

# **AUTOMATED ACADEMIC QUERY MANAGEMENT SYSTEM USING RULE BASED SYSTEM**

**Aishwarya R      (22Z431)**

**19Z013- OPEN SOURCE SYSTEMS**

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING**

**BRANCH: COMPUTER SCIENCE AND ENGINEERING**

Of Anna University



**October 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PSG COLLEGE OF TECHNOLOGY**

**(Autonomous Institution)**

**COIMBATORE – 641 004**

# **CONTENTS**

<b>TABLE OF CONTENT</b>	<b>PAGE</b>
<b>SYNOPSIS</b>	<b>2</b>
<b>1. INTRODUCTION</b>	<b>3</b>
<b>2. REQUIREMENT ANALYSIS</b>	<b>5</b>
<b>3. SYSTEM ANALYSIS</b>	<b>10</b>
<b>4. SYSTEM DESIGN</b>	<b>16</b>
<b>5. IMPLEMENTATION</b>	<b>19</b>
<b>6. RESULTS</b>	<b>22</b>
<b>7. CONCLUSION</b>	<b>26</b>
<b>APPENDIX</b>	<b>27</b>

## **SYNOPSIS**

The Automated Academic Query Management System is designed to streamline and enhance the management of academic queries within educational institutions. The system leverages Java Swing for its graphical user interface, allowing users to interact with the application through an intuitive and user-friendly interface. The Java Database Connectivity (JDBC) connector is utilized to facilitate seamless interaction with the underlying database, ensuring efficient data storage, retrieval, and manipulation. Central to this system is the integration of Drools, a powerful business rule management system (BRMS) that provides automated reasoning capabilities. Drools enables the application to process complex academic queries based on predefined rules and conditions, allowing for intelligent responses and recommendations. This capability not only improves the accuracy of the query handling but also reduces the response time, enhancing the overall user experience.

Key features of the system include:

1. User Authentication
2. Query Submission
3. Automated Response Generation
4. Database Management
5. Reporting and Analytics

This project aims to reduce the administrative burden on academic staff, promote self-service for students, and enhance the efficiency of academic query resolution. By incorporating modern technologies such as Java Swing, JDBC, and Drools, the Automated Academic Query Management System stands to significantly improve the academic experience for all stakeholders involved.

# CHAPTER 1

## INTRODUCTION

### 1.1 Automated Academic Query Management System

In the rapidly evolving landscape of educational technology, the Automated Academic Query Management System stands out as an innovative initiative aimed at streamlining the management of academic queries within educational institutions. This project represents a significant advancement in the realm of academic administration, offering a centralized solution designed to enhance user engagement and efficiency. By utilizing modern technologies such as Java Swing for the user interface, JDBC for database connectivity, and Drools for automated reasoning, the system seeks to create a reliable platform that empowers students and faculty to resolve academic queries seamlessly. The Automated Academic Query Management System signifies a critical step towards a future where academic information is accessible and efficiently managed, prioritizing user satisfaction and operational excellence.

### 1.2 Motivation

In today's digital age, the need for efficient query management in academic settings is more critical than ever. The motivation behind the Automated Academic Query Management System stems from the increasing volume of academic queries and the challenges faced by students and faculty in accessing timely and accurate information. Traditional methods of query resolution often fall short in responsiveness and efficiency, leading to frustration and confusion.

The primary goal of this project is to empower users by providing a secure, user-friendly system for submitting and resolving academic queries. By revitalizing the query management process through the integration of technologies like JDBC and Drools, the system aims to enhance user confidence in accessing academic information. Ultimately, the motivation is to leverage cutting-edge technology to facilitate better communication within educational institutions, ensuring that users can efficiently manage their academic inquiries and receive timely responses.

### 1.3 Problem Statement

The Automated Academic Query Management System aims to address the critical challenges surrounding the management of academic queries, particularly regarding response time and accuracy. Current systems often lack the capability to efficiently handle the growing number of queries, resulting in delays and miscommunications. Students and faculty frequently face difficulties in finding the necessary information, which can hinder their academic progress and engagement. To combat these challenges, the project introduces a comprehensive solution that

integrates Java Swing for the user interface, JDBC for seamless database interaction, and Drools for automated reasoning. This system is designed to provide users with an intuitive platform for submitting queries, ensuring that they can easily access the information they need. By implementing advanced technologies, the project aims to enhance the overall experience in academic query management, promoting a more responsive and user-centric approach.

## 1.4 Objectives

The objectives of the Automated Academic Query Management System are outlined as follows:

1. **Develop an Efficient Query Management System:** The primary aim is to create a centralized platform for managing academic queries, enhancing response times and accuracy.
2. **Integrate Automated Reasoning:** The project seeks to implement Drools to facilitate intelligent processing of academic queries, ensuring users receive accurate responses.
3. **Enhance User Experience:** Through the development of a user-friendly interface using Java Swing, the objective is to simplify the query submission and resolution process.
4. **Utilize Robust Database Management:** By leveraging JDBC, the project aims to ensure efficient handling of academic data, allowing for quick retrieval and updates.
5. **Empower Users:** The goal is to equip students and faculty with the tools necessary to effectively manage their academic inquiries, promoting confidence in the system.

## 1.5 Scope

The scope of the Automated Academic Query Management System encompasses the comprehensive development of a user-friendly platform for managing academic queries. This initiative aims to address the current shortcomings in academic query resolution by implementing innovative solutions that prioritize efficiency and accessibility. Central to the project's scope is the integration of Java Swing for the user interface, JDBC for database management, and Drools for automated reasoning. This combination ensures that users can seamlessly interact with the system while receiving timely and accurate information regarding their academic queries. Furthermore, the scope extends to the development of a responsive and intuitive interface that facilitates easy navigation and interaction with the query management system. The goal is to create an environment where users can quickly find the information they need and receive prompt responses to their inquiries. Ultimately, the project aims to establish a secure and efficient platform for academic query management that empowers users and enhances their overall experience. By addressing the critical challenges of query resolution, the Automated Academic Query Management System aspires to set new standards for efficiency and user satisfaction in educational institutions.

# CHAPTER 2

## REQUIREMENT ANALYSIS

### 2.1 Functional Requirements

#### User Registration and Login

**Description:** Users should be able to register by providing essential details such as name, email, and password. After registration, they can log in to access the system for managing academic queries.

**Actors:** Users (Students, Faculty, Admins).

**Precondition:** Users must have valid credentials to log in.

**Postcondition:** Successfully authenticated users can access their dashboard, while admins can view user submissions and manage the system.

#### Query Submission

**Description:** Users should be able to submit academic queries through a user-friendly interface, attaching any relevant documents if needed. Each submission will be stored in the system for tracking and management.

**Actors:** Users (Students, Faculty).

**Precondition:** Users must be logged in.

**Postcondition:** Users can view their submitted queries, which will be marked as pending for review.

#### Admin Query Review

**Description:** Admins should be able to view submitted queries, respond to them, and either resolve or escalate the query based on its nature.

**Actors:** Admins.

**Precondition:** Admins must have access to the admin panel.

**Postcondition:** Admins can respond to queries, and users will be notified of the resolution or escalation.

#### Response Storage

**Description:** All responses from admins to user queries will be stored in a centralized database for future reference and auditing.

**Actors:** Admins, Users.

**Precondition:** A database must be set up for storing queries and responses.

**Postcondition:** Responses are securely stored and accessible for review by both admins and users.

## 2.2 Non-Functional Requirements

- **User Authentication:** The system should implement secure login mechanisms to prevent unauthorized access to user accounts.
- **Data Integrity:** All submitted queries and responses must be stored securely to ensure data integrity and avoid loss or corruption.
- **Scalability:** The system must be able to scale to handle an increasing number of users and queries without performance degradation.
- **Response Time:** The system should provide a responsive user experience, with minimal lag during query submission and response retrieval.
- **User Interface:** The user interface must be intuitive and user-friendly, making it easy for users to register, log in, submit queries, and track responses.
- **Reliability:** The system must maintain high reliability, with minimal downtime, to ensure users can continuously submit and manage queries.
- **Privacy:** Personal user information, including credentials and submitted queries, must be handled securely and protected from unauthorized access.

## 2.3 Hardware Requirements

**Minimum:** 8 GB RAM, 2 vCPUs, 100 GB SSD (for hosting the backend and database).

**Recommended:** 16 GB RAM, 4 vCPUs, 250 GB SSD.

**User Machines:** Any device with modern web browsers (e.g., Google Chrome, Firefox, or Edge) is sufficient for accessing the platform.

**Stable Internet Connection:** Required for both users and admins.

## 2.4 Software Requirements

- **Operating System:** Windows, Linux, or macOS for hosting the backend and database.
- **Web Framework-** Java with Spring Boot (for backend development) or other Java frameworks as needed.
- **Database-** MySQL or MongoDB (for storing user queries and responses).
- **Version Control-** Git (for code versioning and collaboration).

## 2.5 Technology Stack

**Frontend:**

1. **Swing (JFrame):**

- Used for creating the Graphical User Interface (GUI) of the desktop application.
- Provides windows, forms, buttons, labels, and other components for user interactions like submitting queries, logging in, etc.

