# Final Year Project Design


Ramadhani Yassin Ramadhani | BITA/6/22/079/TZ

Project Title: UNIDOCS – Documents Management System

Supervisor: Mr. Hassan Mohammed

# UNIDOCS project Introduction

**UNIDOCS** - Integrated Document Management System for Universities is built to simplify the process of creating, handling, and managing official documents in higher learning institutions. The existing manual system is slow, prone to mistakes, and inefficient, causing unnecessary delays.

UNIDOCS incorporates An **Automated System** for instant assistance and a structured document library for better access and management. This system automates document requests and keeps track of progress, making processes smoother and more transparent. Built with Angular for the frontend, Spring Boot for the backend, and Postgres for data storage, **UNIDOCS** aims to make university administration more efficient, secure, and accessible. This report covers the purpose, challenges, and methods used in developing the system, focusing on improving administrative workflows in universities.

# Feasibility Study: UNIDOCS Report Summary

- **Economic Feasibility**: The project utilizes open-source technologies (Angular, Spring Boot and Postgres) to minimize costs while maintaining efficiency.

- **Technical Feasibility**: I possess strong skills in web technologies, which ensures the successful implementation of the system as part of my Final Year Project.

- **Operational Feasibility**: **UNIDOCS** addresses real administrative challenges faced by universities, making it highly practical and beneficial for users.

By implementing **UNIDOCS**, universities will transition from slow, manual processes to an intelligent, automated document management system, ultimately enhancing productivity and user satisfaction.

# UNIDOCS Project Methodology

## Approach

✓ Object-Oriented (OOA) for modularity & scalability.

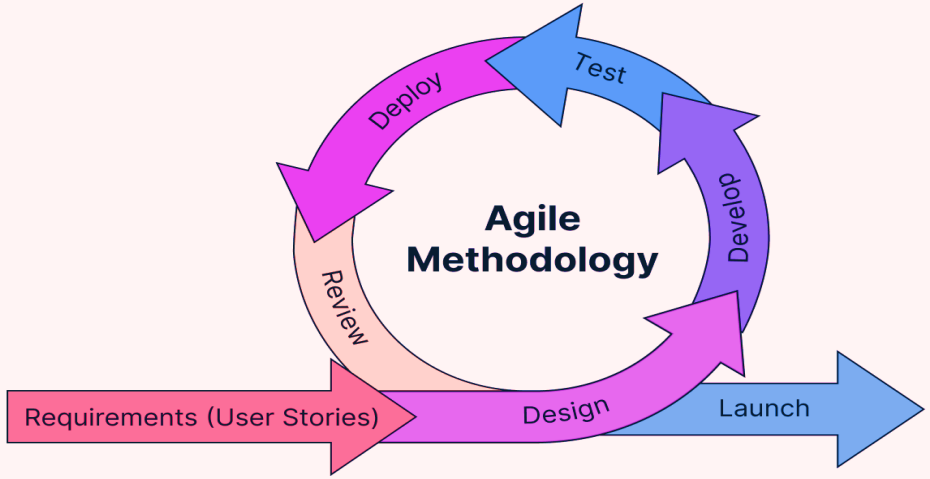
✓ AI Chatbot (CHATBASE API) for assistance.

✓ UML Diagrams (Use Case, Class) for system design.

✓ Bottom-Up Development.

## SDLC Model

✓ Agile Methodology (iterative, user feedback-driven).




## Tools & Components

✓ Angular, Spring Boot, Git & Postman

POSTMAN

# UNIDOCS Functional requirements

1. The system allows students to **Request documents** online and enables administrators to process them through a secure interface.

**2.** The system **Automatically Verifies** student identity and enrollment details before processing any request.

**3.** The system provides **Real-time status updates** and alerts to students via email, SMS, and dashboard notifications.

**4.** The system includes **A chatbot** that guides students through request procedures and answers document-related questions instantly.

**5.** The system **Stores all documents in an encrypted** digital archive, allowing authorized users to retrieve them quickly.

**6.** The system **Generates detailed reports** on request volume, processing times, and system usage trends to support administrative decisions.

# UNIDOCS Non-Functional requirements

**Usability:**

The system interface must be intuitive and easy to navigate for all users.

**Reliability:**

The system must maintain an uptime of **99.9%**, with minimal downtime for maintenance.

**Security:**

All user data must be encrypted and protected against unauthorized access.

Multi-factor authentication should be implemented for sensitive actions.

