# ChatMaps Deliverable 5

Design Pattern and Design Class Diagram

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

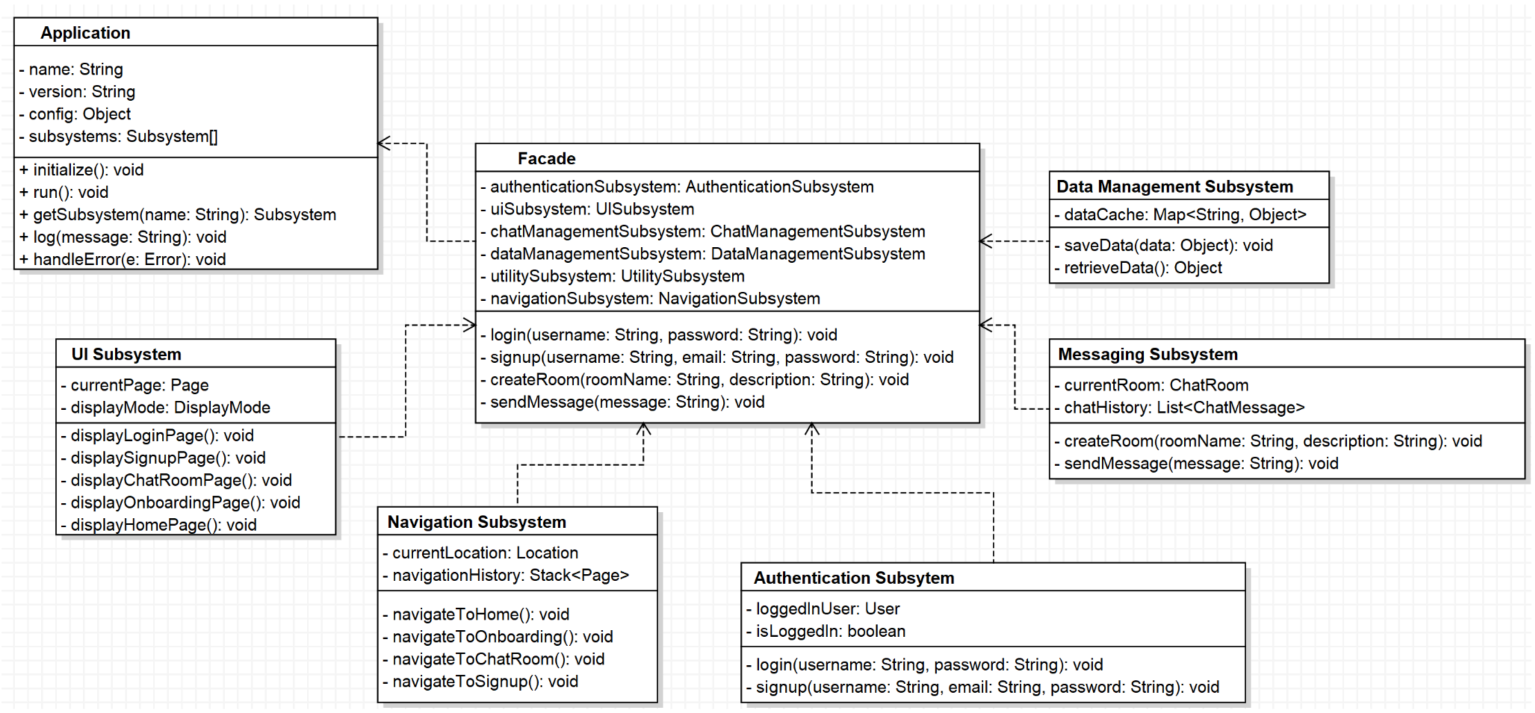
COS420

21 April 2024

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

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

## Design Pattern


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Data Management Subsystem
- dataCache: Map<String, Object>
- saveData(data: Object): void
- retrieveData(): Object
UI Subsystem
- currentPage: Page
- displayMode: DisplayMode
- displayLoginPage(): void
- displaySignupPage(): void
- displayChatRoomPage(): void
- displayOnboardingPage(): void
- displayHomePage(): void
Messaging Subsystem
- currentRoom: ChatRoom
- chatHistory: List<ChatMessage>
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Navigation Subsystem
- currentLocation: Location
- navigationHistory: Stack<Page>
- navigateToHome(): void
- navigateToOnboarding(): void
- navigateToChatRoom(): void
- navigateToSignup(): void
Authentication Subsytem
- loggedInUser: User
- isLoggedIn: boolean
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void

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.

