Yeungnam University

3. Design

Travel Mate

22111138
오원창

[Revision history]

Revision date	Version #	Description	Author
05/19/2025	0.01	1차	오원창
06/03/2025	0.02	완성	

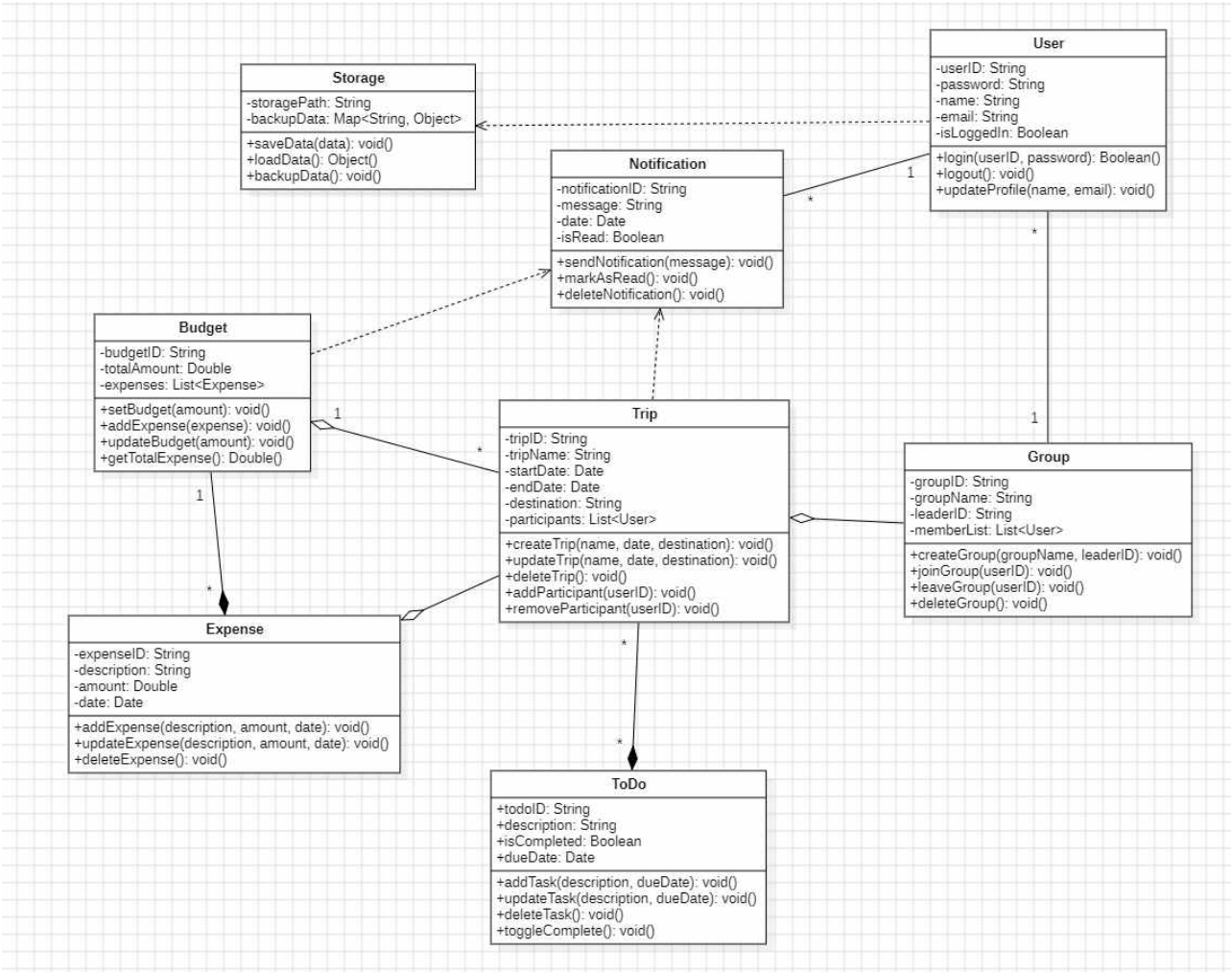
Yeungnam University

= Contents =

1. Introduction

여행을 준비하고 진행하는 과정에서 사용자들은 다양한 앱(캘린더, 가계부, 지도, 메신저 등)을 번갈아 사용해야 하는 불편함을 겪는다. 특히 그룹 여행 시에는 일정 변경, 예산 분배, 비용 정산 등의 복잡한 과정을 효과적으로 관리하기 어려운 문제가 있다.
이를 해결하기 위해 TravelMate는 여행 계획 수립, 예산 관리, 비용 정산, 일정 공유 기능을 하나의 플랫폼에서 통합 제공하는 올인원 여행 관리 앱으로 개발되었다.
TravelMate를 통해 사용자는 여행 관련 모든 정보를 하나의 시스템에서 손쉽게 관리할 수 있으며, 그룹원들과 실시간으로 일정을 공유하고, 예산을 계획하고, 지출을 정산하는 과정을 보다 효율적이고 편리하게 수행할 수 있다.
또한 실시간 알림 및 백업 기능을 통해 여행 중에도 변동사항을 빠르게 인지하고 안정적으로 데이터를 관리할 수 있다.
이번 Design 문서에서는 Class diagram, Sequence disagram, State machine diagram와 함께 이전보다 더욱 세부적인 사항을 설명하고 Implementation requirements을 통해 이 앱을 구현하기 위해 어떤 것이 필요한지 설명할 것이다

2. Class diagram



UI를 구성하는 자바 코드와 관련된 클래스들은 표현하지 않았다.

1. User

Attributes
-userID: String - 사용자 ID를 나타낸다 -password: String - 사용자의 비밀번호를 저장 -name: String - 사용자의 이름을 저장 -email: String - 사용자의 이메일을 저장 -isLoggedIn: Boolean - 로그인 상태를 나타낸다
Methods
+login(userID, password): Boolean - 사용자 ID와 비밀번호를 통해 로그인 수행 +logout(): void - 현재 사용자의 로그아웃을 수행 +updateProfile(name, email): void - 사용자 프로필 정보를 수정

2. Group

Attributes
-groupID: String - 그룹의 고유 ID를 저장 -groupName: String - 그룹 이름을 저장 -leaderID: String - 그룹의 리더 ID를 저장 -memberList: List<User> - 그룹에 속한 멤버들의 목록을 저장
Methods
+createGroup(groupName, leaderID): void - 새로운 그룹을 생성 +joinGroup(userID): void - 그룹에 사용자를 추가 +leaveGroup(userID): void - 그룹에서 사용자를 제거 +deleteGroup(): void - 그룹을 삭제

3. Trip

Attributes
-tripID: String - 여행 계획의 고유 ID를 저장 -tripName: String - 여행 계획의 이름을 저장 -startDate: Date - 여행 시작 날짜를 저장 -endDate: Date - 여행 종료 날짜를 저장 -destination: String - 여행 목적지를 저장 -participants: List<User> - 여행에 참여하는 사용자 목록
Methods
+createTrip(name, date, destination): void - 새로운 여행 계획을 생성 +updateTrip(name, date, destination): void - 여행 계획 정보를 수정 +deleteTrip(): void - 여행 계획을 삭제 +addParticipant(userID): void - 여행에 참여자를 추가 +removeParticipant(userID): void - 여행에서 참여자를 제거

4. ToDo

Attributes
-todoID: String - 할 일의 고유 ID를 저장 -description: String - 할 일 내용을 저장 -isCompleted: Boolean - 할 일 완료 여부를 저장 -dueDate: Date - 할 일의 마감일을 저장
Methods
+addTask(description, dueDate): void - 새로운 할 일을 추가 +updateTask(description, dueDate): void - 기존 할 일 정보를 수정 +deleteTask(): void - 할 일을 삭제 +toggleComplete(): void - 할 일의 완료 상태를 변경

5. Budget

Attributes
-budgetID: String - 예산 관리의 고유 ID를 저장 -totalAmount: Double - 전체 예산 금액을 저장 -expenses: List<Expense> - 지출 목록을 저장
Methods
+setBudget(amount): void - 예산 금액을 설정 +addExpense(expense): void - 새로운 지출 항목을 추가 +updateBudget(amount): void - 예산 금액을 수정 +getTotalExpense(): Double - 현재까지의 총 지출 금액을 계산하여 반환

6. Expense

Attributes
-expenseID: String - 지출 항목의 고유 ID를 저장 -description: String - 지출 항목 설명을 저장 -amount: Double - 지출 금액을 저장 -+date: Date - 지출 날짜를 저장
Methods
+addExpense(description, amount, date): void - 새로운 지출을 추가 +updateExpense(description, amount, date): void - 기존 지출을 수정 +deleteExpense(): void - 지출 항목을 삭제

