Ticket selling system

Analysis and Design Document

Student: Budea Patrick

**Group: 30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

[Application description]

The application represents a ticket managing system which could be used for a variety of ways, such as a music festival. The application is more of an API, since it lacks a dedicated user interface, something which can be further implemented.

# Functional Requirements

*[Present the functional requirements]*

As functional requirements, the system has to allow 2 types of users (admin and cashiers), which can apply CRUD specific operations on a series of entities. The system should also allow the selling of tickets, by a cashier. The user credentials should also be encrypted. Data persistence is made using an SQL based system.

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

As non-functional requirements, the system should handle multiple instances for multiple users, as long as the instances are on different ports. Since the application is using a minimal set of dependencies, the memory footprint is relatively small.

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

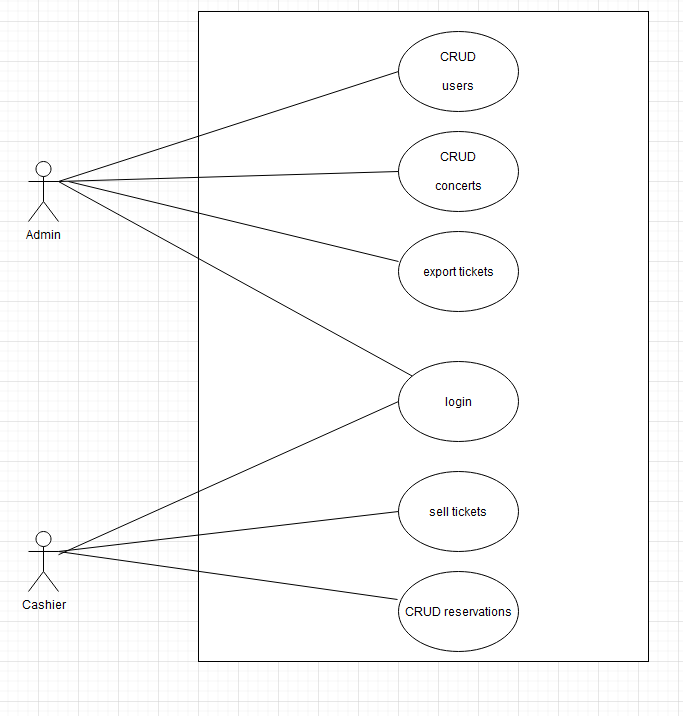
*Use case: <use case goal>*

*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]* 

Use case: sell a ticket to a show

Level: ?

Primary actor: cashier

Main success scenario: the cashier introduces their credentials. If a successful login is achieved, the cashier should be able to select a concert and also assign the number of people on that ticket (could be more than one).

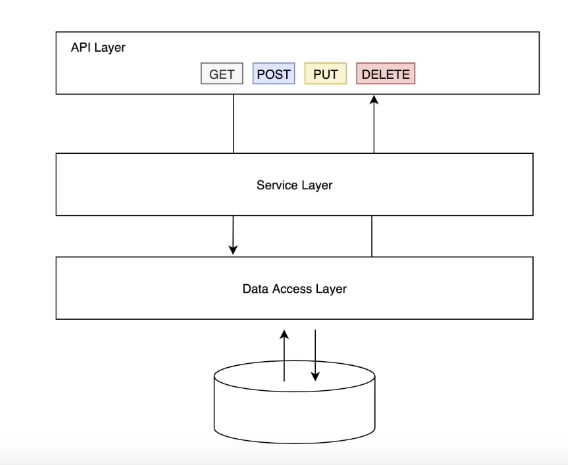
Extensions: the system will prompt the user that their credentials are wrong, or that the number of tickets available do not suffice for the current purchase.

