DreamCatch

Student: Aldea Ovidiu Alexandru

**Group: 30238**

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

Design and implement an application that monitors the various topics we dream of and the quality levels of sleep over time.

The application should be able to allow users to input a short description together with a set of tags to add an entry to a user or application defined category.

Once an entry is created the user is able to evaluate the quality of his/her sleep with the following app metrics: duration, energy level, stress. These measurements are recorded on a scale of 1-5 for easy tracking.

Given existing data the application is capable of aggregating these metrics across various tag categories and is able to present a daily chart for each quality metric given the user inputs a requested category.

# Functional Requirements

* Use an ORM and a DI Container
* Use a layered architecture
* Use the factory pattern for creating the various metric driven reports/charts
* The data will be stored in a database
* All the inputs of the application will be validated against invalid data before submitting

the data and saving it in the database.

# Non-functional Requirements

* Commit the work you do on your Git repository. Do it iteratively as you progress, not all at once (this will incur a penalty on your final mark)
* Use any OOP language you like. Non-exhaustive: Python, C#, Java, Ruby, C/C++, JS+Typescript
* Implement and test the application

2. Use-Case Model

Use case: Login

Level: user-goal level

Primary actor: user

Main success scenario: the user is redirected to the home page

Extensions: a message is displayed to the user in case the data is not valid

Use case: Record sleep

Level: user-goal level

Primary actor: user

Main success scenario: the sleep data is recorded in the database

Extensions: -

Use case: View chart

Level: user-goal level

Primary actor: user

Main success scenario: the user can choose to view the chart for duration/ energy/ stress in the last 7 days.

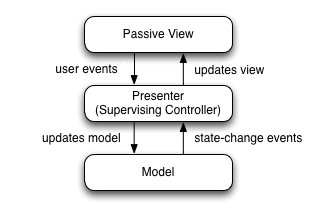
Extensions: -

3. System Architectural Design

**3.1 Architectural Pattern Description**

MVP is a user interface architectural pattern engineered to facilitate automated unit testing and improve the separation of concerns in presentation logic: The model is an interface defining the data to be displayed or otherwise acted upon in the user interface.

**3.2 Diagrams**



4. UML Sequence Diagrams

Sequence diagram for recording sleep data.

*Timeline

Description automatically generated*

5. Class Design

**5.1 Design Patterns Description**

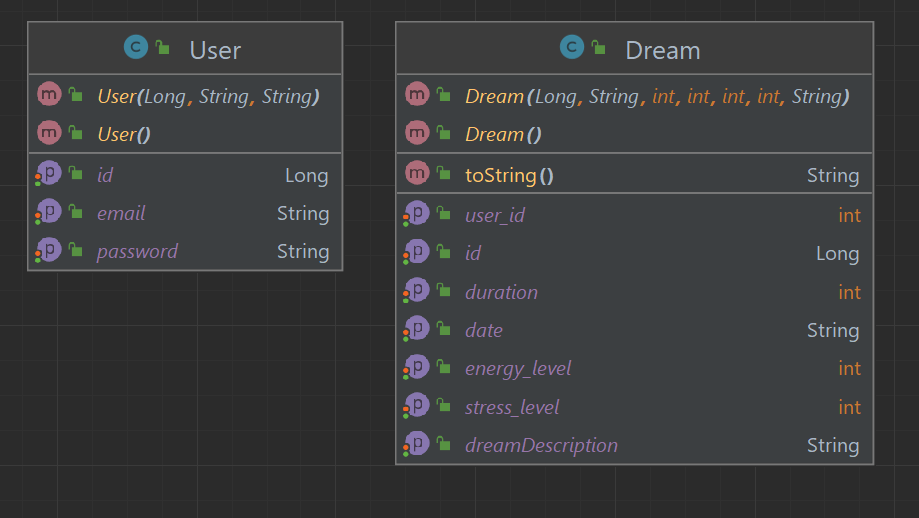
**Factory Method** is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

**5.2 UML Class Diagram**

A screenshot of a video game

Description automatically generated

6. Data Model

**

7. System Testing

The application was not tested.

8. Bibliography

<https://refactoring.guru/design-patterns/factory-method> - Factory Method

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter#:~:text=by%20Derek%20Greer.-,Overview,upon%20in%20the%20user%20interface>. - MVP