7. Notification

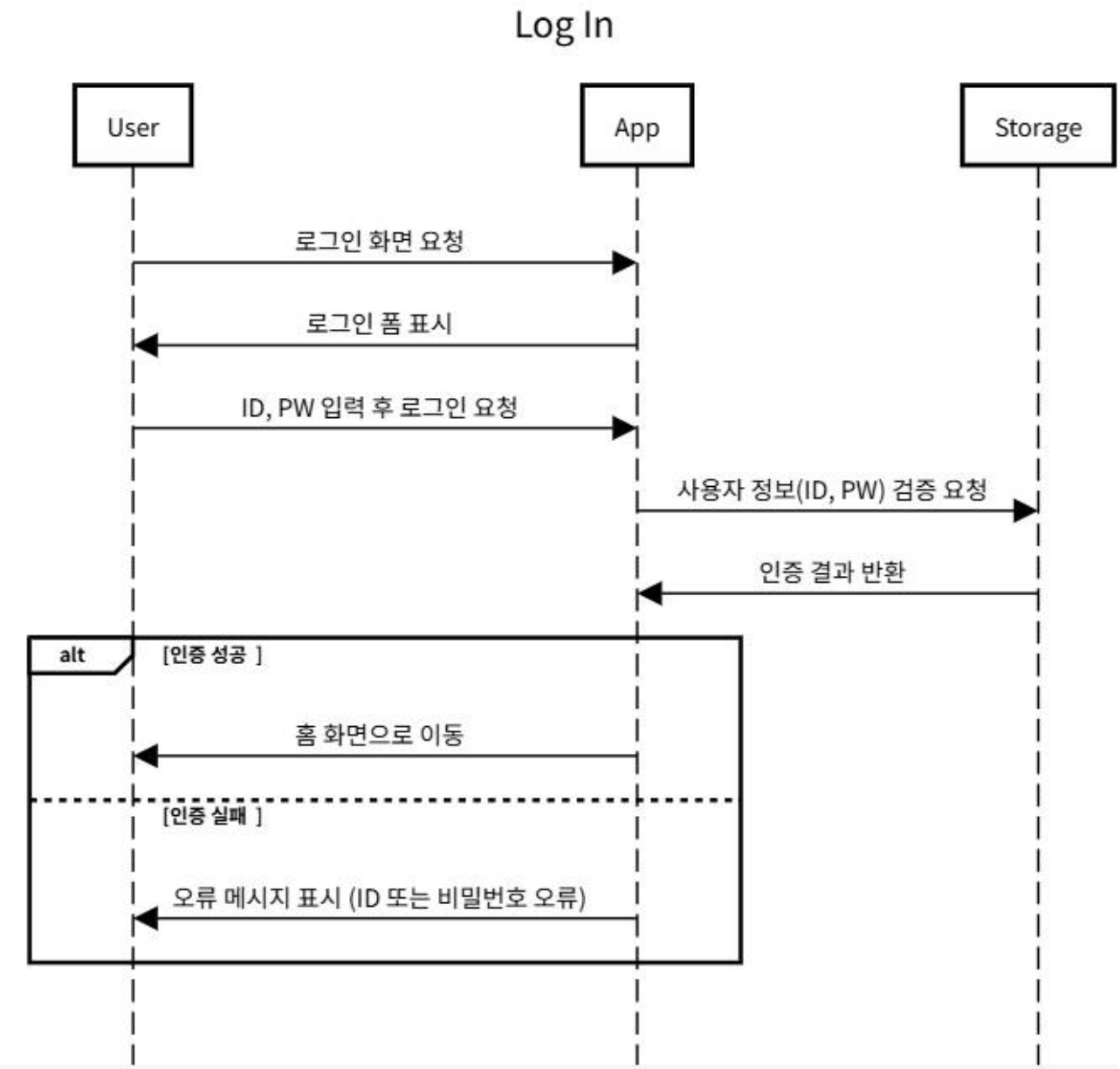
Attributes
-notificationID: String - 알림의 고유 ID를 저장 -message: String - 알림 내용을 저장 -date: Date - 알림 발생 날짜를 저장 -isRead: Boolean - 알림 확인 여부를 저장
Methods
+sendNotification(message): void - 새로운 알림을 전송 +markAsRead(): void - 알림을 읽음 상태로 표시 +deleteNotification(): void - 알림을 삭제

8. Storage

Attributes
-storagePath: String - 데이터가 저장될 경로를 저장 -backupData: Map<String, Object> - 백업 데이터 저장
Methods
+saveData(data): void - 데이터를 저장 +loadData(): Object - 저장된 데이터를 불러옴 +backupData(): void - 데이터를 백업

