## 

## 

## 

## 

## **ChatMaps Deliverable 4**

Design Pattern and Design Class Diagram

Stephen Goodridge, Clark LaChance, Nicholas Pease, Joseph Gallant, Aidan Bradley

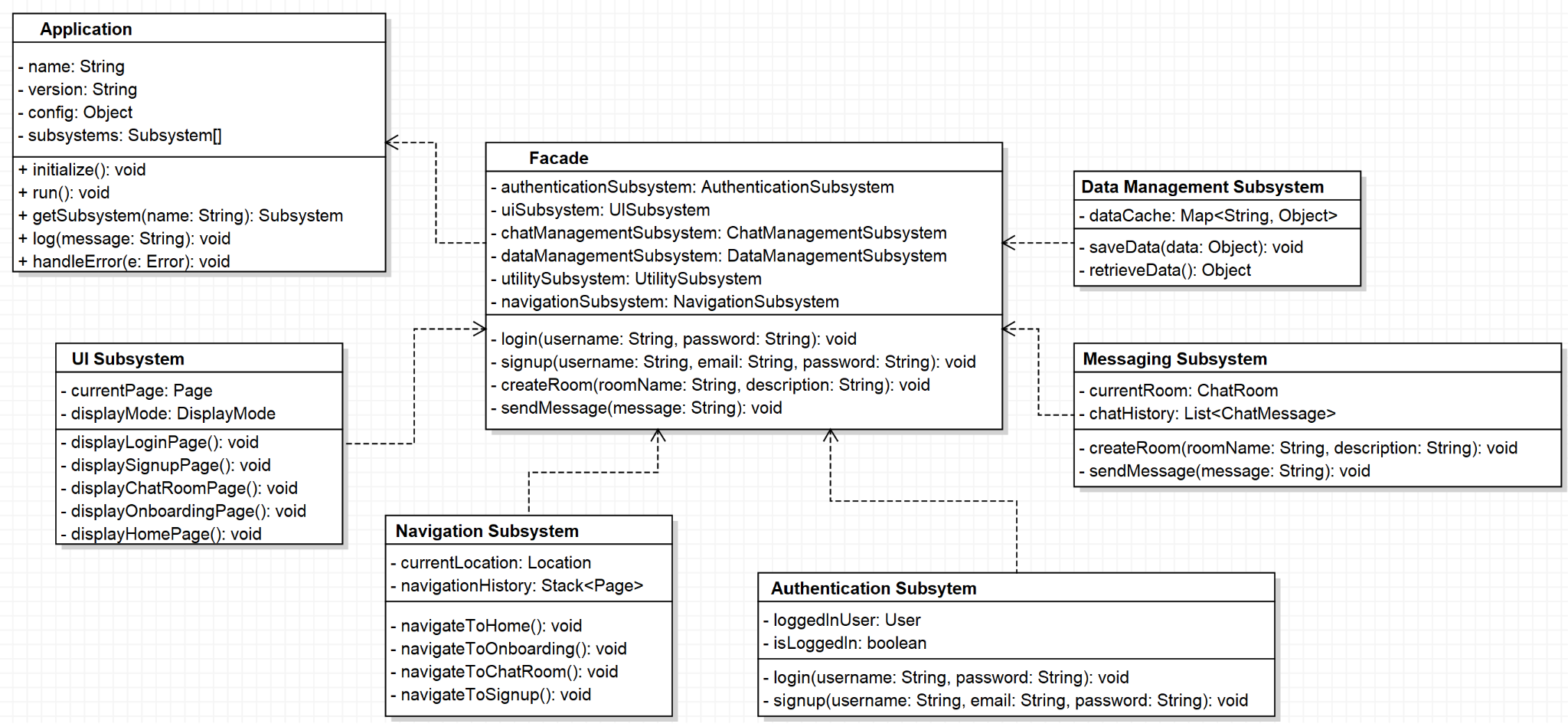
COS420

7 April 2024

Github: <https://github.com/ChatMaps/ChatMaps>

Kanban: <https://trello.com/b/TaygvBv7/chatmaps>

**Design Pattern**



Application - This class serves as the entry point for the application, which will initialize, configure, and manage certain functions. The Facade class is a simplification of the subsystems that the user will interact with, and the Application class will use the Facade class to interact with the subsystems. These subsystems will interact with the Facade class for their respective responsibilities. The relationship between the Application and Facade class is one-to-one.

