

## Problem Statement

ในการพัฒนาแอปพลิเคชันแบบ Real-time นั้น การเลือกใช้เทคโนโลยีและเครื่องมือที่เหมาะสมเป็นสิ่งสำคัญ เพื่อให้สามารถตอบสนองต่อการทำงานแบบออนไลน์ได้อย่างมีประสิทธิภาพ ทีมพัฒนาของเราได้ตัดสินใจใช้ JavaFX สำหรับการพัฒนา Client และ Spring Boot สำหรับการจัดการ Server

เหตุผลหลักที่เลือกใช้ JavaFX เป็นเครื่องมือสำหรับการพัฒนา Client นั้น นอกจากจะช่วยให้สามารถสร้างส่วนติดต่อผู้ใช้ (User Interface) ที่ยืดหยุ่นและมีประสิทธิภาพแล้ว ยังเป็นความท้าทายในการพัฒนาแอปพลิเคชันแบบ Real-time โดยไม่ต้องอาศัย Spring Boot เป็นโครงสร้างหลักของ Client การตัดสินใจเช่นนี้ทำให้ทีมพัฒนาต้องออกแบบโครงสร้างของแอปพลิเคชันให้สามารถทำงานร่วมกันระหว่าง Client (JavaFX) และ Server (Spring Boot) ได้อย่างมีประสิทธิภาพ

เป้าหมายของโครงการนี้คือการสร้างระบบที่สามารถทำงานได้แบบ Online รองรับการสื่อสารระหว่าง Client และ Server ได้อย่างรวดเร็วและมีเสถียรภาพ ซึ่งจะเป็นประสบการณ์ที่ท้าทายและช่วยให้ทีมพัฒนาได้เรียนรู้แนวทางการพัฒนาแอปพลิเคชันแบบ Real-time โดยใช้ Java อย่างเต็มประสิทธิภาพ

## Features

### คุณลักษณะขั้นต่ำ

- ผู้ใช้สามารถลงทะเบียนและเข้าสู่ระบบได้
- รองรับการแชทแบบ 1-ต่อ-1 (Private Chat)
- ใช้งานผ่านเครือข่ายอินเทอร์เน็ตได้
- ระบบแจ้งเตือนเมื่อมีข้อความใหม่

### คุณลักษณะเพิ่มเติม

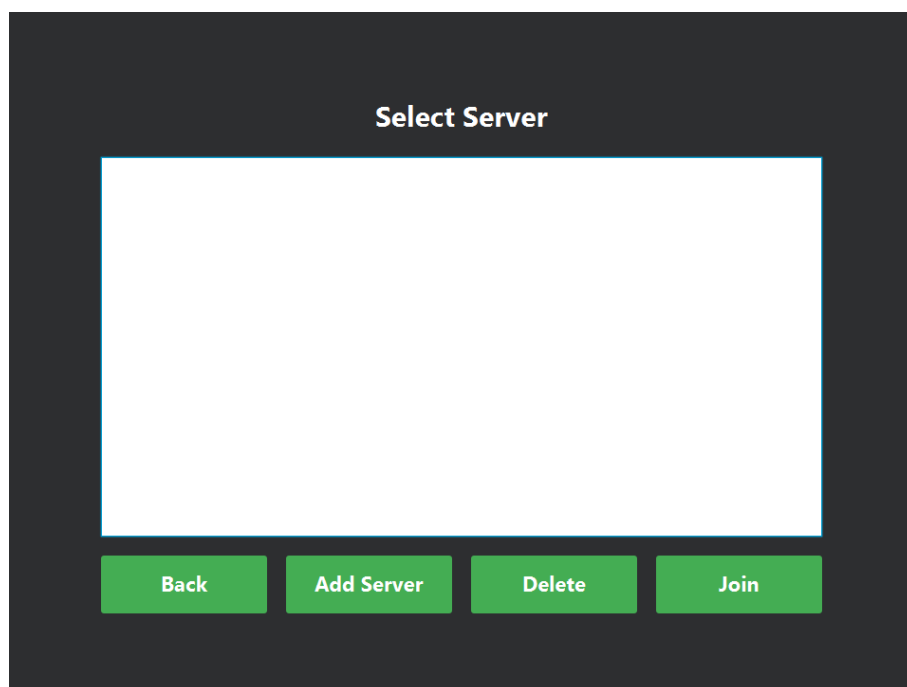
- รองรับการส่งไฟล์และรูปภาพ
- รองรับการแชทเป็นกลุ่ม (Group Chat)
- การเข้ารหัสข้อความเพื่อเพิ่มความปลอดภัย
- ระบบแสดงสถานะออนไลน์/ออฟไลน์ของผู้ใช้
- ระบบจัดการเพื่อน
- แก้ไข/ลบข้อความ

## Program Design

### User Interface



ภาพที่ 1 หน้าต่างแรกเมื่อกดรันโปรแกรม



ภาพที่ 2 หน้าต่างเลือก Server

The screenshot shows a dark-themed window titled "Server Info". It contains two input fields: "Server Name" and "Server Address". The "Server Address" field has a placeholder text "Enter Server Address". At the bottom, there are two green buttons: "Cancel" and "Done".