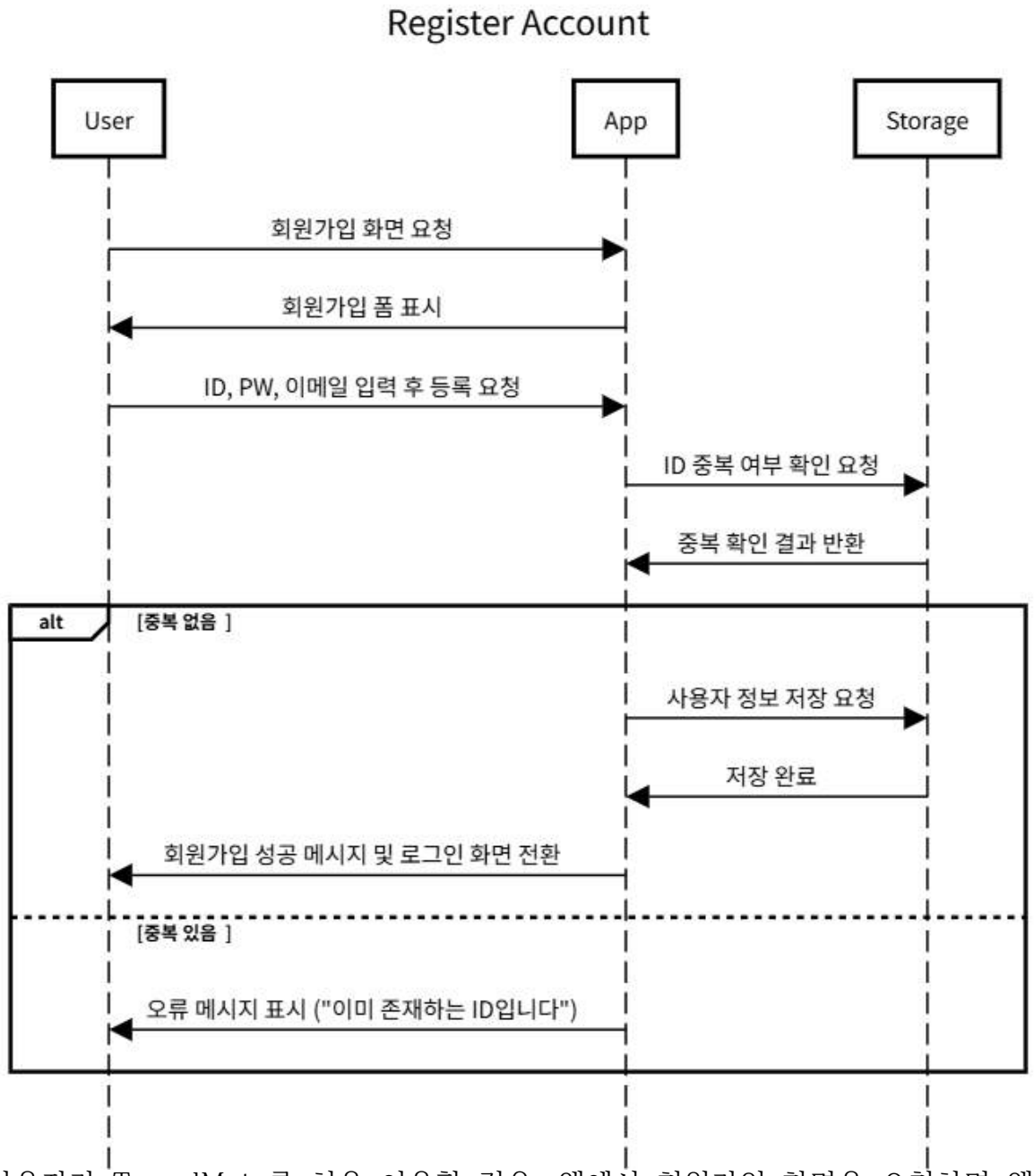
3. Sequence diagram

1. Log In



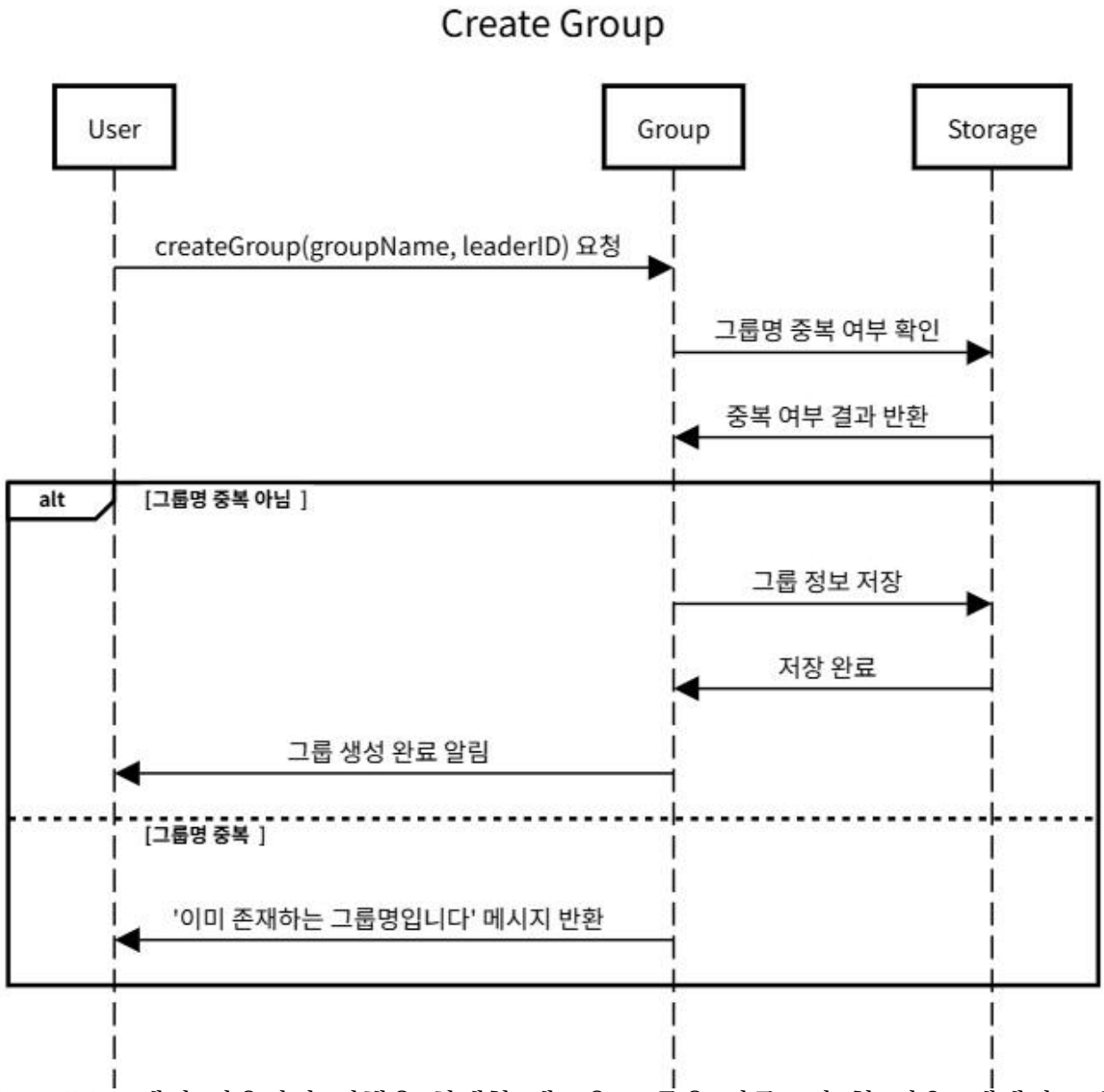
TravelMate 앱이 실행되면 사용자는 로그인 화면을 요청하고, 앱은 사용자에게 로그인 폼을 표시한다. 사용자가 자신의 ID와 비밀번호를 입력한 뒤 로그인 버튼을 누르면, 이 정보는 앱을 통해 서버로 전송된다. 서버는 데이터베이스에 접근하여 해당 ID와 비밀번호가 일치하는 사용자가 존재하는지 확인한다. 데이터베이스로부터 인증 결과를 받은 서버는, 그 결과를 다시 앱으로 전달한다. 만약 인증에 성공했다면 앱은 홈 화면으로 전환되고, 인증에 실패한 경우에는 "ID 또는 비밀번호가 틀렸습니다"라는 오류 메시지를 사용자에게 표시한다. 이 흐름을 통해 사용자는 TravelMate의 주요 기능에 접근할 수 있는 상태로 진입하게 된다.

2. Register Account



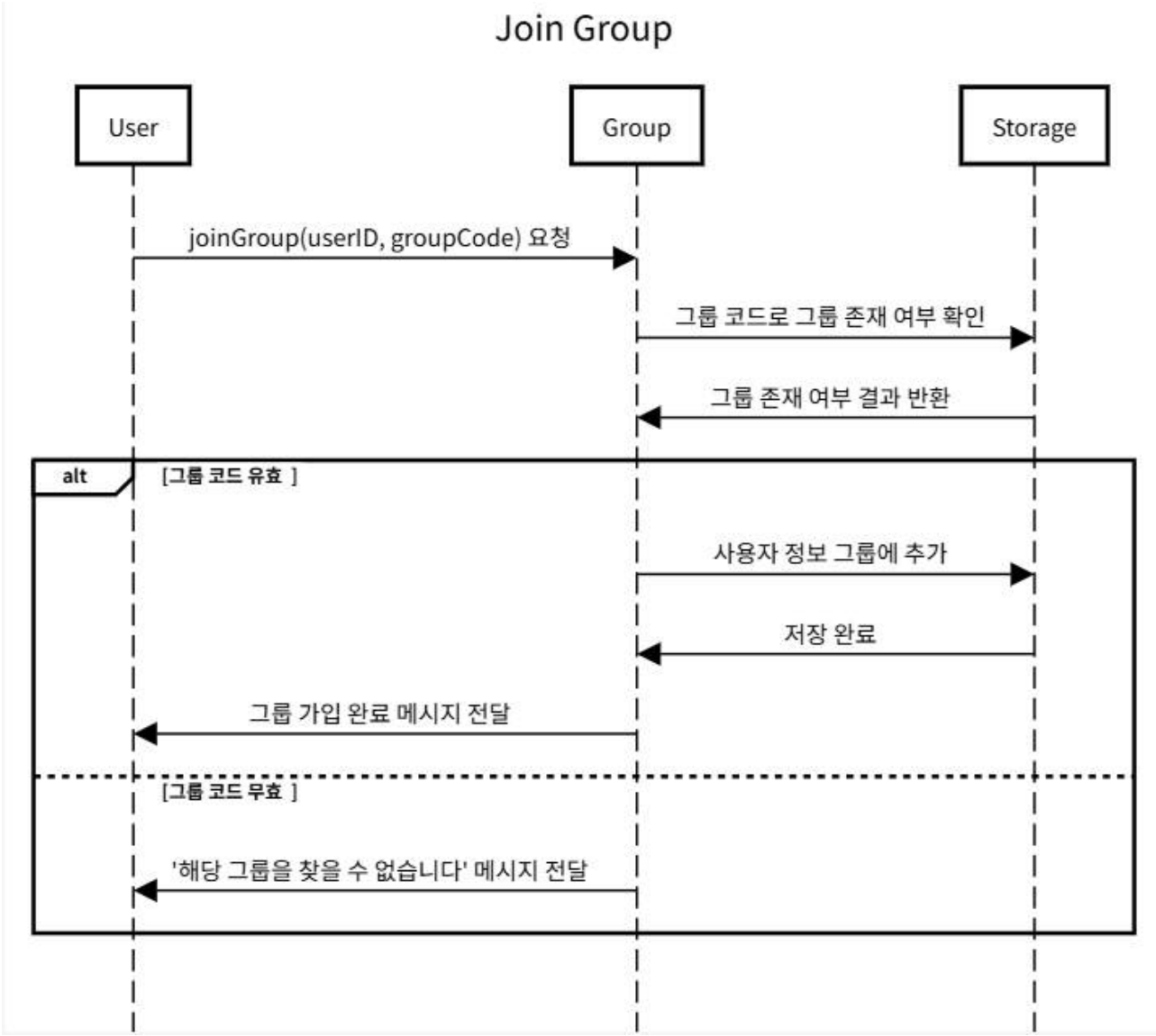
사용자가 TravelMate를 처음 이용할 경우, 앱에서 회원가입 화면을 요청하면 앱은 사용자에게 ID, 비밀번호, 이메일을 입력할 수 있는 폼을 보여준다. 사용자가 모든 정보를 입력하고 등록 버튼을 누르면, 이 정보는 앱을 통해 서버로 전송된다. 서버는 먼저 데이터베이스에 접근하여 입력한 ID가 이미 존재하는지 확인하고, 중복이 없는 경우에만 새 사용자 정보를 데이터베이스에 저장한다. 회원가입이 성공하면 서버는 앱에 성공 메시지를 전달하고, 앱은 로그인 화면으로 전환시킨다. 반면, 입력한 ID가 이미 존재할 경우에는 서버가 오류 메시지를 앱으로 반환하고, 앱은 사용자에게 "이미 존재하는 ID입니다"라는 안내 메시지를 표시하여 다시 입력을 유도한다. 이 과정을 통해 사용자는 TravelMate 계정을 새로 생성할 수 있게 된다.