## **2. HTML/CSS for Rich Text Styling (embedded in JFrames):**

- Basic HTML and CSS can be embedded within `JFrame` components for formatting and styling of messages or labels.
- Example: Styling a success message with `<html><body>` tags inside a `JLabel` or `JTextPane`.

## **3. Java (Swing):**

- Java's Swing library is the primary toolkit for building desktop applications with forms, buttons, and text inputs.
- `JFrame`, `JLabel`, `JButton`, `JTextField`, and `JTextArea` are used to capture user input and display responses.

## **Backend:**

### **1. Java (Core Java):**

- The business logic and rule processing for the system are written in Java.
- User queries and system responses are processed using Java classes, handling the data flow and logic between the frontend (JFrame UI) and the database.

### **2. Java (Spring Boot / Java EE for API):**

- In an alternative approach using web technologies, Spring Boot or Java EE could be utilized to build RESTful APIs for backend logic and HTTP request handling if you migrate to web-based UI in the future.
- Servlets could be used if sticking to Java EE for handling HTTP requests and user interactions with the backend.

## **Database:**

### **1. MySQL:**

- Relational Database Management System used to store user data, queries, and responses.
- Tables might include `users` (for authentication), `queries` (for storing user queries), and possibly `rules` (to store automated reasoning rules).



- JDBC (Java Database Connectivity) is used to connect to the MySQL database and execute SQL queries (insert, update, select).

## **2. MongoDB (Optional for NoSQL):**

- In case of scalability or handling unstructured data, you can use MongoDB to store user submissions and system configurations. It works well if the system needs flexibility in schema and handling large amounts of data.

## **Automated Reasoning / Rule-Based System:**

### **1. Drools:**

- Drools is used as a rule engine for automated decision-making based on predefined rules.
- Example: Assigning queries to faculty based on expertise, workload, and other constraints.
- Drools executes rules to automate and prioritize responses to user queries.

## **Integration & Security:**

### **1. JWT (JSON Web Token) for User Authentication (if web-based):**

- In a web-based solution, JWT can be used for user authentication, ensuring secure sessions and interactions between the frontend and backend.
- RESTful API Integration:
  - i. The backend (Java) must provide RESTful APIs to handle data exchange between the frontend (if using HTML/JS-based UI) and the MySQL database.
  - ii. API endpoints handle query submissions, faculty assignments, and user authentication.

## **How These Components Work Together**

**Frames:** Acts as the frontend, allowing users (students, faculty, or administrators) to interact with the system.

- Students can submit queries.
- Admins can configure faculty availability or workload.
- Faculty can view and respond to queries.

**MySQL:** Stores data related to queries, users, and system configurations.

- Query data is persisted in the database.

- Rules for the reasoning engine can also be stored in the database for easy configuration.

**Automated Reasoning Engine:** Handles decision-making.

- When a query is submitted, the engine checks the rules and constraints (e.g., faculty expertise, availability) to assign the query to the best-fit faculty.
- Could also be used to prioritize or automate responses based on past data.

# CHAPTER 3

## SYSTEM ANALYSIS

### System Architecture:

This chapter presents a detailed analysis of the system requirements for the Automated Academic Query System. The analysis encompasses hardware and software requirements, environment setup, and the functional and non-functional aspects of the system.

### 3.1 Environment Setup

The environment setup for the Automated Academic Query System involves the following steps:

1. **Install Java Development Kit (JDK):** Ensure you have the appropriate version of JDK installed for Java development.
  - Run the installer and follow the prompts.
  - Set the **JAVA\_HOME** environment variable to the JDK installation directory.
2. Download the **JDBC connector** and add the mysql-connector-java-X.X.X.jar to it in the project repository.
3. **Database Setup:** Install MySQL or MongoDB and configure the database for user queries and login. Open mysql workbench and add table queries  
  
create database academicq; use academicq; Create table login
4. **Install Development Tools:**
  - IDE (e.g., IntelliJ IDEA or Eclipse) for Java development.
  - Git for version control.
  - Or VS code- and download all the extensions required
5. **Download Maven and Extract the ZIP file:**
  - Visit the official Maven website: <https://maven.apache.org/download.cgi>.
  - Under **Files**, download the **Binary zip archive**  
(e.g., **apache-maven-3.8.8-bin.zip**).
  - Extract the downloaded ZIP file to a folder where you want to install Maven.

**Set Up Maven Environment Variables- Set **MAVEN\_HOME** Variable:**

- In the **Edit Environment Variable** window, click **New**, and add the Maven **bin** directory:

### Verify Maven Installation

- Press **Windows + R**, type **cmd**, and hit **Enter**. **mvn -v**

```
C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9\bin>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfc97d260186937)
Maven home: C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9
Java version: 19.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-19
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9\bin>mvn archetype:generate -DgroupId=com.academic.querysystem -DartifactId=academic-query-system -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.2.0/maven-clean-plugin-3.2.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.2.0/maven-clean-plugin-3.2.0.pom (5.3 kB at 5.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/35/maven-plugins-35.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/35/maven-plugins-35.pom (9.9 kB at 165 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/35/maven-parent-35.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/35/maven-parent-35.pom (45 kB at 476 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/25/apache-25.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/25/apache-25.pom (21 kB at 467 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-
```

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X
B at 46 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/luben/zstd-jni/1.5.6-3/zstd-jni-1.5.6-3.jar (6.7 MB at 610 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/ibm/icu/icu4j/75.1/icu4j-75.1.jar (14 MB at 983 kB/s)
[INFO] Generating project in Batch mode
Downloading from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml
Downloaded from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml (16 MB at 2.3 MB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (4.3 kB at 20 kB/s)
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9\bin
[INFO] Parameter: package, Value: com.academic.querysystem
[INFO] Parameter: groupId, Value: com.academic.querysystem
[INFO] Parameter: artifactId, Value: academic-query-system
[INFO] Parameter: packageName, Value: com.academic.querysystem
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9\bin\academic-query-system
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 36.918 s
[INFO] Finished at: 2024-10-16T22:21:55+05:30
[INFO] -----

C:\Users\RAMESH\Desktop\academicq\src\apache-maven-3.9.9\bin>
```

## 6. Integrate Maven into Java Project- Create a New Maven Project:

- Open the Command Prompt or IDE terminal.

Maven command to create a new Maven project:

```
mvn archetype:generate -DgroupId=com.academic.querysystem
-DartifactId=academic-query-system
-DarchetypeArtifactId=maven-archetype-quickstart
-DinteractiveMode=false
```

7. This will generate a Maven project structure. Navigate to Project Directory and Open the **pom.xml**: You will now see the generated file in the root directory.

## 8. Choose an Automated Reasoning Library:

Drools: A popular business rule management system (BRMS) that can be used for automated reasoning.

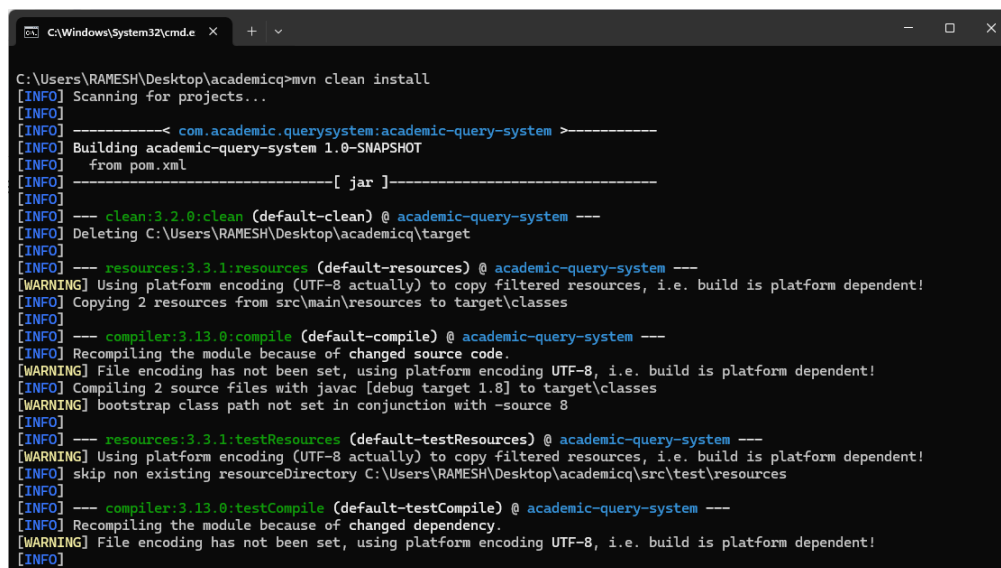
- Download
- Installation: include the required **JAR files** in the project.
- Add the drool dependencies in the **pom.xml** file.

9. **Create Rules (.drl) file**: queries.drl Define rules for query response automation.

## 10. Configure **kmodule.xml** (for Rule Session Configuration)

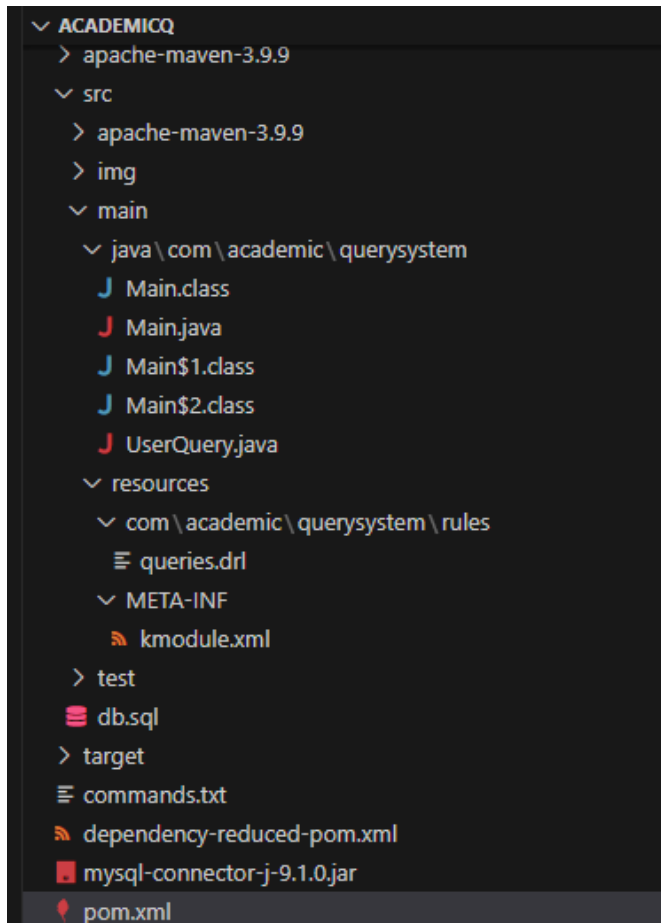
Define Drools session configuration in a **kmodule.xml** file. This file should be placed in the **resources** folder.

## 11. Build and Run the Project



