CAPSTONE PROJECT-1

Knowledge Exchange Platform

Submitted by ,

MOHAMED IQHLAS A

JAVA FULL STACK – (FACEPREP-FA02)

43612030

B.E CSE – AI AND ROBOTICS

INTRODUCTION

Knowledge Exchange is a comprehensive full-stack web application built with Spring Boot and modern enterprise technologies. This placement project demonstrates advanced development practices including secure authentication, role-based access control (RBAC), file upload/download management, threaded discussions, and a professional Bootstrap-based user interface. The backend showcases clean architecture principles, RESTful API design, and scalable system design suitable for collaborative learning platforms.

The system is designed with modular components to ensure maintainability and ease of future enhancements. Robust database modeling with normalized schemas and efficient data access layers support reliable persistence and performance. Overall, Knowledge Exchange reflects industry-standard practices and serves as a strong demonstration of practical full-stack engineering skills.

ABSTRACT

Knowledge Exchange is a knowledge-sharing platform developed using Spring Boot(Version.4), Java 17, Thymeleaf, and MySQL 8.0. The system implements secure authentication with role-based access control (Admin and User), enabling users to post doubts, upload files, and receive threaded replies. Key features include persistent doubt history, file sharing with secure storage, profile management, and an admin dashboard for monitoring user activity.

The backend follows a layered architecture (Controller-Service-Repository pattern) ensuring scalability and maintainability. Additional features include admin analytics, user list management, and secure file handling. The frontend leverages Bootstrap for a professional, responsive interface, ensuring a polished user experience.

TARGET USERS

1.**End Users**: Students and professionals posting doubts, sharing files, and engaging in threaded discussions.

2.**Administrators**: System admins managing users, monitoring doubts, and ensuring platform integrity.

3.**Mentors/Moderators**: Reviewing posted content and guiding discussions.

4.**Recruiters/Evaluators**: Assessing the technical implementation and architecture as a portfolio project.

TECHNOLOGY STACK

Category	Technology	Version	Purpose
Backend	Java	17	Core language
	Spring Boot	3.2.3	Framework
	Spring Security	Latest	Authentication
Database	MySQL	8.0+	Relational DB
	Hibernate/JPA	Latest	ORM
Security	BCrypt	Built-in	Password hashing
Frontend	Thymeleaf	Latest	Template engine
	Bootstrap	5.3+	Responsive UI
Deployment	Maven	3.8+	Build automation
	Docker (opt.)	20.10+	Containerization (future use)

Table 1 : Tech Stack

REQUIREMENTS

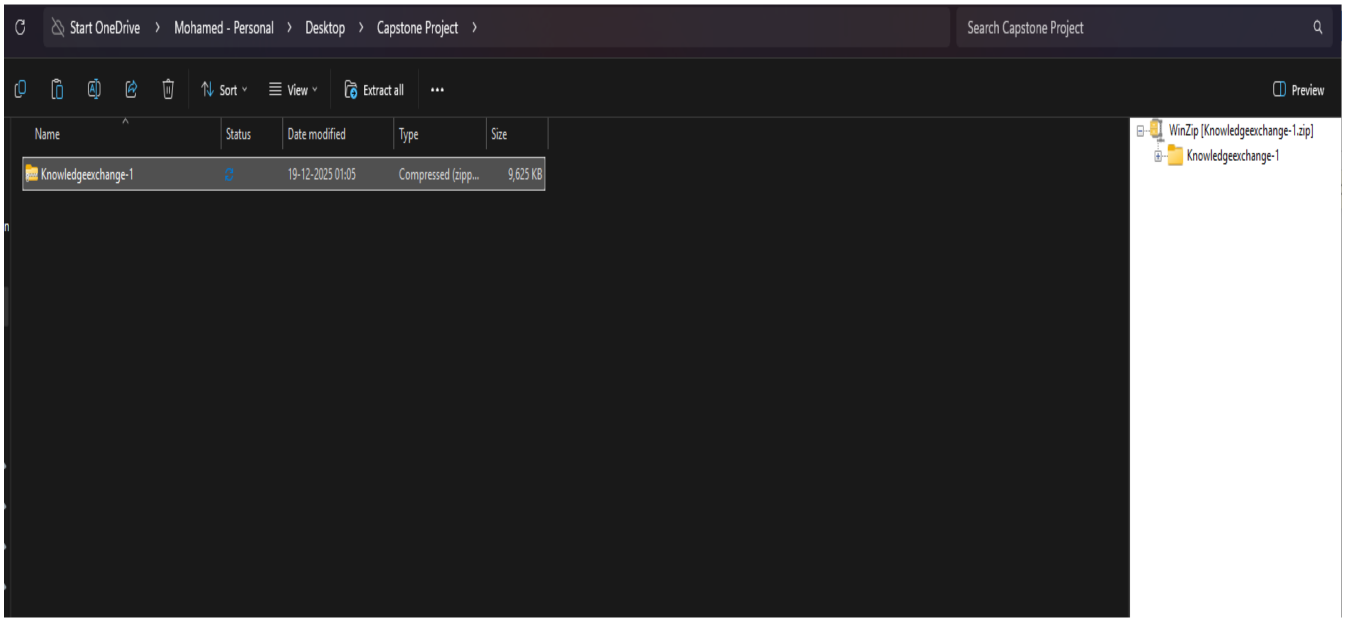
1.1 Hardware

- **RAM**: Minimum 2GB
- **CPU**: 2+ cores
- **Storage**: 5GB+
- **Bandwidth**: 10 Mbps+

1.2 Software

- Java 17 JDK
- Maven 3.8+
- MySQL Server 8.0+
- Docker 20.10+ (for containerization)
- Git 2.30+

1.3 Development Environment Setup(Using Zip Extraction)



Name	Status	Date modified	Type	Size
Knowledgeexchange-1		19-12-2025 01:05	Compressed (zipp...	9,625 KB

Figure 1 : Environment Setup

SYSTEM ARCHITECTURE

The system architecture of Knowledge Exchange follows a layered design to ensure clear separation of responsibilities, scalability, and ease of maintenance. The overall flow of the application, from user interaction to data persistence, is illustrated Below