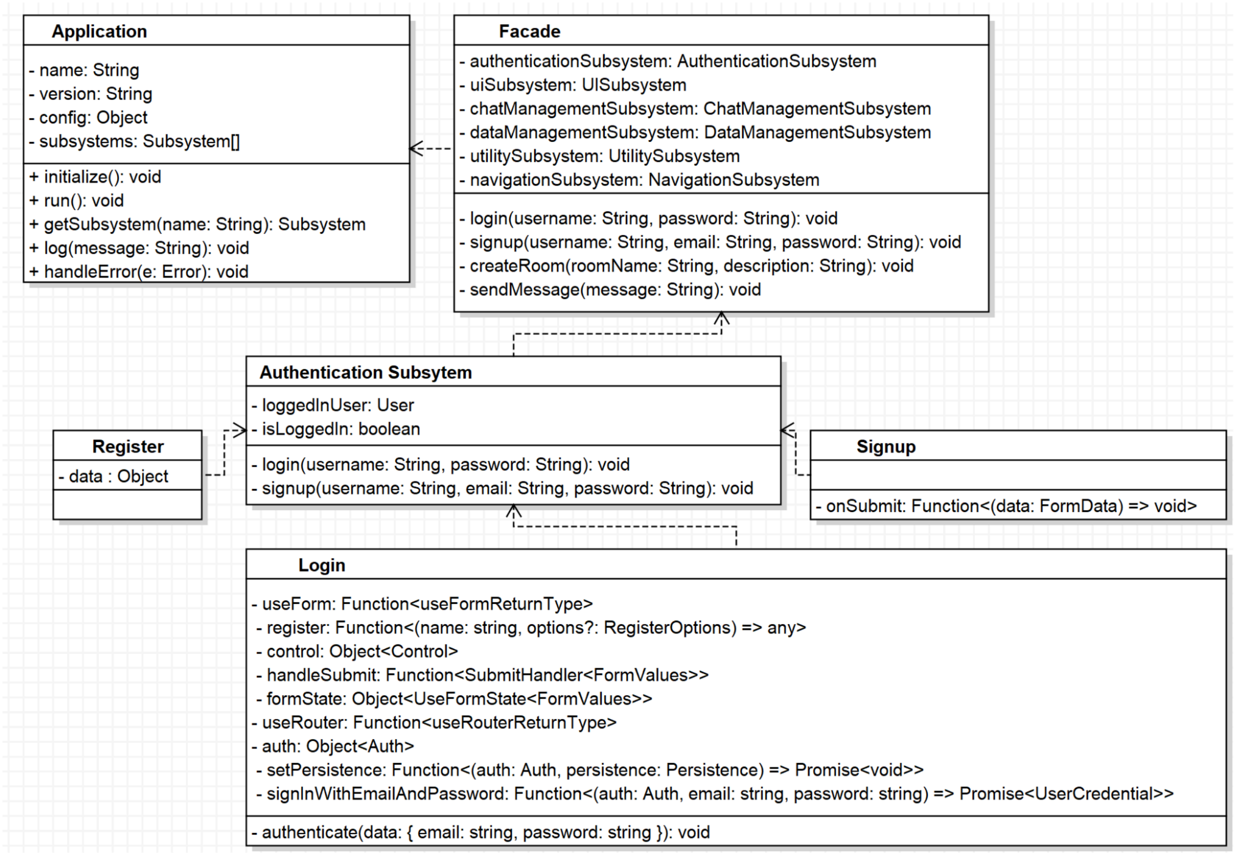
Data Management Subsystem - This subsystem will be responsible for handling all data such as user information, chat messages, and room data, as well as interacting with the database or API for data storage and retrieval. The relationship between the Data Management subsystem class and corresponding classes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