```
C:\Windows\System32\cmd.exe
C:\Users\RAMESH\Desktop\academicq>mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.academic.querysystem:academic-query-system >-----
[INFO] Building academic-query-system 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ academic-query-system ---
[INFO] Deleting C:\Users\RAMESH\Desktop\academicq\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ academic-query-system ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 2 resources from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ academic-query-system ---
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 2 source files with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ academic-query-system ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\RAMESH\Desktop\academicq\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ academic-query-system ---
[INFO] Recompiling the module because of changed dependency.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO]
```

## Project Directory structure



## 3.2 Dataset Considerations

The dataset considerations for the Automated Academic Query System are crucial for ensuring effective query handling and response generation. Utilizing rule-based reasoning through a system like Drools can significantly enhance the performance and accuracy of query responses. Below are the key aspects of dataset considerations for the system:

### Source of Data

- **User Data:**
  - Information collected during user registration, such as username and password which are stored in a secure database. And finally used for authorization of users.
- **Query Data:**
  - User-generated queries that are submitted to the system, encompassing various academic topics and questions. That is kept track of and is used for generating new rules maximizing the system performance.

- **Response Data:**
  - Predefined responses or knowledge base articles that can be matched with user queries for quick resolution.

### 3.3 Rule-Based Reasoning with Drools

- **Knowledge Base Integration:**
  - Drools can be used to create a knowledge base that encapsulates academic rules, best practices, and response templates. This allows the system to infer answers based on predefined criteria and patterns observed in user queries.
- **Rule Definition:**
  - Rules can be defined to automate the response process, ensuring that the system can handle common queries efficiently without human intervention. For example:
    - If a user queries about exam schedules, the system retrieves the latest exam dates from the database.
    - If a query involves multiple subjects, the system can prioritize based on the user's declared major or interests.
- **Dynamic Updates:**
  - The rule set can be dynamically updated as new academic policies or guidelines are introduced, ensuring that the system remains current and responsive to changing user needs.

### 3.4 Advantages of Rule-Based Reasoning

- **Increased Accuracy:**
  - By utilizing a rule-based system, the likelihood of providing accurate responses to user queries increases. The system can evaluate conditions based on user input and apply relevant rules to derive conclusions.
- **Enhanced Efficiency:**
  - Automating response generation through predefined rules reduces the time required for manual review and intervention, leading to faster query resolution.
- **Improved User Experience:**
  - Users receive timely and contextually relevant responses, enhancing their overall experience with the system. This can lead to higher user satisfaction and increased engagement.
- **Scalability:**

- As the volume of queries increases, the rule-based system can easily scale to handle additional load without significant modifications. New rules can be added to the knowledge base to accommodate emerging topics or frequent queries.

### **3.5 Data Security**

- **Confidentiality and Compliance:**

- All user data must be handled in compliance with data protection regulations (e.g., GDPR). User queries and personal information should be encrypted and stored securely to prevent unauthorized access.

- **Data Validation:**

- Implement validation mechanisms to ensure that the data collected from users is accurate and in the expected format before processing or storing it.

Integrating a rule-based reasoning approach using Drools into the Automated Academic Query System provides a robust mechanism for handling user queries effectively. This not only streamlines the response process but also enhances the overall functionality and user satisfaction of the system.



# CHAPTER 4

## SYSTEM DESIGN

### System Architecture

Automated academic query system user flow

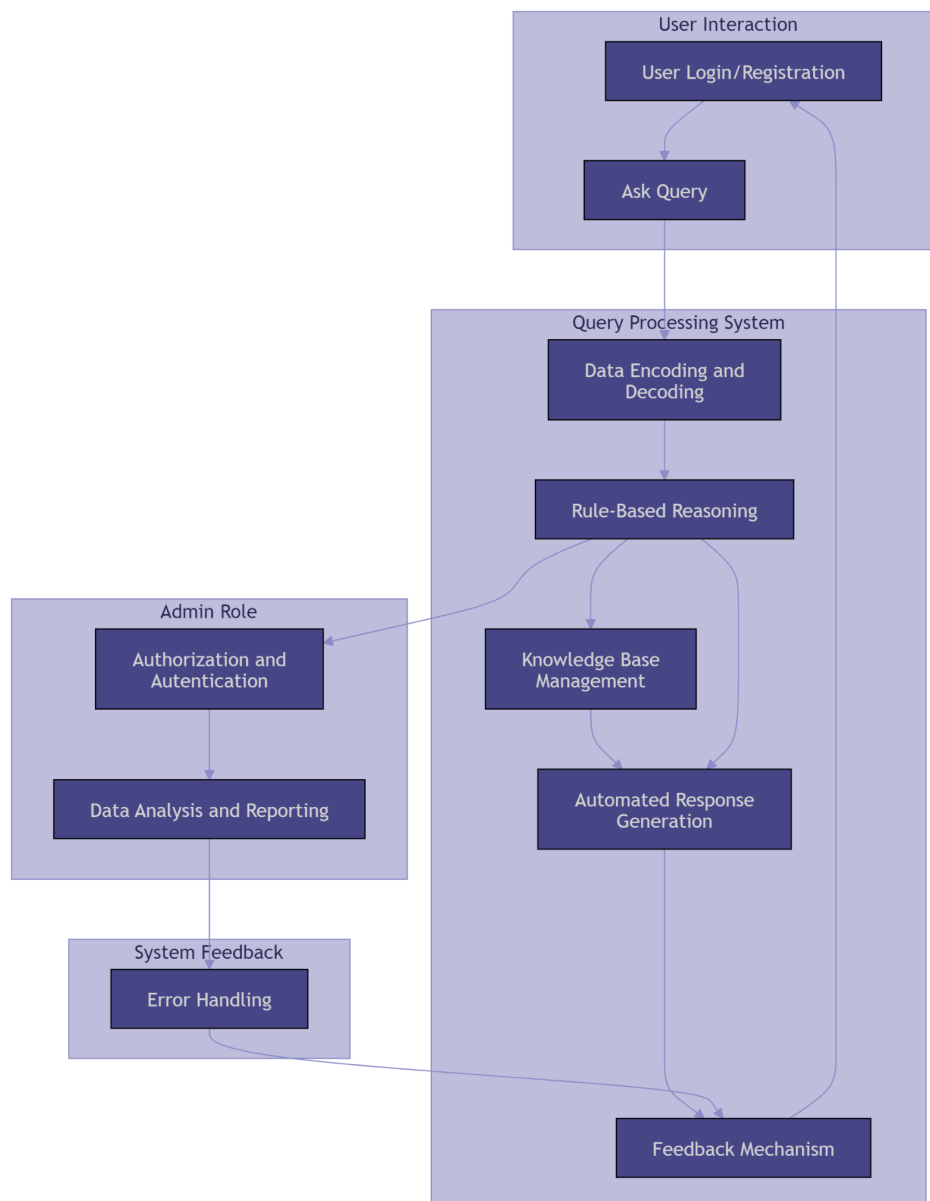


Figure 4.1 User Flow

The architecture described encompasses a user-friendly interface that allows users to log in or sign up for the Automated Academic Query System. Once authenticated, users can initiate queries or access previously answered questions. The heart of the system is the rule-based reasoning mechanism, which uses a knowledge base to efficiently process and respond to user queries. The system ensures data

integrity and security through user authentication, while the admin role focuses on maintaining the system's functionality and responsiveness. The feedback loop fosters continuous improvement, ensuring the system evolves to meet user needs effectively.

#### **User Interaction:**

1. **Login:** The flow begins with the user logging into the Automated Academic Query System using their credentials. Users can also register if they are new to the platform.
2. **Input Query Data:** After successful login, users can submit academic queries. They may also choose to browse previously answered queries for reference.

#### **Query Processing System:**

3. **Data Encoding and Decoding:** The system encodes user queries to ensure security and anonymity. When retrieving responses, it decodes the data for presentation while maintaining confidentiality.
4. **Rule-Based Reasoning:** The core of the system employs a rule-based reasoning engine, such as Drools, which interprets user queries and determines the most relevant responses based on predefined rules. This allows the system to provide accurate and context-aware answers.
5. **Knowledge Base Management:** The system utilizes a centralized knowledge base that contains a repository of common queries and their corresponding responses. The knowledge base is regularly updated to reflect new academic information and user interactions.
6. **Automated Response Generation:** When a query is submitted, the system checks the knowledge base for matching responses. If a direct match is found, the system generates an automated response. If not, it may escalate the query for manual review by administrators or academic advisors.
7. **Feedback Mechanism:** Users can provide feedback on the responses they receive. This feedback is captured and analyzed to improve the accuracy and efficiency of the rule-based reasoning engine, ensuring continuous improvement.

#### **Admin Role:**

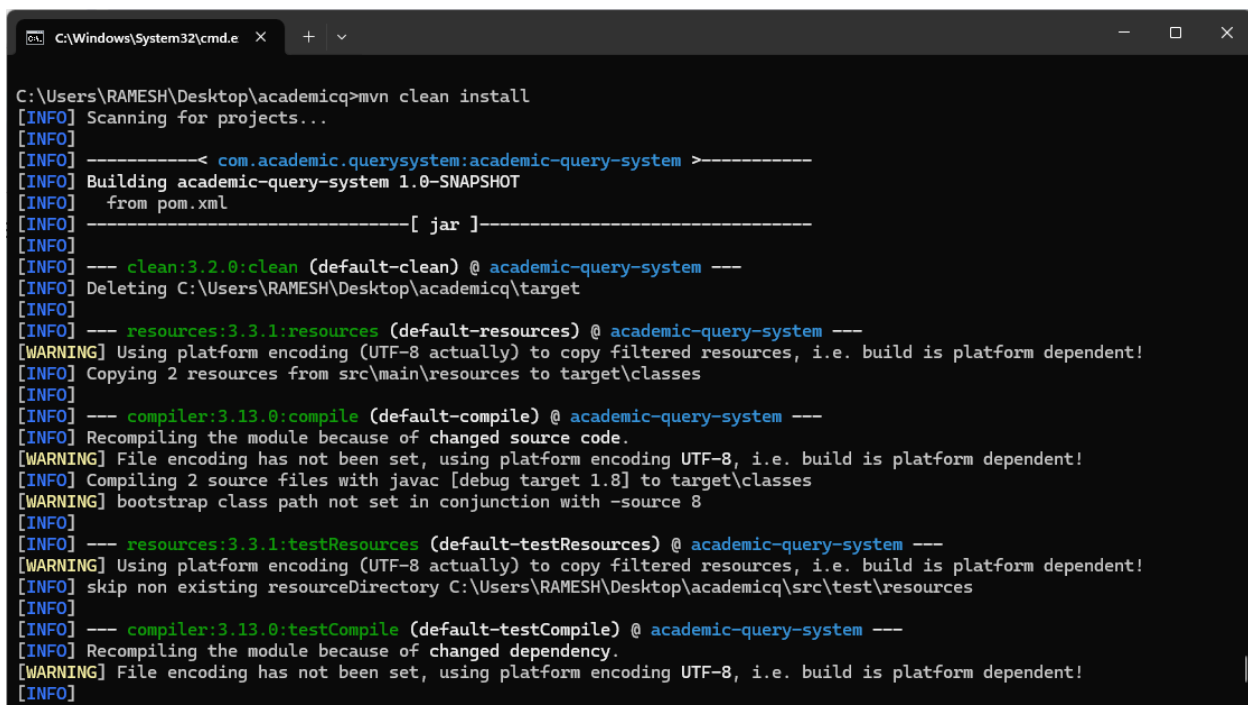
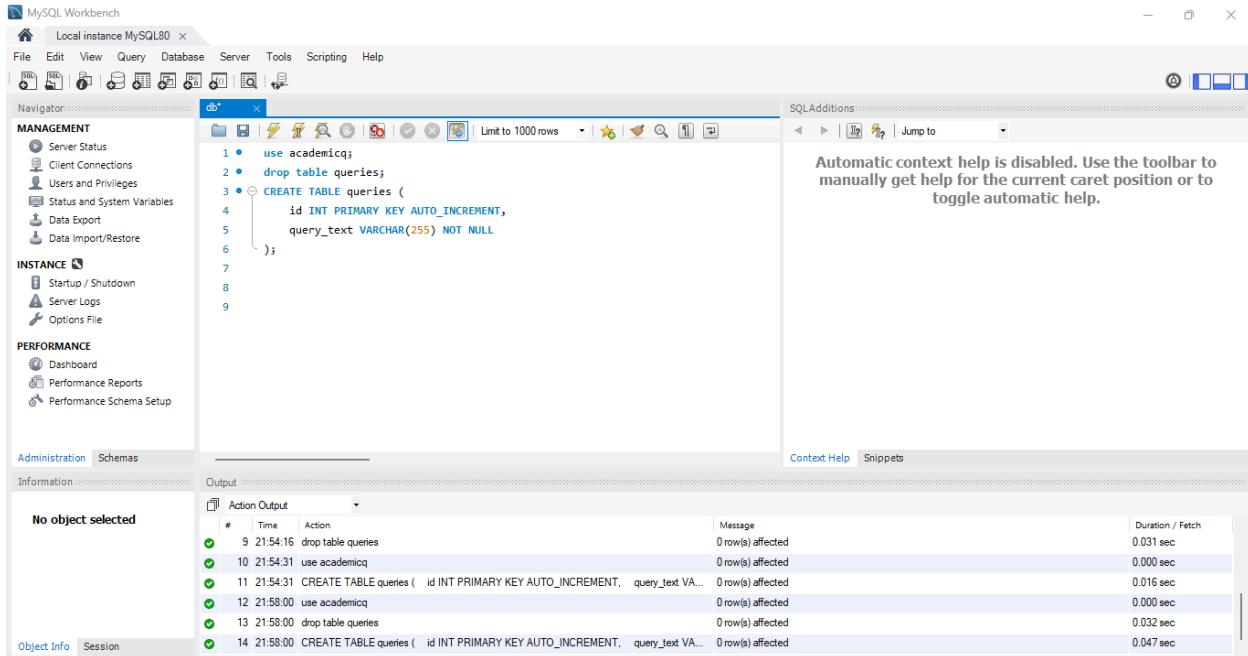
8. **Authentication and authorization:** Admins have the capability to monitor user interactions, query submissions, and response accuracy. They can update the knowledge base and modify the rules within the reasoning engine to enhance system performance and user experience.
9. **Data Analysis and Reporting:** The admin can access reports on system usage, common queries, and user feedback, which can be utilized for strategic decisions and system enhancements.

### **System Feedback:**

10. **Error Handling:** If users encounter issues while using the system, the feedback mechanism captures errors and sends them to the admin for resolution. This ensures that problems are addressed promptly, enhancing user satisfaction and system reliability.

# CHAPTER 5

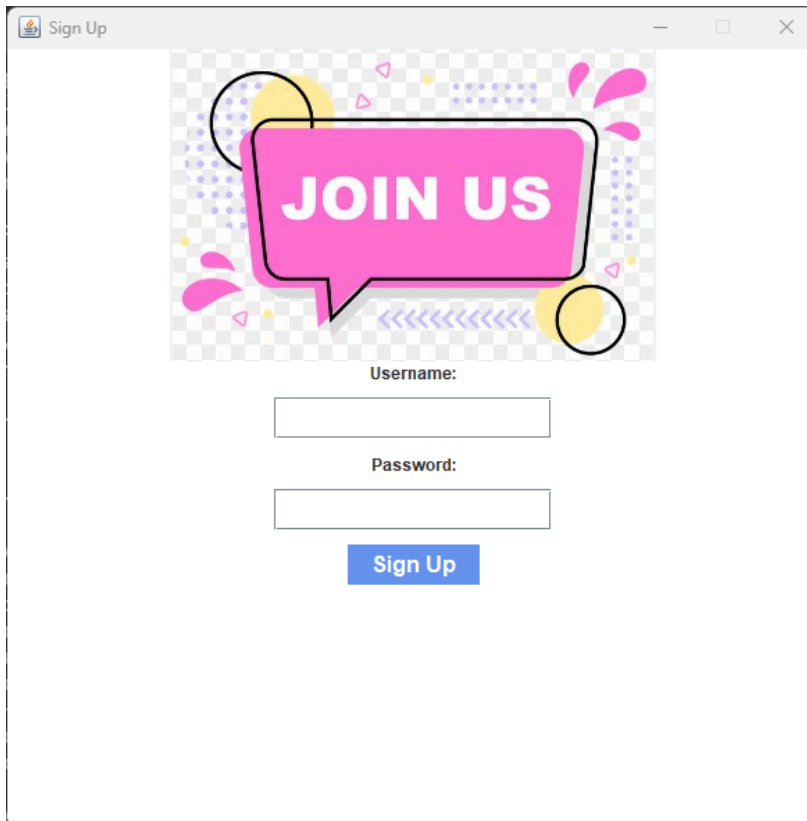
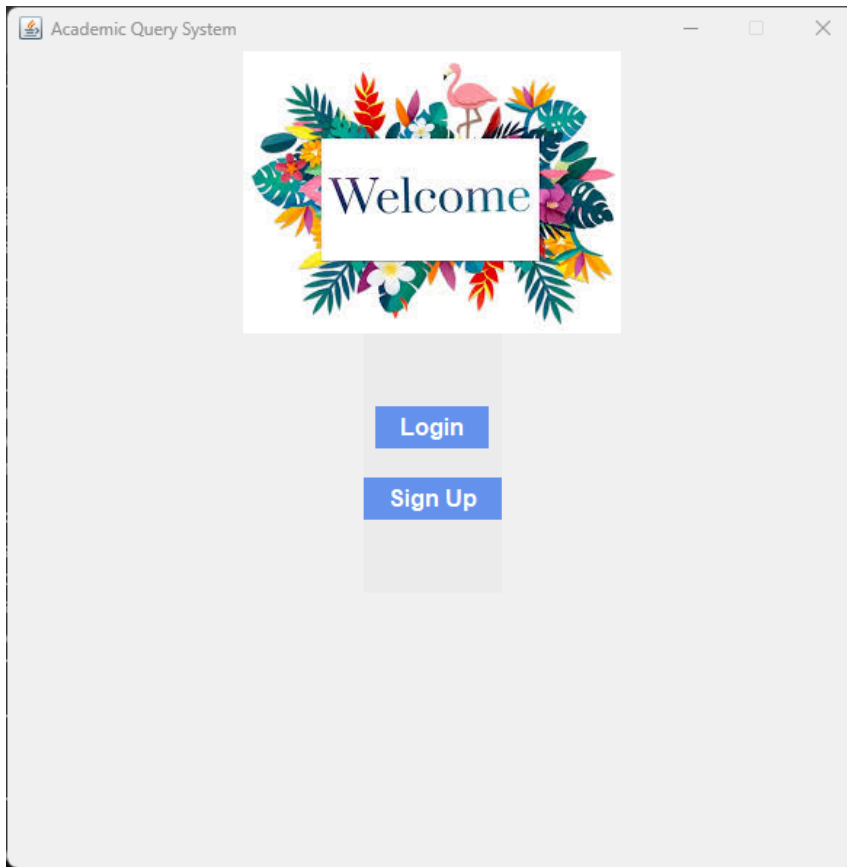
## IMPLEMENTATION