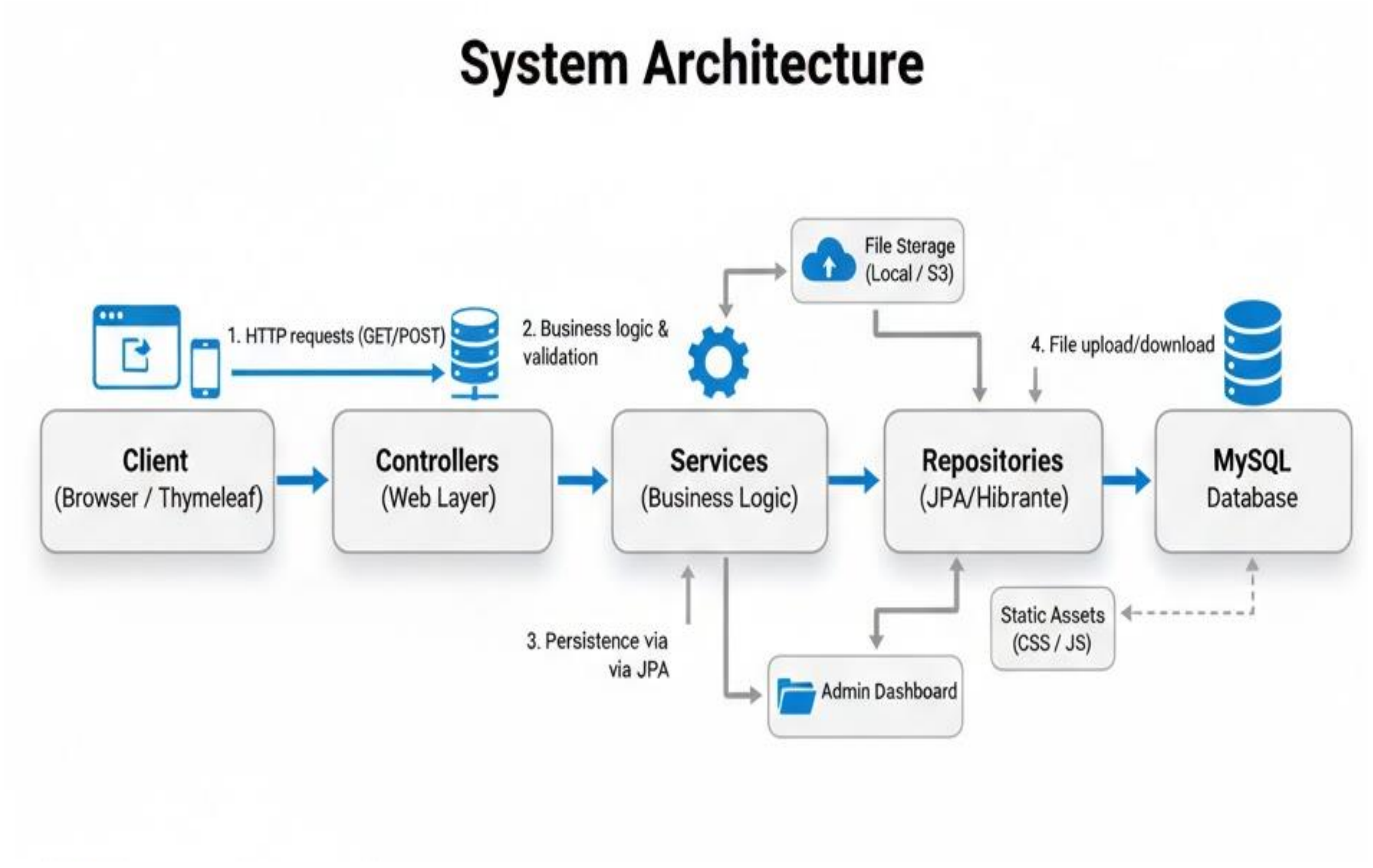

System Architecture
1. HTTP requests (GET/POST)
2. Business logic & validation
File Sterage (Local / S3)
4. File upload/download
Client (Browser / Thymeleaf)
Controllers (Web Layer)
Services (Business Logic)
Repositories (JPA/Hibrante)
MySQL Database
3. Persistence via via JPA
Admin Dashboard
Static Assets (CSS / JS)

Figure 2 : System Architecture

Key Points

1. Users interact via Thymeleaf templates styled with Bootstrap.
2 .Controllers handle requests and delegate to services.
3. Services apply business logic and interact with repositories.
4. Repositories persist data in MySQL using JPA/Hibernate.
5. Files are stored securely and linked to doubts.
6. Admin dashboard provides monitoring and analytics.

The Knowledge Exchange platform uses Spring Boot WebSocket and the STOMP protocol to deliver low-latency, bidirectional updates. The end-to-end flow for a posted message is as follows:

1. **Client submission** — A user submits a doubt or message from the browser (Thymeleaf UI or SPA) via an HTTP POST to /app/chat.sendMessage.
2. **Controller processing** — The MessageHandler controller receives the request, performs input validation and authorization checks, and forwards the validated payload to the service layer.
3. **Service logic and file handling** — The service layer executes business rules, processes multipart file uploads if present (validating types and sizes), prepares metadata, and coordinates persistence.
4. **Persistence** — The repository layer (JPA/Hibernate) persists the message and any file references to the MySQL database, ensuring transactional integrity.
5. **Event publication** — Upon successful persistence, the service publishes an event to the WebSocket broker. The broker uses STOMP to route the event to the appropriate topic.
6. **Broadcast to subscribers** — The broker broadcasts the event on /topic/public (or other topic channels), and all connected clients subscribed to that topic receive the new message in real time.