# Server Info

Server Name

Server Address

Cancel Done

ภาพที่ 3 หน้าต่างกรอกข้อมูล Server

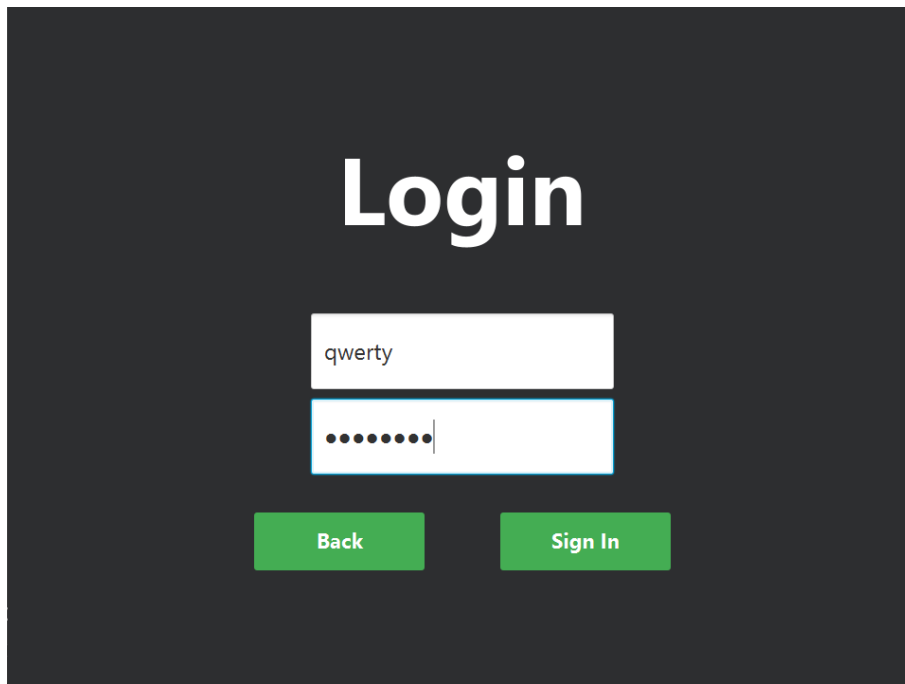
The screenshot shows a dark-themed window titled "Register". It contains three input fields: a username field with the text "qwerty", a password field with 8 dots, and a confirm password field with 8 dots and a cursor. At the bottom, there are two green buttons: "Back" and "Sign Up".

