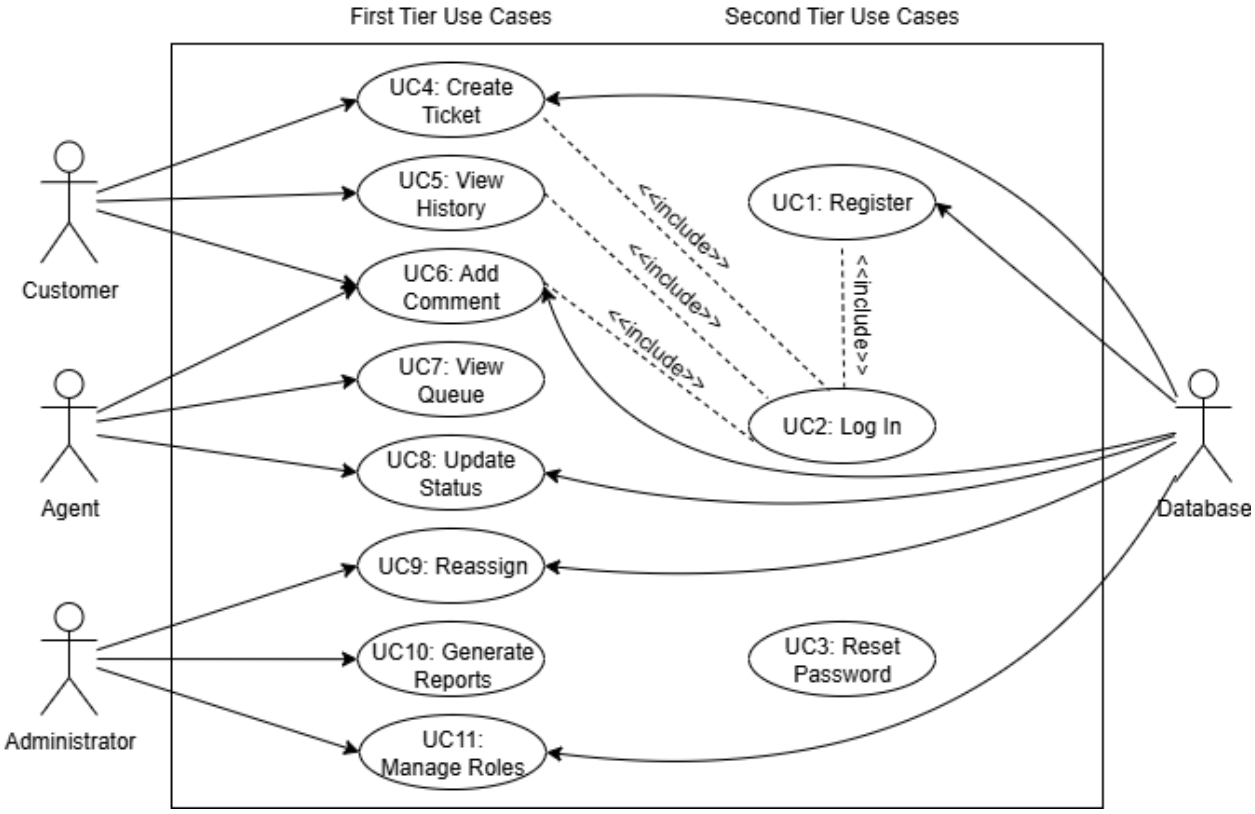
Technical Documentation

3.3.2 Use Case Diagram



3.4 System Sequence Diagrams

A visual representation is helpful to understand what information enters and exits the system in each use case. For the above three significant use cases, a system sequence diagram is given below to serve this purpose.

