Yeungnam University

LIFT ON

BEST ASSISTANT FOR YOUR LIFT LIFE

21912067
김동현
tmdrl583205@naver.com
- Design -

[Revision history]

Revision date	Version #	Description	Author
2023/06/04	0.00	First Documentation	Kim Dong Hyun

= Contents =

1. Introduction

현재 많은 사람들이 피트니스를 즐기고 있다. 이러한 시점에서 해당 프로젝트는, 사람들에게 운동에 관한 정보를 제공, 편리성을 제공하기 위해 만들어졌다.

사람들이 쉽게 피트니스에 접근하지만, 이에 대해 올바른 정보를 얻고, 규칙적으로 수행하는 것은 쉽지 않기에, 이에 조금이라도 더 도움을 주고자 한다.

앞선 단계에서 수행한 Conceptualization, Analysis 단계의 목적을 수행하기 위해 다양한 다이어그램을 이용하고, 이를 기반으로 마지막 단계인 구현 단계에서 프로젝트를 성공적으로 마무리 지을 것이다.

이를 위해서, 해당 Design 단계가 매우 중요하다. 실제로 구현에 쓸 코드들을 설계, 구체화하기 위해, 클래스 다이어그램을 최대한 실제 코드와 가깝게, 사용할 수 있도록 설계하고, 어플리케이션의 동작을 Sequence 다이어그램과 State Machine 다이어그램을 이용하고자 한다.

마지막으로 더 자세히, 원활한 구현과 안정적인 유지 보수를 위해 MVC 패턴을 따르며 Flutter 프레임워크와 Express 프레임워크를 사용할 예정이다. 아래의 단계들은 모두 Flutter 프레임워크의 설계에 초점을 맞추었다.

2. Class diagram

2.1. 서론

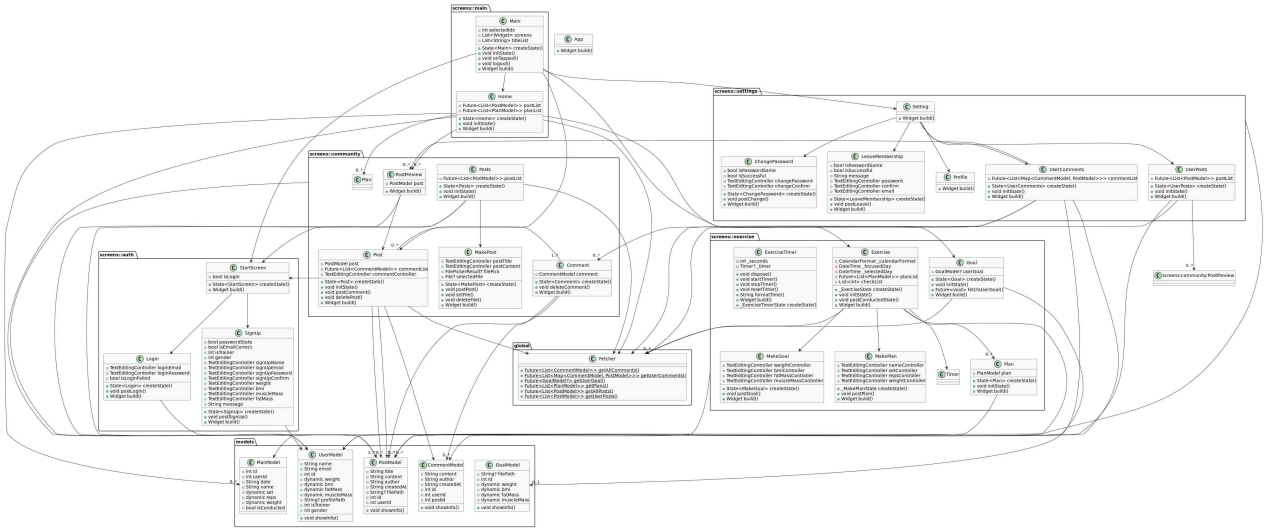
본 어플리케이션의 클래스 다이어그램은 plantuml을 통해 설계하였다. 또한 UML model은 package, class, association을 중점으로 한다. 추가로 Flutter 프레임워크의 Default 패키지, 클래스들은 따로 다이어그램에 표현하지 않았다. 그리고, Flutter 프레임워크의 특성상 Stateful Widget은 부모 클래스와 자식 클래스인 State 클래스로 나뉘는데, 이는 사실상 같은 클래스이기 때문에 한 클래스로 표시했다.

마지막으로, 해당 어플리케이션을 최대한 실용적이며 현실에 가깝게 구현할 예정이기 때문에, 많은 양의 클래스들이 추가되었다.

그러므로, 먼저 전체적인 구조를 보여주고, 그 다음 세부적인 기능에 따른 다이어 그램을 소개할 것이다.

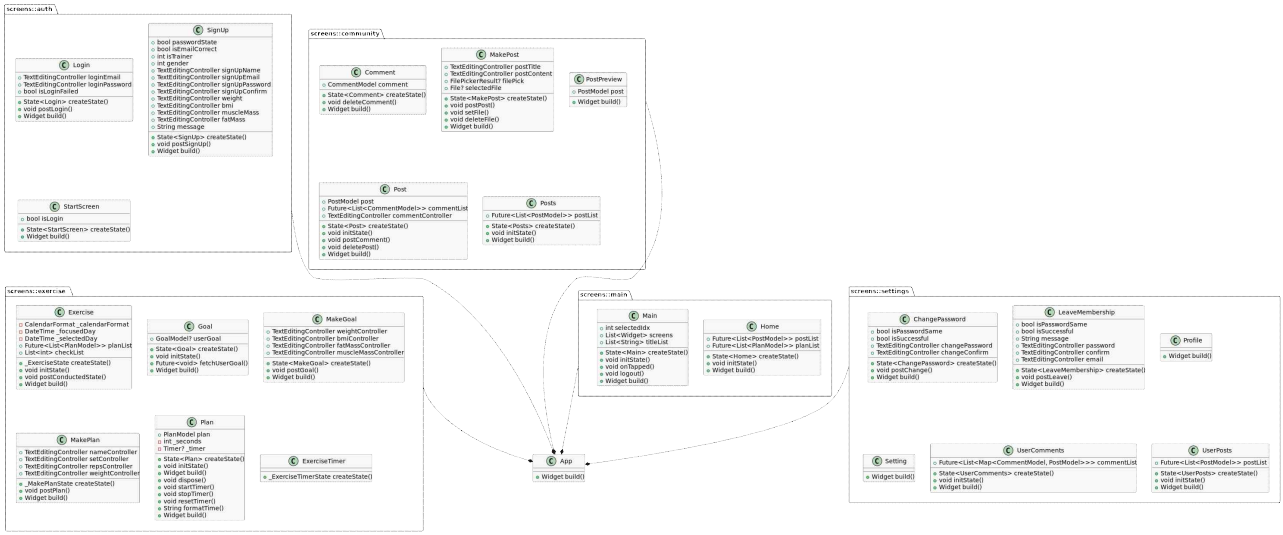
참고할 점이 여러 가지 있는데, 첫 번째로 이 클래스 다이어그램을 볼 때 클래스 하나에 집중하는 것보다는 클래스를 담고 있는 Package, 즉 폴더들의 이름과 구조에 집중하는 것이 좋을 것이다. 다음은 대부분의 클래스들이 가지는 build 메서드인데, 사용자에게 제공할 화면을 렌더링해주기 위한, 즉 보여주기 위한 메서드이다.