**Scalability:**

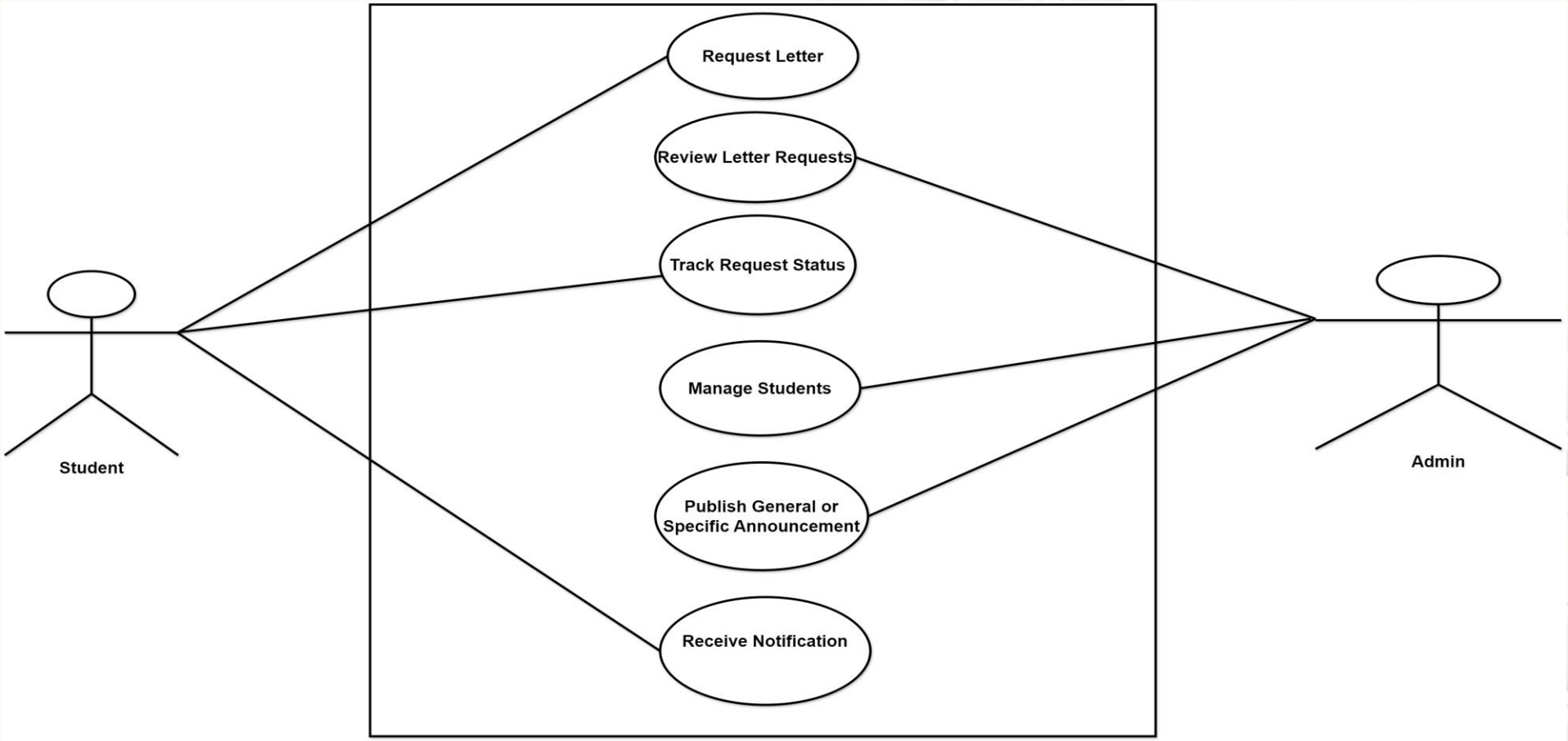
The system must support increasing numbers of users and document requests without performance degradation.

**Performance:**

Requests should be processed within **3 seconds** to ensure a smooth user experience.

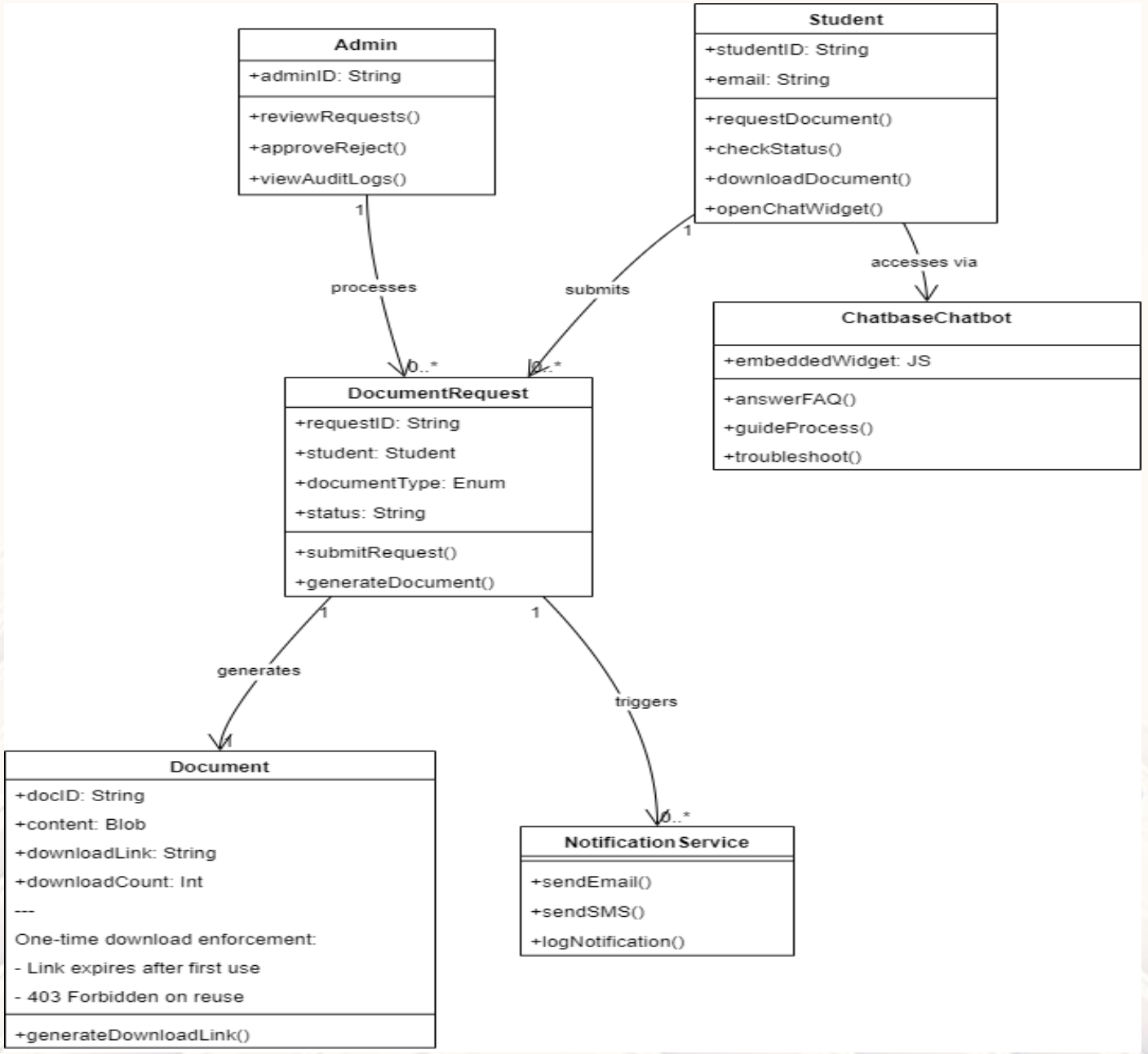
# UNIDOCS Requirement Modeling: Usecase diagram

