AMBLE

Travel. Explore. Draw.

Introduction

LTA is in a large collaborative project with SIT to improve the commuting experience specifically during the First Mile and Last Mile (FMLM) portions of their journey. Our proposed mobile application (referred to as Amble hereafter) aims to improve the entire commute journey of users, specifically during the FMLM segments of walking to and fro from their current location to their intended destination. Amble will serve primarily as a navigational application for the user by providing routing functionality between points on the map. Additionally, through the use of gamification, Amble will allow users to better appreciate their walking experience by enticing them to explore their surroundings more by suggesting different nearby landmarks that they could check out and be able to view and create artworks at those locations, while on their FMLM journey.

Document Objective

This document outlines the architectural and detailed software design for Amble through the usage of a class diagram, a component diagram and sequence diagrams. Rationales for software architectural and programming design patterns adopted will also be justified in this document.

Architectural Design

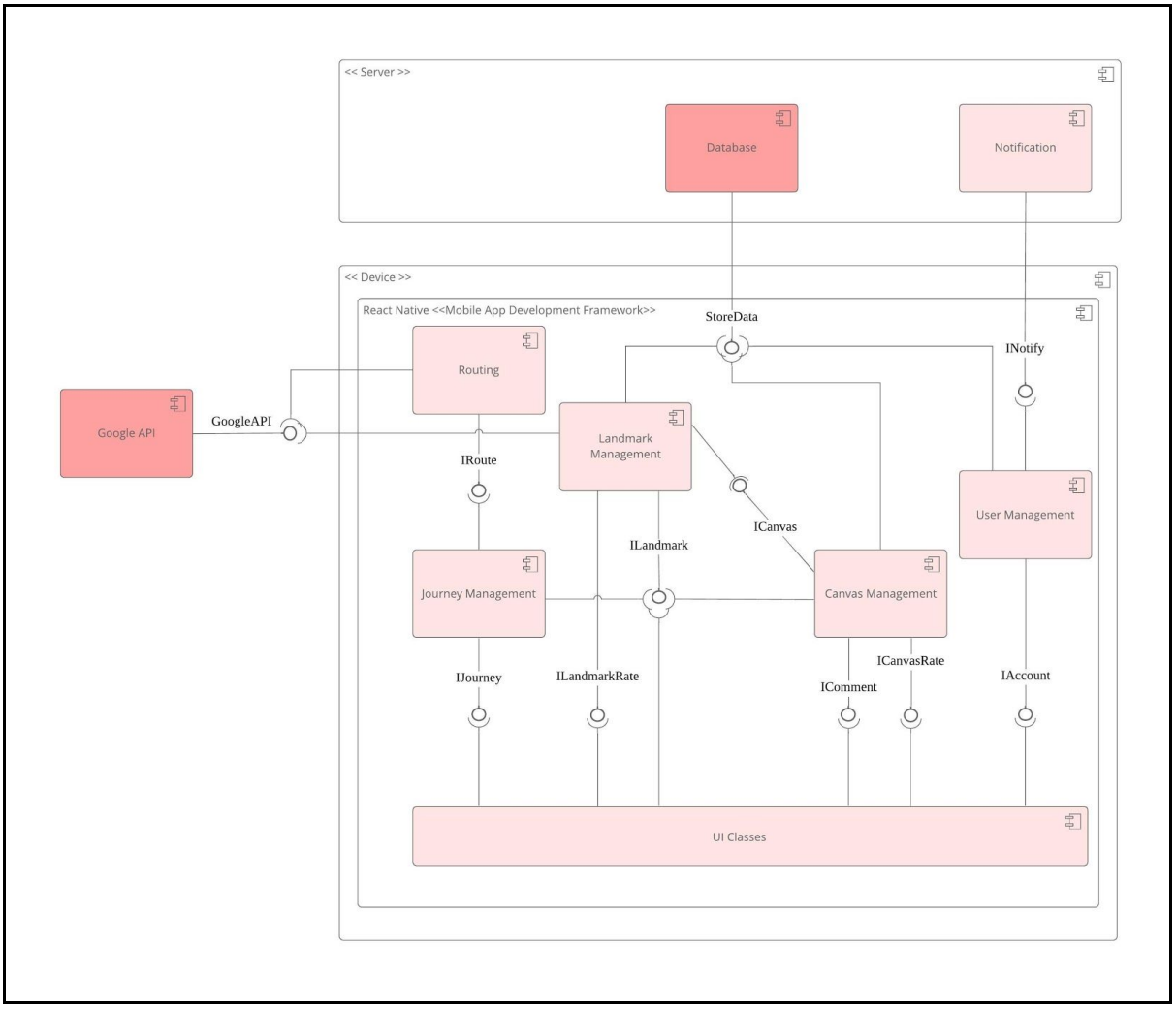

Figure 1. Component Diagram for Amble

Component	**Description**
Database	This component stores all necessary long-lived information of the mobile application and exposes an interface to allow all other necessary components to store data in itself.
User Management	This component encapsulates data and functions with regards to Users. It exposes an IAccount interface, allowing UI classes component to manipulate data from it. At the same time, it requires an interface provided by the Notification component to send emails to users, together with another interface from the Database component to store long-lived information about Users.

Canvas Management	This component encapsulates data and functions with regards to Canvases. It exposes an ICanvas interface, allowing Landmark Management component to manipulate data from it. It also exposes ICanvasRate and IComment interfaces to UI classes component to allow Users to rate and comment on canvases. At the same time, it requires an interface provided by the Database component to store long-lived information about Canvases. It also requires an interface provided by the Landmark Management component to manipulate and store long-lived information about itself with regards to the Landmark.