3. Create Group



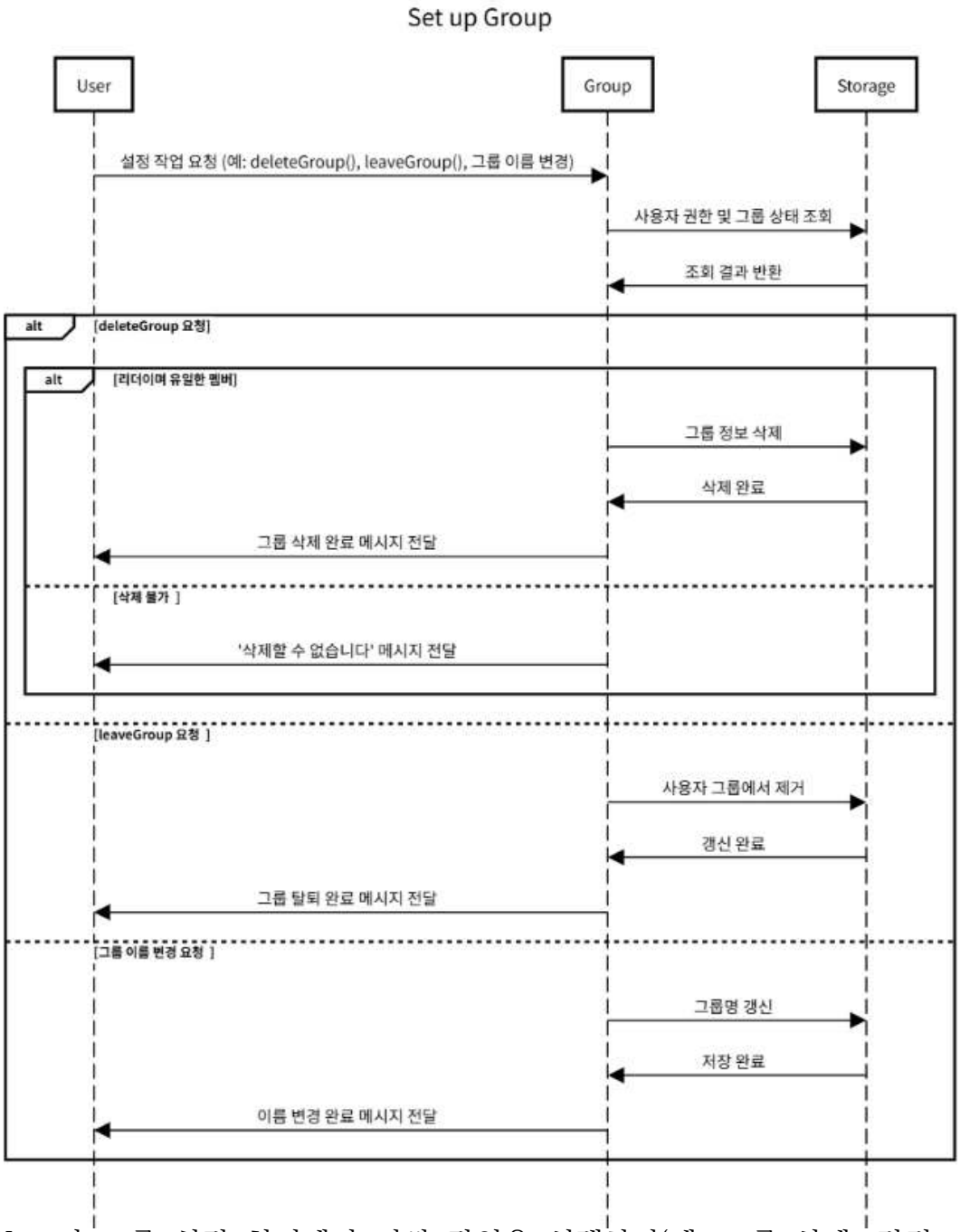
TravelMate에서 사용자가 여행을 함께할 새로운 그룹을 만들고자 할 경우, 앱에서 그룹 생성 화면을 요청하면 그룹명과 설명을 입력할 수 있는 폼이 사용자에게 표시된다. 사용자가 해당 정보를 입력한 후 그룹 생성을 요청하면, 앱은 그 데이터를 서버로 전송한다. 서버는 데이터베이스를 통해 입력된 그룹명이 이미 존재하는지 확인한 뒤, 중복되지 않은 경우 그룹 정보를 저장하고 해당 사용자를 그룹장으로 등록한다. 이후 그룹 생성이 완료되었다는 응답을 앱으로 전송하고, 앱은 그룹 메인 화면으로 전환한다. 반면, 그룹명이 이미 존재할 경우 서버는 오류 메시지를 반환하고, 앱은 사용자에게 "이미 존재하는 그룹명입니다"라는 메시지를 표시하여 다시 입력하도록 유도한다. 이 흐름을 통해 TravelMate는 사용자가 고유한 여행 그룹을 생성할 수 있도록 지원한다.

4. Join Group



TravelMate 앱에서 사용자가 기존 그룹에 참여하려고 할 경우, 앱은 그룹 가입 화면을 통해 그룹 코드 입력 폼을 사용자에게 제공한다. 사용자가 초대받은 그룹 코드를 입력하고 가입 요청을 하면, 앱은 이 코드를 서버로 전송한다. 서버는 데이터베이스를 조회하여 해당 그룹 코드가 유효한지 확인하고, 존재하는 그룹일 경우 해당 사용자를 그룹에 추가한다. 이후 가입 완료 응답을 앱에 전송하며, 앱은 자동으로 그룹의 메인 화면으로 전환된다. 반면, 사용자가 입력한 그룹 코드가 존재하지 않거나 잘못된 경우에는 서버가 오류 메시지를 반환하고, 앱은 사용자에게 "해당 그룹을 찾을 수 없습니다"라는 메시지를 표시하여 다시 입력하도록 유도한다. 이를 통해 사용자는 초대된 그룹에 손쉽게 참여할 수 있게 된다.

5. Set up Group

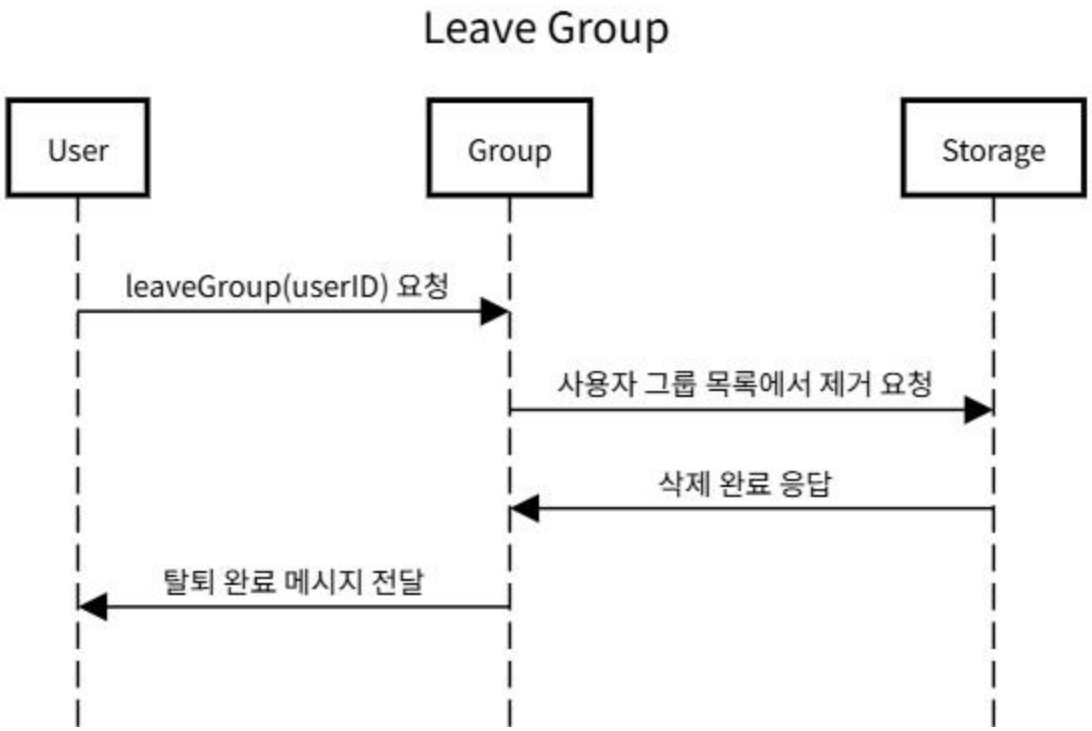


User가 그룹 설정 화면에서 어떤 작업을 선택하면(예: 그룹 삭제, 탈퇴, 이름 변경), 해당 요청은 Group 클래스에 전달된다. Group은 먼저 Storage에 사용자 권한과 그룹 상태를 확인하고, 그 결과에 따라 작업을 처리할 수 있는지 판단한다.