The use case diagram below illustrates the core functionalities of **UNIDOCS** – an integrated system designed to streamline document requests, approvals, and student-admin interactions within a university environment.

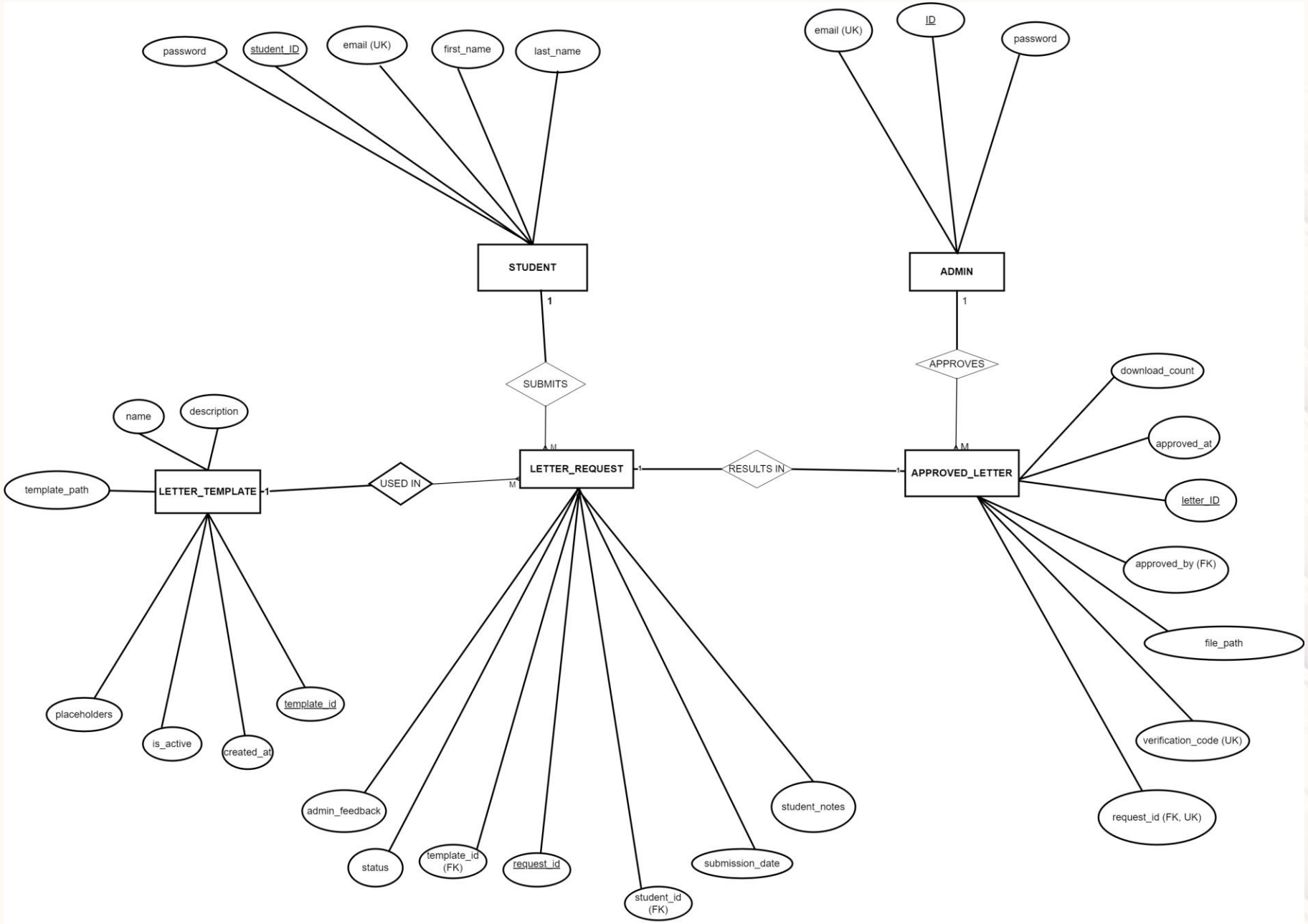


# UNIDOCS Requirement Modeling: Class diagram

The class diagram on the left provides a structural overview of the **UNIDOCS** system, highlighting the main entities, their attributes, and the relationships between them. It captures the backbone of the system's logic by representing how student profiles, administrative users, letter types, and request processes are interlinked.