Landmark Management	This component encapsulates data and functions with regards to Landmarks. It exposes ILandmarkRate and ILandmark interfaces to UI classes component to allow users to rate and access landmarks. At the same time, it requires an interface provided by the Database component to store long-lived information about Landmarks. It also requires an interface provided by Google API component to obtain and manipulate information about itself.
Journey Management	This component encapsulates data and functions with regards to Journeys. It exposes an IJourney interface, allowing UI classes component to manipulate data from it. At the same time, it requires an interface provided by the Route component to manipulate and store long-lived information about itself. It also requires an interface provided by the Landmark Management component to manipulate and store long-lived information about itself with regards to the Landmark.
UI Classes	This component encapsulates data and functions obtained through other components which are subsequently used to display to Users.
Google API	This component encapsulates the data from the various Google APIs that the system requires. It exposes a GoogleAPI interface, allowing Routing and Landmark Management components to manipulate data from it.
Routing	This component encapsulates data and functions with regards to Routes. It exposes an IRoute interface, allowing Journey Management API to manipulate data from it. At the same time, it requires an interface provided by the GoogleMapsAPI to obtain and manipulate data about itself.
Notification	This component encapsulates data and functions with regards to sending notifications to Users. It exposes an INotify interface, allowing User Management component to manipulate data from it.

Architectural Pattern

Model-View-Controller (MVC) Pattern

The MVC Pattern will be adopted as it fits Amble’s purpose of being a highly interactive mobile application that requires interaction from users. This allows for decoupling of the business logic from the view and controller. This aids in speeding up development work as it allows independent working on different components of the system. This also enables scalability of the system as it grows due to how each component exposes an interface to distribute appropriate data and information out for other components who may require them.