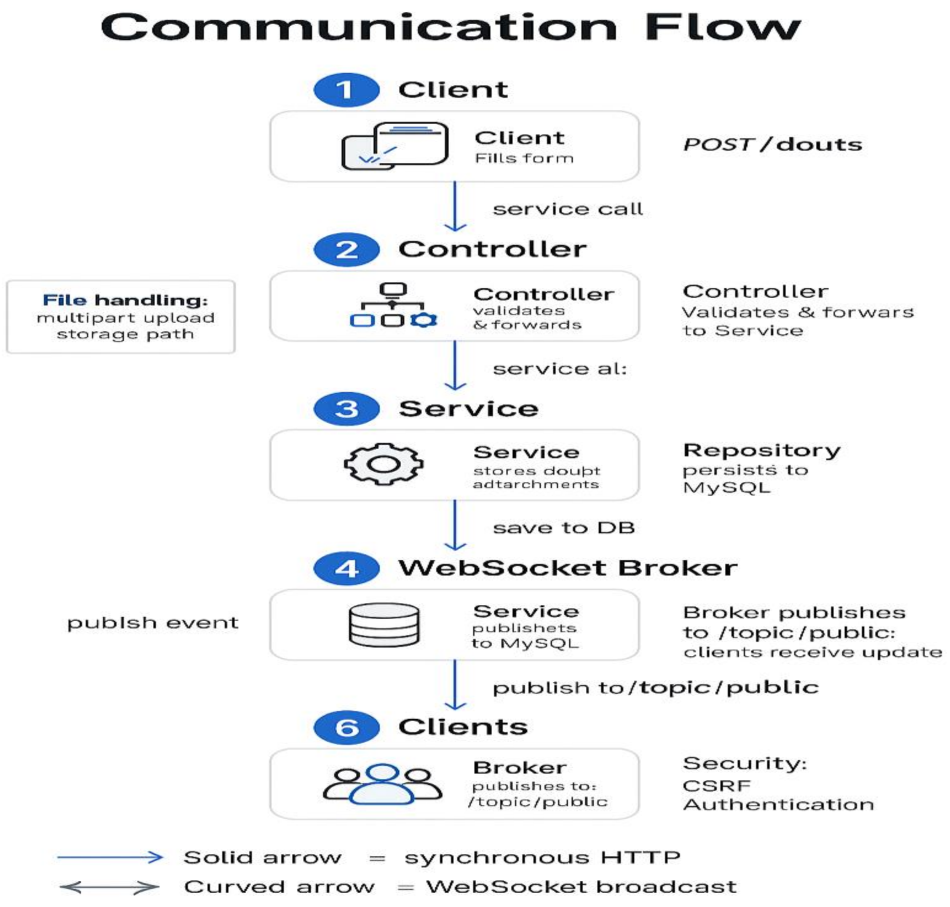


Figure 3 : Communication Flow

Entity-Relationship Diagram

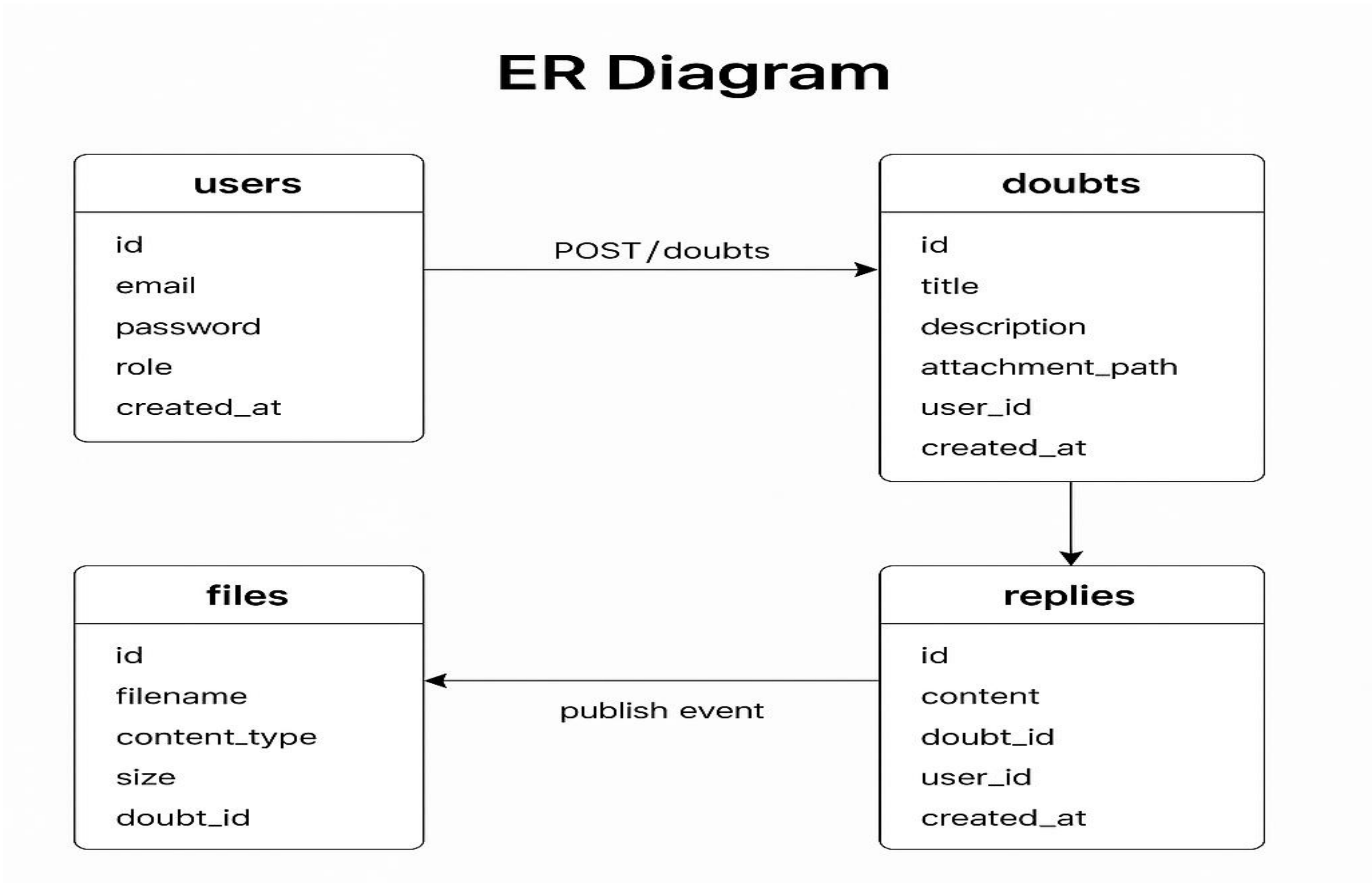


Figure 4 : Database Schema (ER Diagram)

Figure 4 Explains the database schema of the Knowledge Exchange Platform application, illustrating the core entities, their attributes, and the relationships between users, messages, shared doubt files. The diagram highlights one- to-many relationships between users and messages, and users and shared files, as well as a one-to-one relationship between users.

HTTP REQUEST METHODS

HTTP request methods define the actions that a client can perform on server resources.They specify how data should be retrieved, created, updated, or deleted.Each method follows a standardized rule to ensure consistent communication between client and server.

In the Knowledge Exchange Platform, these methods enable structured and secure RESTful API interactions.

HTTP Method	Endpoint	Description
GET	**http://localhost:9096/api/users**	Return the list of all users
GET	http://localhost:9096/api/users/5	Return the user of id
POST	http://localhost:9096/api/auth/register	Create a new user account
POST	http://localhost:9096/api/auth/login	Authenticate user and return JWT
PUT	http://localhost:9096/api/users/profile	Update user profile
DELETE	http://localhost:9096/api/chat/7	Delete message of id 7
DELETE	**http://localhost:9096/api/file/1**	Delete file of id 1

Table 2 : HTTP REQUEST METHOD

MODULES

Module 1: **User Management**

Purpose:

Handles user registration, authentication, profile data, and role assignment to enforce access control across the platform.

Responsibilities

- Register new users with email verification and secure password storage.
- Authenticate users and issue session tokens or maintain server sessions.
- Assign and manage roles (ROLE_USER, ROLE_ADMIN).
- Track user status (online/offline, banned, deactivated).
- Expose profile endpoints for viewing and updating user information.

Key fields

- id — Primary key (UUID or auto-increment).
- name — Display name.
- email — Unique identifier; used for login and notifications.
- password — BCrypt hashed password.
- role — Enum or set: ADMIN, USER.
- is_online — Boolean for presence.
- is_banned — Boolean for moderation state.
- created_at, updated_at — Timestamps.

API endpoints

- POST /register — Create account; validate email, hash password, send verification.
- POST /login — Authenticate and create session; return redirect or token.
- GET /profile — Return current user profile.
- PUT /profile — Update profile fields (name, avatar reference).
- GET /admin/users — Admin only: list users, filter by role/status.

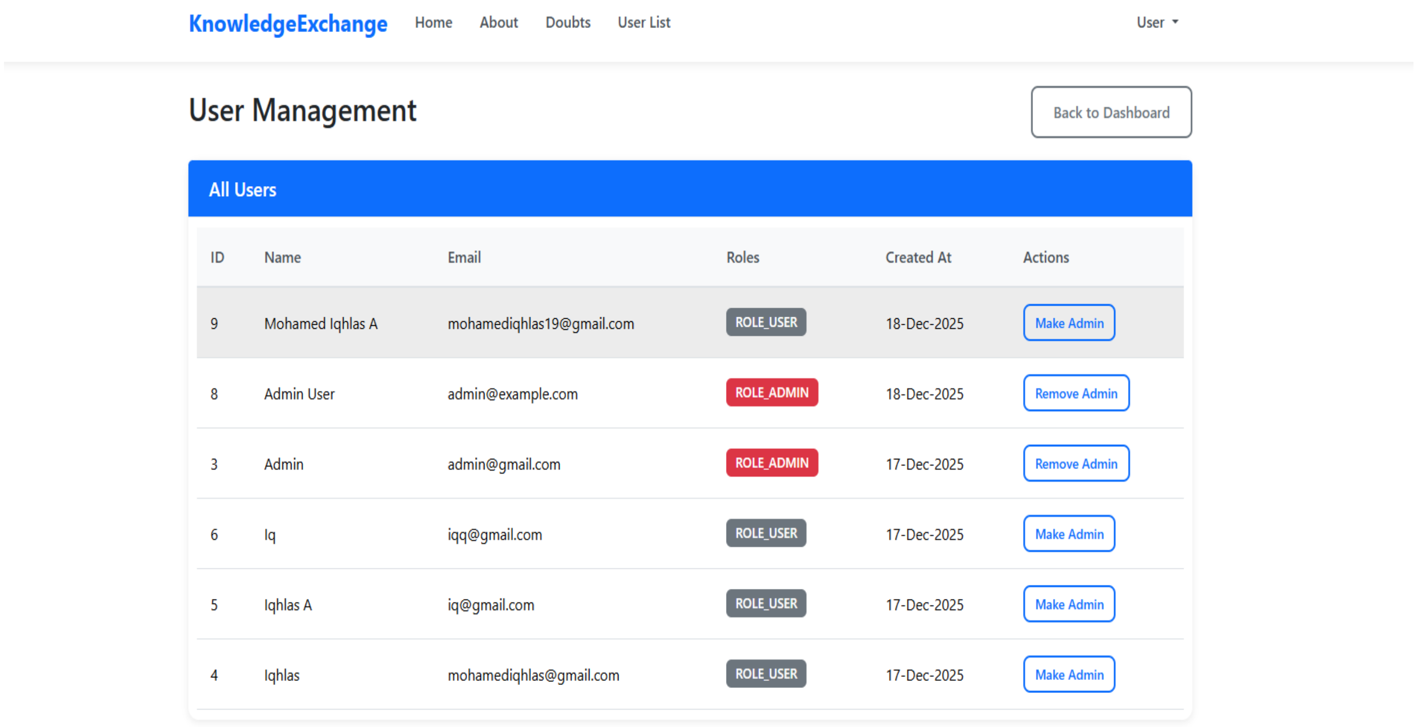
- PUT /admin/users/{id}/role — Admin only: change role.
- PUT /admin/users/{id}/ban — Admin only: ban/unban user.

Validation and security

- Enforce strong password rules and rate limit login attempts.
- Use BCrypt for hashing and never store plaintext passwords.
- Require email verification before granting full privileges.
- Protect endpoints with authentication and role checks.
- Log critical events (failed logins, role changes, bans).

UI considerations

- Registration and login forms with clear validation messages.
- Profile page with avatar preview and edit controls.
- Admin user list with search, filters, and bulk actions.