2.2. 전체 다이어그램



다양한 기능을 구현함에 따라 다양한, 방대한 클래스들이 추가되었기 때문에 해당 목차에서는 이러한 기능들을 일일이 설명할 수 없다. 앞서 말했듯이, 전체적인 모습을 보여주고 자세한 설명은 아래에서 진행하겠다.

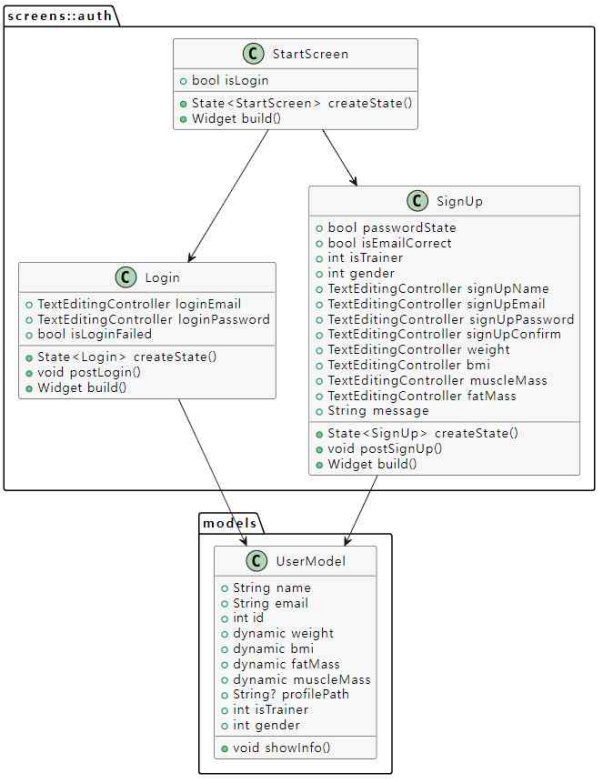
2.3 Screens



먼저, Flutter 프레임워크는 프론트 엔드에서 주로 사용하는 프레임워크이며, 위젯이라고 부르는 클래스들의 관계가 매우 중요하다. 클래스들을 어플리케이션의 용도에 맞게, 잘 꾸며 놓음으로서 유저 인터페이스의 구성을 알맞게 구현할 수 있다. 앞서 말했듯, 하나의 클래스에 집중하기보다는 그 클래스들을 가지는 패키지에 집중할 것이기 때문에 모든 설명은 패키지를 중심으로 하되, 필요시 클래스에 대해 자세히 설명하겠다. 일단 해당 다이어그램에서는, 전체적인 구조를 살펴볼 수 있다. 가장 중요한 점은 모든 클래스, 패키지가 App 클래스 위에 놓여져 있다는 점이다.

2.4 Auth

- 사용자가 로그인, 회원가입을 진행할 수 있는 클래스들의 모음이다.

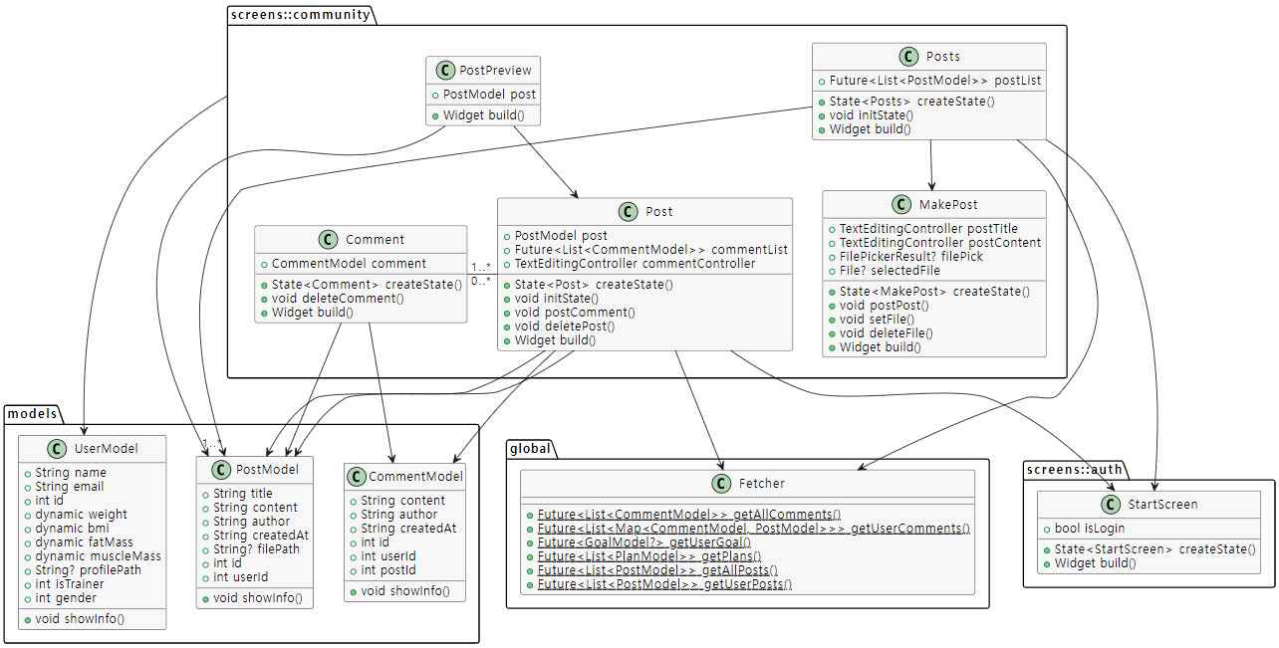


Package : Class	Explanation
models : UserModel	- 사용자의 모든 정보가 담겨져 있는 클래스이다.
Auth : StartScreen	- 시스템의 여러 가지 기능을 다루기 위해서는 먼저 로그인을 해야 하는데, 해당 클래스에서 로그인 또는 회원가입 화면을 제공해준다. - isLogin 변수를 통해 사용자가 로그인 화면을 선택할지, 회원가입 화면을 선택할지 정할 수 있다. - 위의 기능을 구현하기 위해서 동일 Package의 로그인, 회원가입 클래스를 사용한다.
Auth : Login	- 사용자가 로그인할 화면이다. Default 클래스인 TextEditingController를 통해 사용자의 입력을 받을 수 있다. 사용자의 입력을 받고 버튼을 클릭하면, postLogin 메서드를 통해 서버에 로그인을 요청한다. - isLoginFailed 변수를 통해 사용자가 로그인에 성공했는지 실패했는지의 상태를 저장하고 이에 따라 화면에 표시할 수 있다.
Auth : SignUp	- 앞서 말한 로그인과 비슷하게 TextEditingController

	를 이용해 사용자의 여러 가지 입력을 받아낸다. 회원가입 클래스에서는 사용자에게 입력받아야 할 정보들이 많다. 따라서 변수들도 매우 많이 추가되었다. - isTrainer, gender 변수를 통해 각각 사용자가 일반 사용자인지, 트레이너인지 그리고, 남성인지 여성인지의 정보를 저장한다. - passwordState 변수를 통해 사용자의 비밀번호와 확인용 비밀번호가 일치하는지 확인할 정보를 저장한다. - isEmailCorrect 변수를 통해 사용자의 이메일이 이메일 양식에 맞는지를 확인한다.