Facade - This class uses subsystems to carry out the functions from the Application class. It relies on the functions from subsystems to be able to work. The relationship between the Facade class and the Application class is one-to-one, and between the Facade class and Subsystems is one-to-many.

UI Subsystem - This subsystem handles the presentation layer of the application, such as the Home class, the MainTab class, and the Sidebar classes. It will carry out functions associated with presenting information to the user. The relationship between the UI subsystem class and corresponding classes is one-to-one.

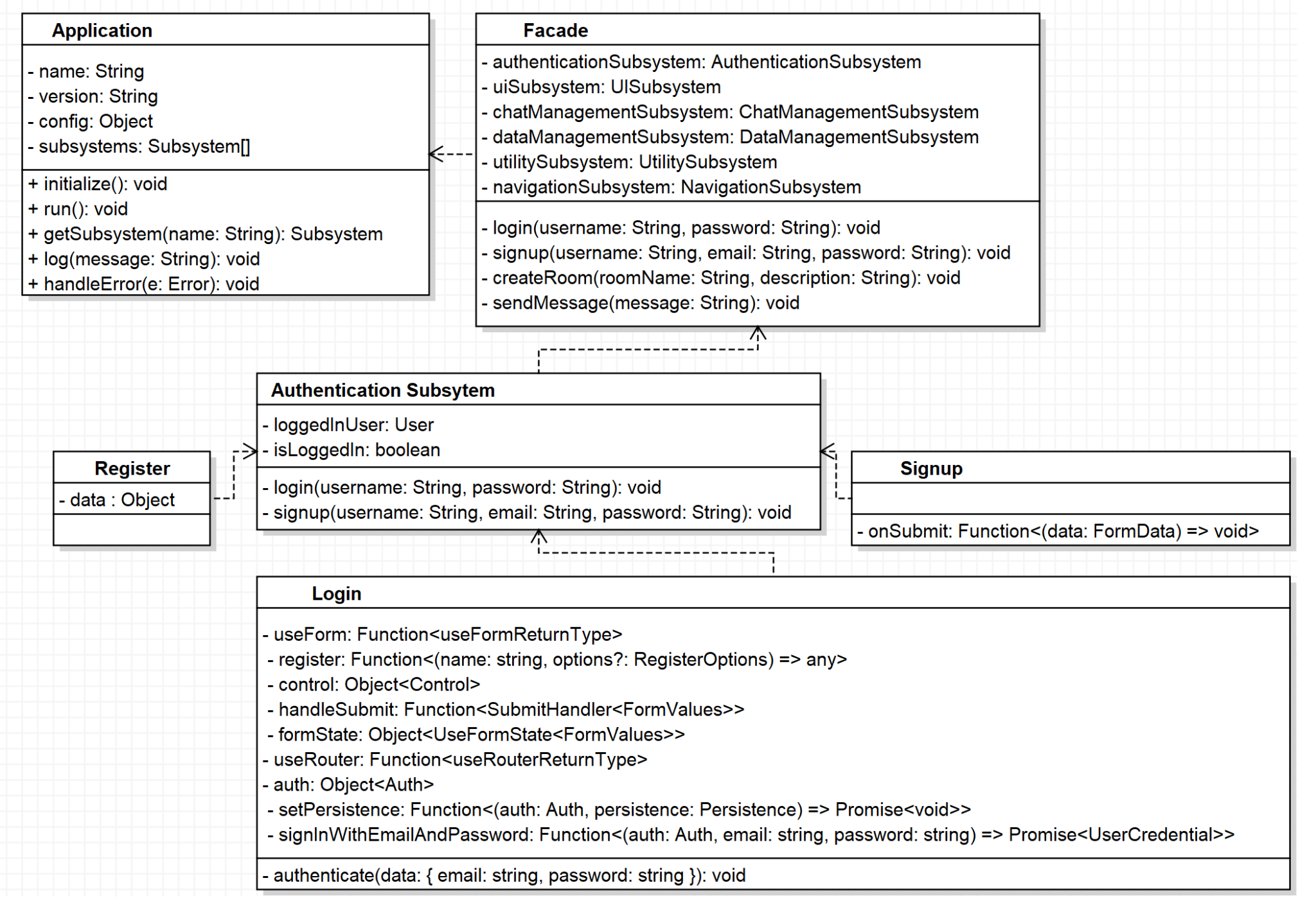
Messaging Subsystem - This subsystem will be responsible for handling chat messages, system messages, and sending and receiving chats, as well as chat rooms. The relationship between the Messaging subsystem class and corresponding class is one-to-one.