Detailed Design

Class Diagram

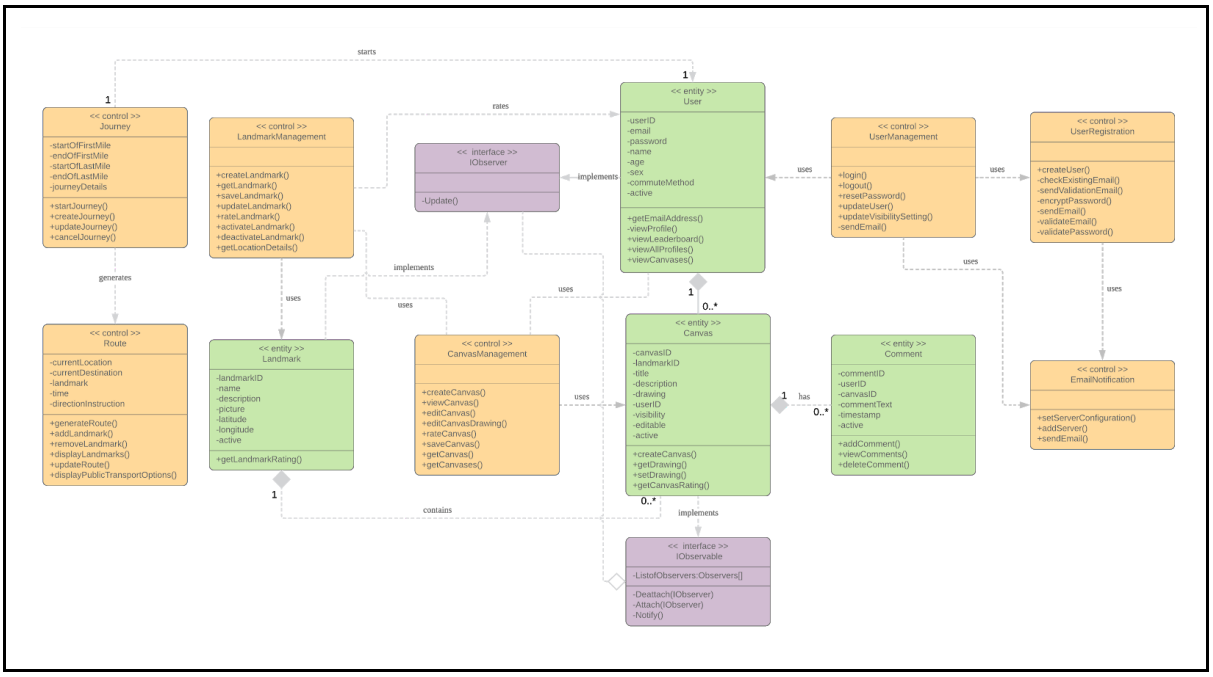

Figure 3. Class Diagram for Amble

Class	Type	Description
User	Entity	This class contains the long-lived information of the user profile, and is the main entity in the system. <u>Relationships</u> This class will use the following classes: ● UserManagement control class for controlling data related to User. ● Journey control class for controlling data related to the Journey. [1-to-1 relationship] ● LandmarkManagement control class for controlling data related to the Landmarks. ● CanvasManagement control class for controlling data related to the Canvas. This class is related to the Canvas entity which holds long-lived information of the Canvas. [1-to-0 or many relationship] This class implements the IObserver class to observe the IObservable interface class for Canvas.

UserManagement	Control	This class controls the general management actions of User accounts like User authentication and updating of information. <u>Relationships</u> This class will use the following classes: ● UserRegistration control class for controlling the creation of new User accounts, which has been decomposed from UserManagement to scale down the complexity of the control class. ● EmailNotification control class for controlling the sending of emails for password resets.
UserRegistration	Control	This class controls the creation of new User accounts. <u>Relationships</u> This class will use the EmailNotification control class to send verification of email address messages to the user's email address.
EmailNotification	Control	This class controls the configuration of email server, and interfaces with the email server to send emails to the user.
Journey	Control	This class controls the data related to the Journey of the User. <u>Relationship</u> This class will use the Route control class for controlling details of the individual Routes within the Journey for the User.
Route	Control	This class controls the data related to the individual Routes within the Journey of the User.
Landmark	Entity	This class contains the long-lived information of individual Landmarks. <u>Relationship</u> This class is related to the Canvas entity which holds long-lived information of the Canvas. This class uses the LandmarkManagement control class for controlling the data of Landmarks. This class implements the IObserver class to observe the IObservable interface class for Canvas.
LandmarkManagement	Control	This class controls the data related to Landmarks.

Canvas	Entity	This class contains the long-lived information for each Canvas created by Users. Relationship This class is related to the Comment entity, which contains long-lived information of a User's Comment. [1-to-0 or many relationship] This class implements the IObservable class to manage and notify classes that implemented the IObserver interface class to observe the Canvas.
CanvasManagement	Control	This class controls the data related to an individual Canvas.
Comment	Entity	This class contains the long-lived information for a Comment created by a User for a Canvas.
IObservable	Interface	This class contains a list of references to subscriber objects, allowing for different subscribers to be updated when any events happen to the publisher.
IObserver	Interface	This class contains a common method for different subscribers from different classes to be notified about any events that happen to the object they are observing.

Design Pattern

Observer Pattern

The Observer Pattern will be adopted as it enables scalability of the system as it grows due to how a change in state of data in an object in a class can be notified to an open-ended number of other classes easily. This enables code reusability as different classes only have to rely on implementing the same IObserver interface to get all necessary updates from Observables.