2.5 Community

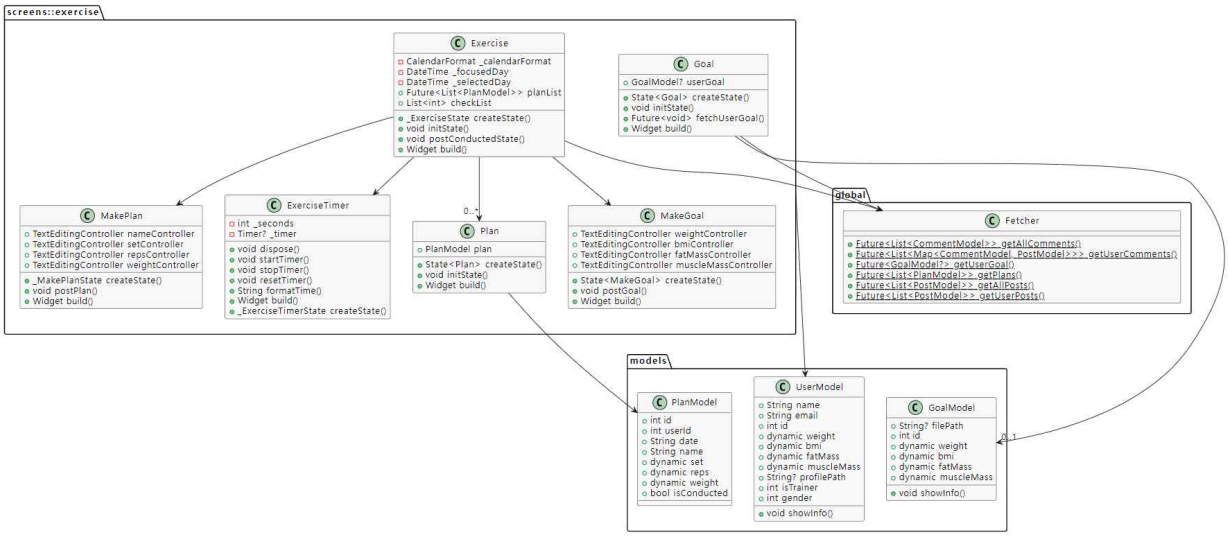
- 사용자가 커뮤니티 활동을 할 수 있도록 지원하는 클래스들의 모음이다.



Package : Class	Explanation
Global : Fetcher	- 각 클래스들에 필요한 여러 가지 정보들을 서버로부터 받아와 클래스들에 가져다주는 클래스이다. 주로 정적 메소드들이 주를 이룬다.
Auth : StartScreen	- 사용자가 커뮤니티 활동을 하며 게시글을 등록하고 댓글을 쓰기 위해서는 로그인이 필요하다. 이를 위해 로그인과 회원가입을 지원하는 클래스이다.
Models : UserModel	- 현재 로그인되어있는 사용자를 나타내는 클래스이다.
Models : PostModel	- 게시글의 정보를 가지고 있는 클래스이다. - Post 클래스가 하나의 PostModel 클래스를 가진다.
Models : CommentModel	- 게시글에 등록되어 있는 댓글의 정보를 가지고 있는 클래스이다. - Post 클래스가 여러 개의 CommentModel을 가진다.
Community : Posts	- 사용자가 커뮤니티 화면에 입장했을 때, Post들의 전체적인 Preview를 나타낸다. 여러 개의 PostModel과 그에 따른 여러 개의 PostPreview를 가진다.
Community : Post	- Fetcher를 통해 해당 게시글의 정보를 가지고 오며, 그 정보는 PostModel 클래스가 가진다. 해당 정보

	를 화면에 띄워주는 클래스이다. - 또한 여러 개의 댓글을 가지며 이를 Comment 클래스를 통해 화면에 띄워준다.
Community : MakePost	- Posts 클래스에서 버튼을 누르면 MakePost 클래스로 넘어오는데, 해당 클래스에서 사용자가 게시글을 등록할 수 있다. - Flutter 프레임워크의 Default 클래스인 File, FilePickerResult 클래스를 통해 이미지 파일도 서버에 등록할 수 있다.
Community : Comment	- 해당 클래스는 하나의 CommentModel을 가지고, 그 클래스가 가지고 있는 정보를 가져와 사용자가 볼 수 있도록 화면에 띄워준다. 반드시 하나 이상의 Post에 종속되어 있다.
Community : PostPreview	- PostModel의 정보를 가져와 사용자에게 표시해주는데, Preview 목적을 가지는 클래스임으로 일부의 정보만 띄워주고 해당 클래스를 화면에서 클릭하면 Post 클래스로 넘어가 모든 내용을 살펴볼 수 있도록 한다.

