**Realisation document: Pass The Ball**

**By**: Simeon Markov

**Institution Name**: Fontys UAS

**Course/Class**: ICT/EN08

**Date:** 2025-10-09

# Introduction

This document highlights the project development phase, detailing the technical structure, implementation strategy and development workflow. It includes explanation of the core logic, explaining major parts of the architecture like controllers, views, components, writing mysql queries and dealing with migrations (database changes, modifications).

# Project structure

This section presents the system architecture and main components.

**System architecture:**

**Architecture:** Inertia.js hybrid SPA (no separate API, no client-side routing)

* **Backend:** Laravel handles routing, controllers, models, validation
* **Frontend:** Vue 3 SFCs with Composition API + TypeScript
* **Bridge:** Inertia.js manages page transitions without full page reloads

**Directory structure:**

pass\_the\_ball/

│

├── app/ # Laravel Backend (PHP)

│ ├── Console/ # Artisan CLI commands

│ ├── Enums/ # Type-safe constants (notification types, roles)

│ ├── Events/ # Broadcasting events for real-time updates

│ ├── Http/

│ │ ├── Controllers/ # Route handlers returning Inertia responses

│ │ ├── Middleware/ # Request filters (auth, Inertia shared data)

│ │ ├── Requests/ # FormRequest validation classes

│ │ └── Resources/ # API transformers (Eloquent → JSON)

│ ├── Models/ # Eloquent ORM models (User, Post, Comment, Group, etc.)

│ ├── Notifications/ # Email/database/broadcast notification classes

│ ├── Policies/ # Authorization logic (user permissions)

│ ├── Providers/ # Service provider bootstrapping

│ └── Services/ # Business logic (image optimization, AI enhancement)

│

├── bootstrap/ # Application initialization

│ ├── app.php # Creates Laravel instance

│ ├── providers.php # Registers service providers

│ └── cache/ # Cached config/routes (auto-generated)

│

├── config/ # Configuration files (.env values)

│ ├── app.php # Core settings (timezone, locale, debug)

│ ├── auth.php # Authentication guards

│ ├── broadcasting.php # Pusher/Echo WebSocket config

│ ├── database.php # Database connections (SQLite default)

│ ├── filesystems.php # Storage disks (local, S3, public)

│ ├── fortify.php # Authentication features (2FA, password reset)

│ ├── inertia.php # Inertia.js server config

│ ├── openai.php # OpenAI API for AI features

│ ├── purifier.php # HTML sanitization (XSS protection)

│ ├── queue.php # Queue drivers for background jobs

│ └── services.php # Third-party service credentials

│

├── database/

│ ├── migrations/ # Version-controlled database schema

│ ├── factories/ # Faker factories for test data

│ └── seeders/ # Database seeders for initial/demo data

│

├── docs/ # Technical documentation (markdown guides)

│ ├── FLASH\_MESSAGES\_FLOW.md

│ ├── GROUPS\_FEATURE\_GUIDE.md

│ ├── NOTIFICATIONS\_SYSTEM.md

│ ├── PHOTO\_GALLERY\_FEATURE.md

│ ├── POLYMORPHIC\_REACTIONS.md

│ └── ... (20+ feature/architecture docs)

│

├── public/ # Web root (publicly accessible)

│ ├── index.php # Entry point for all HTTP requests

│ ├── build/ # Compiled Vite assets (JS/CSS bundles)

│ ├── images/ # Static image assets

│ └── storage/ # Symlink to /storage/app/public (user uploads)

│

├── resources/

│ ├── css/ # Global CSS/Tailwind entry points

│ ├── js/ # Vue 3 Frontend (TypeScript)

│ │ ├── actions/ # Reusable Inertia form actions

│ │ ├── components/

│ │ │ ├── ui/ # Reka UI primitives (Button, Dialog, Input)

│ │ │ ├── app/ # Feature components (CreatePost, PostList)

│ │ │ └── groups/ # Group-specific components

│ │ ├── composables/ # Reusable composition functions (useFlashMessage)

│ │ ├── layouts/ # Page layouts (AppLayout, AuthLayout)

│ │ ├── lib/ # Utility libraries (cn() for class merging)

│ │ ├── pages/ # Inertia page components (Dashboard, Profile)

│ │ ├── routes/ # Auto-generated TypeScript route helpers (Wayfinder)

│ │ ├── types/ # TypeScript type definitions (User, Post, etc.)