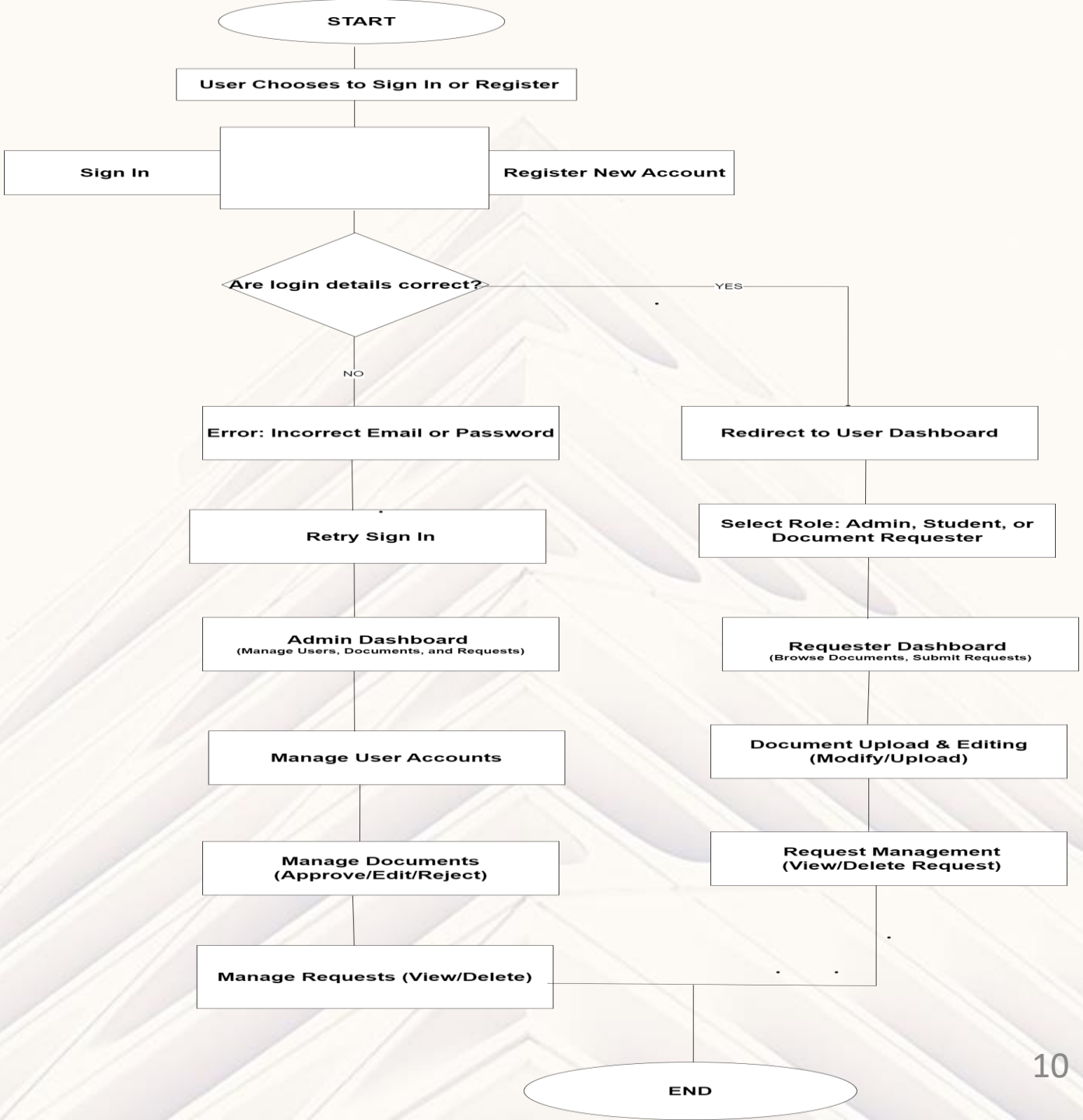
# UNIDOCS Requirement Modeling: ER-Diagram




The ER diagram on the Right presents the core data structure of the **UNIDOCS** system, focusing on five main entities: Student, Admin, Letter_Request, Letter_Template and Approved_Letter.