User Management

All Users

ID	Name	Email	Roles	Created At	Actions
9	Mohamed Iqhlas A	mohamediqhlas19@gmail.com	ROLE_USER	18-Dec-2025	Make Admin
8	Admin User	admin@example.com	ROLE_ADMIN	18-Dec-2025	Remove Admin
3	Admin	admin@gmail.com	ROLE_ADMIN	17-Dec-2025	Remove Admin
6	Iq	iqq@gmail.com	ROLE_USER	17-Dec-2025	Make Admin
5	Iqhlas A	iq@gmail.com	ROLE_USER	17-Dec-2025	Make Admin
4	Iqhlas	mohamediqhlas@gmail.com	ROLE_USER	17-Dec-2025	Make Admin

Figure 4: User Management

Module 2: **Doubt Management**

Purpose :
Manage creation, retrieval, update, and deletion of doubts (questions) posted by users, including support for attachments and threaded replies.

Responsibilities

- Create and store doubts with title, description, tags, and optional attachments.
- Retrieve lists of doubts with pagination, sorting, and filtering (by tag, unanswered, recent).
- Provide detailed view for a single doubt including replies and attachments.
- Allow owners and admins to edit or delete doubts.

Key fields

- **id** — Primary key.
- **title** — Short descriptive title.
- **description** — Full text of the doubt.
- **created_by** — Foreign key to users.
- **created_at, updated_at** — Timestamps.
- **status** — e.g., OPEN, ANSWERED, CLOSED.
- **tags** — Optional list for categorization.
- **attachment_refs** — References to files stored in File Management.

API endpoints

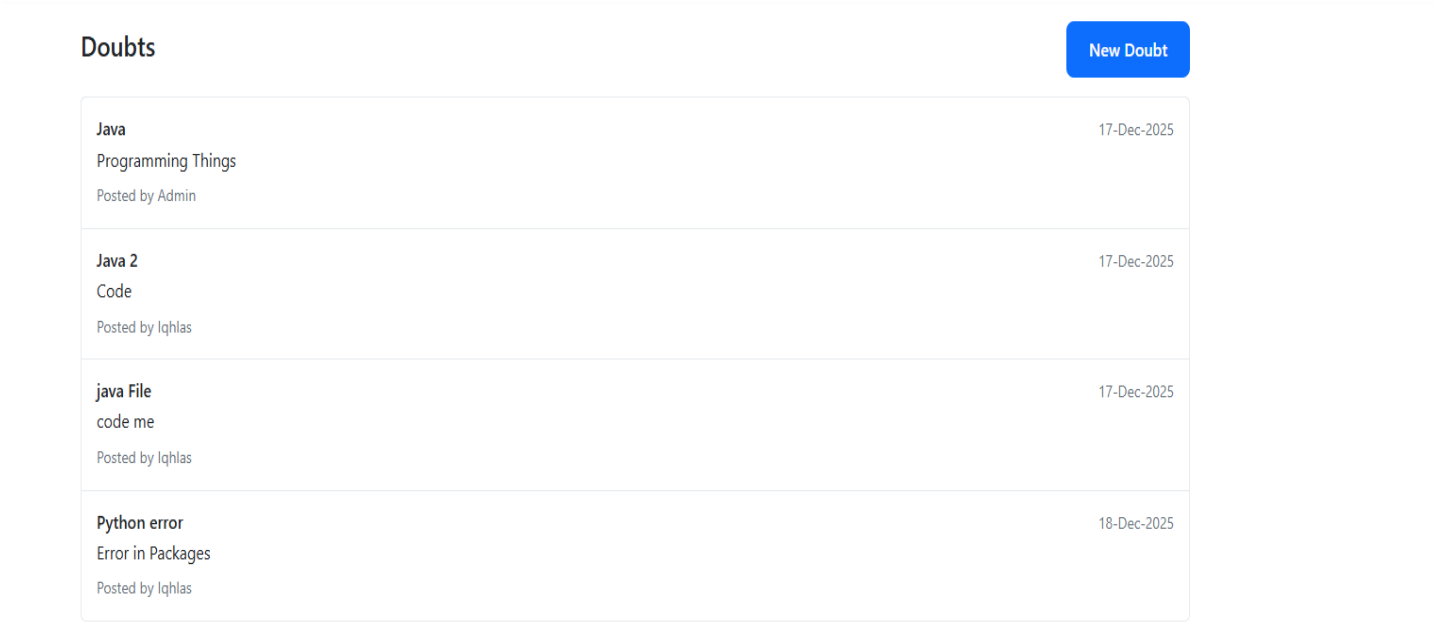
- GET /doubts — List doubts with query params: page, size, tag, status.
- POST /doubts — Create a new doubt (multipart if attachments).
- GET /doubts/{id} — Get doubt details with replies and attachments.
- PUT /doubts/{id} — Update doubt (owner or admin).
- DELETE /doubts/{id} — Delete doubt (owner or admin).

Business rules

- Only authenticated users can post doubts.
- Owners and admins can edit/delete; moderators (if present) can moderate content.
- Tagging helps discoverability; validate tags against allowed list.
- Mark doubt as answered when a reply is accepted.

UI considerations

- Compact list view with title, snippet, author, date, and status badge.
- Detail view with threaded replies, attachments, and quick-reply box.
- New doubt modal or page with file upload and tag suggestions.



Java
Programming Things
Posted by Admin
17-Dec-2025

Java 2
Code
Posted by Iqhlas
17-Dec-2025

java File
code me
Posted by Iqhlas
17-Dec-2025

Python error
Error in Packages
Posted by Iqhlas
18-Dec-2025

Figure 5: Doubt Management

Module 3: File Management

Purpose :
Securely handle file uploads and downloads associated with doubts and user profiles, storing metadata and safe references rather than exposing raw storage paths.

Responsibilities

- Accept multipart uploads, validate file type and size, and store files in chosen storage (local filesystem or cloud object store).
- Persist file metadata and a secure storage reference in the database.
- Serve files via authenticated endpoints with proper content-type and range support if needed.
- Support deletion and cleanup of orphaned files.

Key fields

- **id** — Primary key.
- **original_filename** — Original name provided by user.

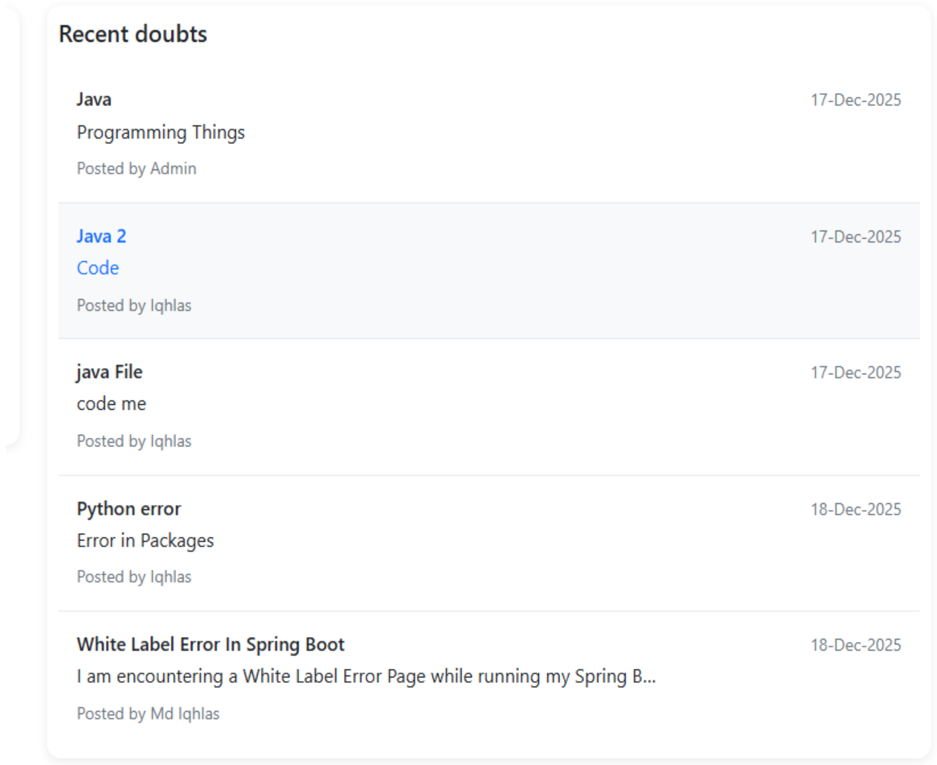
- **storage_path** — Secure reference or key to the stored file.
- **content_type** — MIME type.
- **file_size** — Size in bytes.
- **uploaded_by** — Foreign key to users.
- **doubt_id** — Foreign key to associated doubt (nullable).
- **created_at** — Upload timestamp.