사용자가 그룹 삭제를 요청한 경우, Group은 해당 사용자가 그룹의 리더이며 유일한 멤버일 때만 Storage에 그룹 삭제를 요청한다. 삭제가 완료되면 사용자에게 성공 메시지를 전달하지만, 그렇지 않으면 “삭제할 수 없습니다”라는 메시지를 반환한다.

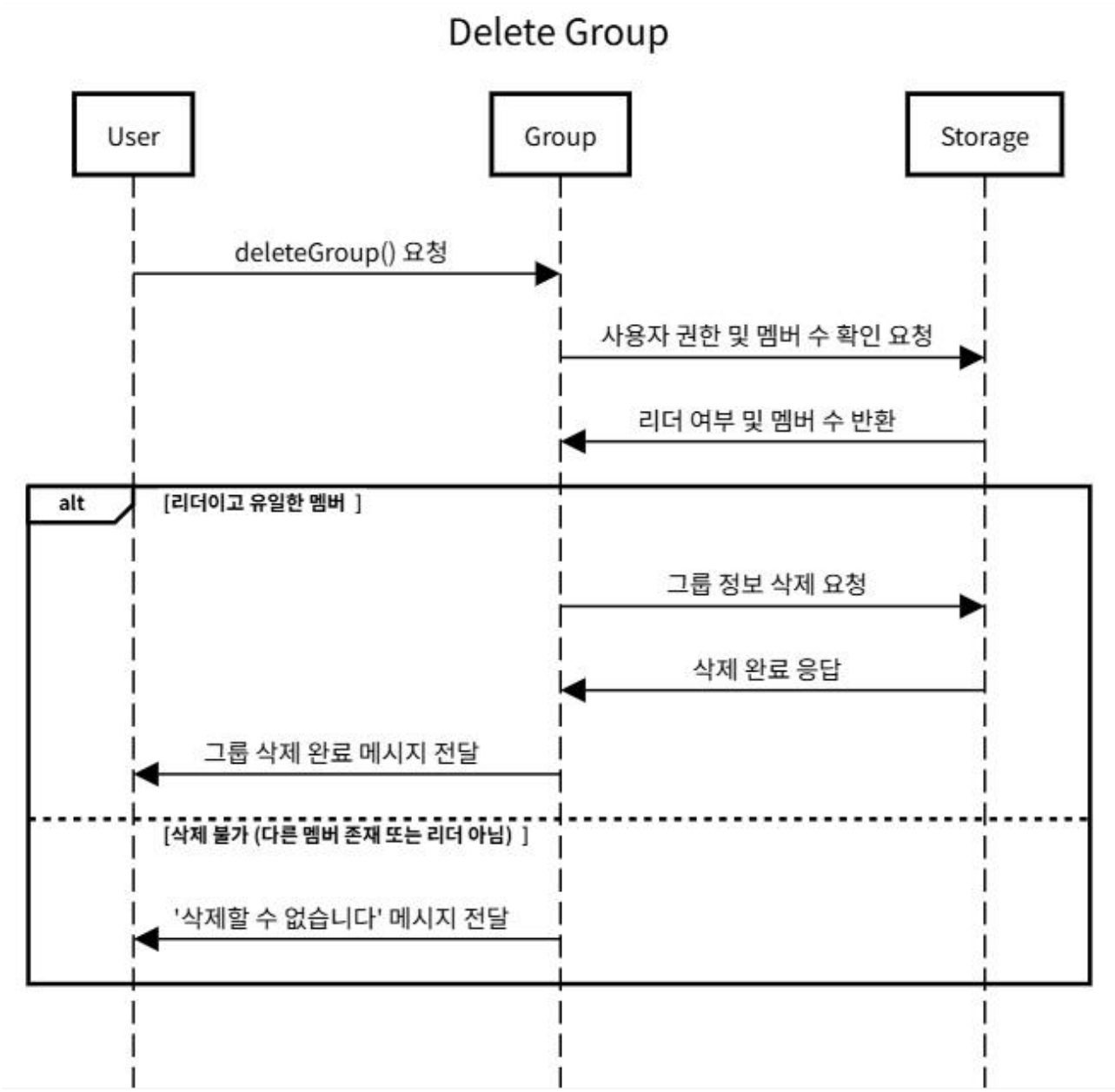
그룹 탈퇴의 경우, 사용자를 그룹 멤버 목록에서 제거하는 작업이 이루어지고, 그룹 이름 변경은 그룹명을 갱신한 뒤 사용자에게 완료 메시지를 반환한다.

6. Leave Group



User가 TravelMate에서 현재 소속된 그룹을 탈퇴하려고 하면, leaveGroup(userID) 요청을 Group 클래스에 보낸다. Group 클래스는 해당 요청을 받아서 Storage에 사용자 정보를 그룹 멤버 목록에서 제거해달라고 요청하고, Storage는 데이터에서 해당 사용자를 성공적으로 삭제한 후 응답을 반환한다. Group 클래스는 이 결과를 사용자에게 전달하고, 그룹 탈퇴가 완료되었다는 메시지를 보여준다.

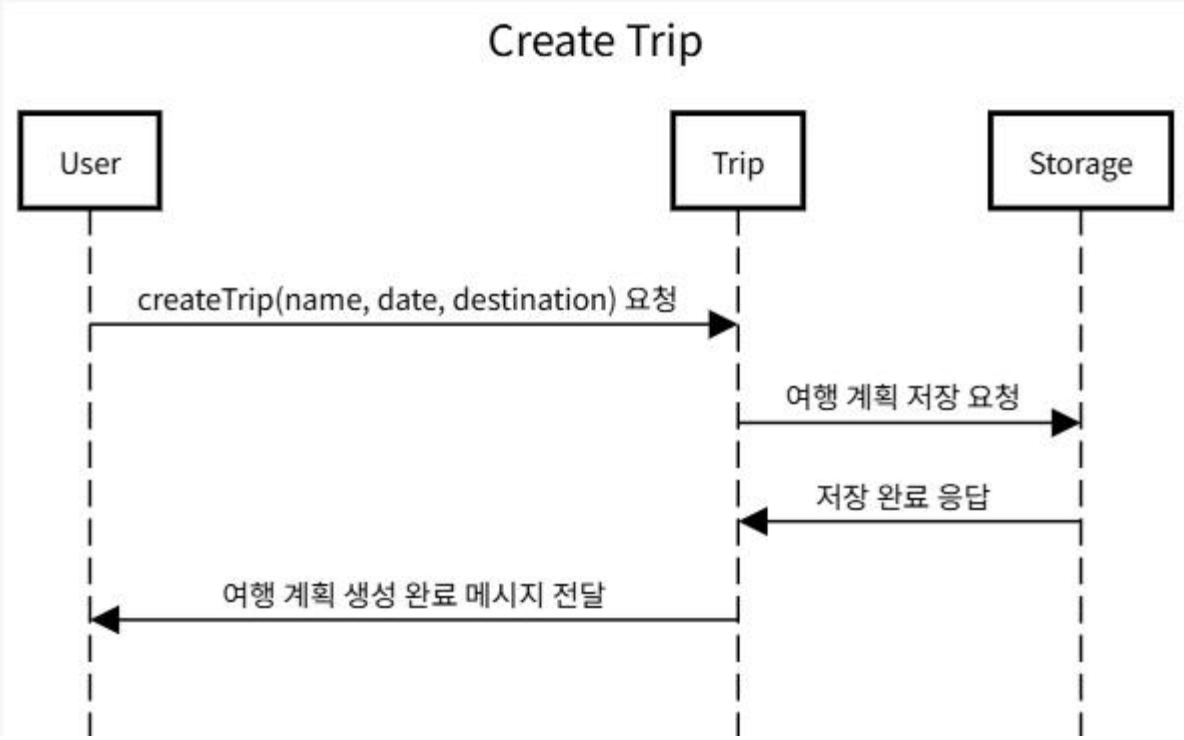
7. Delete Group



사용자가 TravelMate에서 그룹을 삭제하고자 하면, deleteGroup() 요청을 Group 클래스에 보낸다. Group 클래스는 이 요청을 처리하기 전에 Storage에 사용자 권한(리더인지 여부)과 그룹 내 멤버 수를 조회한다. Storage는 리더 여부와 현재 그룹에 몇 명이 소속되어 있는지를 반환한다.