3. System Architectural Design

**3.1 Architectural Pattern Description**

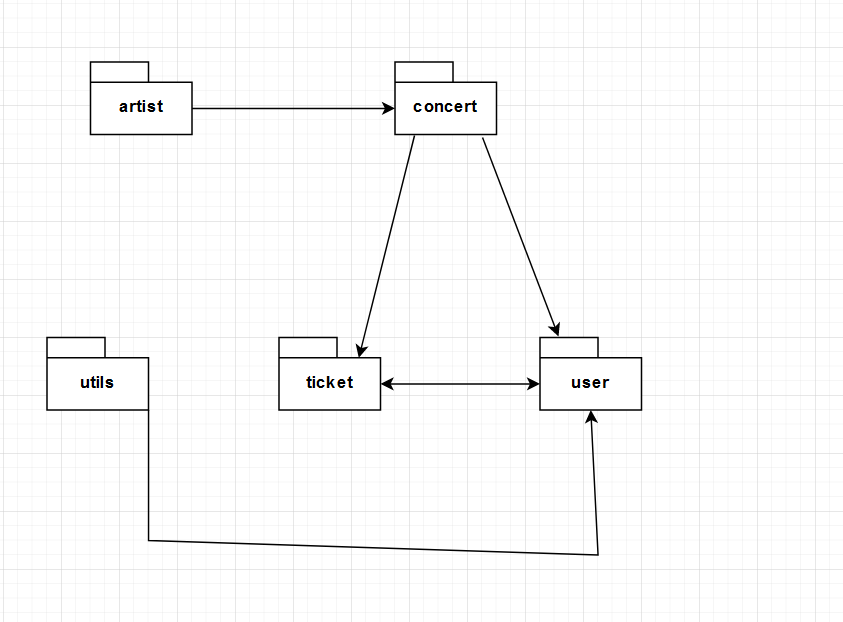
*[Describe briefly the used architectural patterns.]*

The architectural pattern employed is the MVC architecture. The system lacks a user interface, but the object modelling and the request controller is present, allowing for API calls on endpoints, specific to each operation.



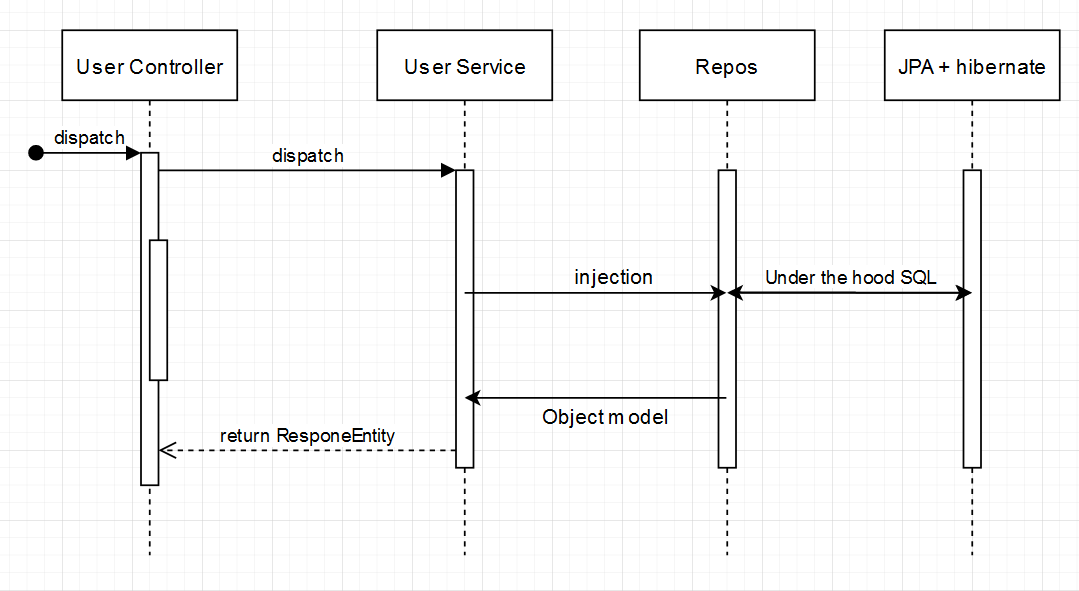
**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*