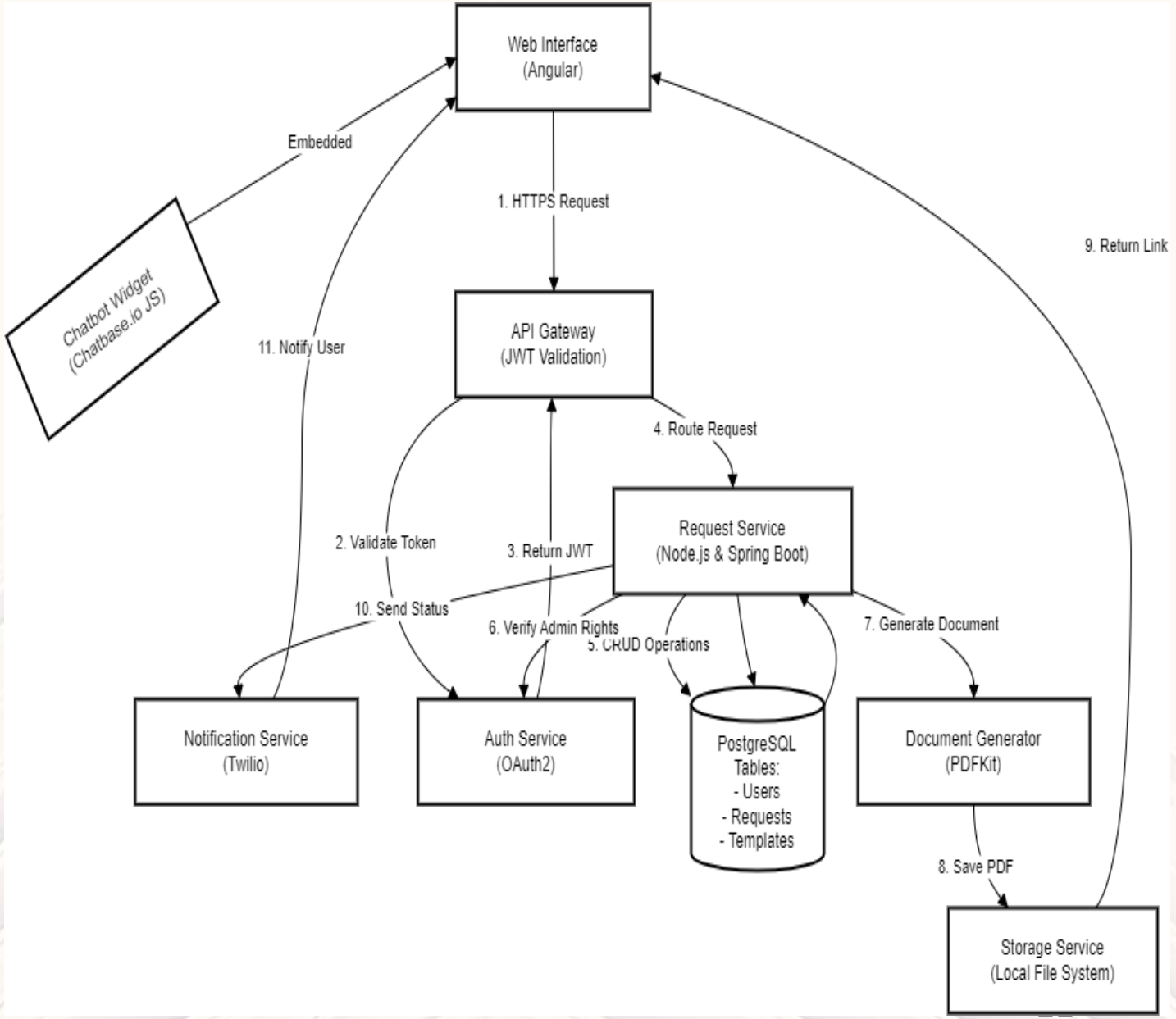
# UNIDOCS Data Flow Diagram

The data flow diagram (DFD) below illustrates how information moves within the UNIDOCS system – a centralized platform developed to manage document requests, approvals, and communication between students and university administrators.



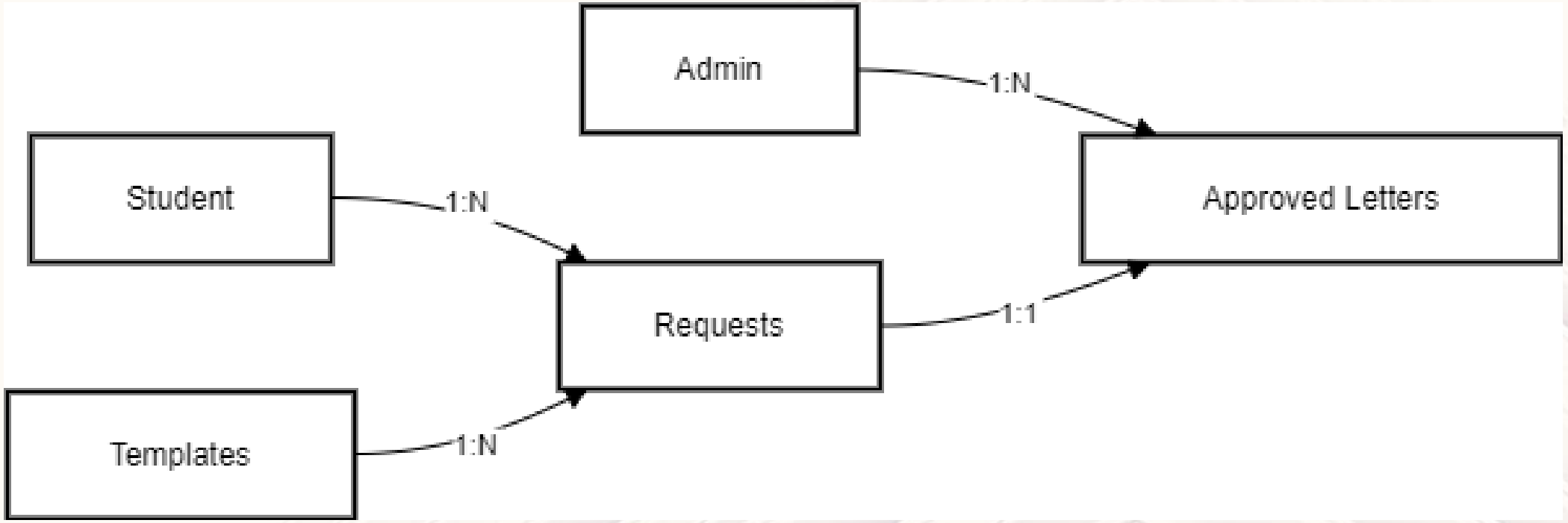
# UNIDOCS Architecture (Process Flow)