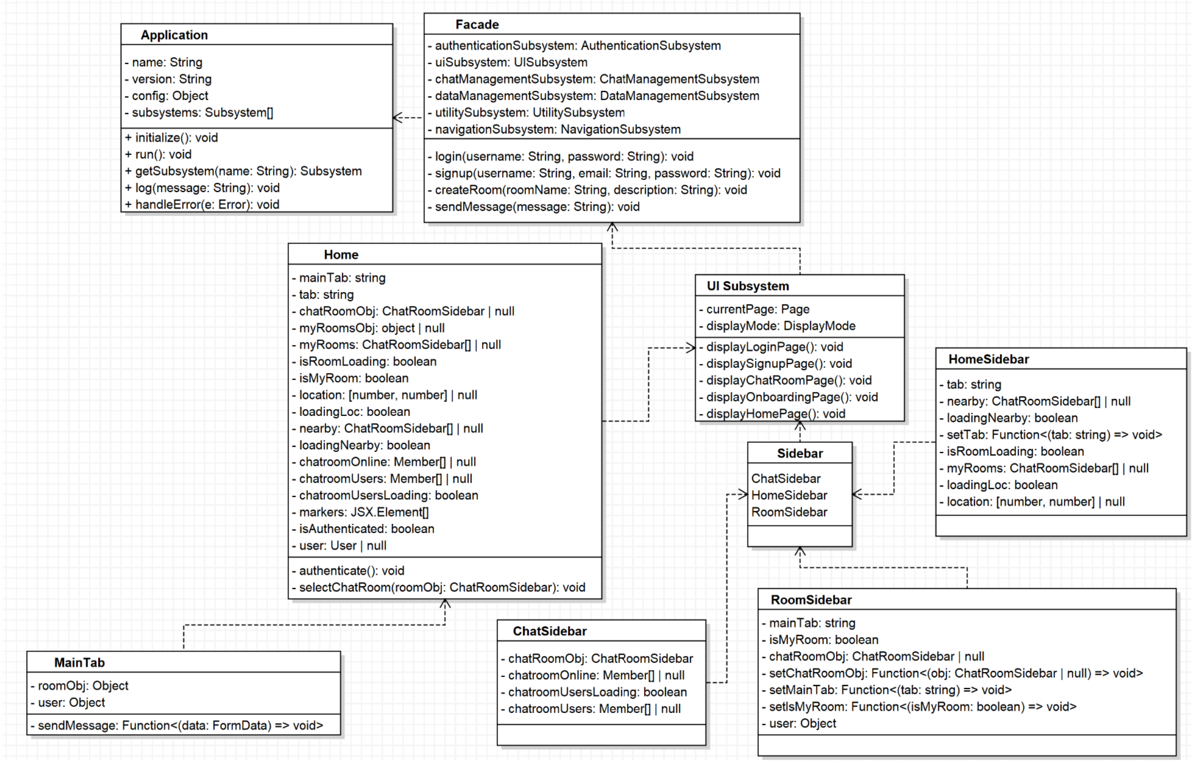
The reasoning behind using the Facade subclass of the Structural design pattern is mainly to simplify the application. It makes using the application easier as well as hiding the complexity of one subsystem. The client side will be separate from the complexity of the subsystem, making for easier maintenance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

## Design Class Diagrams

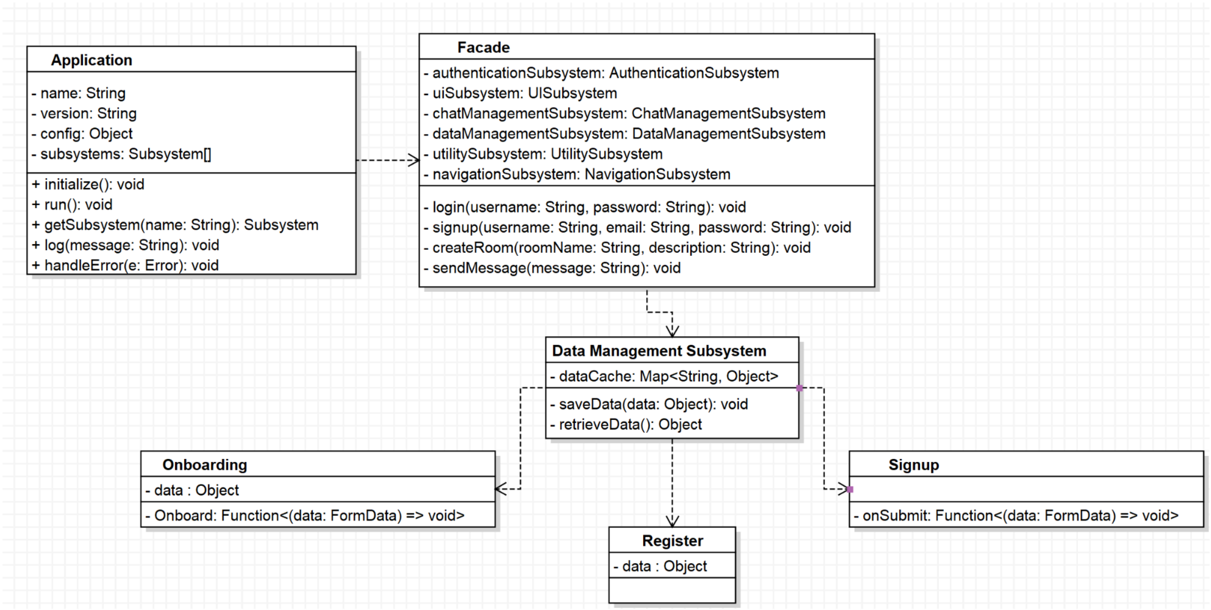
## Authentication Subsystem :