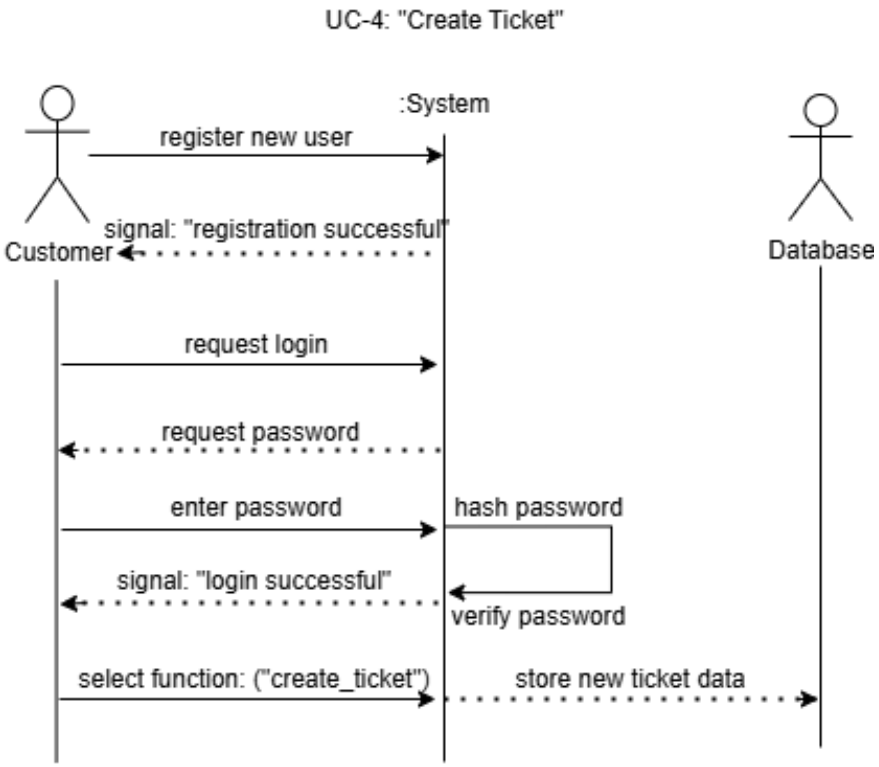


Figure 7.1: Sequence Diagram – Submit Ticket

Description:

This diagram shows how the user submits a ticket and how the system handles it. The user fills out a form, and the TicketController checks if the data is valid. If everything is okay, it saves the ticket in the database and creates a ticket ID. The system then sends a notification to the user that the ticket was created and updates the reports before showing a confirmation message.

Design Principle Used:

This diagram follows the idea of **Single Responsibility**, meaning each part of the system does one main job. It also keeps **low coupling**, so every controller works on its own task without depending too much on others. This makes the system easier to update or fix later.

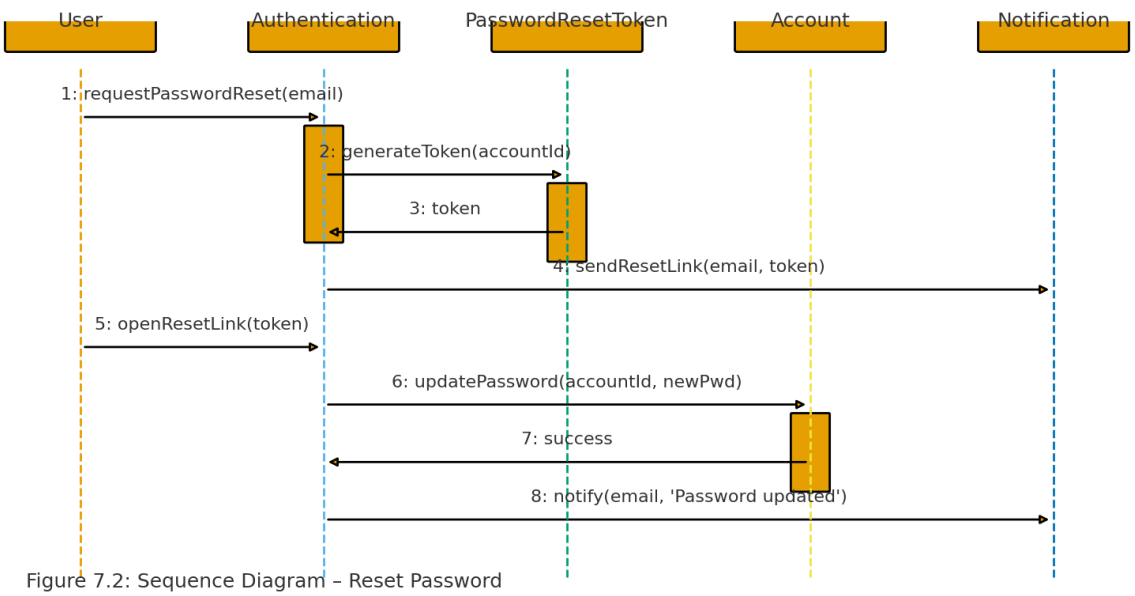


Figure 7.2: Sequence Diagram – Reset Password

Figure 7.2: Sequence Diagram – Password Reset

Description:

This diagram explains what happens when a user resets their password. The user asks for a password reset, and the system checks their account and makes a reset token. The token is sent to the user's email. When the user opens the link, they can enter a new password, and the system updates it and sends a message saying the password was successfully changed.