API endpoints

- POST /files — Upload file (multipart); returns file id/reference.
- GET /files/{id} — Download file (authorization enforced).
- DELETE /files/{id} — Delete file (owner or admin).

Validation and security

- Enforce allowed MIME types and maximum file size.
- Scan or sanitize file names; avoid storing user-supplied paths.
- Store files outside web root or use signed URLs for cloud storage.
- Authorize downloads: only owners, admins, or authorized users can access.
- Maintain transactional consistency: if DB persist fails, remove uploaded file.



Recent doubts

Java
Programming Things
Posted by Admin
17-Dec-2025

Java 2
Code
Posted by Iqhlas
17-Dec-2025

java File
code me
Posted by Iqhlas
17-Dec-2025

Python error
Error in Packages
Posted by Iqhlas
18-Dec-2025

White Label Error In Spring Boot
I am encountering a White Label Error Page while running my Spring B...
Posted by Md Iqhlas
18-Dec-2025

Figure 6: File Management

Module 4: Reply Management
Purpose:
Support threaded discussions beneath each doubt, enabling users to reply, edit, and view conversation history in chronological order.
Responsibilities

- Create, update, and delete replies linked to doubts and users.
- Support nested replies or a single-level thread depending on UX choice.
- Track reply metadata and moderation status.

Key fields

- **id** — Primary key.
- **doubt_id** — Foreign key to doubts.
- **user_id** — Foreign key to users.
- **content** — Reply text.
- **parent_reply_id** — Nullable for nested replies.
- **created_at, updated_at** — Timestamps.
- **is_edited** — Boolean flag.
- **is_flagged** — Moderation flag.

API endpoints

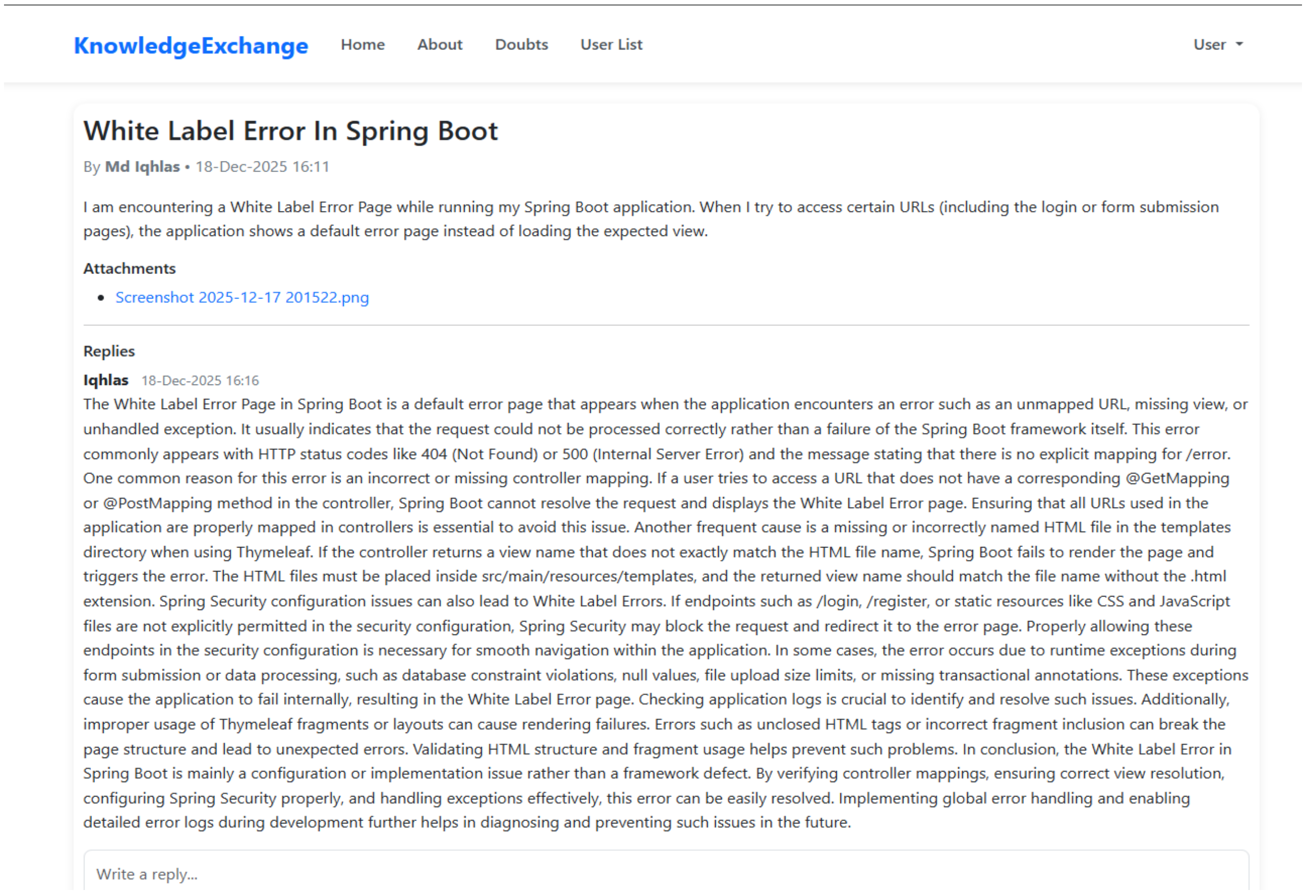
- POST /doubts/{id}/replies — Add reply to a doubt.
- GET /doubts/{id}/replies — List replies for a doubt (paginated).
- PUT /replies/{id} — Edit reply (owner or admin).
- DELETE /replies/{id} — Delete reply (owner or admin).
- POST /replies/{id}/flag — Flag reply for moderation.

Business rules

- Only authenticated users can reply.
- Owners and admins can edit/delete replies; flagged replies enter moderation queue.
- Limit nesting depth to maintain readability (e.g., 2–3 levels).
- Notify doubt owner when a new reply is posted (optional email or in-app notification).

UI considerations

- Inline reply composer beneath each doubt.
- Collapsible nested replies for long threads.
- Visual indicators for edited or flagged replies.


KnowledgeExchange Home About Doubts User List User

White Label Error In Spring Boot

By **Md Iqhlas** • 18-Dec-2025 16:11

I am encountering a White Label Error Page while running my Spring Boot application. When I try to access certain URLs (including the login or form submission pages), the application shows a default error page instead of loading the expected view.