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Authentication Subsytem
- loggedInUser: User
- isLoggedIn: boolean
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
Register
- data : Object
Signup
- onSubmit: Function<(data: FormData) => void>
Login
- useForm: Function<useFormReturnType>
- register: Function<(name: string, options?: RegisterOptions) => any>
- control: Object<Control>
- handleSubmit: Function<SubmitHandler<FormValues>>
- formState: Object<UseFormState<FormValues>>
- useRouter: Function<useRouterReturnType>
- auth: Object<Auth>
- setPersistence: Function<(auth: Auth, persistence: Persistence) => Promise<void>>
- signInWithEmailAndPassword: Function<(auth: Auth, email: string, password: string) => Promise<UserCredential>>
- authenticate(data: { email: string, password: string }): void

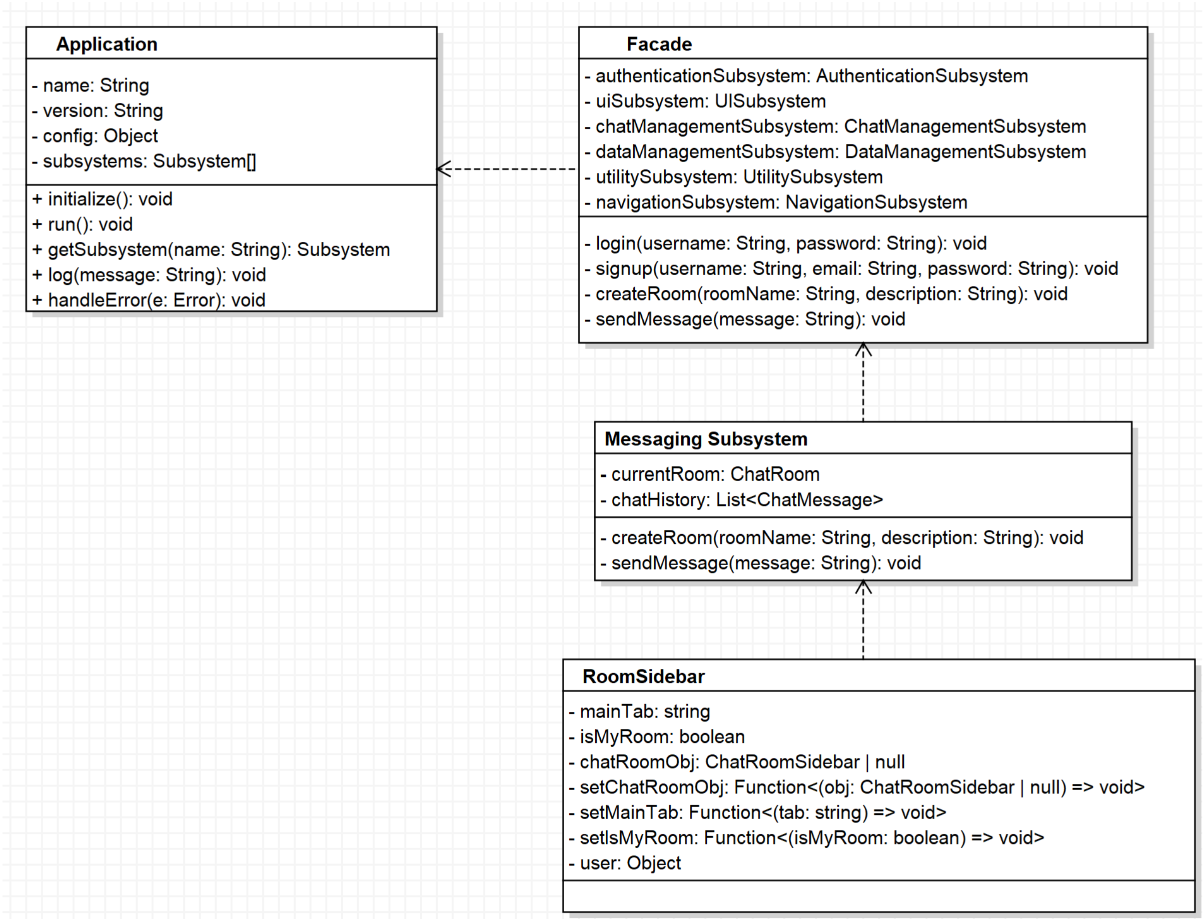
## UI Subsystem :


