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 6

5. Class Design 6

6. Data Model 8

7. System Testing 8

8. Bibliography 8

9. Video 8

1. Requirements Analysis

# Assignment Specification

The project's goal is to develop and execute an application that tracks the various subjects we dream about as well as the quality of our sleep over time.

The application should be able to accept a brief summary, as well as a set of categories in order to record sleep data. Once a record is made, the user can assess the quality of his or her sleep using the app metrics: duration, energy level, and stress.

Given existing data, the application can aggregate these metrics across different tag categories and show a daily chart for each quality measure.

# Functional Requirements

The user should be able to login

The user can record sleep details

The user can see daily sleep records as charts

# Non-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.

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: -

Diagram, text, application

Description automatically generated

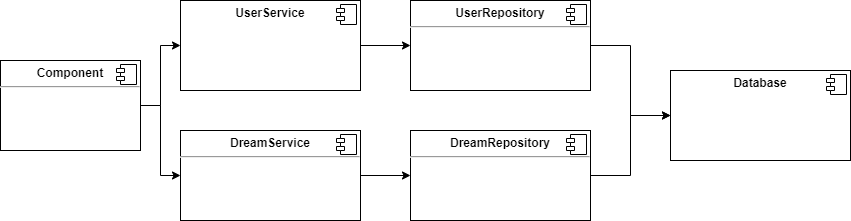
3. System Architectural Design

**3.1 Architectural Pattern Description**

The application is based on a client-server architecture. The backend server was made using Spring Boot and the front-end using React framework. The database used in this project is MySql.

**3.2 Diagrams**

**Component diagram**

****

A screenshot of a computer

Description automatically generated with low confidenceGraphical user interface, application, chat or text message

Description automatically generated

4. UML Sequence Diagrams

Sequence diagram for recording sleep data.