2.6 Exercise

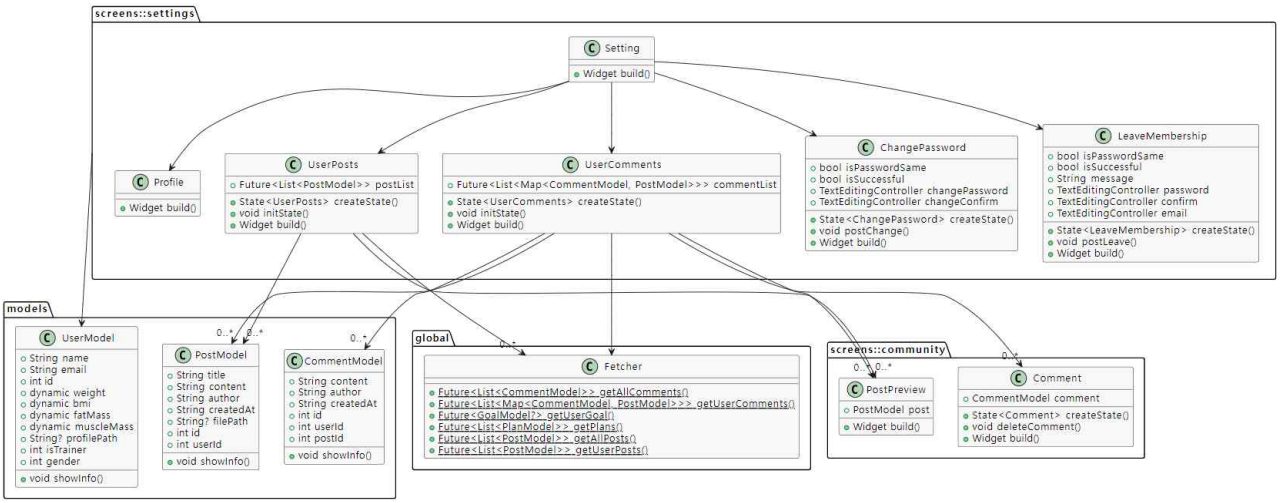


Package : Class	Explanation
Global : Fetcher	- 각 클래스들에 필요한 여러 가지 정보들을 서버로부터 받아와 클래스들에 가져다주는 클래스이다. 주로 정적 메소드들이 주를 이룬다.
Models : UserModel	- 현재 로그인되어있는 사용자를 나타내는 클래스이다. 사용자가 원활히 운동에 관한 활동을 하기 위해 필요하다.
Models : PlanModel	- 사용자의 운동 계획에 관한 클래스이다.
Models : GoalModel	- 사용자의 운동 목표에 관한 클래스이다.
Exercise : Exercise	- 어플리케이션의 운동에 관련된 모든 클래스, 기능의 중점이 되는 클래스이다. 해당 클래스에서, 운동 목표 작성, 계획 작성 등으로 넘어갈 수 있으며, 캘린더와 다른 Goal, Plan 클래스를 통해 다양한 정보를 확인할 수 있다.
Exercise : Goal	- 해당 클래스는 Exercise 화면에서 띄어지며, fetchUserGoal 메서드를 통해 먼저 서버로부터 사용자의 운동 목표에 관한 데이터를 가져온 뒤 Exercise 화면에 띄어준다.
Exercise : MakeGoal	- 사용자의 운동 목표를 작성하는 클래스로, TextEditingController 클래스를 통해 사용자의 입력을 받고, postGoal 메서드를 통해 서버로 데이터를 보내 저장할 수 있다.
Exercise : Plan	- Exercise 메인 화면에 띄어지는 클래스로, Exercise 클래스가 정보를 Fetcher 클래스를 통해 받아오면

	그 정보를 PlanModel에 저장하고 Plan 클래스가 그 PlanModel의 정보를 화면에 띄어준다.
Exercise : ExerciseTimer	- 사용자의 운동 시간을 측정하는 클래스이다. Start, Stop, Reset 등의 다양한 버튼을 가지고 사용자에게 편리한 기능을 제공한다.
Exercise : MakePlan	- 사용자의 운동 계획을 작성하는 클래스로, TextEditingController 클래스를 통해 사용자의 입력을 받고, postPlan 메서드를 통해 서버로 데이터를 보내 저장할 수 있다.

2.7 Settings

- 사용자가 계정 탈퇴, 비밀번호 변경 등의 정보를 수정하는 활동 또는 자신의 커뮤니티 활동 등을 볼 수 있는 클래스들의 모음이다.

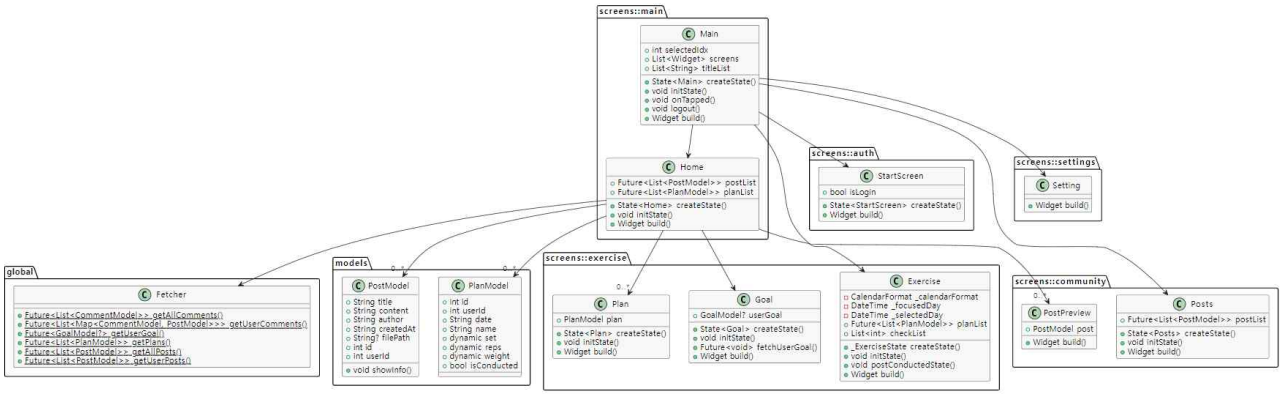


Package : Class	Explanation
Global : Fetcher	- 각 클래스들에 필요한 여러 가지 정보들을 서버로부터 받아와 클래스들에 가져다주는 클래스이다. 주로 정적 메소드들이 주를 이룬다.
Models : UserModel	- 현재 로그인 되어있는 사용자를 나타내는 클래스이다. Setting 클래스에서는, 당연히 사용자가 누구인지 알아야하고 따라서 로그인되어 있어야 하므로 해당 클래스가 필요하다.
Models : PostModel	- 게시물에 관한 정보를 가지는 클래스이다.
Models : CommentModel	- 게시물의 댓글의 정보를 가지는 클래스이다.
Settings : Setting	- Setting에 관련된 모든 위젯, 클래스를 띄어주는 클래스이다. - Setting 클래스는 동일 패키지의 다른 모든 클래스들을 가진다.
Settings : Profile	- 사용자의 프로필을 띄어주기 위한 클래스이다.
Settings : ChangePassword	- 사용자가 비밀번호를 변경하는 클래스이다. - isPasswordSame 변수를 통해 새로운 비밀번호와 확인용 비밀번호가 일치하는지 확인하고 이에 따른 메시지를 화면에 띄어준다. - postChange 메서드를 통해 서버에 변경된 비밀번호를 저장한다.
Settings : LeaveMembership	- 사용자의 회원탈퇴를 지원하는 클래스이다. - 사용자의 입력을 TextEditingController 클래스로 입

	력받고 이에 따라 패스워드가 일치하는지의 여부를 화면에 띄어준다. - 비밀번호 변경 로직과 비슷하게 isPasswordSame 변수를 통해 비밀번호 일치 여부를 확인하고 그에 따른 메시지를 화면에 띄어준다. - postLeave 메서드를 통해 서버로 사용자의 회원탈퇴를 알리는 메시지를 전송한다.
Settings : UserPosts	- 사용자가 지금까지 작성한 게시물들을 볼 수 있는 클래스이다. PostModel을 하나도 가지지 않거나, 또는 여러 PostModel들을 가질 수 있다. - PostModel의 정보를 Community 패키지의 PostPreview 클래스를 통해 화면에 띄어주고, 사용자가 이를 확인할 수 있다. PostPreview 클래스를 클릭하면, Post 클래스를 통해 더 자세히 게시글에 관한 정보를 확인할 수 있다.
Settings : UserComments	- 사용자가 지금까지 작성한 댓글들을 볼 수 있는 클래스이다. ComentModel을 하나도 가지지 않거나, 또는 여러 ComentModel을 가질 수 있다. - 댓글이 등록되어 있는 게시글의 정보도 함께 화면에 띄어주기 위해, PostModel의 정보도 UserComment 클래스에서 사용한다.

2.8 Main

- 모든 화면의 시작 또는 기초가 되는 클래스들의 모음이며, 다른 패키지들의 여러 클래스들을 사용한다.