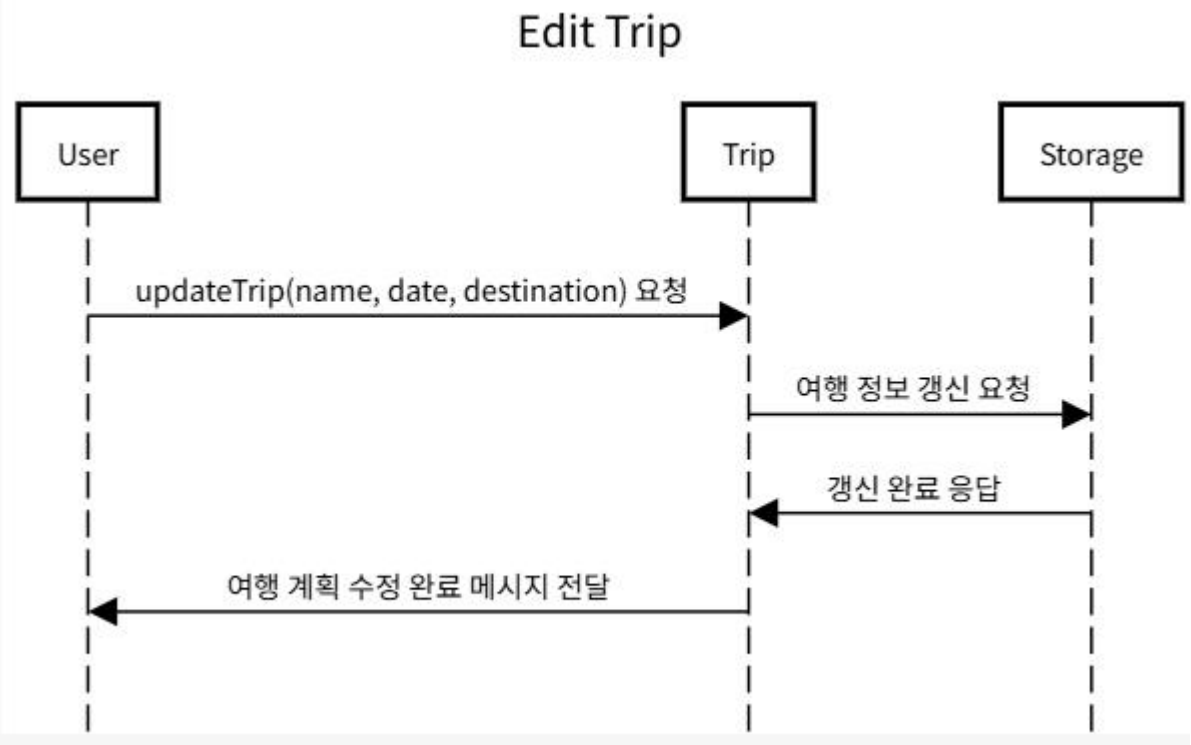
그 결과, 사용자가 그룹의 리더이고 유일한 멤버일 경우에만 Group 클래스는 Storage에 그룹 삭제를 요청한다. 삭제가 성공적으로 이루어지면 Group은 사용자에게 그룹이 삭제되었음을 알린다. 반대로 그룹에 다른 멤버가 남아 있거나, 사용자가 리더가 아닌 경우에는 "삭제할 수 없습니다"라는 메시지를 사용자에게 전달하여 작업을 거부한다.

8. Create Trip



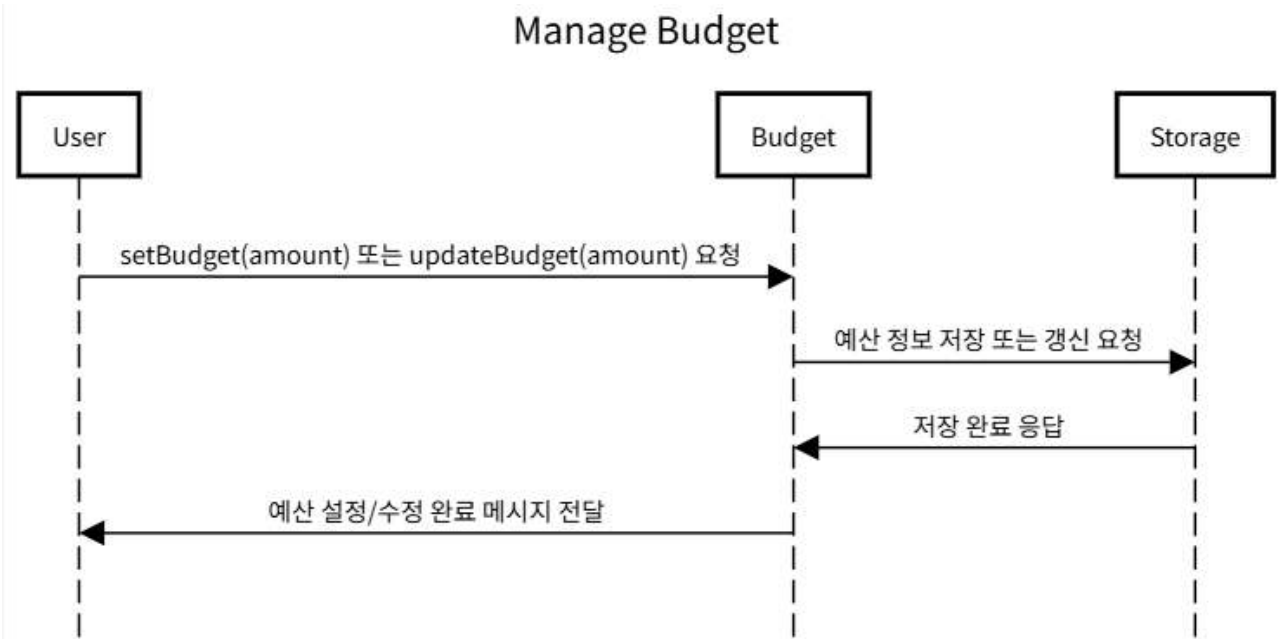
TravelMate에서 사용자가 새로운 여행 계획을 세우기 위해 여행 이름, 날짜, 목적지 등을 입력하고 createTrip() 요청을 하면, 이 요청은 Trip 클래스에 전달된다. Trip 클래스는 입력된 정보를 바탕으로 새로운 여행 객체를 생성하고, 이를 Storage에 저장 요청한다. Storage는 이 정보를 데이터베이스에 저장한 뒤 저장 완료 응답을 Trip에 반환하고, Trip은 사용자에게 여행 계획이 성공적으로 생성되었다는 메시지를 전달한다.