```
C:\Windows\System32\cmd.e X + v
[INFO] --- shade:3.2.4:shade (default) @ academic-query-system ---
[INFO] Including org.kie:kie-api:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie.soup:kie-soup-maven-support:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.slf4j:slf4j-api:jar:1.7.30 in the shaded jar.
[INFO] Including org.drools:drools-core:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie:kie-internal:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie.soup:kie-soup-xstream:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.drools:drools-core-reflective:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.drools:drools-core-dynamic:jar:7.66.0.Final in the shaded jar.
[INFO] Including commons-codec:commons-codec:jar:1.15 in the shaded jar.
[INFO] Including mysql:mysql-connector-j:jar:9.1.0 in the shaded jar.
[INFO] Including org.slf4j:slf4j-simple:jar:1.7.30 in the shaded jar.
[INFO] Including org.drools:drools-compiler:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie:kie-memory-compiler:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.drools:drools-ecj:jar:7.66.0.Final in the shaded jar.
[INFO] Including organtlr:antlr-runtime:jar:3.5.2 in the shaded jar.
[INFO] Including com.thoughtworks.xstream:xstream:jar:1.4.19 in the shaded jar.
[INFO] Including io.github.x-stream:mxparser:jar:1.2.2 in the shaded jar.
[INFO] Including xmlpull:xmlpull:jar:1.1.3.1 in the shaded jar.
[INFO] Including org.kie:kie-spring:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.jbpm:jbpm-flow:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.drools:drools-mvel:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.drools:drools-serialization-protobuf:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie.soup:kie-soup-commons:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie.soup:kie-soup-project-datamodel-commons:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie.soup:kie-soup-project-datamodel-api:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie:kie-dmn-api:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie:kie-dmn-feel:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.kie:kie-dmn-model:jar:7.66.0.Final in the shaded jar.
[INFO] Including org.antlr:antlr4-runtime:jar:4.9.2 in the shaded jar.
[WARNING] - META-INF/spring.tooling
[WARNING] spring-aop-5.3.14.jar, spring-beans-5.3.14.jar, spring-context-5.3.14.jar, spring-core-5.3.14.jar, spring-expression-5.3.14.jar, spring-jcl-5.3.14.jar, spring-tx-5.3.14.jar define 2 overlapping resources:
[WARNING] - META-INF/license.txt
[WARNING] - META-INF/notice.txt
[WARNING] maven-shade-plugin has detected that some class files are
[WARNING] present in two or more JARs. When this happens, only one
[WARNING] single version of the class is copied to the uber jar.
[WARNING] Usually this is not harmful and you can skip these warnings,
[WARNING] otherwise try to manually exclude artifacts based on
[WARNING] mvn dependency:tree -Ddetail=true and the above output.
[WARNING] See http://maven.apache.org/plugins/maven-shade-plugin/
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing C:\Users\RAMESH\Desktop\academicq\target\academic-query-system-1.0-SNAPSHOT.jar with C:\Users\RAMESH\Desktop\academicq\target\academic-query-system-1.0-SNAPSHOT-shaded.jar
[INFO] Dependency-reduced POM written at: C:\Users\RAMESH\Desktop\academicq\dependency-reduced-pom.xml
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ academic-query-system ---
[INFO] Installing C:\Users\RAMESH\Desktop\academicq\dependency-reduced-pom.xml to C:\Users\RAMESH\.m2\repository\com\academic\querysystem\academic-query-system\1.0-SNAPSHOT\academic-query-system-1.0-SNAPSHOT.pom
[INFO] Installing C:\Users\RAMESH\Desktop\academicq\target\academic-query-system-1.0-SNAPSHOT.jar to C:\Users\RAMESH\.m2\repository\com\academic\querysystem\academic-query-system\1.0-SNAPSHOT\academic-query-system-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 10.232 s
[INFO] Finished at: 2024-10-18T01:58:29+05:30
[INFO]
C:\Users\RAMESH\Desktop\academicq>
```

Run the .jar to open the application