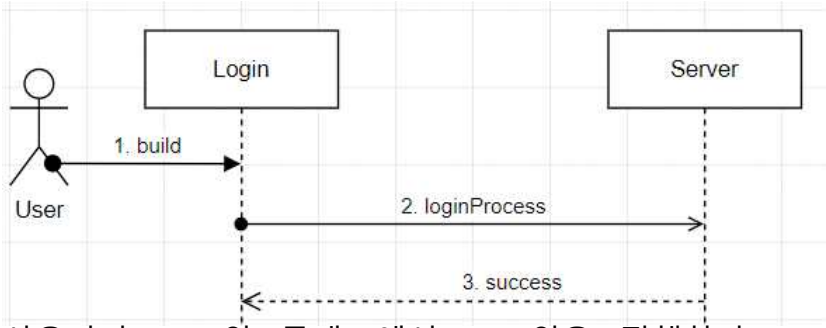


Package : Class	Explanation
Global : Fetcher	- 각 클래스들에 필요한 여러 가지 정보들을 서버로부터 받아와 클래스들에 가져다주는 클래스이다. 주로 정적 메소드들이 주를 이룬다.
Models : PostModel	- 게시물에 관한 정보를 가지는 클래스이다. - 홈 화면에서 가장 최근의 게시글들을 띄어주기 위해 필요하다.
Models : PlanModel	- 사용자의 운동 계획에 관한 정보를 가지는 클래스이다. - 홈 화면에서 여러 계획들을 미리 띄어주기 위해 필요하다.
Exercise : Plan	- 홈 화면에서 사용자에게 미리 여러 계획들을 띄어주는 클래스이다.
Exercise : Goal	- 홈 화면에서 사용자에게 미리 사용자의 목표를 띄어주는 클래스이다.
Exercise : Exercise	- 메인 화면이 가지는 클래스들 중 하나이다. 운동 관련 클래스이며, 메인 화면 위에서 작동한다.
Settings : Setting	- 메인 화면이 가지는 클래스로, 설정 관련 클래스이다. 메인 화면 위에서 작동한다.
Auth : StartScreen	- 메인 화면 또는 홈 화면에서 필요 시에 로그인 또는 회원가입을 지원하는 클래스이다.
Community : Posts	- 메인 클래스가 가지는 클래스로, 커뮤니티의 게시글들을 확인 할 수 있다. 마찬가지로 메인화면 위에서 작동한다.
Main : Main	- 어플리케이션의 시각적인 부분에서 가장 메인이 되는 클래스로, 모든 위젯이 해당 클래스 위에서 작동한다. 화면 맨 밑의 하단 바를 통해 클래스들이 선

	택될 수 있다.
Main : Home	- 어플리케이션을 맨 처음 구동할 때 가장 처음에 띄어지는 클래스, 화면이다. 다양한 정보를 띄우며, 사용자에게 편리성을 제공한다.

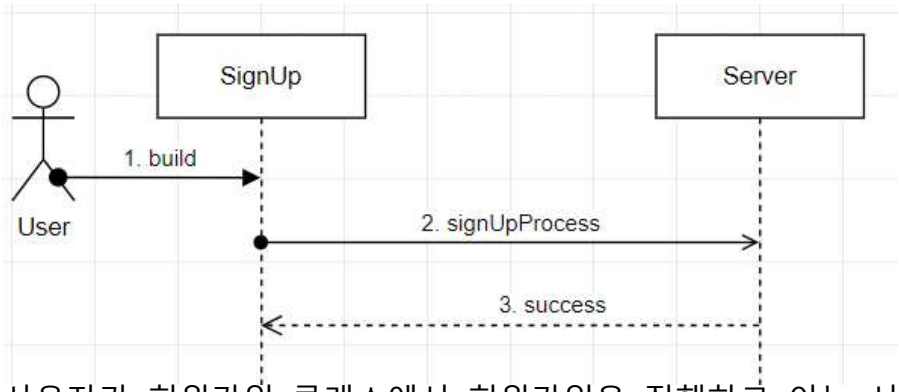
3. Sequence diagram

3.1 Login Sequence Diagram



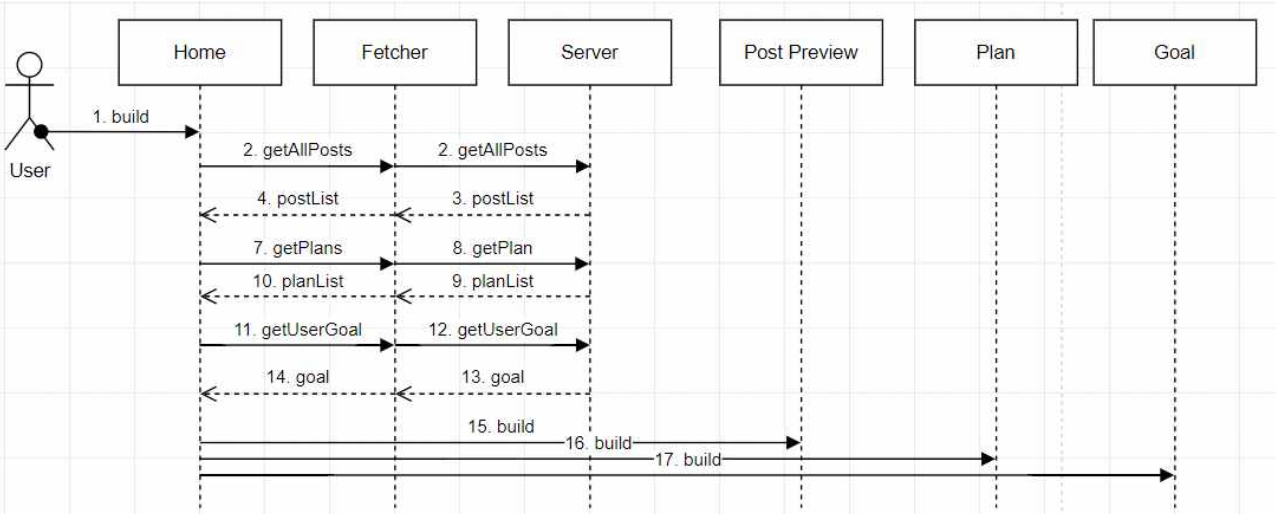
사용자가 로그인 클래스에서 로그인을 진행한다. 로그인 정보는 서버로 전달되고, success 변수를 통해 로그인 성공, 실패 여부를 확인한다.