9. Edit Trip



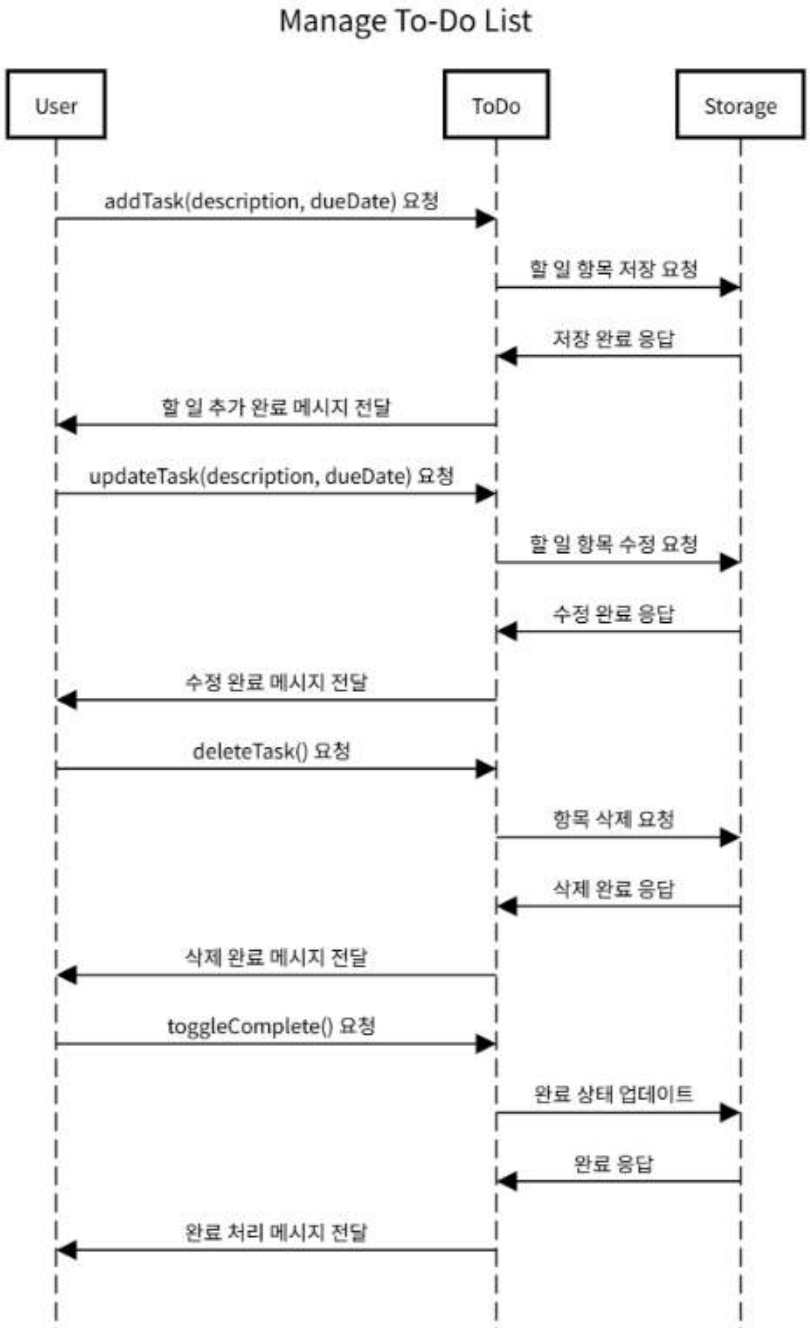
사용자가 TravelMate에서 기존에 생성한 여행 계획의 일정이나 목적지를 변경하고자 하면, updateTrip() 요청을 Trip 클래스에 전달한다. Trip 클래스는 변경된 정보를 받아 Storage에 갱신을 요청하고, Storage는 해당 데이터를 데이터베이스에서 수정한 후 성공 응답을 반환한다. 이후 Trip은 사용자에게 여행 계획이 성공적으로 수정되었다는 메시지를 전달하여 변경사항이 반영되었음을 알린다.

10. Manage Budget



사용자가 TravelMate에서 여행에 사용할 예산을 입력하거나 기존 예산을 수정하려고 하면, setBudget() 또는 updateBudget() 요청을 Budget 클래스에 전달한다. Budget 클래스는 사용자의 요청에 따라 예산 총액이나 항목을 갱신하고, 이 변경사항을 Storage에 저장 또는 갱신 요청한다. 저장이 완료되면 Storage는 응답을 반환하고, Budget은 사용자에게 예산 설정 혹은 수정이 완료되었다는 메시지를 전달하여 결과를 알린다.

11. Manage To-Do List

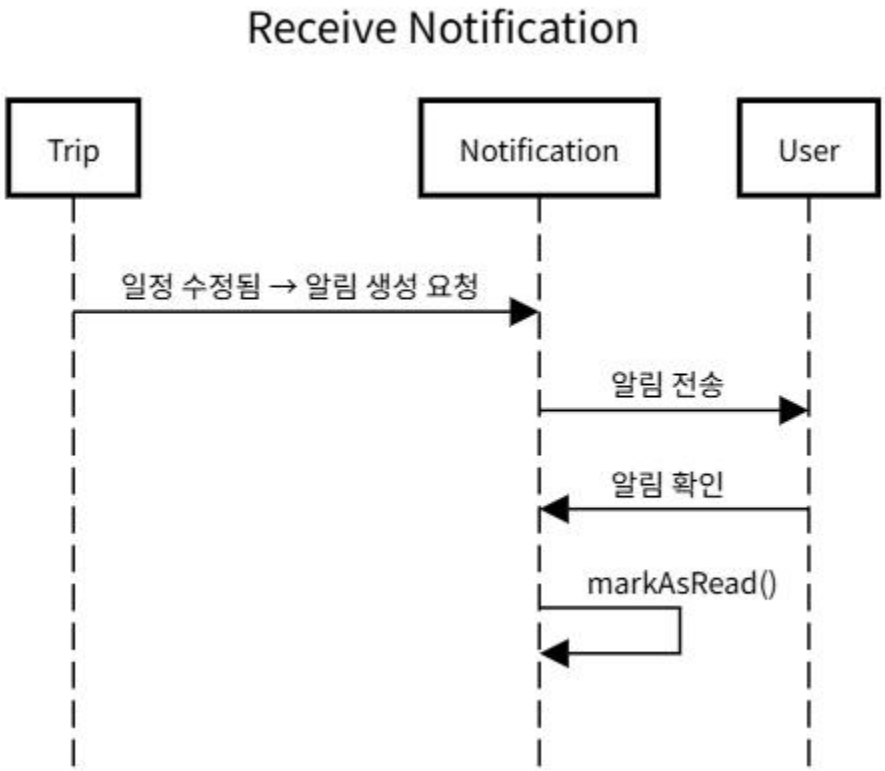


사용자가 TravelMate에서 할 일 항목을 추가하거나 관리할 때, addTask() 요청을 통해 할 일의 설명과 마감일을 ToDo 클래스에 전달한다. ToDo 클래스는 해당 정보를 Storage에 저장하고, 저장 완료 응답을 받은 뒤 사용자에게 작업 완료 메시지를 전달한다.

기존 할 일을 수정하거나(updateTask()), 삭제하거나(deleteTask()), 완료 표시를 전환할 때(toggleComplete()), 각각 ToDo 클래스가 해당 작업을 처리하고 Storage에 반영한 후 그 결과를 사용자에게 알려준다.

이를 통해 사용자는 자신의 여행 일정에 맞춰 할 일을 유연하게 추가하고 관리할 수 있다.