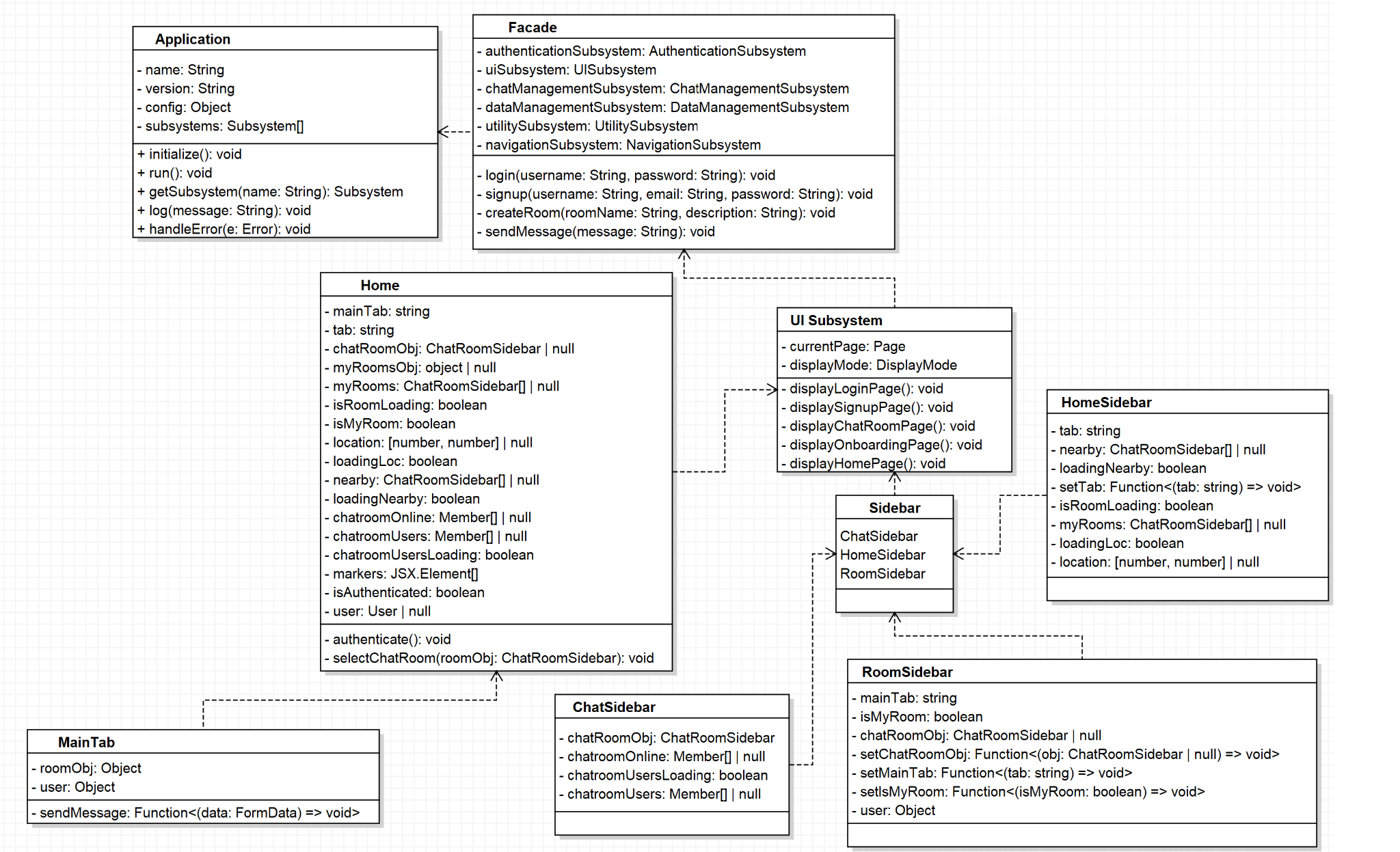
Data Management Subsystem - This subsystem will be responsible for handling all data such as user information, chat messages, and room data, as well as interacting with the database or API for data storage and retrieval. The relationship between the Data Management subsystem class and corresponding classes is one-to-one.