```
C:\Users\RAMESH\Desktop\academicq>java -jar target\academic-query-system-1.0-SNAPSHOT.jar
```



The image shows a web browser window titled "Sign Up". The main content area features a large, colorful graphic with the text "JOIN US" in white, bold, uppercase letters inside a pink speech bubble. The background of the graphic is a checkered pattern with various colorful shapes and patterns. Below the graphic, there are two input fields: "Username:" and "Password:". Below the password field, there is a blue button with white text: "Sign Up".

Username:

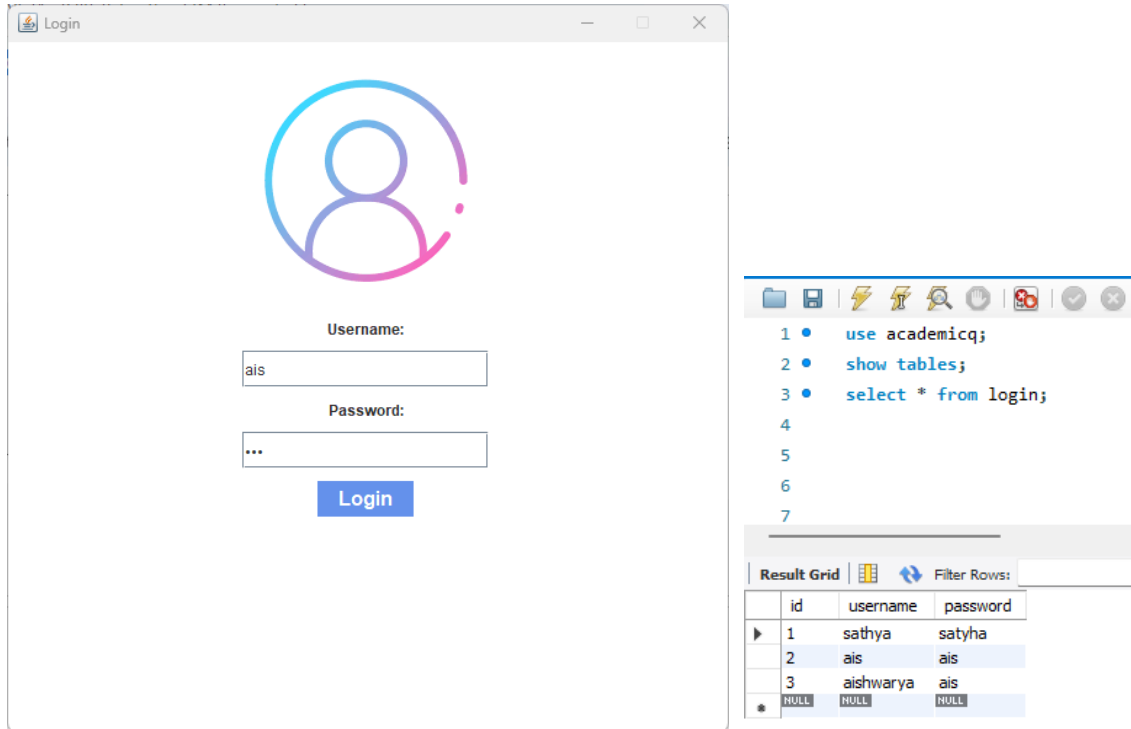
Password:

Sign Up

## CHAPTER 6

### RESULTS

#### 1. Login using credentials



Username:

ais

Password:

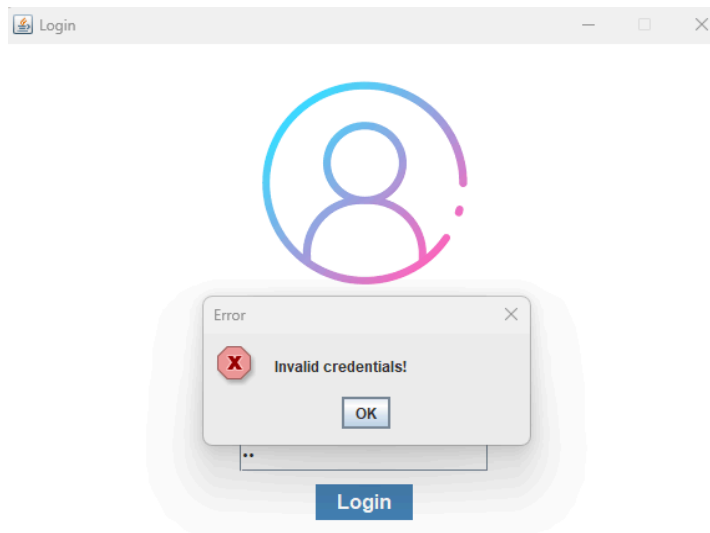
...

Login