The architecture diagram on the left outlines the process flow of the **UNIDOCS** system – a streamlined digital platform for managing academic document requests and approvals within the university. It visually maps out the sequence of operations, starting from student login and letter request submission to administrative review, approval, and final letter generation.

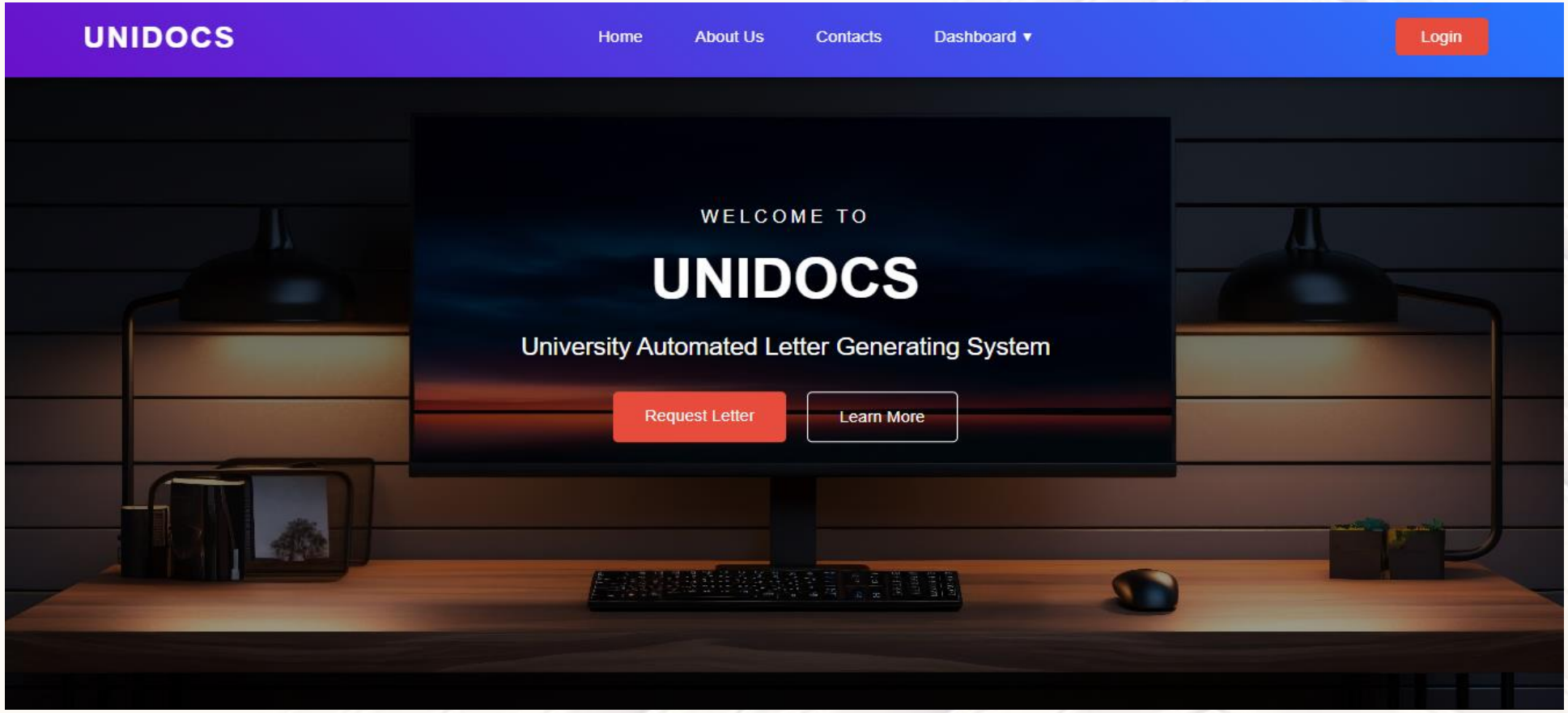


# UNIDOCS Database Entities Relationship

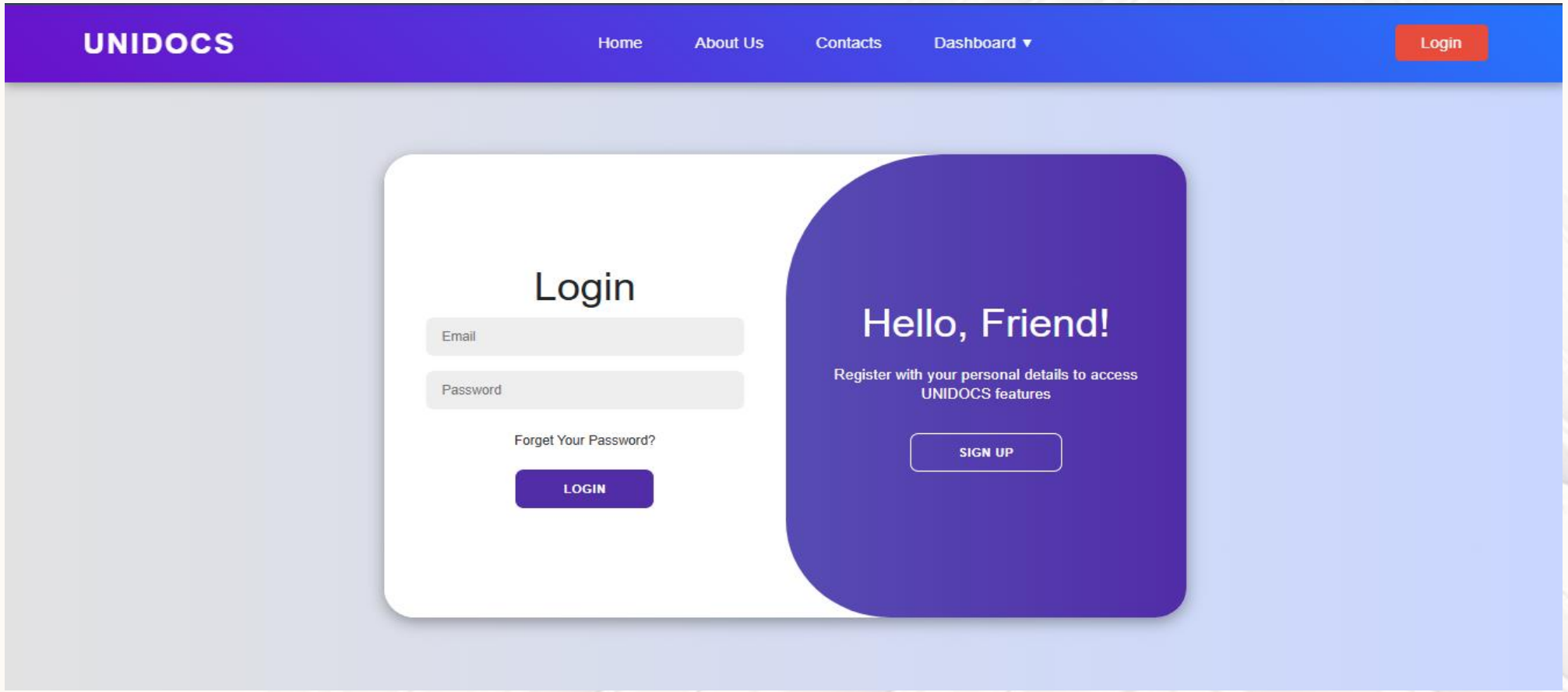