3.2 SignUp Sequence Diagram



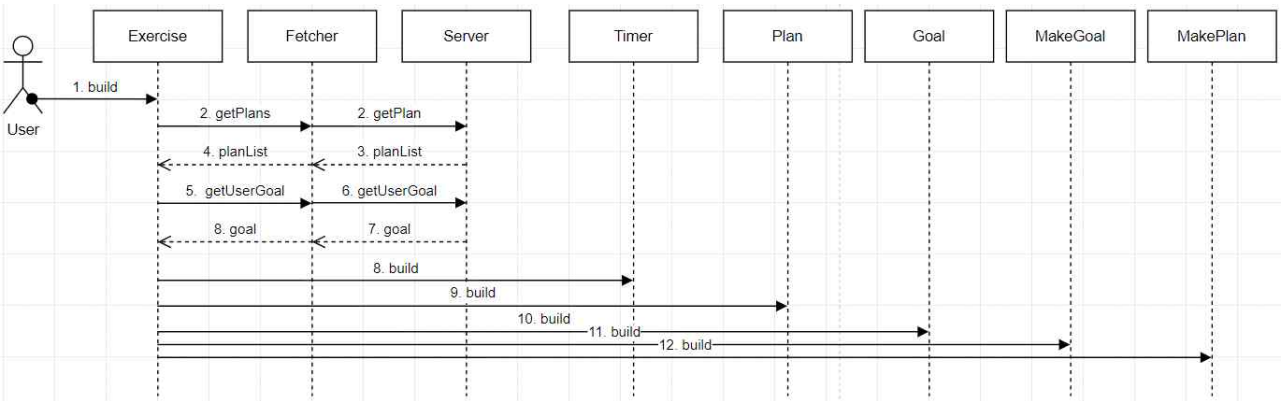
사용자가 회원가입 클래스에서 회원가입을 진행하고 이는 서버로 전달된다. 성공, 실패 여부는 success 변수를 통해 확인한다.

3.3 Home Sequence Diagram



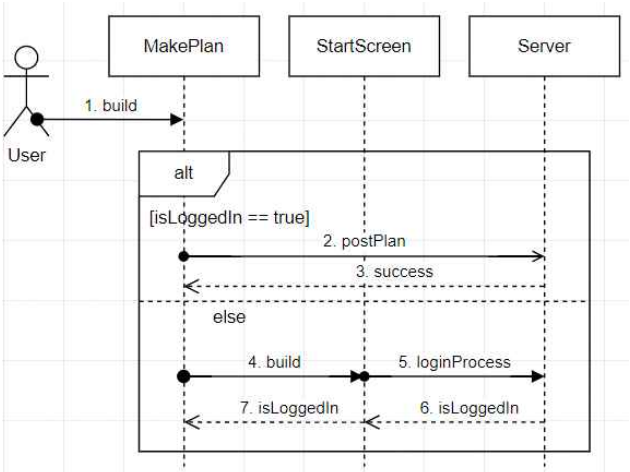
사용자가 맨 처음 어플리케이션의 시작에서 홈 화면에 온 시점이다. 먼저 사용자에게 다양한 화면을 띄어주기 위해, Fetcher 클래스를 호출하여 Posts, Plan, Goal을 가져오고, Fetcher는 서버에 정보를 요청한다. 이러한 데이터들을 성공적으로 받으면, Post Preview, Plan, Goal 클래스에 데이터를 보내 렌더링하고, 사용자에게 보여준다.

3.4 Exercise Sequence Diagram



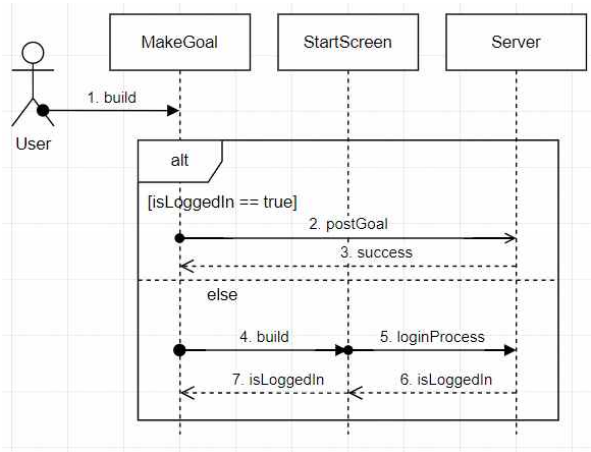
사용자가 홈 화면 다음 Exercise 화면으로 넘어오게 되면, 홈 화면과 마찬가지로 사용자의 데이터를 띄어주기 위해 Fetcher 클래스를 통해 서버에 다양한 정보를 요청한다. Plan, Goal을 요청하고 Fetcher가 Server에 이 데이터를 받아 Exercise 클래스에 넘겨준다. Exercise는 이 데이터들을 각각 Plan, Goal 클래스에 넘겨주고 build 메소드를 호출해 화면에 렌더링한다. 또한 사용자의 편의와 기능을 위한 Timer, Make Goal, Make Plan 클래스를 렌더링하여 사용자에게 보여준다.

3.5 Make Plan Sequence Diagram



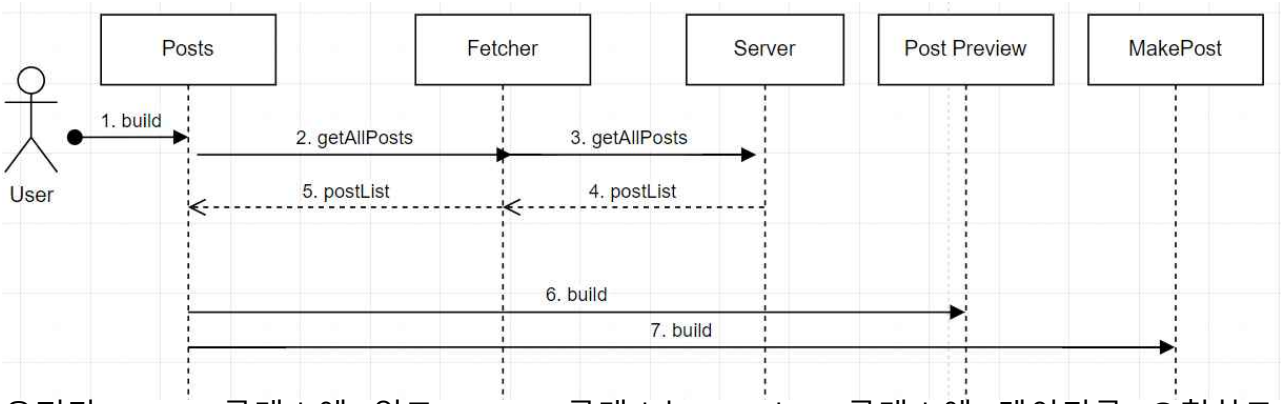
사용자가 Exercise 클래스에서 MakePlan을 누르면, 사용자는 자신의 계획을 등록할 수 있다. Exercise 화면에서 선택한 날짜를 바탕으로, 자신의 계획을 세우고, 이를 서버에 등록할 수 있다. 하지만 이는 기본적으로 로그인이 되어있다는 전제하에 수행되는 것들이기 때문에 isLoggedIn 변수를 통해 로그인이 되어있지 않다면 StartScreen 클래스로 넘어가 로그인을 수행할 것이다.