# Register

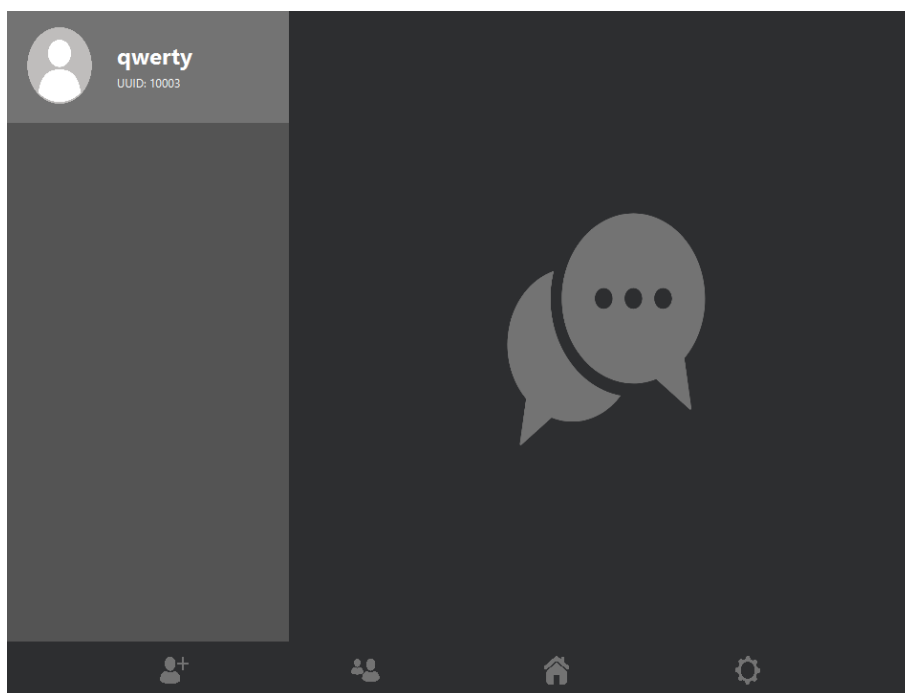
qwerty

Back Sign Up

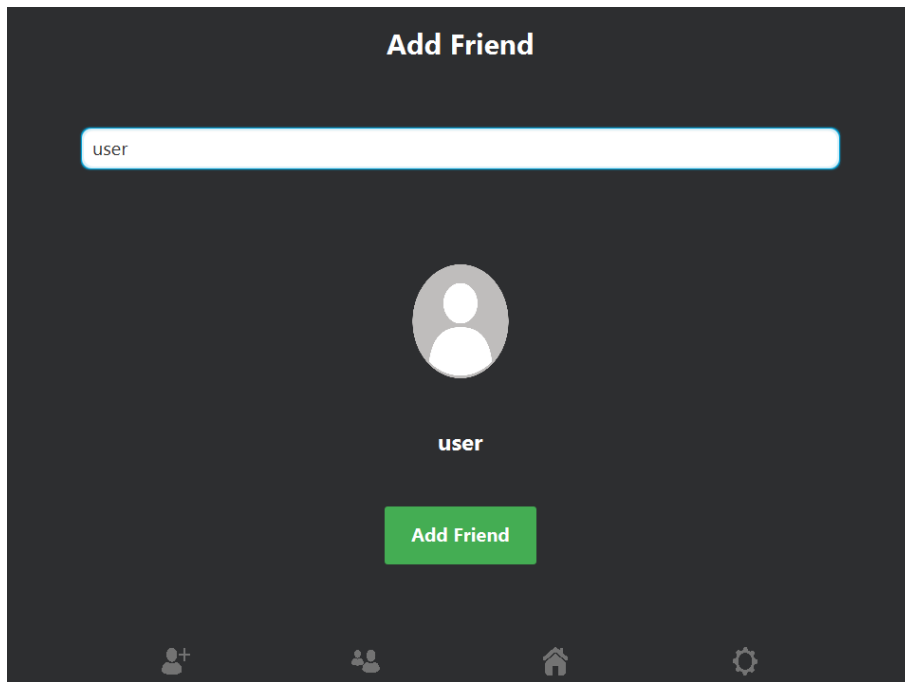
ภาพที่ 4 หน้าต่างการสมัครสมาชิก



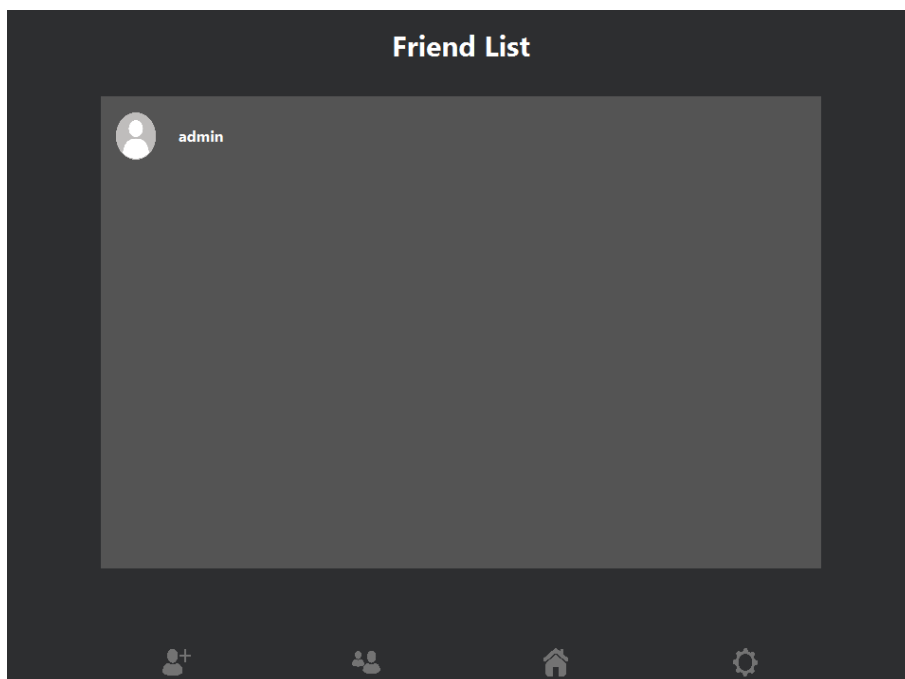
ภาพที่ 5 หน้าต่างการเข้าสู่ระบบ



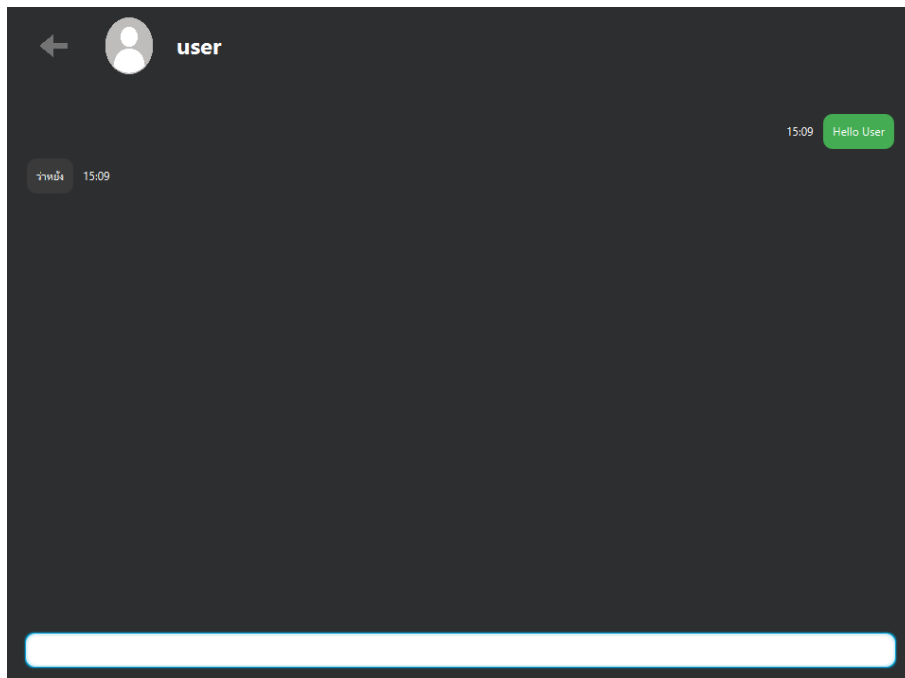
ภาพที่ 6 หน้าหลักของแชท



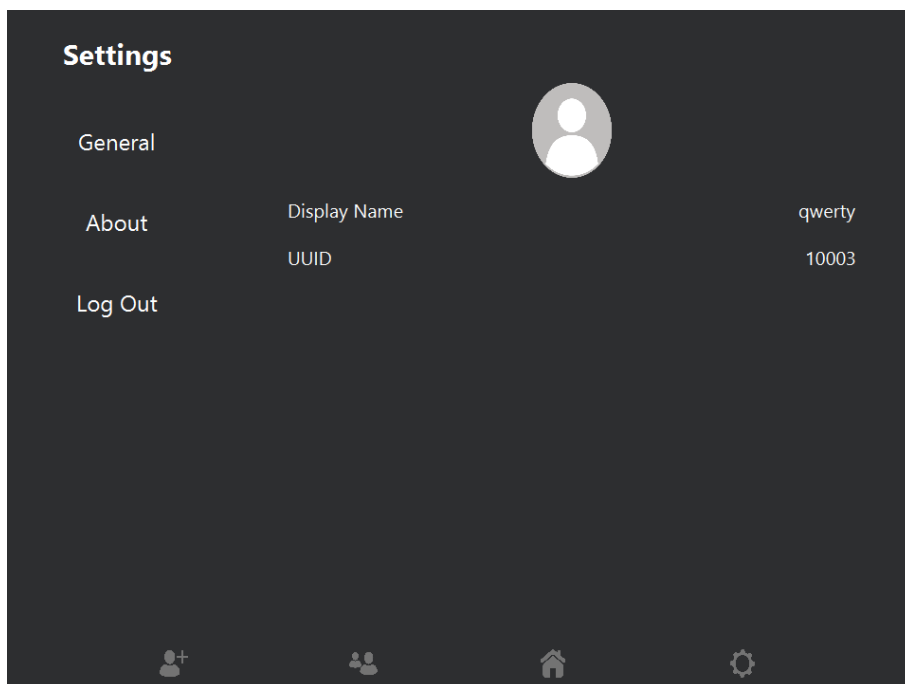
ภาพที่ 7 หน้าต่างเพิ่มเพื่อน



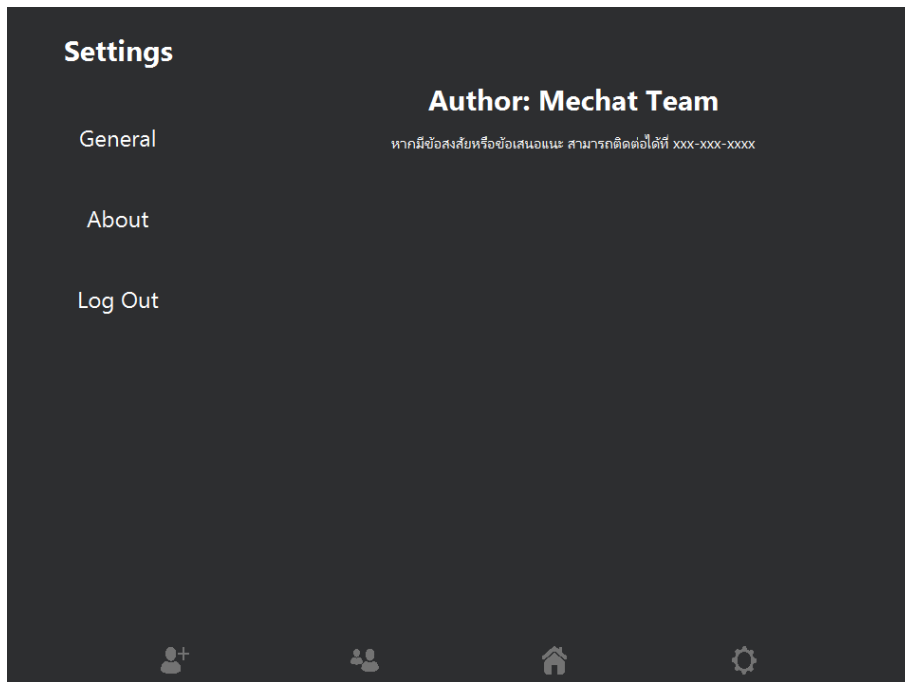
ภาพที่ 8 หน้าต่างรายชื่อเพื่อน



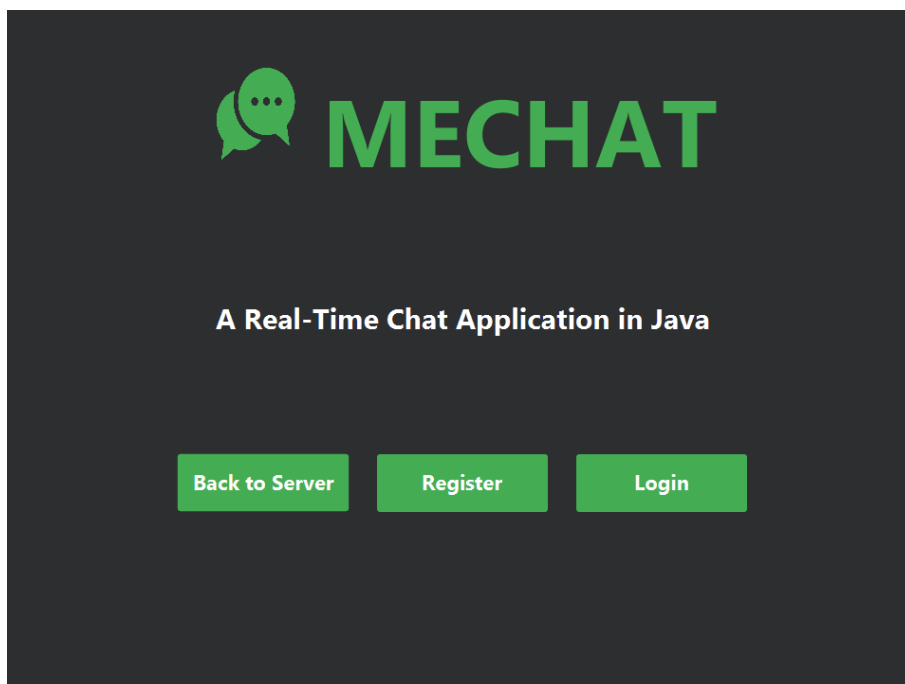
ภาพที่ 9 หน้าต่างแชทสนทนา



ภาพที่ 10 หน้าต่างการตั้งค่าทั่วไป



ภาพที่ 11 หน้าต่างแสดงเกี่ยวกับโปรแกรม