The diagram below illustrates the relationships between the core database entities of the **UNIDOCS** system, highlighting how data is interconnected within the platform. It outlines the associations between the key entities: **Admin**, **Letter_Template**, **Letter_Request**, **Student**, and **Approved_Letter**. The Student entity is linked to the Letter_Request entity, representing the student's submissions for various document types.



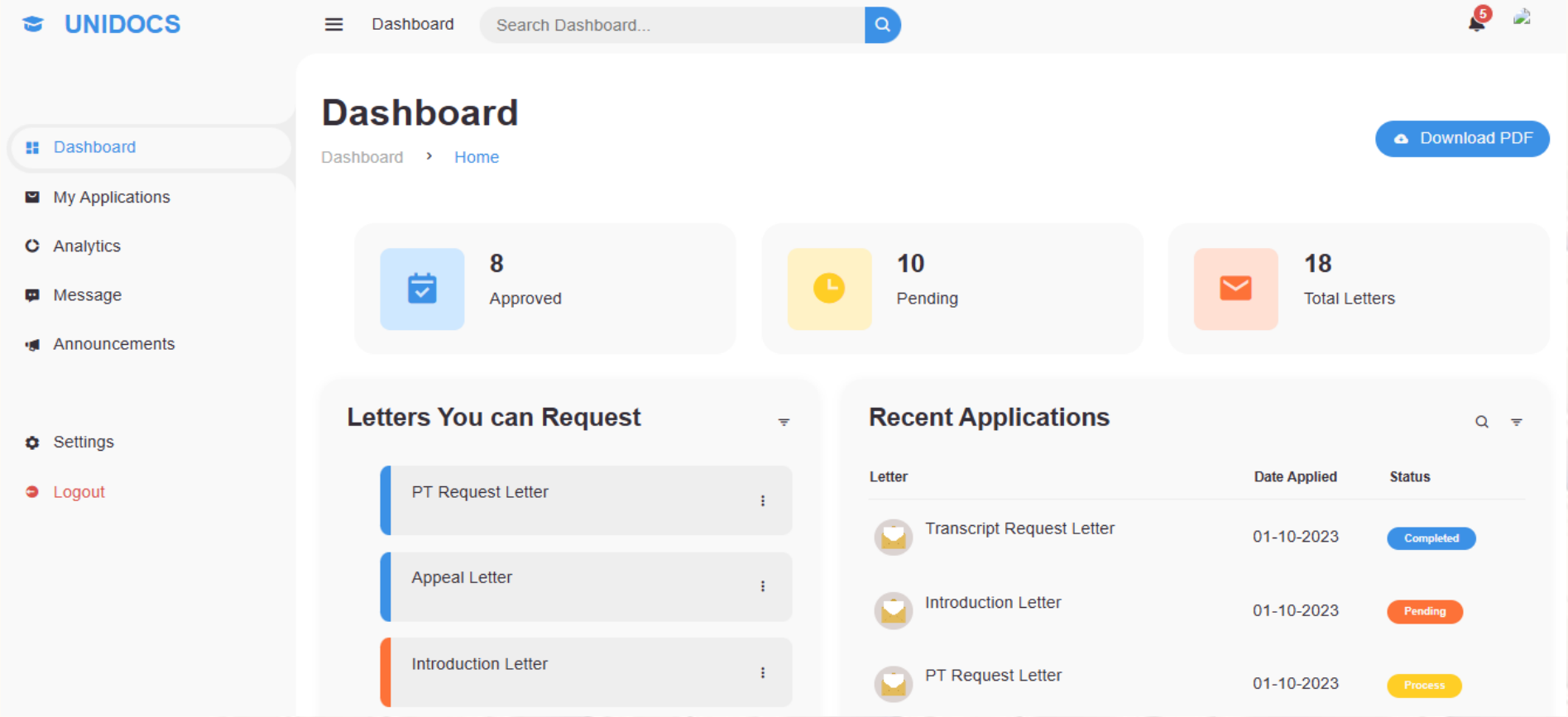
# Design Demonstration: System layout sample, Report samples & simulations