Navigation Subsystem - This subsystem will handle routing between different screens or components. The relationship between the Navigation subsystem class and corresponding classes is one-to-one.

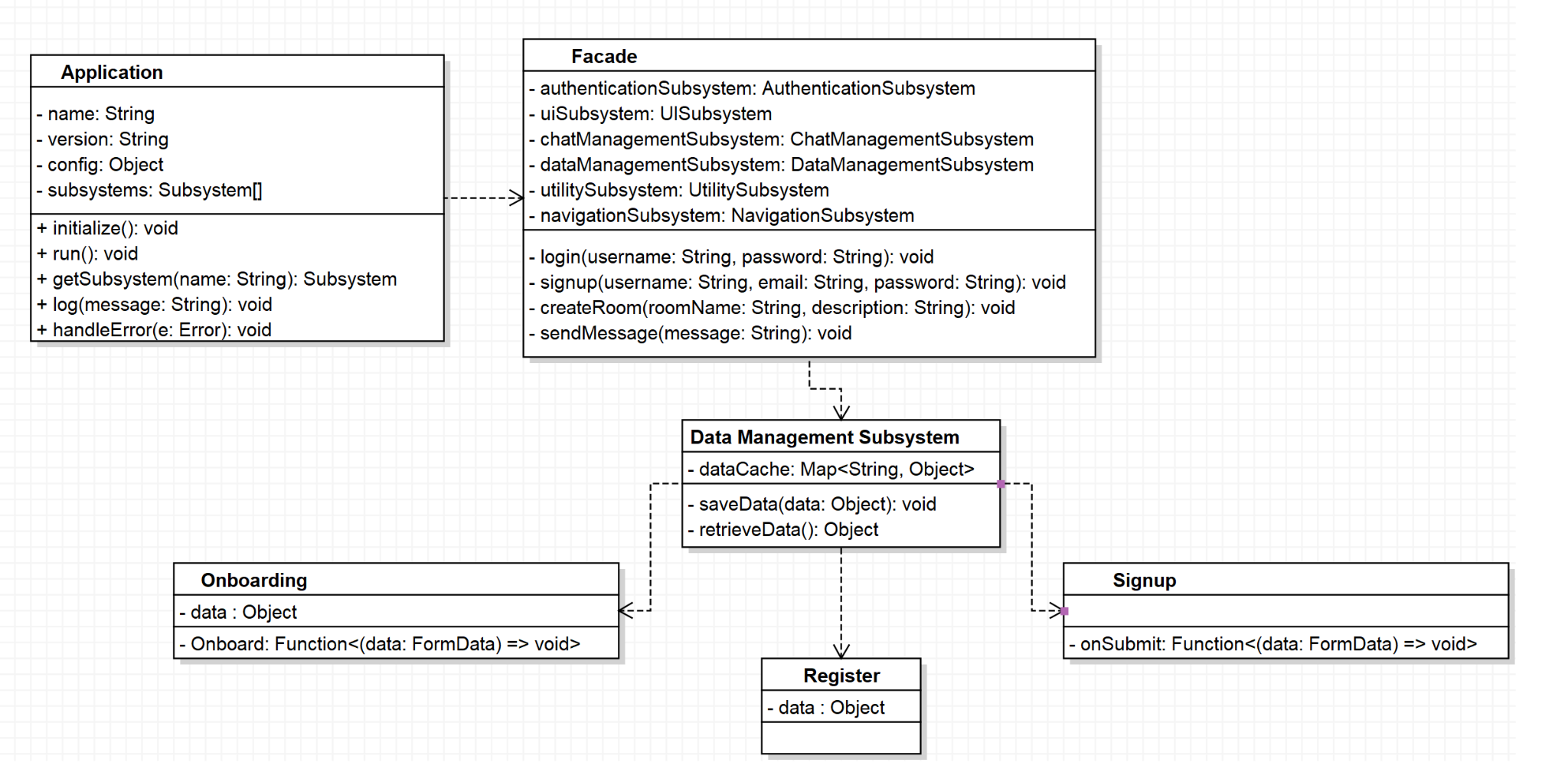
The reasoning behind using the Facade subclass of the Structural design pattern is mainly to simplify the application. It makes using the application easier as well as hiding the complexity of one subsystem. The client side will be separate from the complexity of the subsystem, making for easier maintenance and low coupling. The ability for better maintenance, subsystem encapsulation, and simplification make this pattern a suitable choice for implementation.