4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*



5. Class Design

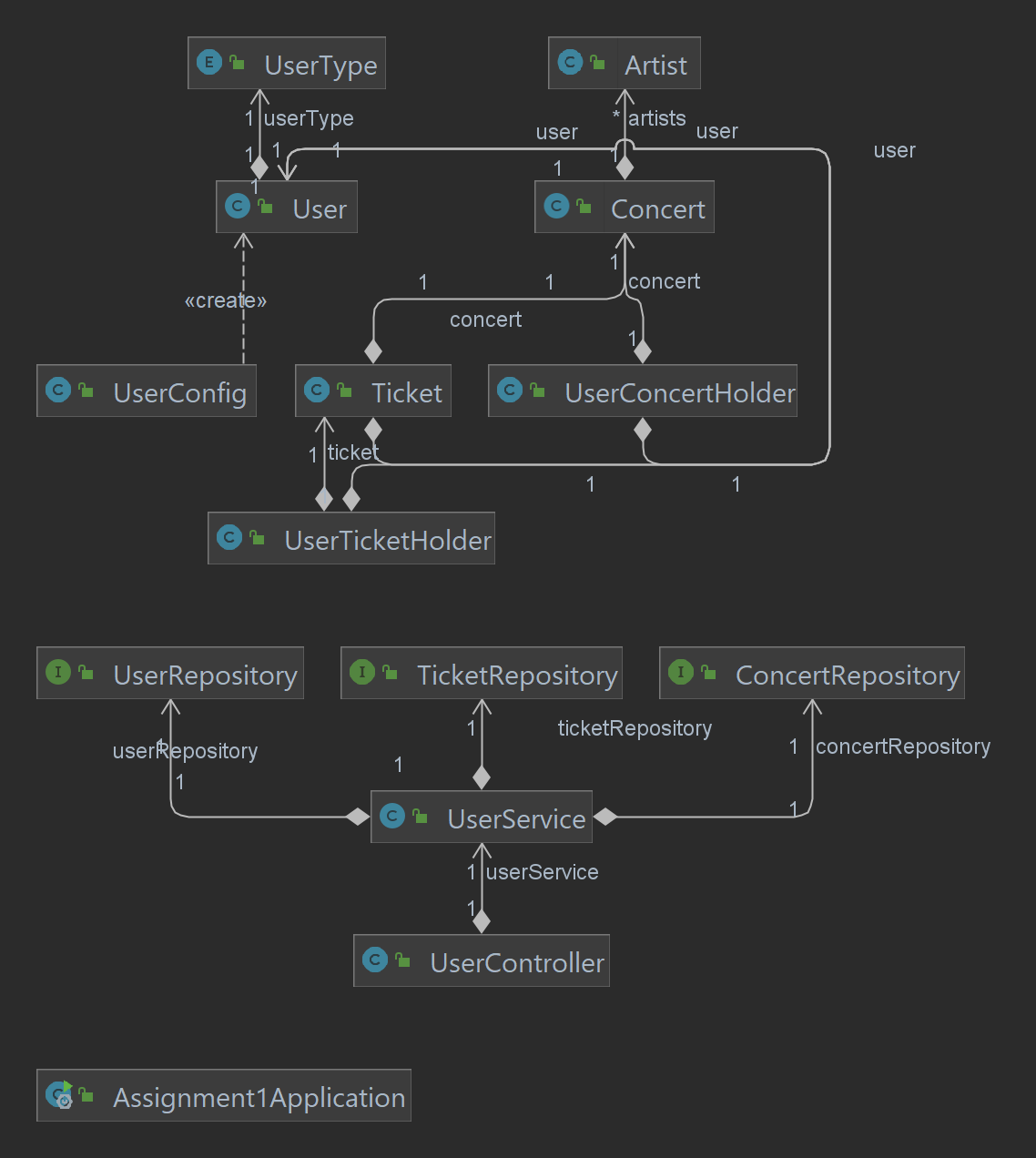
**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

The pattern that has been used is dependency injection. This is a form of inversion of control and aims to separate the concerns of constructing objects and using them, leading to loosely coupled programs.

**5.2 UML Class Diagram**

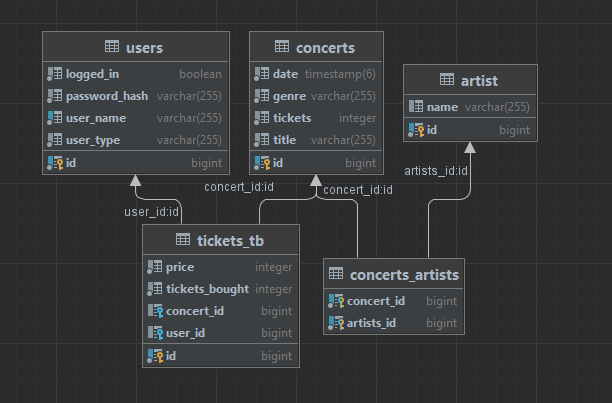
*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



The pattern has been used to inject the service into the REST controller and also to inject into service the specific repositories for each entity for CRUD operations and specific methods.

6. Data Model

*[Present the data models used in the system’s implementation.]*



7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

The system employs a few unit tests for password encryption and database connection. There was also an attempt to implement a max tickets limit reached unit test.

8. Bibliography