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Home
- mainTab: string
- tab: string
- chatRoomObj: ChatRoomSidebar | null
- myRoomsObj: object | null
- myRooms: ChatRoomSidebar[] | null
- isRoomLoading: boolean
- isMyRoom: boolean
- location: [number, number] | null
- loadingLoc: boolean
- nearby: ChatRoomSidebar[] | null
- loadingNearby: boolean
- chatroomOnline: Member[] | null
- chatroomUsers: Member[] | null
- chatroomUsersLoading: boolean
- markers: JSX.Element[]
- isAuthenticated: boolean
- user: User | null
- authenticate(): void
- selectChatRoom(roomObj: ChatRoomSidebar): void
UI Subsystem
- currentPage: Page
- displayMode: DisplayMode
- displayLoginPage(): void
- displaySignupPage(): void
- displayChatRoomPage(): void
- displayOnboardingPage(): void
- displayHomePage(): void
HomeSidebar
- tab: string
- nearby: ChatRoomSidebar[] | null
- loadingNearby: boolean
- setTab: Function<(tab: string) => void>
- isRoomLoading: boolean
- myRooms: ChatRoomSidebar[] | null
- loadingLoc: boolean
- location: [number, number] | null
Sidebar
ChatSidebar
HomeSidebar
RoomSidebar
RoomSidebar
- mainTab: string
- isMyRoom: boolean
- chatRoomObj: ChatRoomSidebar | null
- setChatRoomObj: Function<(obj: ChatRoomSidebar | null) => void>
- setMainTab: Function<(tab: string) => void>
- setIsMyRoom: Function<(isMyRoom: boolean) => void>
- user: Object
ChatSidebar
- chatRoomObj: ChatRoomSidebar
- chatroomOnline: Member[] | null
- chatroomUsersLoading: boolean
- chatroomUsers: Member[] | null
MainTab
- roomObj: Object
- user: Object
- sendMessage: Function<(data: FormData) => void>