Attachments

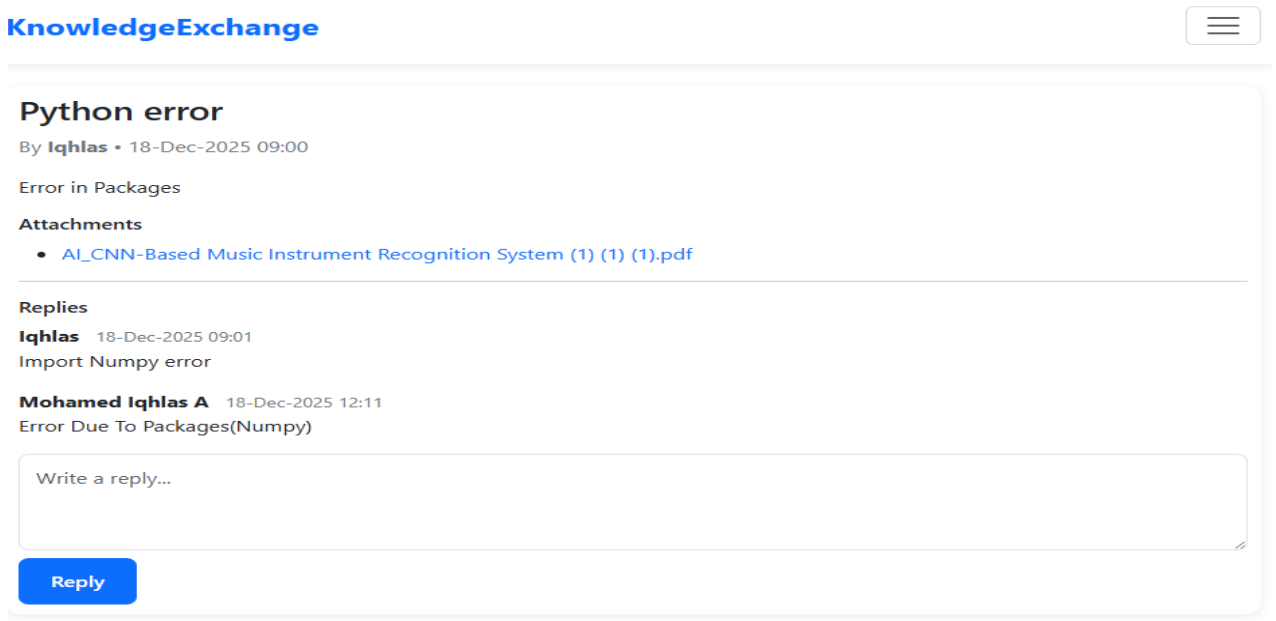
- Screenshot 2025-12-17 201522.png

Replies

Iqhlas 18-Dec-2025 16:16

The White Label Error Page in Spring Boot is a default error page that appears when the application encounters an error such as an unmapped URL, missing view, or unhandled exception. It usually indicates that the request could not be processed correctly rather than a failure of the Spring Boot framework itself. This error commonly appears with HTTP status codes like 404 (Not Found) or 500 (Internal Server Error) and the message stating that there is no explicit mapping for /error. One common reason for this error is an incorrect or missing controller mapping. If a user tries to access a URL that does not have a corresponding @GetMapping or @PostMapping method in the controller, Spring Boot cannot resolve the request and displays the White Label Error page. Ensuring that all URLs used in the application are properly mapped in controllers is essential to avoid this issue. Another frequent cause is a missing or incorrectly named HTML file in the templates directory when using Thymeleaf. If the controller returns a view name that does not exactly match the HTML file name, Spring Boot fails to render the page and triggers the error. The HTML files must be placed inside src/main/resources/templates, and the returned view name should match the file name without the .html extension. Spring Security configuration issues can also lead to White Label Errors. If endpoints such as /login, /register, or static resources like CSS and JavaScript files are not explicitly permitted in the security configuration, Spring Security may block the request and redirect it to the error page. Properly allowing these endpoints in the security configuration is necessary for smooth navigation within the application. In some cases, the error occurs due to runtime exceptions during form submission or data processing, such as database constraint violations, null values, file upload size limits, or missing transactional annotations. These exceptions cause the application to fail internally, resulting in the White Label Error page. Checking application logs is crucial to identify and resolve such issues. Additionally, improper usage of Thymeleaf fragments or layouts can cause rendering failures. Errors such as unclosed HTML tags or incorrect fragment inclusion can break the page structure and lead to unexpected errors. Validating HTML structure and fragment usage helps prevent such problems. In conclusion, the White Label Error in Spring Boot is mainly a configuration or implementation issue rather than a framework defect. By verifying controller mappings, ensuring correct view resolution, configuring Spring Security properly, and handling exceptions effectively, this error can be easily resolved. Implementing global error handling and enabling detailed error logs during development further helps in diagnosing and preventing such issues in the future.

Write a reply...


KnowledgeExchange

Python error

By **Iqhlas** • 18-Dec-2025 09:00

Error in Packages

Attachments

- AI_CNN-Based Music Instrument Recognition System (1) (1) (1).pdf

Replies

Iqhlas 18-Dec-2025 09:01

Import Numpy error

Mohamed Iqhlas A 18-Dec-2025 12:11

Error Due To Packages(Numpy)

Write a reply...

Reply

Figure 7: Reply Management

Module 5**: Admin Dashboard**

Purpose :

Provide administrators with tools to monitor platform health, manage users and content, and access analytics for informed moderation and platform decisions**.**

Core features

- User Management — Searchable user list, role assignment, ban/unban, deactivate/reactivate.
- Doubt Moderation — Queue of flagged doubts/replies, quick actions (approve, edit, remove), and bulk moderation tools.
- Analytics and Metrics — Overview cards for total users, active users (7-day), total doubts, unanswered doubts, average response time, top tags.
- Activity Feed — Recent actions: new doubts, replies, file uploads, admin actions.
- Export and Reporting — CSV export of user lists and activity logs (admin-only).
- Audit Log — Record of admin actions with timestamps and actor identity**.**

UI components

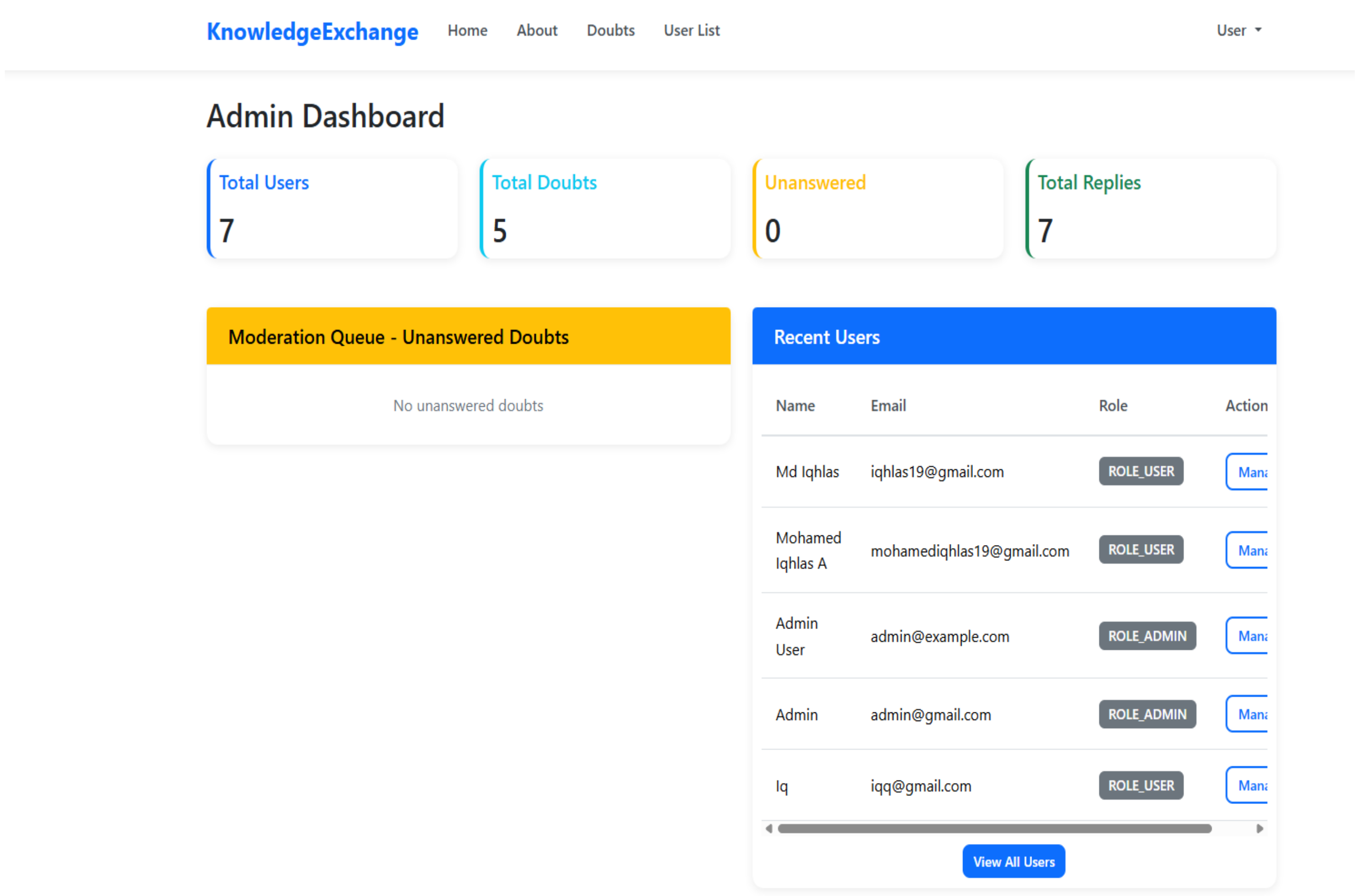
- Dashboard overview with KPI cards and trend sparklines.
- Moderation table with filters (flagged, unanswered, by tag).
- User detail modal with activity history and quick actions.
- Visual charts for tag distribution and response time.