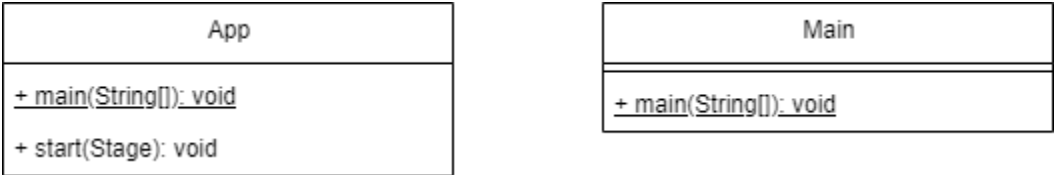


ภาพที่ 12 หน้าต่างเมื่อกด Log Out



Diagram

- Client



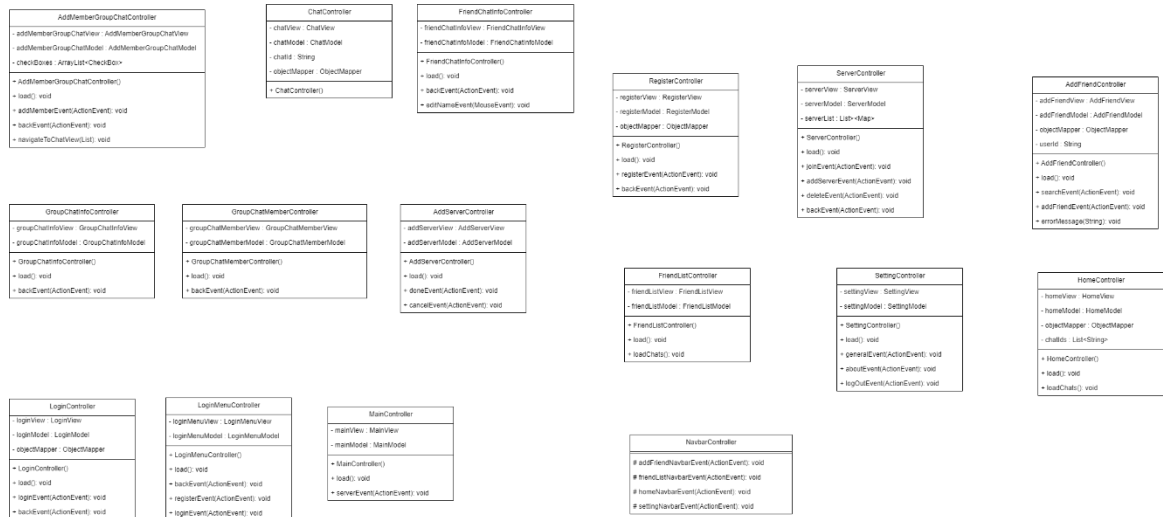
Client/App



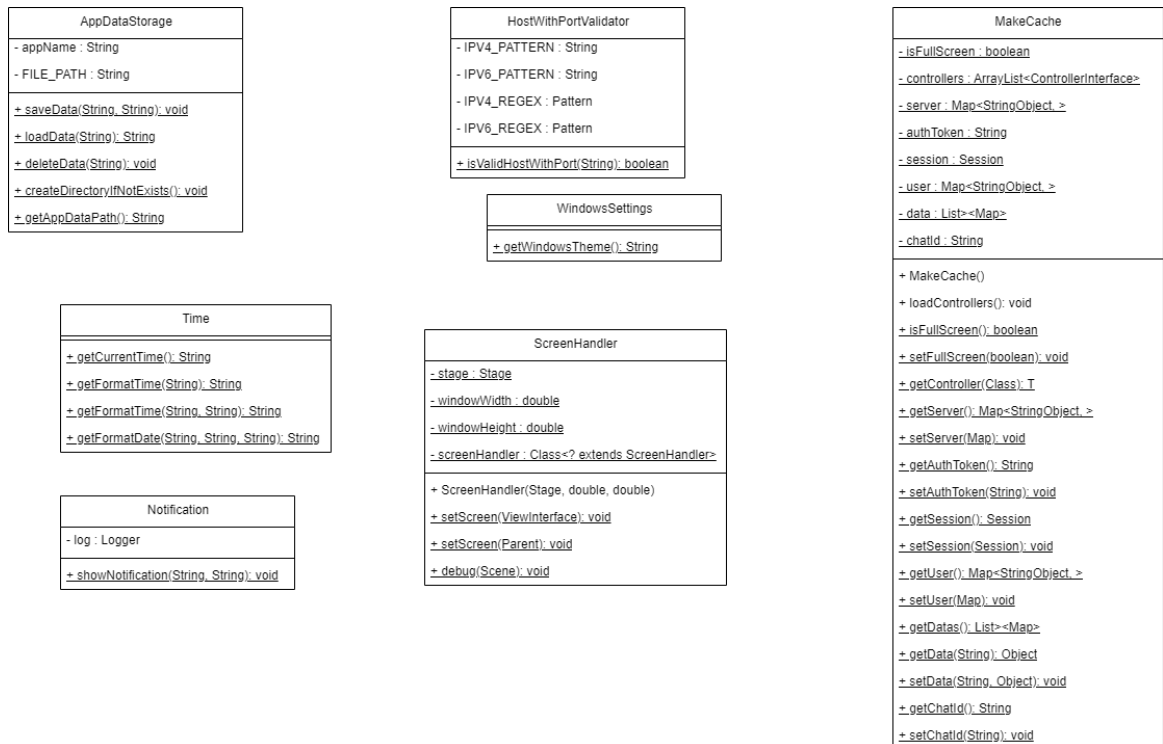
Client/Interface



Client/Websocket



## Client/Controller

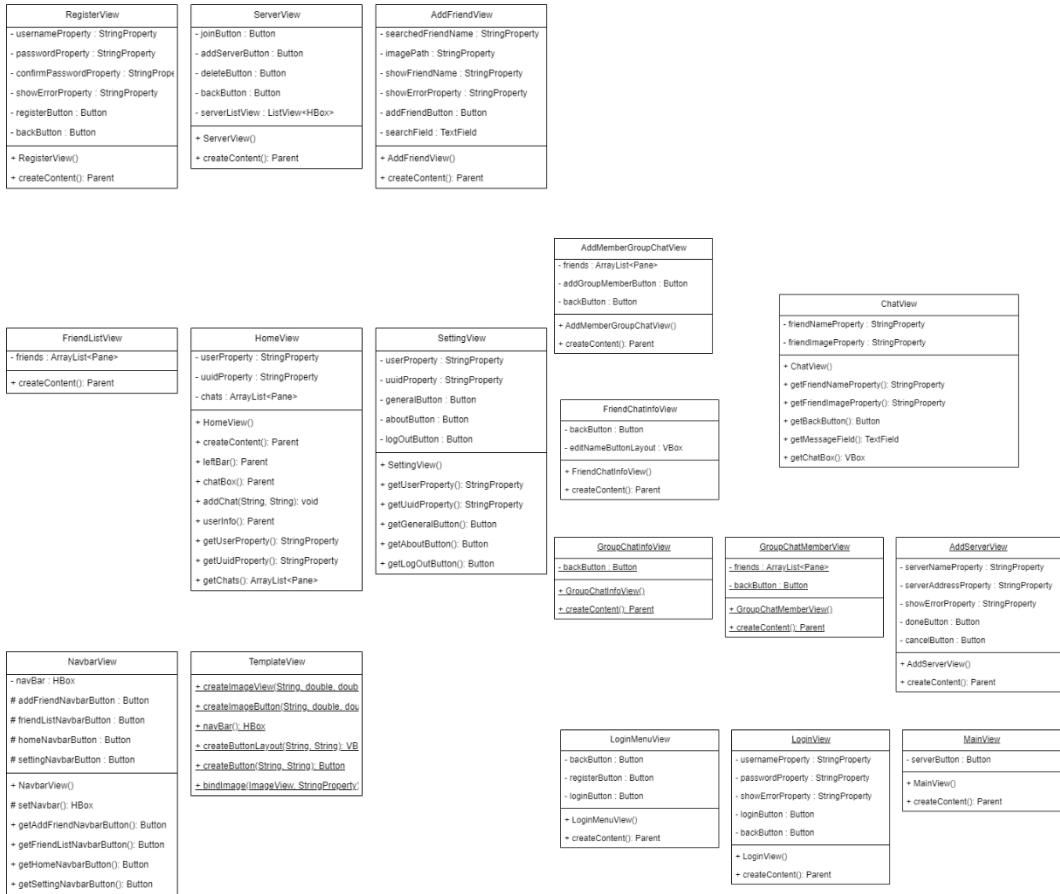


## Client/Utils

RequestMessage	ResponseMessage
- op : int - t : int - d : Map<StringObject, > - objectMapper : ObjectMapper	- op : int - t : int - d : Map<StringObject, > - objectMapper : ObjectMapper
+ RequestMessage(String) + getOp(): int + getT(): int + getD(): Map<StringObject, >	+ ResponseMessage(int, int) + getOp(): int + getT(): int + getD(): Map<StringObject, > + put(String, Object): void + raw(): String

RestApiService
- log : Logger <u>- webClient : WebClient</u>
<u>+ connect(): void</u> <u>+ disconnect(): void</u> <u>+ getConnection(String, String): Mono&lt;String&gt;</u> <u>+ login(String, String): Mono&lt;String&gt;</u> <u>+ register(String, String): Mono&lt;String&gt;</u> <u>+ getUser(String): Mono&lt;String&gt;</u> <u>+ getUserByUsername(String): Mono&lt;String&gt;</u> <u>+ getChat(String): Mono&lt;String&gt;</u> <u>+ getChatHistory(String): Mono&lt;String&gt;</u>

Client/ Service



Client/View

- Server

BaseDTO
- id : Long - createdAt : LocalDateTime - updatedAt : LocalDateTime
+ getId(): Long + setId(Long): void + getCreatedAt(): LocalDateTime + setCreatedAt(LocalDateTime): void + getUpdatedAt(): LocalDateTime + setUpdatedAt(LocalDateTime): void

UserDTO
- username : String - displayName : String - avatar : String
+ getUsername(): String + setUsername(String): void + getDisplayName(): String + setDisplayName(String): void + getAvatar(): String + setAvatar(String): void

Server/Dto

ChatService
- userService : UserService - chatRepository : ChatRepository - chatHistoryRepository : ChatHistoryRepository - chatMemberRepository : ChatMemberRepository
+ getChatsById(Long): List<Chat> + getChatById(Long): Chat + getChatByUserId(Long, Long): Chat + addChat(Chat, List): Chat + updateChat(Chat, List, List): void + getChatHistory(Chat): List<ChatHistory> + saveChatHistory(Chat, UserDTO, String): void + updateChatHistory(Chat, Long, String): void + deleteChatHistory(Chat, Long): void + getChatUser(Chat, UserDTO): UserDTO + getChatUsers(Chat): List<ChatMember> + getChatUsers(Chat, UserDTO): List<User> + saveChatUser(Chat, UserDTO): void