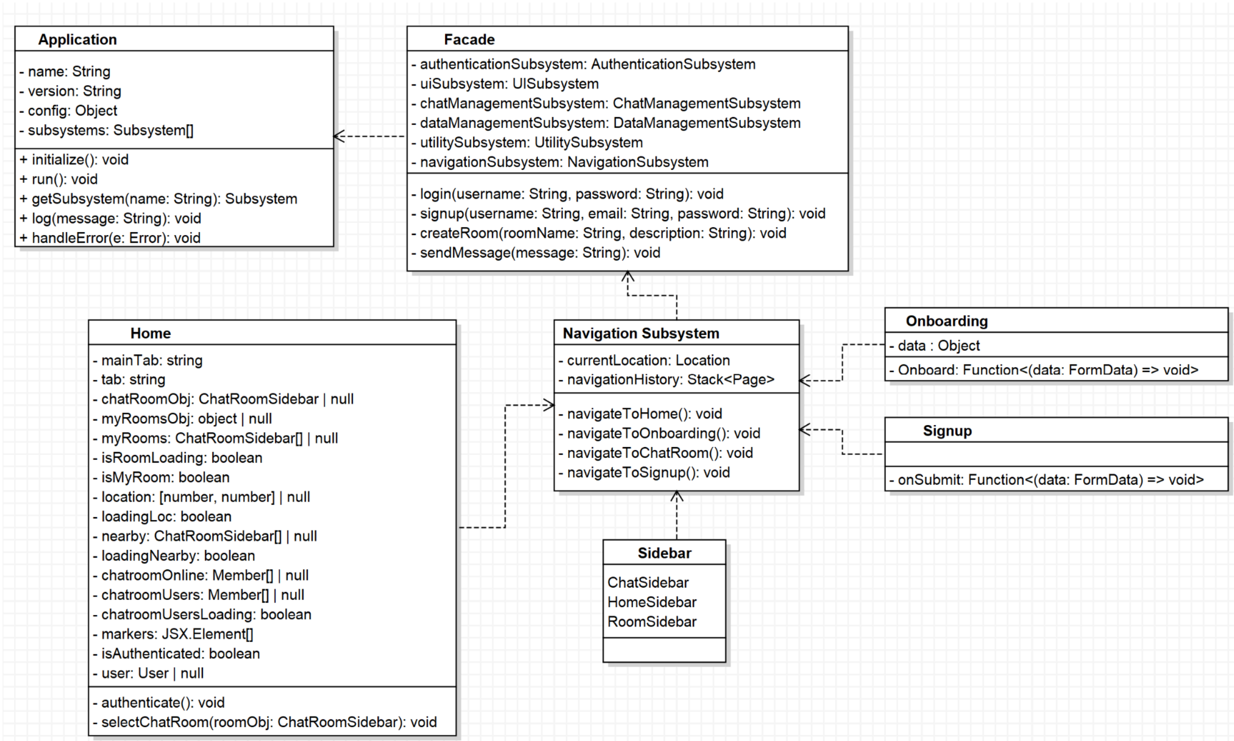
## Data Management Subsystem :


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Data Management Subsystem
- dataCache: Map<String, Object>
- saveData(data: Object): void
- retrieveData(): Object
Onboarding
- data : Object
- Onboard: Function<(data: FormData) => void>
Signup
- onSubmit: Function<(data: FormData) => void>
Register
- data : Object

## Messaging Subsystem :


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Messaging Subsystem
- currentRoom: ChatRoom
- chatHistory: List<ChatMessage>
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
RoomSidebar
- mainTab: string
- isMyRoom: boolean
- chatRoomObj: ChatRoomSidebar | null
- setChatRoomObj: Function<(obj: ChatRoomSidebar | null) => void>
- setMainTab: Function<(tab: string) => void>
- setIsMyRoom: Function<(isMyRoom: boolean) => void>
- user: Object

## Navigation Subsystem :


Application
- name: String
- version: String
- config: Object
- subsystems: Subsystem[]
+ initialize(): void
+ run(): void
+ getSubsystem(name: String): Subsystem
+ log(message: String): void
+ handleError(e: Error): void
Facade
- authenticationSubsystem: AuthenticationSubsystem
- uiSubsystem: UISubsystem
- chatManagementSubsystem: ChatManagementSubsystem
- dataManagementSubsystem: DataManagementSubsystem
- utilitySubsystem: UtilitySubsystem
- navigationSubsystem: NavigationSubsystem
- login(username: String, password: String): void
- signup(username: String, email: String, password: String): void
- createRoom(roomName: String, description: String): void
- sendMessage(message: String): void
Home
- mainTab: string
- tab: string
- chatRoomObj: ChatRoomSidebar | null
- myRoomsObj: object | null
- myRooms: ChatRoomSidebar[] | null
- isRoomLoading: boolean
- isMyRoom: boolean
- location: [number, number] | null
- loadingLoc: boolean
- nearby: ChatRoomSidebar[] | null
- loadingNearby: boolean
- chatroomOnline: Member[] | null
- chatroomUsers: Member[] | null
- chatroomUsersLoading: boolean
- markers: JSX.Element[]
- isAuthenticated: boolean
- user: User | null
- authenticate(): void
- selectChatRoom(roomObj: ChatRoomSidebar): void
Navigation Subsystem
- currentLocation: Location
- navigationHistory: Stack<Page>
- navigateToHome(): void
- navigateToOnboarding(): void
- navigateToChatRoom(): void
- navigateToSignup(): void
Onboarding
- data : Object
- Onboard: Function<(data: FormData) => void>
Signup
- onSubmit: Function<(data: FormData) => void>
Sidebar
ChatSidebar
HomeSidebar
RoomSidebar