3.6 Make Goal Sequence Diagram



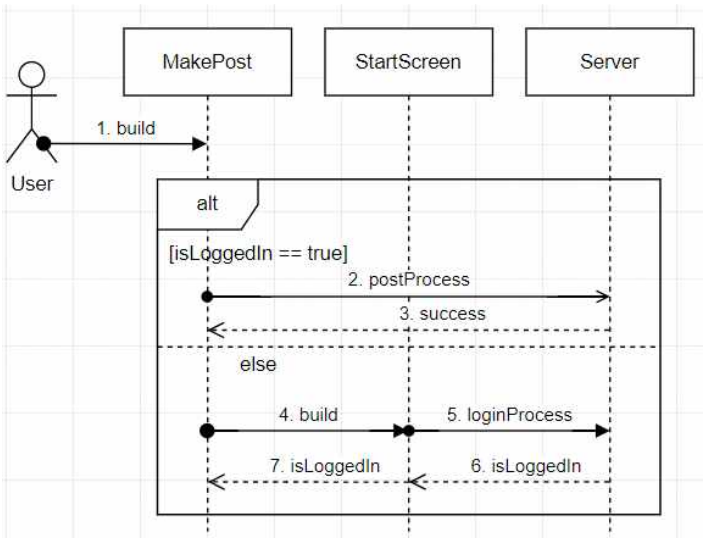
사용자가 Exercise 클래스에서 MakeGoal 버튼을 누르면, 사용자는 자신의 운동 목표를 서버에 등록할 수 있다. 하지만 Make Plan과 마찬가지로그인이 되어있다는 전제하에 수행되는 것들이기 때문에 isLoggedIn 변수를 통해 로그인이 되어있지 않다면 StartScreen 클래스로 넘어가 로그인을 수행할 것이다.

3.7 Posts Sequence Diagram



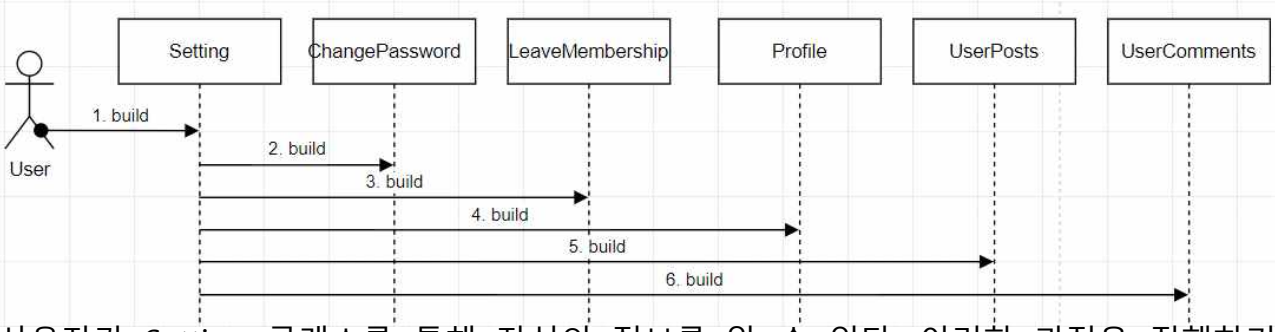
유저가 Posts 클래스에 있고, Posts 클래스는 Fetcher 클래스에 데이터를 요청하고, Fetcher 클래스는 다시 서버에 데이터를 요청하고 반환받아 이를 그대로 Posts 클래스에 가져다 준다. Posts 클래스는 Post Preview 클래스에 데이터를 전달하고 Post Preview 클래스는 이를 렌더링하여 사용자에게 보여준다. 또한 Make Post 위젯으로 넘어가기 위한 Make Post 버튼도 빌드하여 렌더링한다.

3.8 Make Post Sequence Diagram



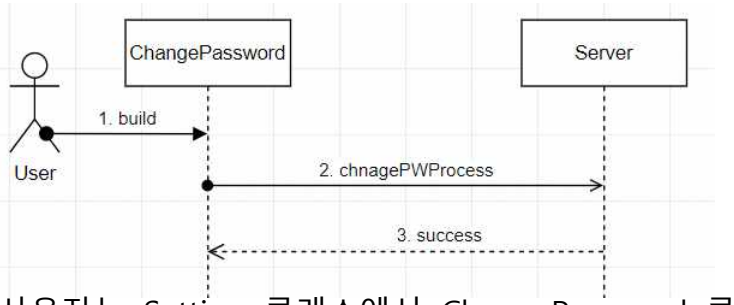
사용자가 Posts 클래스에서 Make Post 버튼을 누르고 사용자가 게시글을 작성해 이에 대한 정보를 서버로 보내고, 서버는 이를 저장한 뒤 게시글 저장 성공 여부를 반환한다. 만약 로그인되어 있지 않다면 게시글을 작성할 수 없고 먼저 StartScreen 클래스를 통해 로그인 또는 회원가입을 우선적으로 진행하게 된다.

3.9 Setting Sequence Diagram



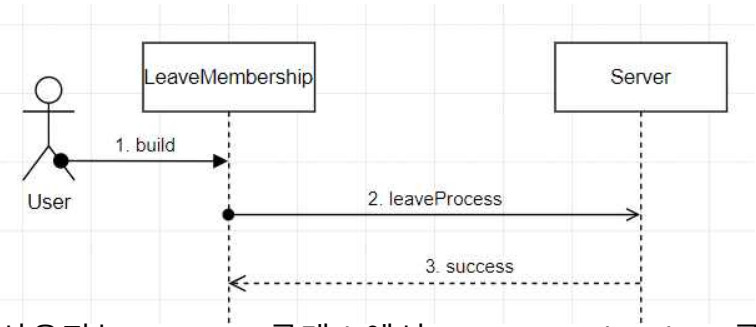
사용자가 Setting 클래스를 통해 자신의 정보를 알 수 있다. 이러한 과정을 진행하기 위해 다른 클래스들을 우선 build 한다.

3.10 ChangePassword Sequence Diagram



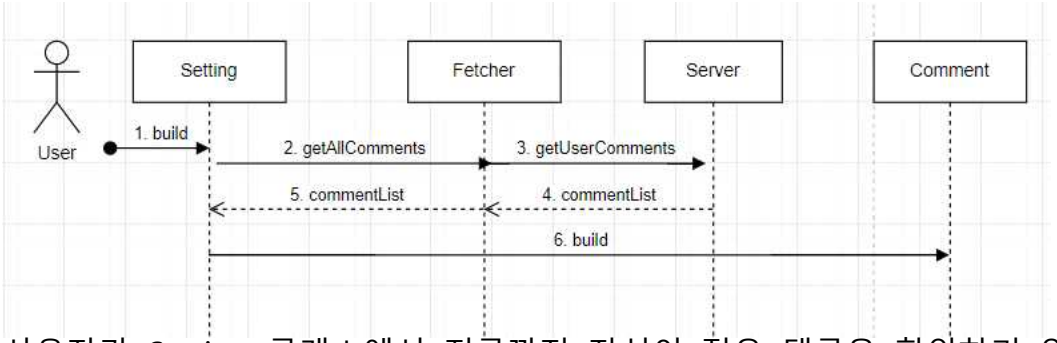
사용자는 Setting 클래스에서 ChangePassword 클래스로 넘어오고, 변경할 비밀번호를 입력한 다음 서버에 이러한 데이터를 보내고, 서버는 변경된 비밀번호를 저장한다. 마지막으로 서버가 성공 여부를 반환한다.