```
1 • use academicq;
2 • show tables;
3 • select * from login;
4
5
6
7
```

	id	username	password
▶	1	sathya	satya
	2	ais	ais
	3	aishwarya	ais
*	NULL	NULL	NULL

#### 2. If invalid credentials



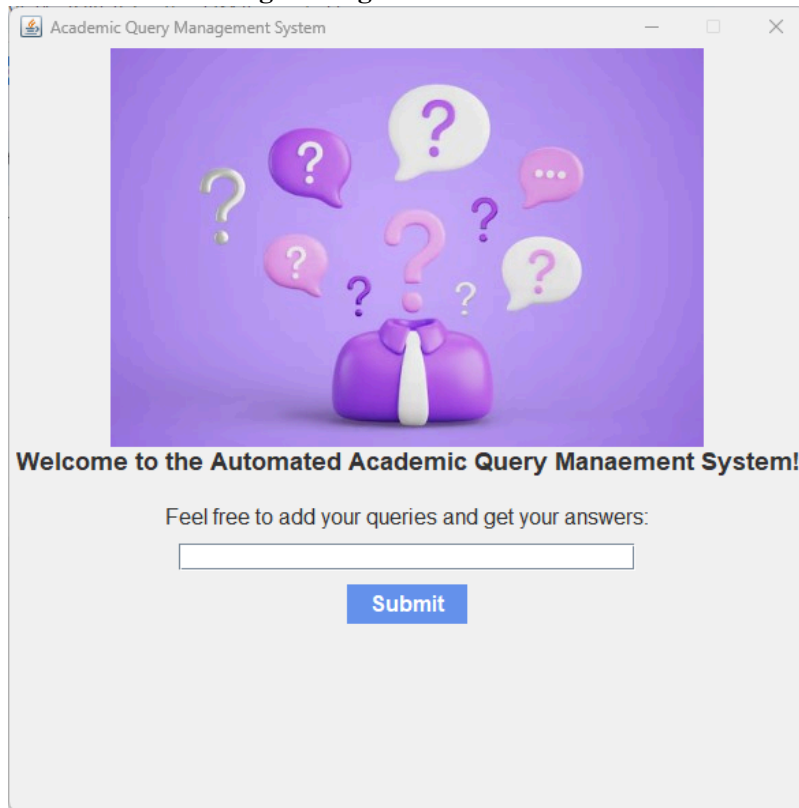
Error

Invalid credentials!

OK

Login

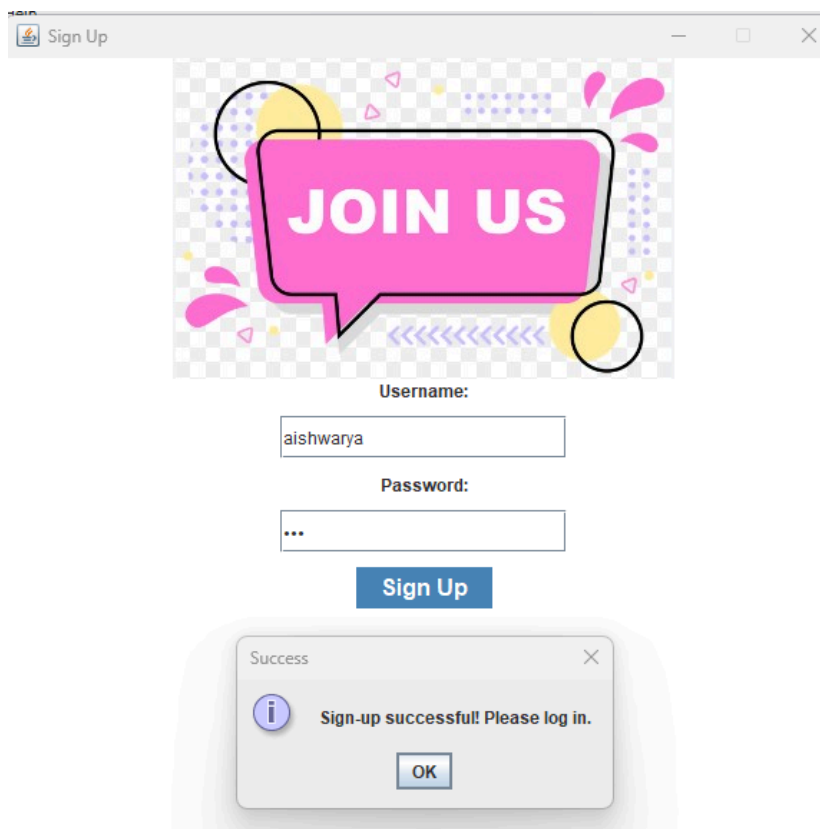
### 3. Successful login using credentials



The screenshot shows a web browser window titled "Academic Query Management System". The main content area has a purple background with a 3D illustration of a purple question mark character wearing a white tie, surrounded by several floating question marks in various colors. Below the illustration, the text "Welcome to the Automated Academic Query Manaement System!" is displayed. Underneath, it says "Feel free to add your queries and get your answers:" followed by a text input field and a blue "Submit" button.

### 4. Sign up

If successful sign up redirected to login page



The screenshot shows a web browser window titled "Sign Up". The main content area features a colorful graphic with a pink speech bubble saying "JOIN US" on a checkered background. Below the graphic, there are two input fields: "Username:" with the text "aishwarya" and "Password:" with masked characters "...". A blue "Sign Up" button is positioned below the password field. At the bottom of the window, a "Success" dialog box is open, displaying an information icon and the message "Sign-up successful! Please log in." with an "OK" button.



## 5. Add a query- Store in mysql

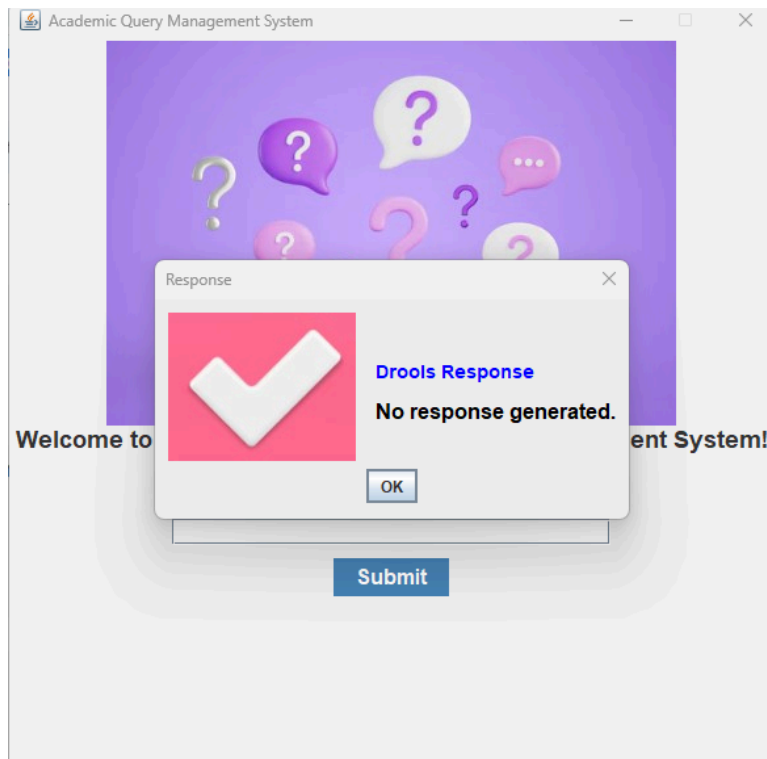
The screenshot displays the 'Academic Query Management System' window. A 'Success' dialog box is centered, featuring a large white checkmark on a pink background and the text 'Success Query submitted successfully!' with an 'OK' button. Below the dialog, a text input field contains the word 'library', and a blue 'Submit' button is positioned below it. The background of the main window shows a purple gradient with several question marks. At the bottom, a SQL query editor shows the following code:

```
1 • use academicq;  
2 • show tables;  
3 • select * from queries;  
4  
5  
6  
7
```

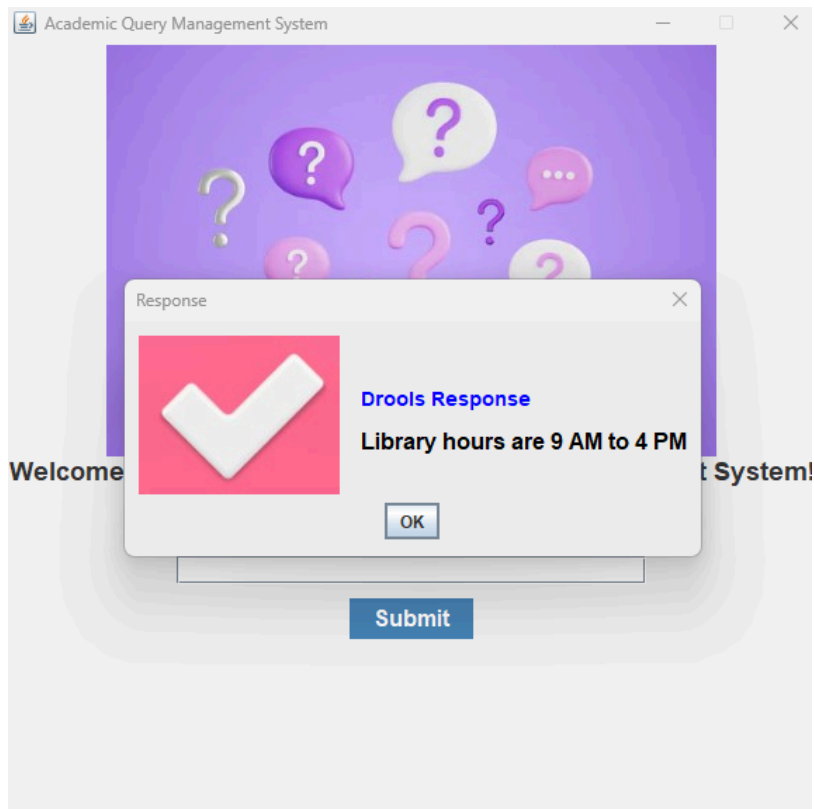
Below the editor is a 'Result Grid' section with a table containing two rows:

	id	query_text
▶	1	When is the next exam?
	2	Library hours

## 6. Response- If not in predefined rules set



## 7. If present



## **CHAPTER 7**

### **CONCLUSION**

The Automated Academic Query Management System has been successfully developed using Java Swing for the user interface and JDBC for database connectivity, ensuring smooth and efficient interaction between users and the system. The core functionality of the system leverages Drools, a rule-based reasoning engine, to automate the processing of academic queries, providing fast and accurate responses based on predefined rules. This implementation enhances query handling by reducing the need for manual intervention, enabling a more responsive and scalable solution.

The system's integration of a rule-based reasoning approach ensures that users receive contextually relevant answers while maintaining flexibility for future updates. The use of Java Swing provides a clean and intuitive interface, ensuring a user-friendly experience for both students and administrators. Additionally, JDBC allows for robust database operations, facilitating seamless query storage, retrieval, and processing.

With features like automated response generation, a feedback mechanism, and admin management, the system is well-equipped to handle a wide range of academic queries efficiently. The combination of these technologies ensures a reliable and scalable platform that can easily adapt to evolving academic needs. This successful implementation demonstrates the power of integrating rule-based systems into academic environments, improving both user satisfaction and system performance.

## **APPENDIX**

### **queries.drl**

```
import com.academic.querysystem.UserQuery;
rule "Check Library Hours"
when
    q : UserQuery(queryText matches ".*[Ll]ibrary.*[Hh]ours.*")
then
    q.setResponse("Library hours are 9 AM to 4 PM");
end
rule "Check Course Availability"
when
    q : UserQuery(queryText matches ".*[Cc]ourse.*[Aa]vailability.*")
then
    q.setResponse("Available courses include: Course A, Course B, Course C.");
end
rule "Check Exam Schedule"
when
    q : UserQuery(queryText matches ".*[Ee]xam.*[Ss]chedule.*")
then
    q.setResponse("The next exams are scheduled for next week on Wednesday and Friday. Dates
willbe announced soon.");
end
rule "Check Assignment Deadlines"
when
    q : UserQuery(queryText matches ".*[Aa]ssignment.*[Dd]eadline.*")
then
    q.setResponse("The upcoming assignment deadlines are: Assignment 1 - October 25,
Assignment 2 - November 1.");
end
rule "Check Registration Dates"
when
    q : UserQuery(queryText matches ".*[Rr]egistration.*[Dd]ates.*")
then
    q.setResponse("Registration for the next semester opens on November 1 and closes on
November 30.");
end
rule "Check Library Services"
when
    q : UserQuery(queryText matches ".*[Ll]ibrary.*[Ss]ervices.*")
then
    q.setResponse("Library services include book lending, research assistance, and study
rooms.");
end
```

### **Main.java**

```
<?xml version="1.0" encoding="UTF-8"?>
<kmodule xmlns="http://www.drools.org/xsd/kmodule">
    <kbase name="kbase" packages="com.academic.querysystem.rules">
        <ksession name="ksession-rules" default="true"/>
    </kbase>
```

```

</kmodule>
package com.academic.querysystem;
import javax.swing.*;
import org.kie.api.KieServices;
import org.kie.api.runtime.KieContainer;
import org.kie.api.runtime.KieSession;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class Main {
    // MySQL connection setup
    private static Connection connect() {
        Connection connection = null;
        try {
            String url = "jdbc:mysql://localhost:3306/academicq";
            String user = "root";
            String password = "Root@123";
            connection = DriverManager.getConnection(url, user, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Main::showOptionPage);
    }

    // Method to show option page for Login or Sign Up
    public static void showOptionPage() {
        JFrame frame = createFrame("Academic Query System", new Color(240, 240, 240));
        ImageIcon welcomeImage = new
        ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\welcome.jpg"); // Replace
        with the correct path
        JLabel imageLabel = new JLabel(welcomeImage);
        imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align the image
        frame.add(imageLabel); // Light gray background

        JButton loginButton = createButton("Login");
        JButton signupButton = createButton("Sign Up");

        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS)); // Vertical box layout
        panel.add(Box.createRigidArea(new Dimension(0, 50))); // Space at the top
        panel.add(loginButton);
        panel.add(Box.createRigidArea(new Dimension(0, 20))); // Space between buttons
        panel.add(signupButton);
        panel.add(Box.createRigidArea(new Dimension(0, 50))); // Space at the bottom

        frame.add(panel);
    }
}

```

```

        frame.setVisible(true);
        loginButton.addActionListener(e -> {
            frame.dispose();
            loginPage(); // Open login page
        });
        signupButton.addActionListener(e -> {
            frame.dispose();
            signUpPage(); // Open sign-up page
        });
    }
    // Method for login page with reduced text box size
    public static void loginPage() {
        JFrame frame = createFrame("Login", Color.WHITE); // White background
        ImageIcon loginImage = new
        ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\login.jpg"); // Replace with
        the correct path
        JLabel imageLabel = new JLabel(loginImage);
        imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align the
        image
        frame.add(imageLabel)
        JLabel userLabel = new JLabel("Username:");
        JTextField userField = new JTextField(20);
        userField.setMaximumSize(new Dimension(200, 30)); // Reduced size
        JLabel passLabel = new JLabel("Password:");
        JPasswordField passField = new JPasswordField(20);
        passField.setMaximumSize(new Dimension(200, 30)); // Reduced size
        JButton loginButton = createButton("Login");
        addComponentsToFrame(frame, userLabel, userField, passLabel, passField, loginButton);
        loginButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());

            if (validateLogin(username, password)) {
                frame.dispose(); // Close the login window
                queryPage(); // Open the query submission page
            } else {
                JOptionPane.showMessageDialog(frame, "Invalid credentials!", "Error",
                JOptionPane.ERROR_MESSAGE);
            }
        });

        frame.setVisible(true);
    }

    // Method for signup page
    public static void signUpPage() {
        JFrame frame = createFrame("Sign Up", Color.WHITE); // White background

```

```

        ImageIcon sImage = new
ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\signup.jpg"); // Replace with
the correct path
        JLabel imageLabel = new JLabel(sImage);
        imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align the
image
        frame.add(imageLabel);