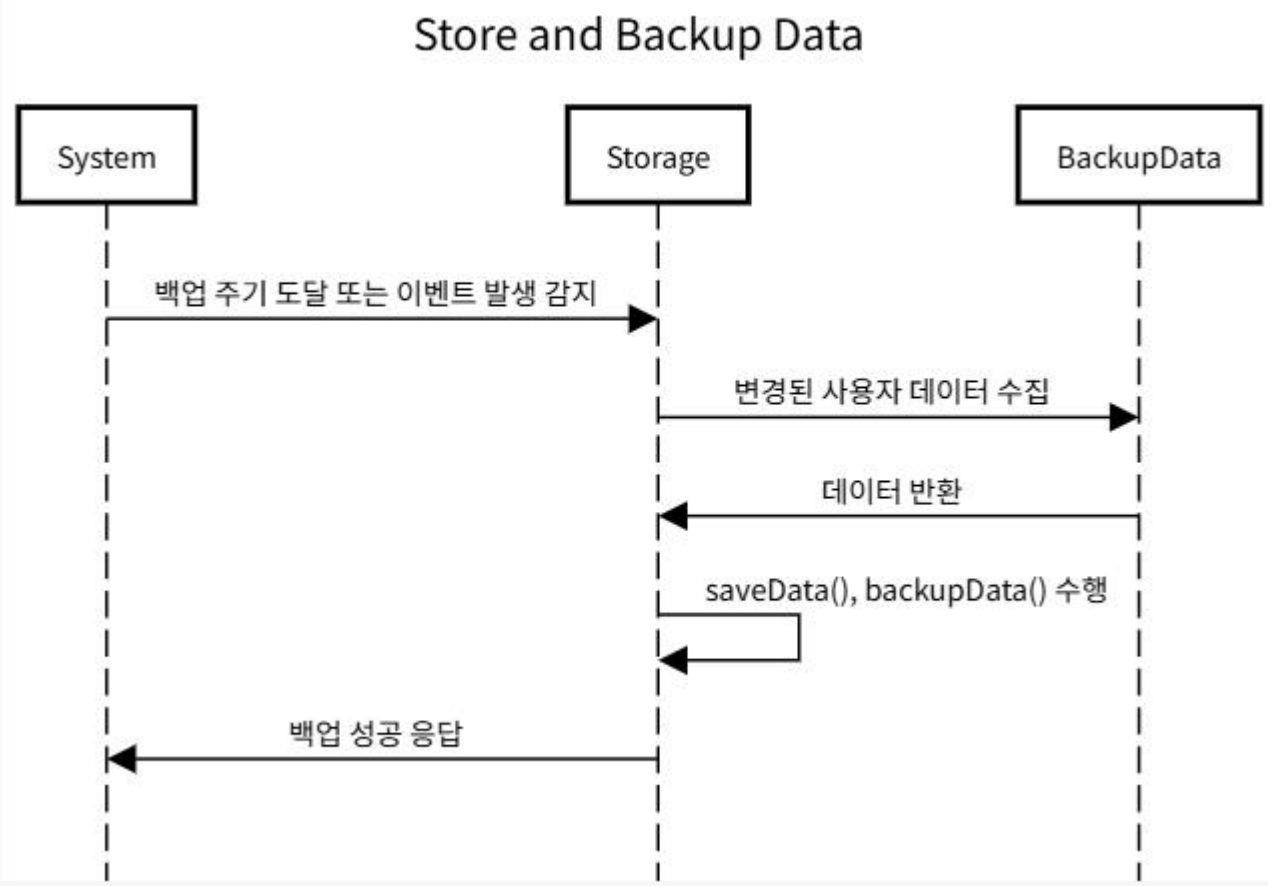
12. Receive Notifications



TravelMate에서 여행 일정이 수정되거나 할 일이 추가되는 등 중요한 이벤트가 발생하면, 해당 이벤트를 감지한 클래스(예: Trip, ToDo, Budget)는 Notification 클래스에 알림 생성을 요청한다. Notification 클래스는 적절한 메시지를 생성하여 관련된 사용자에게 알림을 전송한다.

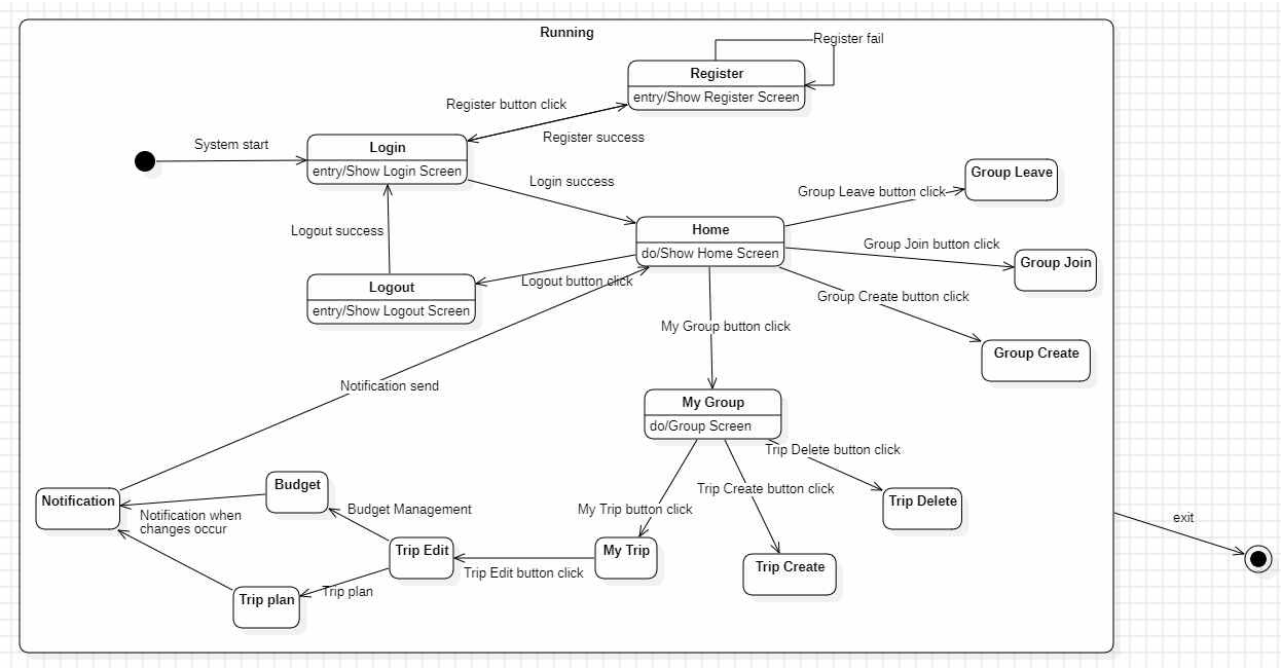
사용자가 알림을 확인하면, 알림은 markAsRead() 메서드를 통해 읽음 상태로 전환되며, 이후 중복되지 않도록 처리된다. 이를 통해 TravelMate는 사용자에게 중요한 여행 변경사항이나 주의사항을 실시간으로 전달할 수 있다.

13. Store and Backup Data



System은 일정 시간 간격이 도달하거나 사용자의 특정 작업(예: 여행 계획 저장, 예산 수정 등)이 발생했을 때 이를 감지하고 자동으로 백업 프로세스를 시작한다.
Storage는 BackupData 클래스에 접근하여 변경된 사용자 데이터를 수집한 뒤, 해당 데이터를 기반으로 saveData() 및 backupData() 작업을 실행한다. 이 작업은 로컬 저장소 및 클라우드 백업 등 여러 경로로 저장될 수 있으며, 작업이 완료되면 System에 성공 여부를 응답한다.
이 자동 저장 구조는 사용자의 명시적 입력 없이도 데이터를 안정적으로 보호하고, 앱이 예기치 않게 종료되더라도 손실을 방지하는 역할을 한다.

4. State machine diagram



State Machine Diagram은 TravelMate 애플리케이션의 클라이언트 측 동작 과정을 상태 기반으로 시각화한 것이다. 시스템은 사용자가 앱을 실행하면서 시작되며, 로그인 화면에 진입한다. 로그인 성공 시 홈 화면으로 전이되며, 회원가입 시도는 Register 상태를 통해 수행된다. 홈에서는 그룹 생성, 가입, 탈퇴가 가능하며, 각각 Group Create, Group Join, Group Leave 상태로 이동한다. 사용자가 My Group 화면으로 진입하면, Trip Create, Trip Delete, My Trip 등의 하위 상태로 이동할 수 있고, 여행 계획 및 수정, 예산 관리 등의 기능도 제공된다. 또한 변경 사항이 발생할 경우 Notification 상태로 이동하여 알림이 전송된다. 사용자는 언제든지 Logout 상태를 통해 앱을 종료할 수 있으며, 최종적으로 Final State로 전이되어 시스템이 종료된다. 모든 상태는 사용자 인터랙션에 기반한 버튼 클릭 이벤트를 중심으로 전이되며, 전체 앱의 흐름은 Running 이라는 복합 상태 안에서 관리된다.

5. Implementation requirements

개발 환경

IDE	Android Studio 또는 IntelliJ IDEA
Programming Language	Java 17 또는 Kotlin 1.8
Version Management	Git (GitHub 사용)

클라이언트 환경

Mobile OS	Android 8.0 (Oreo) 이상
Screen Resolution Support	최소 720x1280, 권장 1080x1920 이상
Required Permissions	인터넷, 저장소 접근

6. Glossary

용어	정의
User	TravelMate 애플리케이션을 사용하는 일반 사용자로, 로그인, 여행 계획 작성, 예산 설정 등의 기능을 수행합니다.
Group	여러 사용자가 모여 여행을 함께 계획하고 정보를 공유하는 단위입니다. 그룹 생성, 가입, 탈퇴 등의 기능이 포함됩니다.
Trip	여행의 계획 정보를 나타내며, 여행 이름, 날짜, 목적지, 참여자 목록 등으로 구성됩니다.
ToDo	여행 중 해야 할 일을 관리하는 항목으로, 체크리스트 형태로 작성되며 완료 여부를 표시할 수 있습니다.
Budget	여행 전체의 예산을 설정하고, 각 카테고리별로 지출을 계획하거나 기록할 수 있는 기능입니다.
Expense	실제 지출 항목에 대한 정보를 저장하고, 예산과 비교하여 확인할 수 있는 기능입니다.
Notification	그룹이나 여행 계획에서 변경 사항 또는 알림이 필요한 이벤트 발생 시 사용자에게 메시지를 전달합니다.
Storage	사용자 데이터(여행 정보, 예산, 그룹 등)를 안전하게 저장하고 백업하는 기능을 담당합니다.

7. References

정재곤, 『Do it! 안드로이드 앱 프로그래밍 (개정판 8판)』, 이지스퍼블리싱, 2023
PlantUML Sequence Diagram Guide
UML Reference Manual, James Rumbaugh et al., Addison-Wesley, 2004.