│ │ ├── app.ts # Inertia app initialization

│ │ ├── bootstrap.ts # Axios/Laravel Echo setup

│ │ ├── echo.ts # WebSocket configuration

│ │ └── ssr.ts # Server-side rendering entry

│ └── views/

│ └── app.blade.php # Root HTML template (mounts Vue app)

│

├── routes/ # Laravel route definitions

│ ├── web.php # Main app routes (posts, groups, profiles)

│ ├── auth.php # Authentication routes (Fortify)

│ ├── settings.php # User settings routes

│ └── console.php # Artisan command routes

│

├── storage/ # Private file storage

│ ├── app/

│ │ └── public/ # User uploads (symlinked to /public/storage)

│ ├── framework/ # Cache, sessions, compiled views

│ └── logs/ # Application logs (daily rotation)

│

├── tests/ # Pest PHP test suite

│ ├── Feature/ # End-to-end tests (HTTP, database)

│ ├── Unit/ # Isolated unit tests (models, services)

│ ├── Pest.php # Pest configuration

│ └── TestCase.php # Base test class

│

├── vendor/ # Composer dependencies (gitignored)

│

├── composer.json # PHP dependencies & autoloading

├── package.json # Node.js dependencies & scripts

├── vite.config.ts # Vite build configuration

├── tsconfig.json # TypeScript compiler options

├── eslint.config.js # ESLint code quality rules

├── components.json # Reka UI component library config

├── phpunit.xml # Pest PHP test configuration

└── .env # Environment variables (not in repo)

# Key technologies

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Backend** | Laravel 12 | MVC framework, routing, ORM, authentication |
| **Frontend** | Vue 3 + TypeScript | Reactive UI with Composition API |
| **Bridge** | Inertia.js 2 | SPA experience without client-side routing |
| **Styling** | Tailwind CSS 4 | Utility-first CSS framework |
| **UI Components** | Reka UI, HeadlessUI | Radix Vue wrapper components |
| **Database** | MariaDB | Relational database |
| **Real-time** | Laravel Echo + Pusher | WebSocket broadcasting |
| **Build Tool** | Vite 7 | Fast frontend asset bundling |
| **Testing** | Pest PHP | Modern PHP testing framework |
| **Image Processing** | Intervention Image | Image optimization/resizing |
| **Rich Text** | CKEditor 5 | WYSIWYG editor with HTML sanitization |
| **AI** | OpenAI SDK | AI-powered post enhancements |

Table : Key technologies used

**Techstack decision**

# Frontend Architecture

* Leveraging Vue.js reusable components alongside Laravel packages helps in boosting the frontend, ensuring clean architecture.
* Laravel’s reactive data binding system allows for real-time updates, useful for user feeds, notification counters, and social interactions without requiring full page reloads.
* Vue.js has proven particularly effective for social media applications, supporting features such as dynamic feeds, user profiles, and real-time chat functionality.
* Inertia.js serves as a bridge between Laravel's server-side capabilities and Vue.js frontend framework, eliminating the need for a separate API while maintaining SPA-like functionality. The approach benefits from simplified data flow, enhanced performance.

# Backend Architecture

* PHP offers several advantages for social network development, including excellent scalability, fast loading times, and strong database connectivity. The language's ability to handle high-traffic websites makes it particularly suitable for social platforms that may experience rapid user growth.
* Laravel's design patterns, including Factory, Observer, and Strategy patterns, provide a solid architectural foundation for complex social network features.
* The combination of JavaScript and TypeScript in the techstack provides flexibility for both client side interactivity and server-side operations.

# Database Architecture

• MariaDB was chosen due to its superior performance compared to MySQL, particularly in scenarios requiring high concurrency and complex queries.

|  |  |  |
| --- | --- | --- |
| *Feature* | *MySQL* | *MariaDB* |
| Real-time analytics | Limited | Limited |
| Scalability | Moderate (manual tuning needed) | Moderate |
| Data type support | Primarily structured | Structured + semistructured |
| Vector search | No | No |
| Performance | Good for small datasets | Good for small to medium datasets |