Design Principle Used:

This diagram uses **Separation of Concerns**, where each object (like Authentication, Token, or Notification) has its own job. It also uses the **Information Expert** idea, so the object that has the needed information (like the token or account) is the one that handles it.

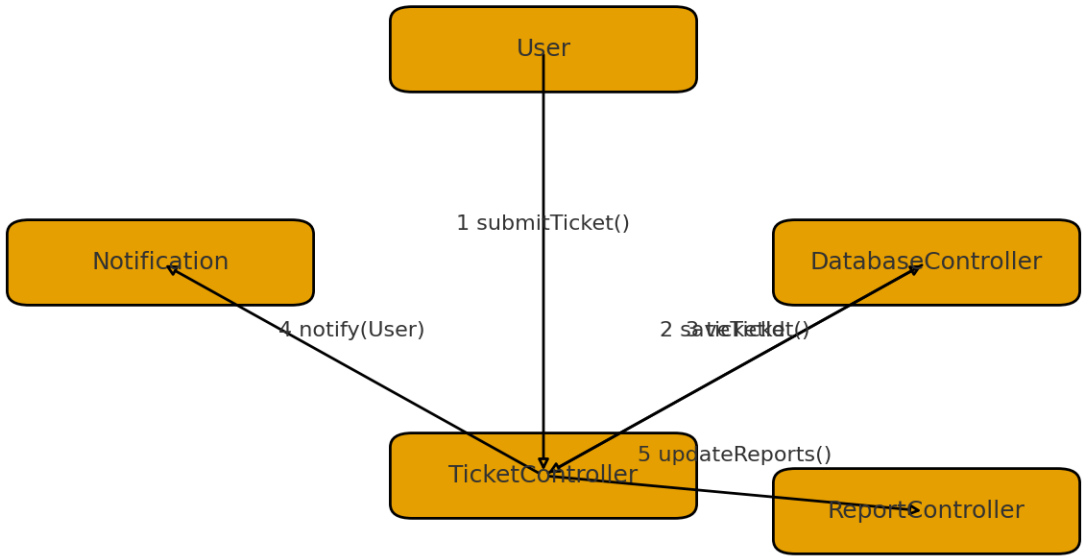
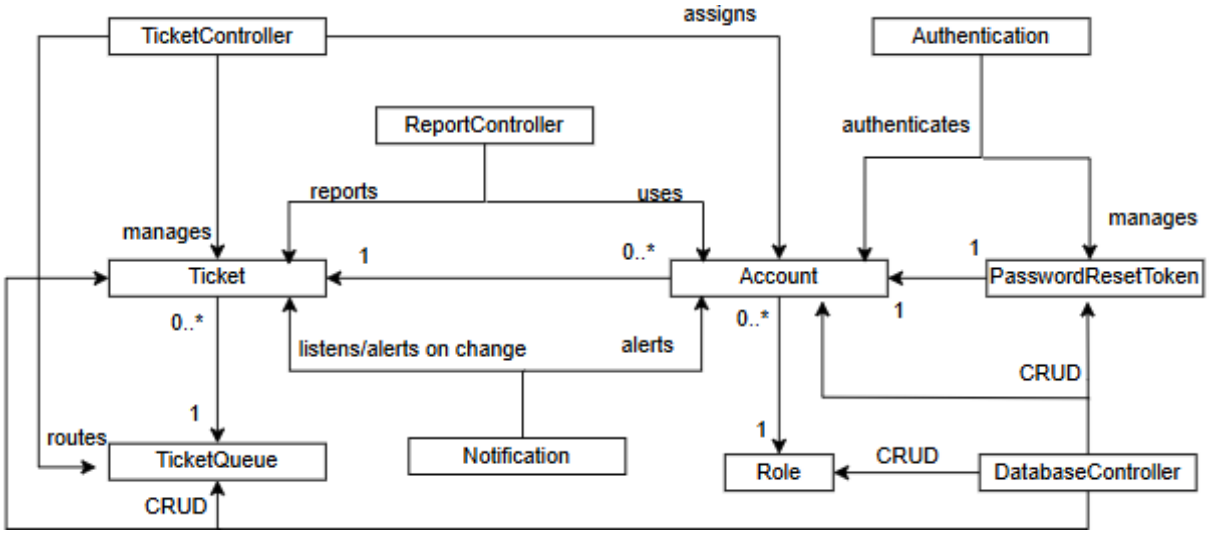


Figure 7.3: Communication Diagram – Submit Ticket

8.1 Class Diagram



Database Schema