        JLabel userLabel = new JLabel("Username:");
        JTextField userField = new JTextField(20);
        userField.setMaximumSize(new Dimension(200, 30)); // Reduced size
        JLabel passLabel = new JLabel("Password:");
        JPasswordField passField = new JPasswordField(20);
        passField.setMaximumSize(new Dimension(200, 30)); // Reduced size
        JButton signUpButton = createButton("Sign Up");

        addComponentsToFrame(frame, userLabel, userField, passLabel, passField, signUpButton);

        signUpButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());

            if (createNewUser(username, password)) {
                JOptionPane.showMessageDialog(frame, "Sign-up successful! Please log in.",
"Success", JOptionPane.INFORMATION_MESSAGE);
                frame.dispose();
                loginPage(); // Redirect to login page after successful sign-up
            } else {
                JOptionPane.showMessageDialog(frame, "Username already exists!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        frame.setVisible(true);
    }

    // Method to validate login credentials from MySQL database
    public static boolean validateLogin(String username, String password) {
        boolean valid = false;
        try (Connection connection = connect()) {
            if (connection != null) {
                String sql = "SELECT * FROM login WHERE username = ? AND password = ?";
                PreparedStatement statement = connection.prepareStatement(sql);
                statement.setString(1, username);
                statement.setString(2, password);

                ResultSet resultSet = statement.executeQuery();
                valid = resultSet.next(); // If a record exists, login is valid
            }
        }
    }

```

```

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return valid;
}

// Method to create a new user in the database
public static boolean createUser(String username, String password) {
    boolean created = false;
    try (Connection connection = connect()) {
        if (connection != null) {
            String sql = "INSERT INTO login (username, password) VALUES (?, ?)";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, username);
            statement.setString(2, password);

            statement.executeUpdate();
            created = true; // User created successfully
        }
    } catch (SQLException e) {
        if (e.getErrorCode() == 1062) { // Duplicate entry error (username exists)
            System.out.println("Username already exists!");
        } else {
            e.printStackTrace();
        }
    }
    return created;
}

// Query submission page
public static void queryPage() {
    JFrame frame = createFrame("Academic Query Management System", new Color(240, 240,
240)); // Light gray background

    ImageIcon qImage = new
ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\q.jpg"); // Replace with the
correct path
    JLabel imageLabel = new JLabel(qImage);
    imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align the
image
    frame.add(imageLabel);

    // Welcome message
    JLabel welcomeLabel = new JLabel("Welcome to the Automated Academic Query
Manaement System!");
    welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));
    welcomeLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    frame.add(welcomeLabel);
    frame.add(Box.createRigidArea(new Dimension(0, 20))); // Space below welcome message

```



```

JLabel label = new JLabel("Feel free to add your queries and get your answers:");
label.setFont(new Font("Arial", Font.PLAIN, 16));
label.setAlignmentX(Component.CENTER_ALIGNMENT);
frame.add(label);
frame.add(Box.createRigidArea(new Dimension(0, 10))); // Space below label

JTextField queryField = new JTextField(30); // Consistent textbox size
queryField.setMaximumSize(queryField.getPreferredSize()); // Maintain size
frame.add(queryField);
frame.add(Box.createRigidArea(new Dimension(0, 10))); // Space below text box

JButton submitButton = createButton("Submit");
frame.add(submitButton);
frame.add(Box.createRigidArea(new Dimension(0, 20))); // Space at the bottom

submitButton.addActionListener(e -> {
    String query = queryField.getText().trim(); // Remove leading/trailing whitespace

    // Validate query input
    if (query.isEmpty()) {
        JOptionPane.showMessageDialog(frame, "Please enter a query!", "Empty Query",
JOptionPane.WARNING_MESSAGE);
        return; // Exit if query is empty
    }

    submitQuery(query, frame);
    queryField.setText(""); // Clear the input field
    triggerDroolsResponse(query, frame); // Trigger Drools for reasoning
});

frame.setVisible(true);
}

// Method to insert query into the database
private static void submitQuery(String queryText, JFrame frame) {
    UserQuery query = new UserQuery(queryText); // Create a Query object
    try (Connection connection = connect()) {
        if (connection != null) {
            String sql = "INSERT INTO queries (query_text) VALUES (?)";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, query.getQueryText());

            statement.executeUpdate();
            // Custom success icon
            ImageIcon successIcon = new
ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\response.jpg");

            // HTML styled success message
            String successMessage = "<html><body style='font-family:sans-serif; font-size:12px;'>"
+ "<h2 style='color:green;'>Success</h2>"

```

```

+ "<p style='color:black;'>Query submitted successfully!</p>"
+ "</body></html>";

// Display the dialog
JOptionPane.showMessageDialog(
    frame,
    successMessage,
    "Success",
    JOptionPane.INFORMATION_MESSAGE,
    successIcon
);

    }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(frame, "Failed to submit the query!", "Error",
JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    }
}

// Method to trigger Drools for reasoning
private static void triggerDroolsResponse(String query, JFrame frame) {
    KieServices kieServices = KieServices.Factory.get();
    KieContainer kieContainer = kieServices.getKieClasspathContainer();
    KieSession kieSession = kieContainer.newKieSession("ksession-rules");

    try {
        UserQuery userQuery = new UserQuery(query); // Create a new UserQuery object
        kieSession.insert(userQuery); // Insert the UserQuery object into the KieSession
        kieSession.fireAllRules(); // Execute the rules

        // After firing rules, retrieve the response
        String response = userQuery.getResponse();
        if (response == null) {
            response = "No response generated."; // Fallback message if no response was set
        }
        // Custom icon
        ImageIcon icon = new
        ImageIcon("C:\\Users\\RAMESH\\Desktop\\academicq\\src\\img\\response.jpg");

        // HTML styled message
        String styledMessage = "<html><body style='font-family:sans-serif; font-size:12px;'>"
            + "<h3 style='color:blue;'>Drools Response</h3>"
            + "<p style='color:black;'>" + response + "</p>"
            + "</body></html>";

        // Display the dialog
        JOptionPane.showMessageDialog(
            frame,
            styledMessage,

```

```

        "Response",
        JOptionPane.INFORMATION_MESSAGE,
        icon
    );

    } finally {
        kieSession.dispose(); // Clean up KieSession
    }
}

// Method to create a frame with a specified title and background color
private static JFrame createFrame(String title, Color backgroundColor) {
    JFrame frame = new JFrame(title);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(600, 600);
    frame.setResizable(false);
    frame.getContentPane().setBackground(backgroundColor);
    frame.setLayout(new BorderLayout(frame.getContentPane(), BorderLayout.Y_AXIS)); //
Vertical box layout
    frame.setLocationRelativeTo(null); // Center on screen
    return frame;
}

// Method to create a styled button
private static JButton createButton(String text) {
    JButton button = new JButton(text);
    button.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align button
    button.setPreferredSize(new Dimension(150, 40));
    button.setFont(new Font("Arial", Font.BOLD, 16));
    button.setBackground(new Color(100, 149, 237)); // Cornflower blue background
    button.setForeground(Color.WHITE); // White text color
    button.setFocusPainted(false); // Remove focus outline
    button.setBorderPainted(false); // Remove border
    button.setOpaque(true);

    button.addMouseListener(new MouseAdapter() {
        public void mouseEntered(MouseEvent e) {
            button.setBackground(new Color(70, 130, 180)); // Darker blue on hover
        }
        public void mouseExited(MouseEvent e) {
            button.setBackground(new Color(100, 149, 237)); // Revert to original color
        }
    });

    return button;
}

// Helper method to add multiple components to a frame
private static void addComponentsToFrame(JFrame frame, JComponent... components) {
    for (JComponent component : components) {

```

```

        component.setAlignmentX(Component.CENTER_ALIGNMENT); // Center align
components
        frame.add(component);
        frame.add(Box.createRigidArea(new Dimension(0, 10))); // Space between components
    }
}
}

```

### **UserQuery.java**

```

package com.academic.querysystem;

public class UserQuery {
    private String queryText;
    private String response;

    public UserQuery(String queryText) {
        this.queryText = queryText;
    }

    public String getQueryText() { // Ensure this method exists
        return queryText;
    }

    public String getResponse() { // Ensure this method exists
        return response;
    }

    public void setResponse(String response) { // Ensure this method exists
        this.response = response;
    }
}

```

### **Pom.xml**

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.academic.querysystem</groupId>
    <artifactId>academic-query-system</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>academic-query-system</name>
    <url>http://maven.apache.org</url>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>

```

```

    <artifactId>maven-shade-plugin</artifactId>
    <version>3.2.4</version>
    <executions>
      <execution>
        <phase>package</phase>
        <goals>
          <goal>shade</goal>
        </goals>
        <configuration>
          <artifactSet>
            <excludes>
              <exclude>org.apache.maven.plugins:maven-jar-plugin</exclude> <!--
Prevents conflict -->
            </excludes>
          </artifactSet>
          <transformers>
            <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
              <mainClass>com.academic.querysystem.Main</mainClass>
            </transformer>
          </transformers>
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>

<dependencies>
  <!-- Drools Core -->
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>7.66.0.Final</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-core</artifactId>
    <version>7.66.0.Final</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>9.1.0</version> <!-- Correct version for MySQL -->
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.7.30</version> <!-- or another compatible version -->
  </dependency>

```

```
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-compiler</artifactId>
  <version>7.66.0.Final</version>
</dependency>
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-spring</artifactId>
  <version>7.66.0.Final</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.8.1</version>
  <scope>test</scope>
</dependency>
</dependencies>

</project>
```