*Timeline

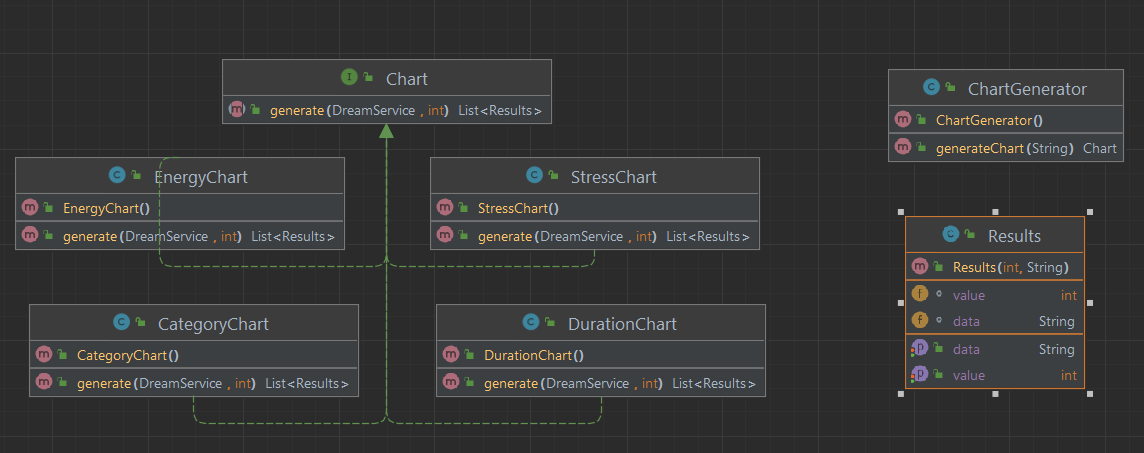
Description automatically generated*

5. Class Design

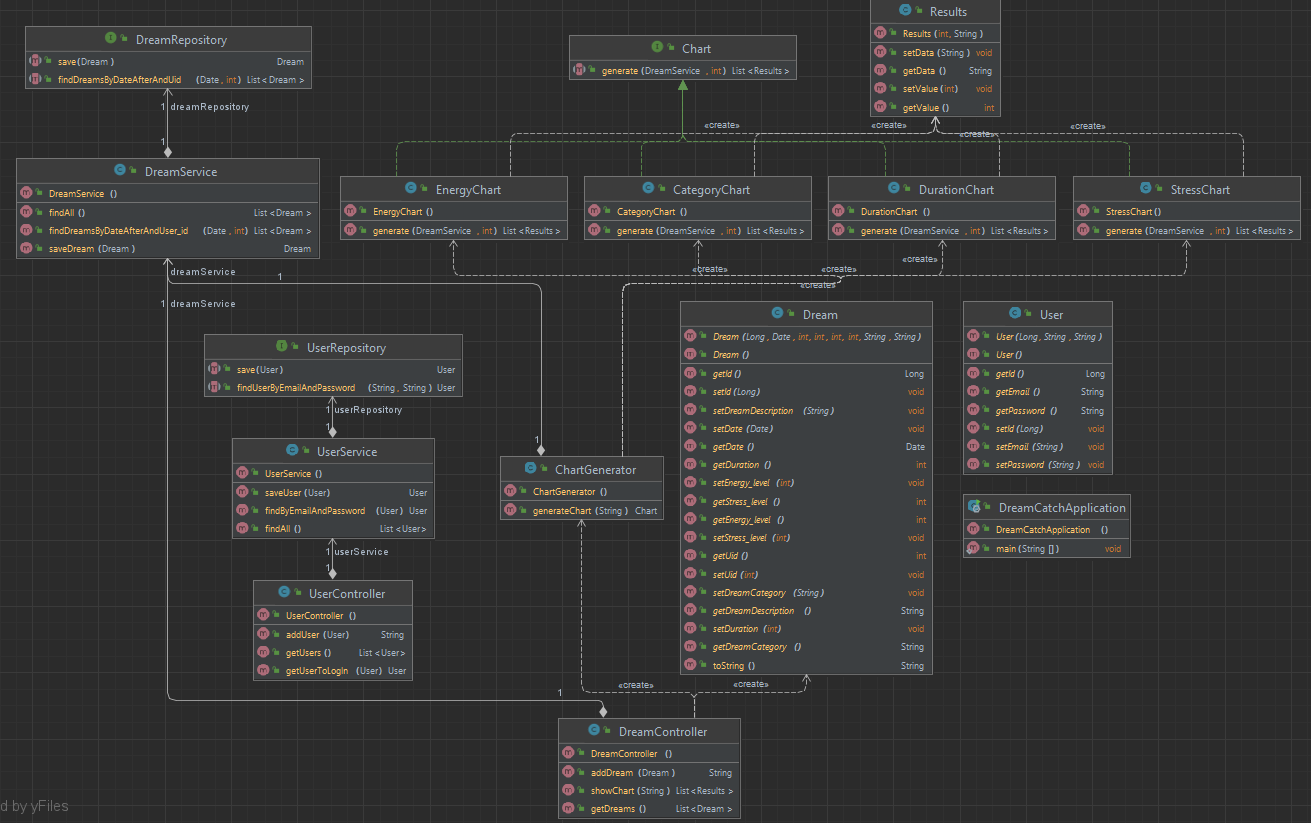
**5.1 Design Patterns Description**

**One of the non-functional requirements of this project was to use the Factory Method design pattern. Factory Method Pattern, implies that you should only specify an interface or abstract class for making an object and let the descendants determine which class to instantiate. The Factory Method Pattern enables sub-classes to select the sort of object to create. It encourages loose coupling by removing the need to connect application-specific classes into the code.**

**The Factory Pattern was used in the creation of the charts that show the user a report of the last 7 days.**

****

**5.2 UML Class Diagram**

****

6. Data Model

Graphical user interface

Description automatically generated

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

9. Video

https://youtu.be/GCYE\_y41pPQ