Security and access

- Dashboard endpoints protected by ROLE_ADMIN.
- Two-factor authentication recommended for admin accounts.
- Audit logging for all admin actions to ensure accountability.

Operational notes

- Rate limit admin actions to prevent accidental mass changes.
- Provide role-based sub-permissions if multiple admin levels are needed (e.g., moderator vs super-admin).
- Include a safe preview mode for content edits to avoid accidental public changes.



Admin Dashboard

Total Users 7

Total Doubts 5

Unanswered 0

Total Replies 7

Moderation Queue - Unanswered Doubts

No unanswered doubts

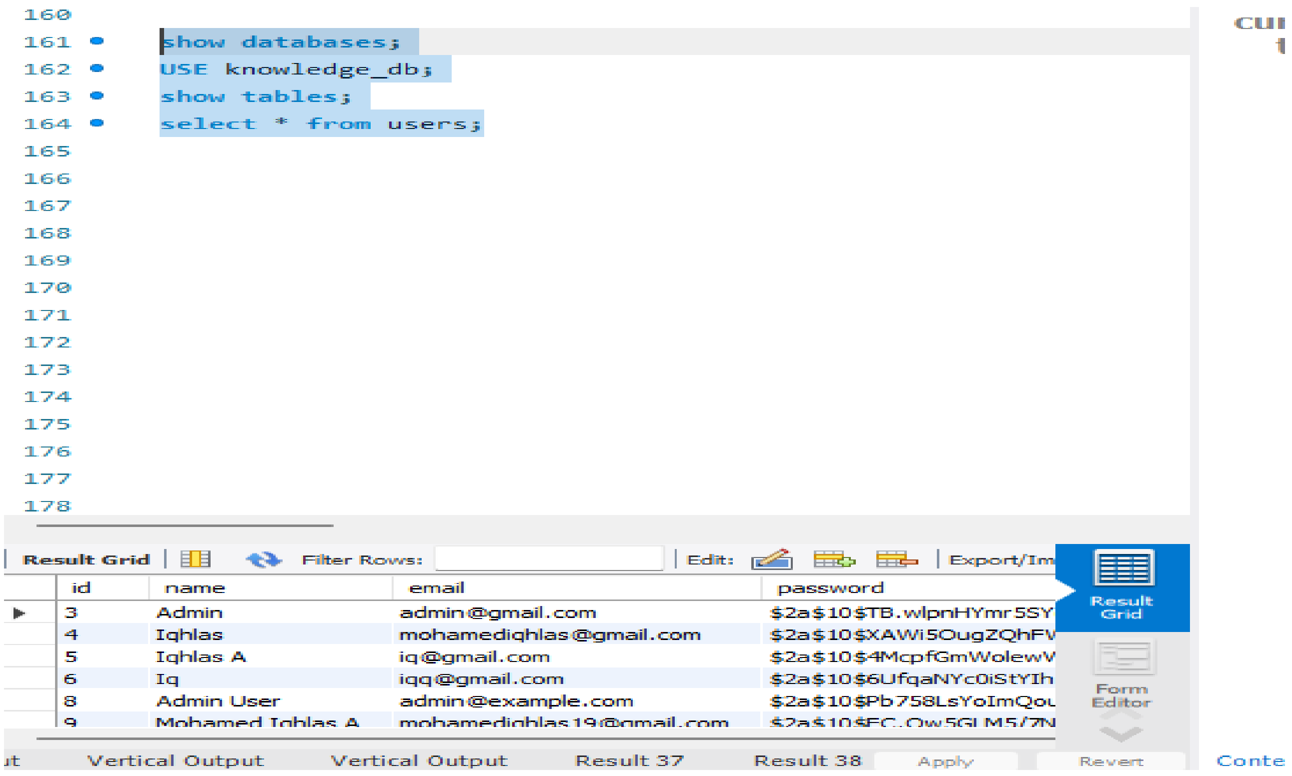
Recent Users

Name	Email	Role	Action
Md Iqhlas	iqhlas19@gmail.com	ROLE_USER	Mana
Mohamed Iqhlas A	mohamediqhlas19@gmail.com	ROLE_USER	Mana
Admin User	admin@example.com	ROLE_ADMIN	Mana
Admin	admin@gmail.com	ROLE_ADMIN	Mana
Iq	iqq@gmail.com	ROLE_USER	Mana

Figure 8: Admin Management.

PROJECT OUTPUT – DATABASE SCHEMA

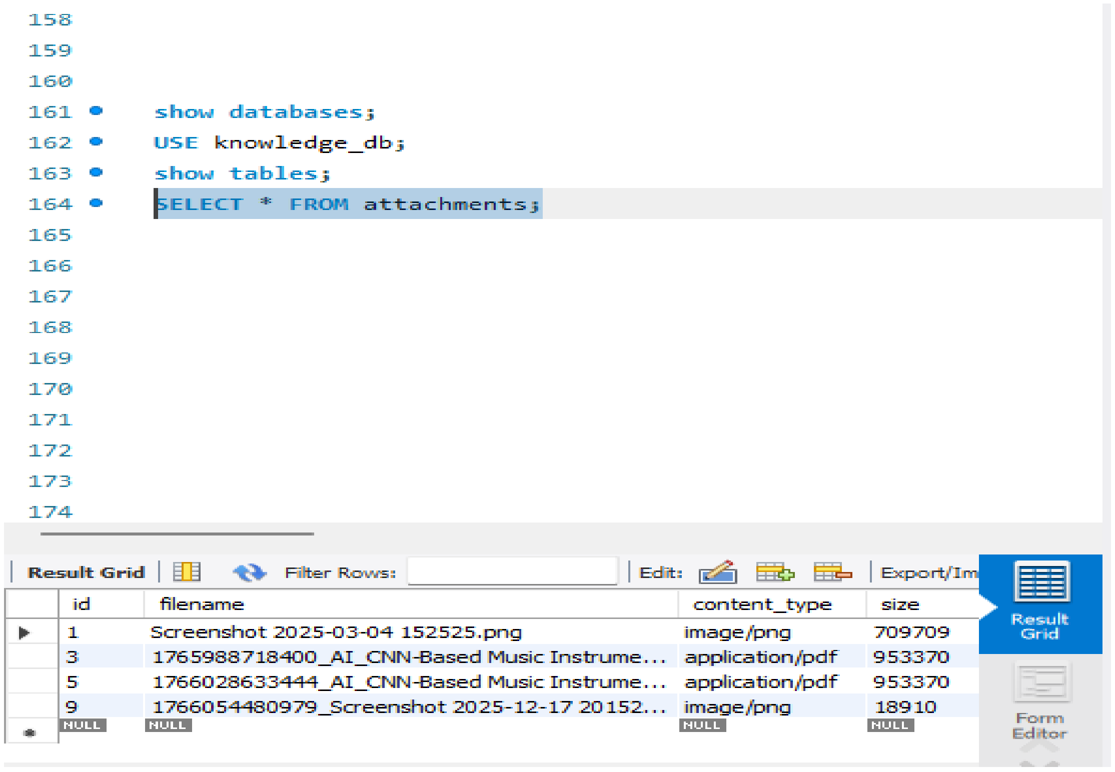
1) users Table :



```
show databases;
USE knowledge_db;
show tables;
select * from users;
```

id	name	email	password
3	Admin	admin@gmail.com	$2a$10$TB.wlpnHYmr5SY
4	Iqhlas	mohamediqhlas@gmail.com	$2a$10$XAWi5OugZQhFV
5	Iqhlas A	iq@gmail.com	$2a$10$4McpfGmWolewV
6	Iq	iqq@gmail.com	$2a$10$6UfqaNYc0iStYIh
8	Admin User	admin@example.com	$2a$10$Pb758LsYoImQou
9	Mohamed Iqhlas A	mohamediqhlas19@gmail.com	$2a$10$FC.Ow5GLM5/7N