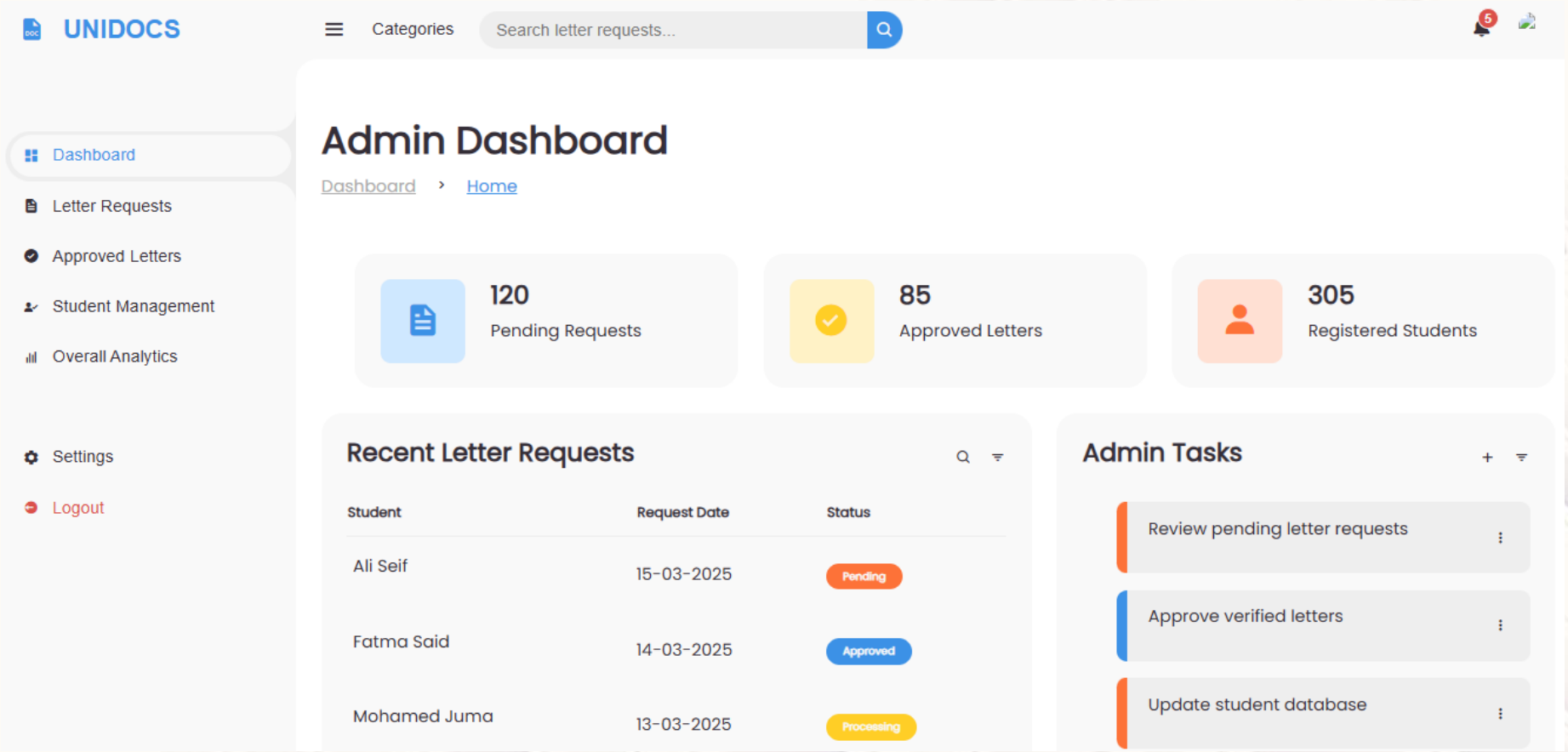
# Design Demonstration: System layout sample, Report samples & simulations



# Design Demonstration: System layout sample, Report samples & simulations



# Design Demonstration: System layout sample, Report samples & simulations



# THE END