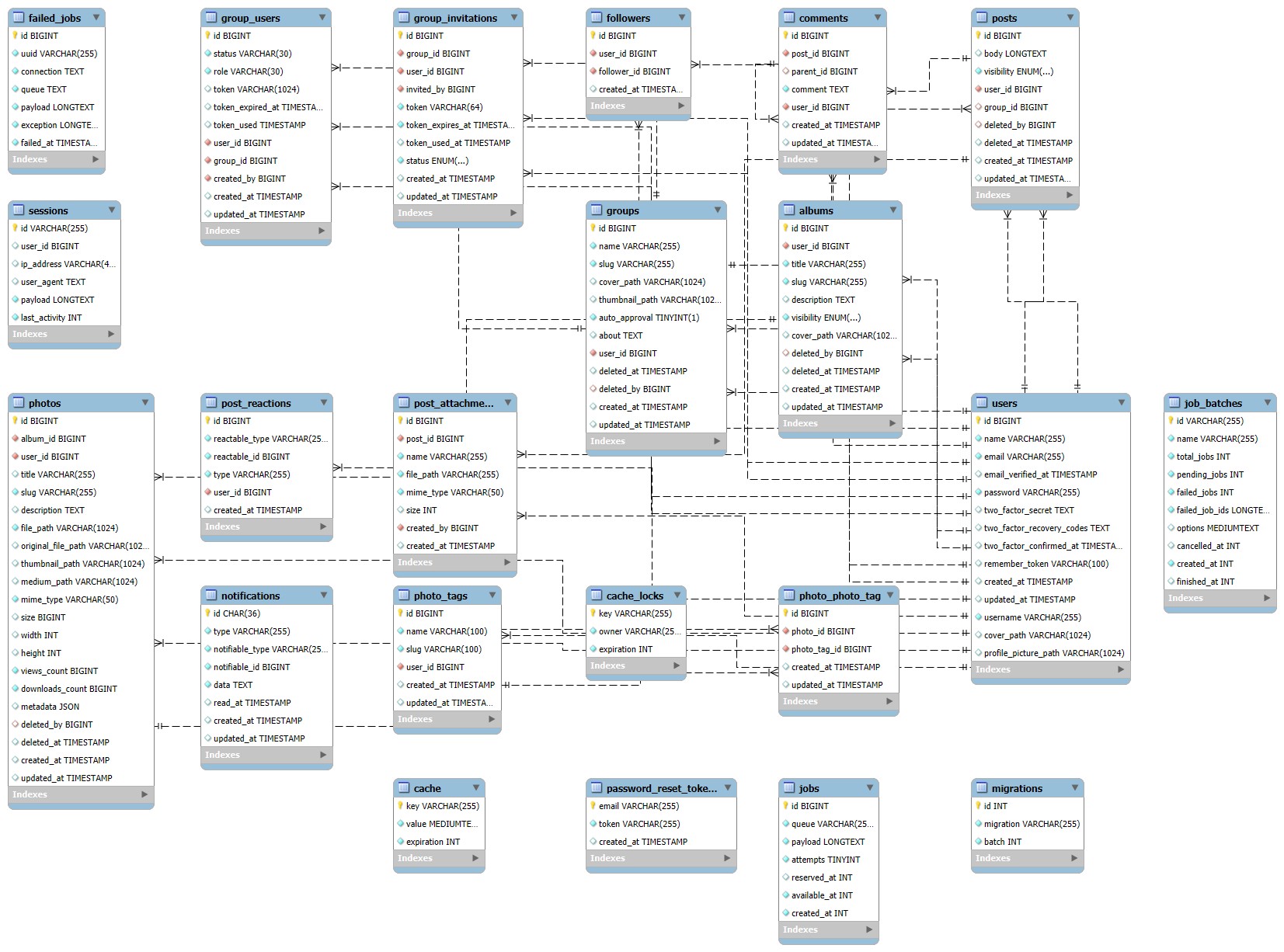
*Table 2: Comparison between MariaDB & MySQL*

## Elasticsearch for Advanced Search

* *Real-time Search:* Instantaneous search results across user profiles, posts, and content.
* *Social Search Features:* Advanced filtering and ranking based on user connections and social graphs.
* *Scalable Architecture:* Distributed processing for handling large-scale social media data.
* *Complex Query Support:* Boolean queries, aggregations, and relevance scoring for sophisticated social search features.

# 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 groupuser 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.](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.
* BM Coder. (2025, August 22). *Building a social networking platform with*
* *Laravel*[. **https://www.bmcoder.com/blog/building-a-social-networking-platform-with-laravel**](https://www.bmcoder.com/blog/building-a-social-networking-platform-with-laravel)
* WitQuick. (n.d.). *Pros and cons of the VueJS*[. **https://witquick.hashnode.dev/pros-and-consofthe-vuejs**](https://witquick.hashnode.dev/pros-and-cons-of-the-vuejs)
* Pattem Digital. (n.d.). *Vue JS Application: Benefits & Industry Use*
* *Cases*[. **https://pattemdigital.com/insight/vue-js-applications-and-case-study/**](https://pattemdigital.com/insight/vue-js-applications-and-case-study/)
* SingleStore. (2024, September 18). MariaDB vs. MySQL.
* [**https://www.singlestore.com/blog/mariadb-vs-mysql/**](https://www.singlestore.com/blog/mariadb-vs-mysql/)
* Apiumhub. (n.d.). Elastic Search; advantages, case studies & books. [**https://apiumhub.com/tech-blog-barcelona/elastic-search-advantages-books/**](https://apiumhub.com/tech-blog-barcelona/elastic-search-advantages-books/)
* AI transperancy: PerplexityAI for extensive in-depth research, summarizing articles, papers.
* AI Transperancy: Copilot/Manus AI for information summary.