3.11 Leave Membership Sequence Diagram



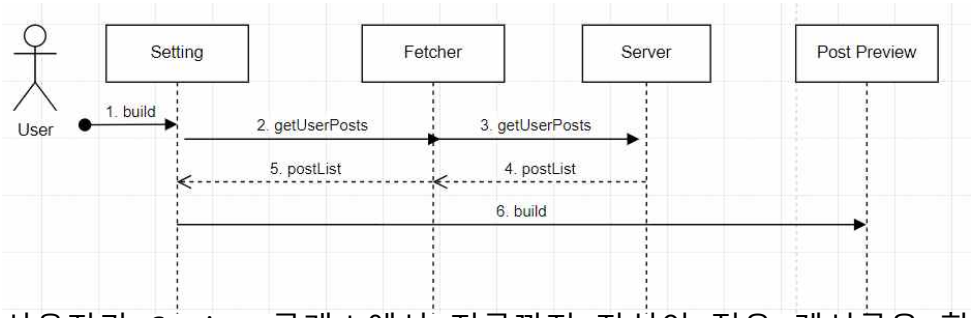
사용자는 Setting 클래스에서 LeaveMembership 클래스로 넘어오고, 사용자가 자신의 개인 정보를 확인한 뒤, 서버에 회원 탈퇴를 요청하면 서버가 그 유저에 관한 정보를 지우고 회원 탈퇴 처리한다. 마찬가지로 성공 여부는 서버가 반환한다.

3.12 UserComments Sequence Diagram



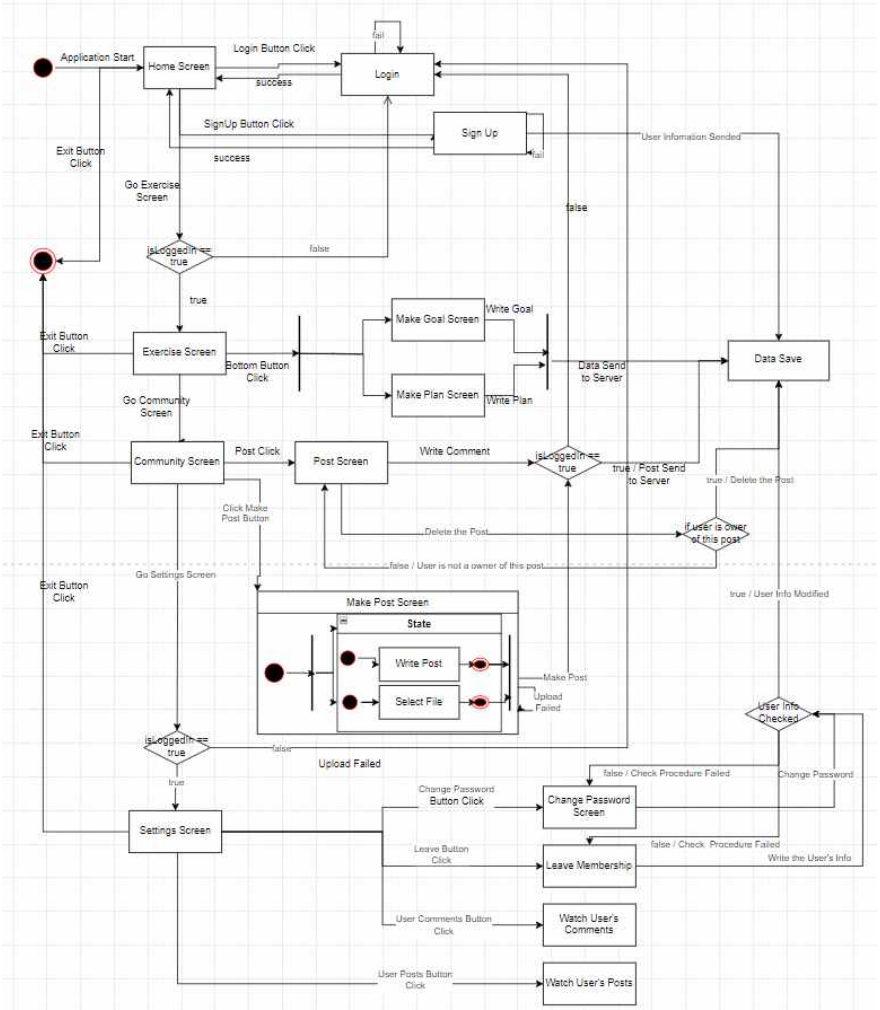
사용자가 Setting 클래스에서 지금까지 자신이 적은 댓글을 확인하기 위해, Fetcher 클래스를 통해 서버로 자신의 모든 댓글 정보를 요청하고, 서버는 이를 반환한다. 이 데이터를 Comment 클래스로 보내 빌드, 렌더링해 화면에 띄운다.

3.13 UserPosts Sequence Diagram



사용자가 Setting 클래스에서 지금까지 자신이 적은 게시글을 확인하기 위해, Fetcher 클래스를 통해 서버로 자신의 모든 게시글 정보를 요청하고, 서버는 이를 반환한다. 이 데이터를 Post Preview 클래스로 보내 빌드, 렌더링해 화면에 띄운다.

4. State machine diagram



어플리케이션 실행시 맨 처음 홈 화면에서 시작한다. 홈 화면에서 상단바를 누를 시 로그인 또는 회원가입으로 이동한다. 그 다음 Exercise 화면으로 넘어가는데, 이 때 로그인이 되어 있어야 정보를 볼 수 있기 때문에 로그인 화면으로 넘어가거나, 로그인이 되어있다면 Exercise 화면으로 넘어간다. 이 때 하단의 버튼을 클릭하면, 운동 계획을 짜거나 운동 목표를 설정할 수 있다. 데이터를 작성하고 이를 서버에 보내 저장한다. 다음으로는 커뮤니티 화면인데, 커뮤니티에서 게시물 자체를 열람하는 것은 로그인할 필요가 없다. 하지만, 게시글을 작성할 시 로그인이 필요하다. 로그인 후 게시글을 작성할 때, 글을 작성하고 파일을 선택하고, 데이터를 서버로 보내 저장한다.

Setting 화면에서는, 비밀번호 변경, 회원 탈퇴, 유저의 댓글 열람, 유저의 게시글 열람 등을 선택할 수 있다. 비밀번호 변경과 회원 탈퇴에서는 유저의 개인 정보를 확인한 후, 서버에 데이터를 보내게 된다. 마지막으로, 어떤 화면에서든 어플리케이션을 닫게 되면, 어플리케이션이 종료된다.

5. Implementation requirements

- Flutter Framework
- node js
- npm
- Visual Studio Code

6. Glossary

- MVC 패턴 : Model, View, Controller 패턴으로, 사용자 인터페이스, 데이터 및 논리 제어를 구현하는데 널리 사용되는 소프트웨어 디자인 패턴
- Flutter 프레임워크 : 구글에서 출시한 모바일/웹/데스크톱 크로스 플랫폼 GUI SDK
- Node js : 오픈 소스 JavaScript 엔진인 크롬 V8에 비동기 이벤트 처리 라이브러리인 libuv를 결합한 플랫폼
- Express js : Node js를 기반으로 하며 Node.js의 핵심 모듈인 http와 Connect 컴포넌트를 기반으로 하는 웹 프레임워크

7. References

- 소프트웨어 설계 강의 자료 6,7,8,9장
- plantuml 문서: https://plantuml.com/ko/class-diagram
- plantuml 에디터 : https://plantuml-editor.kkeisuke.dev/