', 'admin', 'moderator').	Not Null

token	VARCHAR(1024)	A token associated with the user's group membership (e.g., for invitations).	Nullable
token_expired_at	TIMESTAMP	Timestamp indicating when the token expires.	Nullable
token_used	TIMESTAMP	Timestamp indicating when the token was used.	Nullable
user_id	BIGINT	Foreign key, references the ID of the user.	Foreign Key (users.id)
group_id	BIGINT	Foreign key, references the ID of the group.	Foreign Key (groups.id)
created_by	BIGINT	Foreign key, references the ID of the user who added this user to the group.	Foreign Key (users.id)
created_at	TIMESTAMP	Timestamp indicating when the user was added to the group.	Not Null

Table 12: Group users (pivot) table

'job_batches':

Column	Type	Description	Constraints
id	VARCHAR(255)	Primary key, unique identifier for the job batch.	Primary Key
name	VARCHAR(255)	A descriptive name for the job batch.	Not Null
total_jobs	INT	The total number of jobs in the batch.	Not Null
pending_jobs	INT	The number of jobs in the batch that are still pending.	Not Null
failed_jobs	INT	The number of jobs in the batch that have failed.	Not Null
failed_job_ids	LONGTEXT	A serialized list of IDs of the failed jobs within the batch.	Not Null
options	MEDIUMTEXT	Configuration options for the job batch.	Nullable
cancelled_at	INT	Unix timestamp indicating when the job batch was cancelled.	Nullable

created_at	INT	Unix timestamp indicating when the job batch was created.	Not Null
finished_at	INT	Unix timestamp indicating when the job batch was completed.	Nullable

Table 13: Jobs baches table

'jobs':

Column	Type	Description	Constraints
id	BIGINT	Primary key, unique identifier for the job.	Primary Key
queue	VARCHAR(25.)	The name of the queue the job belongs to.	Not Null
payload	LONGTEXT	The serialized payload of the job, containing all necessary data for execution.	Not Null
attempts	TINYINT	The number of times the job has been attempted.	Not Null
reserved_at	INT	Unix timestamp indicating when the job was reserved for processing.	Nullable
available_at	INT	Unix timestamp indicating when the job becomes available for processing.	Not Null
created_at	INT	Unix timestamp indicating when the job was created.	Not Null

Table 14: Jobs table

'post_attachments':

Column	Type	Description	Constraints
id	BIGINT	Primary key, unique identifier for the attachment.	Primary Key
post_id	BIGINT	Foreign key, references the ID of the post the attachment belongs to.	Foreign Key (posts.id)
name	VARCHAR(255)	The original name of the attached file.	Not Null
file_path	VARCHAR(255)	The path where the attached file is stored.	Not Null

mime_type	VARCHAR(20)	The MIME type of the attached file (e.g., 'image/jpeg', 'video/mp4').	Not Null
created_by	BIGINT	Foreign key, references the ID of the user who uploaded the attachment.	Foreign Key (users.id)
created_at	TIMESTAMP	Timestamp indicating when the attachment was created.	Not Null

Table 15: Post attachments table

'post_reactions':

Column	Type	Description	Constraints
id	BIGINT	Primary key, unique identifier for the reaction.	Primary Key
post_id	BIGINT	Foreign key, references the ID of the post being reacted to.	Foreign Key (posts.id)
type	VARCHAR(255)	The type of reaction (e.g., 'like', 'love').	Not Null
user_id	BIGINT	Foreign key, references the ID of the user who made the reaction.	Foreign Key (users.id)
created_at	TIMESTAMP	Timestamp indicating when the reaction was made.	Not Null

Table 16: Post reactions table

'password_reset_tokens':

Column	Type	Description	Constraints
email	VARCHAR(255)	The email address associated with the password reset request.	Not Null, Unique
token	VARCHAR(255)	The unique token generated for the password reset.	Not Null, Unique
created_at	TIMESTAMP	Timestamp indicating when the password reset token was created.	Not Null

Table 17: Password reset tokens table

References

Nikitins, Nikiti. (Published on 09/04/2024). Code First Approach vs. Database First Approach: Which Is Best? https://builtin.com/articles/code-first-vs-database-first-approach.

Tim, Kadlec. (2012). Implementing Responsive Design: Building Sites for an Anywhere, Everywhere Web (Voices That Matter) 1st ed. New Riders Pub.

AI Transperancy: Copilot/Manus AI for information summary.