FriendService
- friendRepository : FriendRepository - userService : UserService
+ getFriends(Long): List<Friend> + getFriends(Long, Long): Friend + addFriend(UserDTO, UserDTO): void + updateFriend(UserDTO, UserDTO, Friend): void

UserService
- userRepository : UserRepository
+ convertToDTO(User): UserDTO + convertToEntity(UserDTO): User + getAllUsers(): List<UserDTO> + getUserById(Long): UserDTO + getUserByUsername(String): UserDTO + login(String, String): UserDTO + createUser(User): User + updateUser(Long, User): User + deleteUser(Long): void

Server/Service

BoolToIntConverter
+ convertToDatabaseColumn(Boolean): Integer
+ convertToEntityAttribute(Integer): Boolean

Crypt
- saltRounds : int
+ encrypt(String): String
+ decrypt(String, String): boolean

JWT
- SECRET_KEY : String
- EXPIRATION_TIME : long
- key : SecretKey
+ header(): Header
+ generateToken(Map): String
+ validateToken(String): boolean
+ getPayload(String): Map<StringObject

YearConverter
- THAI_BUDDHIST_OFFSET : int
+ convertToDatabaseColumn(LocalDateTime): LocalDateTime
+ convertToEntityAttribute(LocalDateTime): LocalDateTime

CustomBanner
- out : PrintStream
- meChatVersion : String
- javaVersion : String
- springFrameworkVersion : String
- springBootVersion : String
- hibernateVersion : String
- mariadbVersion : String
+ printBanner(Environment, Class, PrintStream): void
+ printLogo(): void
+ printCaption(): void
+ print(String, String): void

Server/ Utils

BaseEntity
- id : Long - createdAt : LocalDateTime - updatedAt : LocalDateTime
+ getId(): Long + setId(Long): void + getCreatedAt(): LocalDateTime + setCreatedAt(LocalDateTime): void + getUpdatedAt(): LocalDateTime + setUpdatedAt(LocalDateTime): void # onCreate(): void # onUpdate(): void

Chat
- name : String - type : Type
+ getName(): String + setName(String): void + getType(): Type + setType(Type): void

ChatAttachments
- chatHistory : ChatHistory - filePath : String
+ getChatHistory(): ChatHistory + setChatHistory(ChatHistory): void

ChatHistory
- chat : Chat - user : User - message : String - edited : Boolean - deleted : Boolean
+ getChat(): Chat + setChat(Chat): void + getUser(): User + setUser(User): void + getMessage(): String + setMessage(String): void + getEdited(): Boolean + setEdited(Boolean): void + getDeleted(): Boolean + setDeleted(Boolean): void

ChatMember
- chat : Chat - user : User
+ getChat(): Chat + setChat(Chat): void + getUser(): User + setUser(User): void

Friend
- user : User - friend : User - status : Status - acceptedAt : LocalDateTime
+ getUser(): User + setUser(User): void + getFriend(): User + setFriend(User): void + getStatus(): Status + setStatus(Status): void + getAcceptedAt(): LocalDateTime

User
- username : String - password : String - displayName : String - avatar : String
+ getUsername(): String + setUsername(String): void + getPassword(): String + setPassword(String): void + verifyPassword(String): boolean + getDisplayName(): String + setDisplayName(String): void + getAvatar(): String + setAvatar(String): void

Server/Entity

## Manual

1. ติดตั้ง Java JDK 21 จาก [Java Archive Downloads - Java SE 21](#)
2. โหลด Source Code จาก [Releases · MeChat/256702-F2-Team06-108-175](#)
3. ขั้นตอนการติดตั้ง Server
  - 3.1. รันสคริปต์ build.sh หรือ build.bat เพื่อสร้างไฟล์ server.jar ภายในโฟลเดอร์ Build  
ตัวอย่างการใช้งาน  

```
$ sh build.sh (Shell Script)
```

```
$ build.bat (Command Prompt)
```
  - 3.2. เปลี่ยนชื่อไฟล์ application.yml.example และ .env.example ให้นำ .example ออก
  - 3.3. กำหนดข้อมูลใน .env และ application.yml
  - 3.4. พิมพ์คำสั่ง  

```
$ java -jar server.jar
```
4. ขั้นตอนการติดตั้ง Client
  - 4.1. รันสคริปต์ build.sh หรือ build.bat เพื่อสร้างไฟล์ client.jar ภายในโฟลเดอร์ Build  
ตัวอย่างการใช้งาน  

```
$ sh build.sh (Shell Script)
```

```
$ build.bat (Command Prompt)
```
  - 4.2. รันสคริปต์ mechat.bat เพื่อเริ่มต้นใช้งาน Client