**Design Class Diagrams**

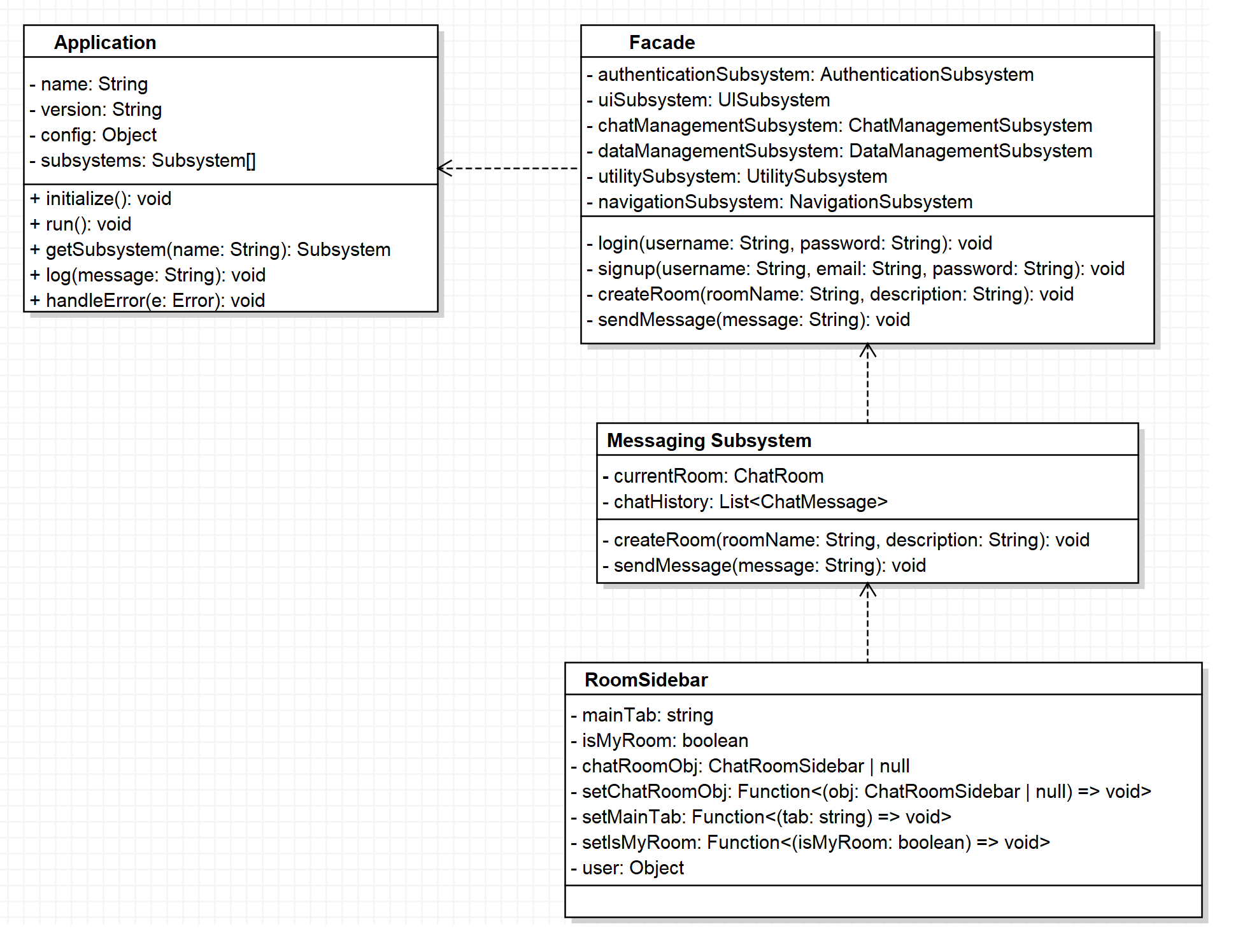
Authentication Subsystem : 

UI Subsystem : 

Data Management Subsystem :



Messaging Subsystem :



Navigation Subsystem :