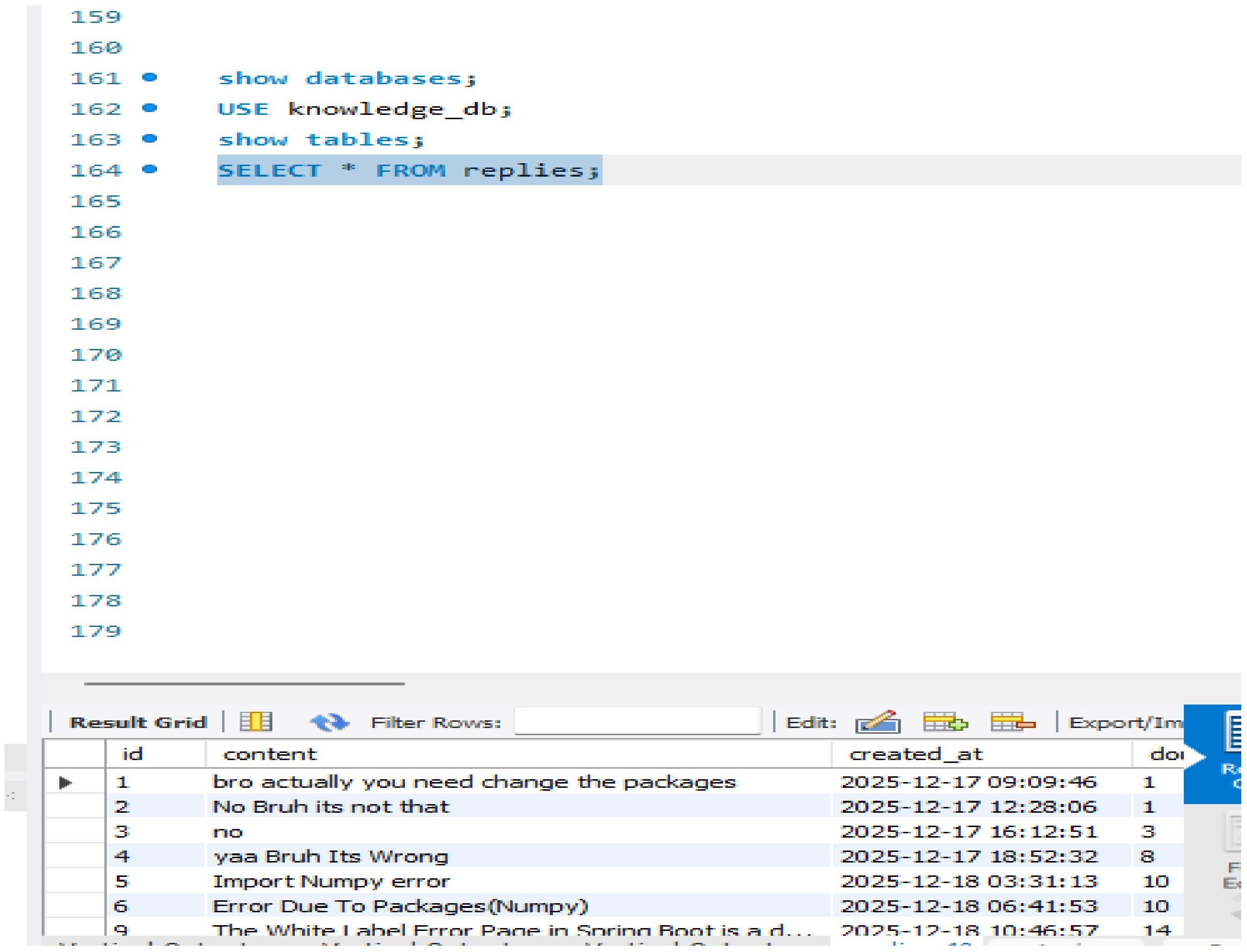
2)File attachment Table:



```
show databases;
USE knowledge_db;
show tables;
SELECT * FROM attachments;
```

id	filename	content_type	size
1	Screenshot 2025-03-04 152525.png	image/png	709709
3	1765988718400_AI_CNN-Based Music Instrume...	application/pdf	953370
5	1766028633444_AI_CNN-Based Music Instrume...	application/pdf	953370
9	1766054480979_Screenshot 2025-12-17 20152...	image/png	18910
NULL	NULL	NULL	NULL

3)Replay messages Table:

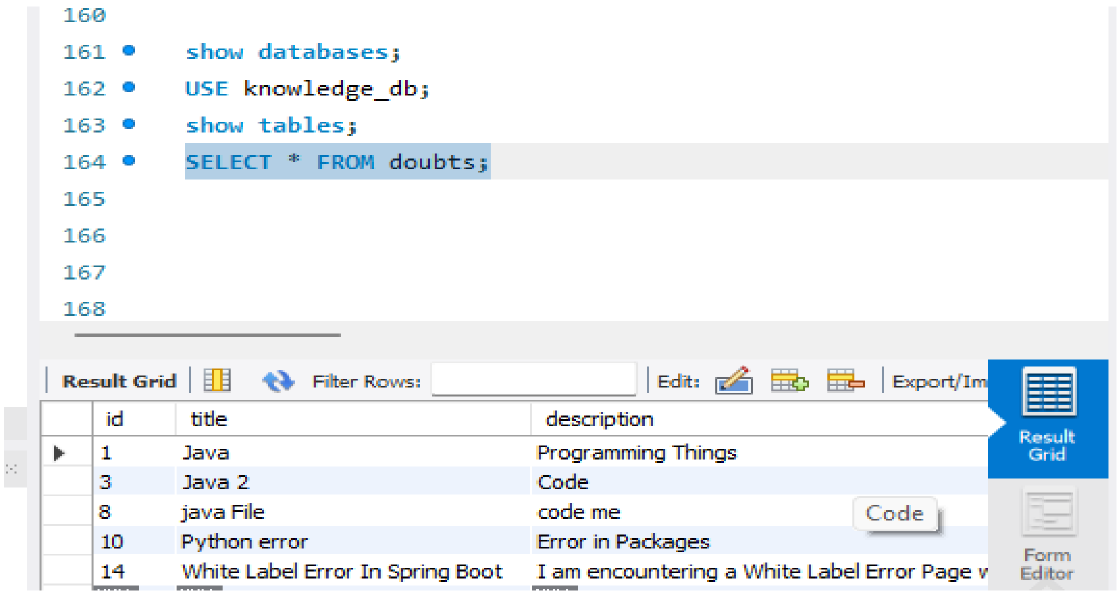


```
show databases;
USE knowledge_db;
show tables;
SELECT * FROM replies;
```

Result Grid | Filter Rows: | Edit: | Export/Im

	id	content	created_at	do
▶	1	bro actually you need change the packages	2025-12-17 09:09:46	1
	2	No Bruh its not that	2025-12-17 12:28:06	1
	3	no	2025-12-17 16:12:51	3
	4	yaa Bruh Its Wrong	2025-12-17 18:52:32	8
	5	Import Numpy error	2025-12-18 03:31:13	10
	6	Error Due To Packages(Numpy)	2025-12-18 06:41:53	10
	9	The White Label Error Page in Spring Boot is a d...	2025-12-18 10:46:57	14

4)Doubts Table:



```
show databases;
USE knowledge_db;
show tables;
SELECT * FROM doubts;
```

Result Grid | Filter Rows: | Edit: | Export/Im

	id	title	description
▶	1	Java	Programming Things
	3	Java 2	Code
	8	java File	code me
	10	Python error	Error in Packages
	14	White Label Error In Spring Boot	I am encountering a White Label Error Page w

CONCLUSION

Knowledge Exchange is a robust and user-friendly platform for posting doubts, sharing files, and engaging in threaded discussions. It implements secure authentication and role-based access control to protect user data and restrict administrative functions. The application follows a clean, layered architecture that promotes maintainability and scalability. File handling and data persistence are implemented with transactional integrity and secure storage practices. Administrative tools provide monitoring and moderation capabilities to keep the platform reliable and well-managed. Overall, the project delivers a practical, extensible foundation suitable for real-world deployment and future enhancements.

DRIVE LINK (for project demo):
https://drive.google.com/file/d/17il2OzrBOLDti26pwMMY-YQ4_Vjql1f6/view?usp=sharing

THANK YOU