Sequence Diagram

UC3.1 - Entering Landmark Area

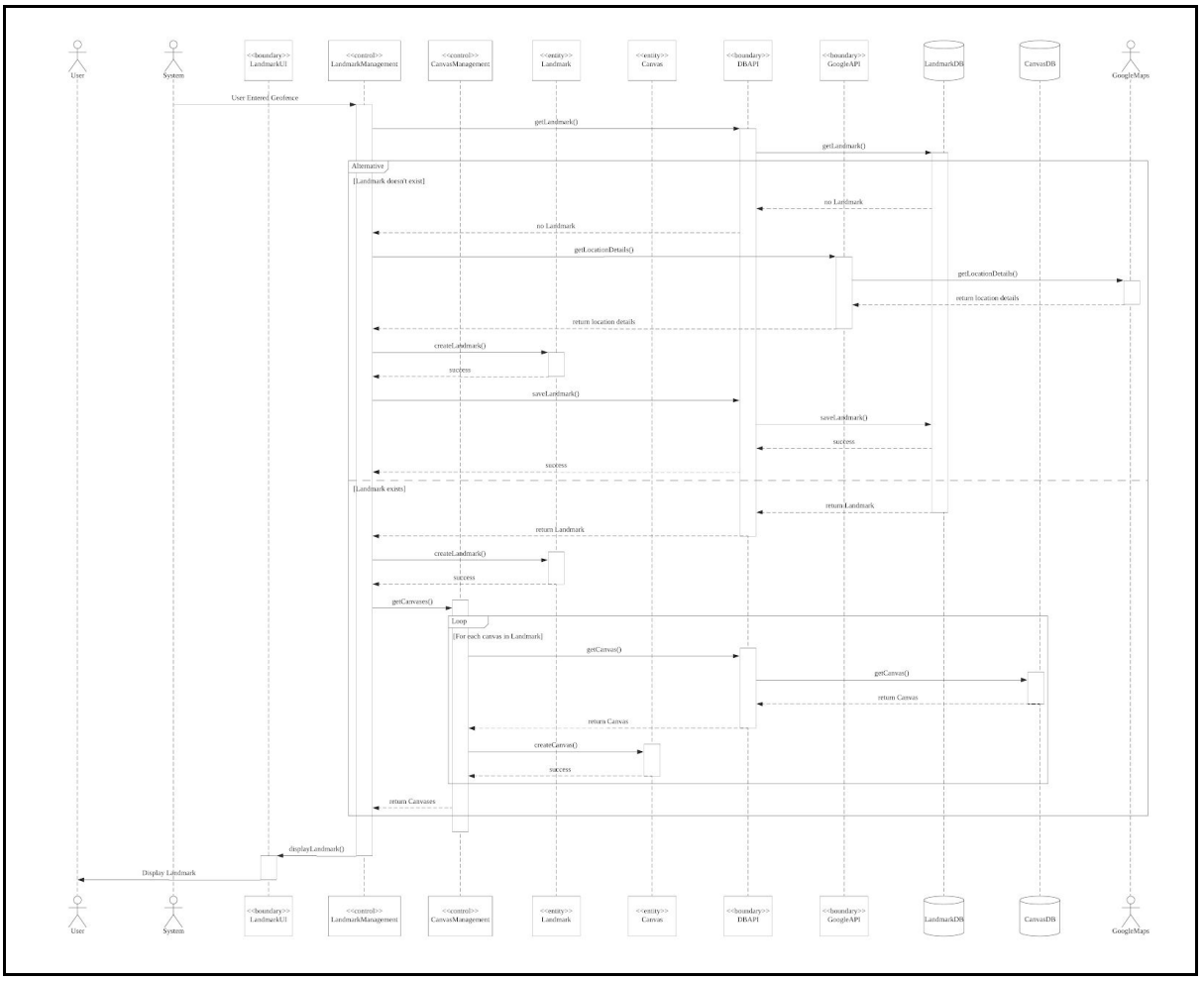

Figure 4. Sequence Diagram for Amble UC3.1 (Entering Landmark Area)

When the System detects that the User has entered a Landmark Area, it will attempt to get the Landmark details. If Landmark exists, it will get the details from the database. If the Landmark has yet to exist, it will gather the details from the Google API before creating the Landmark and saving the details to the LandmarkDB. Subsequently, it will then display the Landmark details to the User.

UC4.0 - Create Canvas

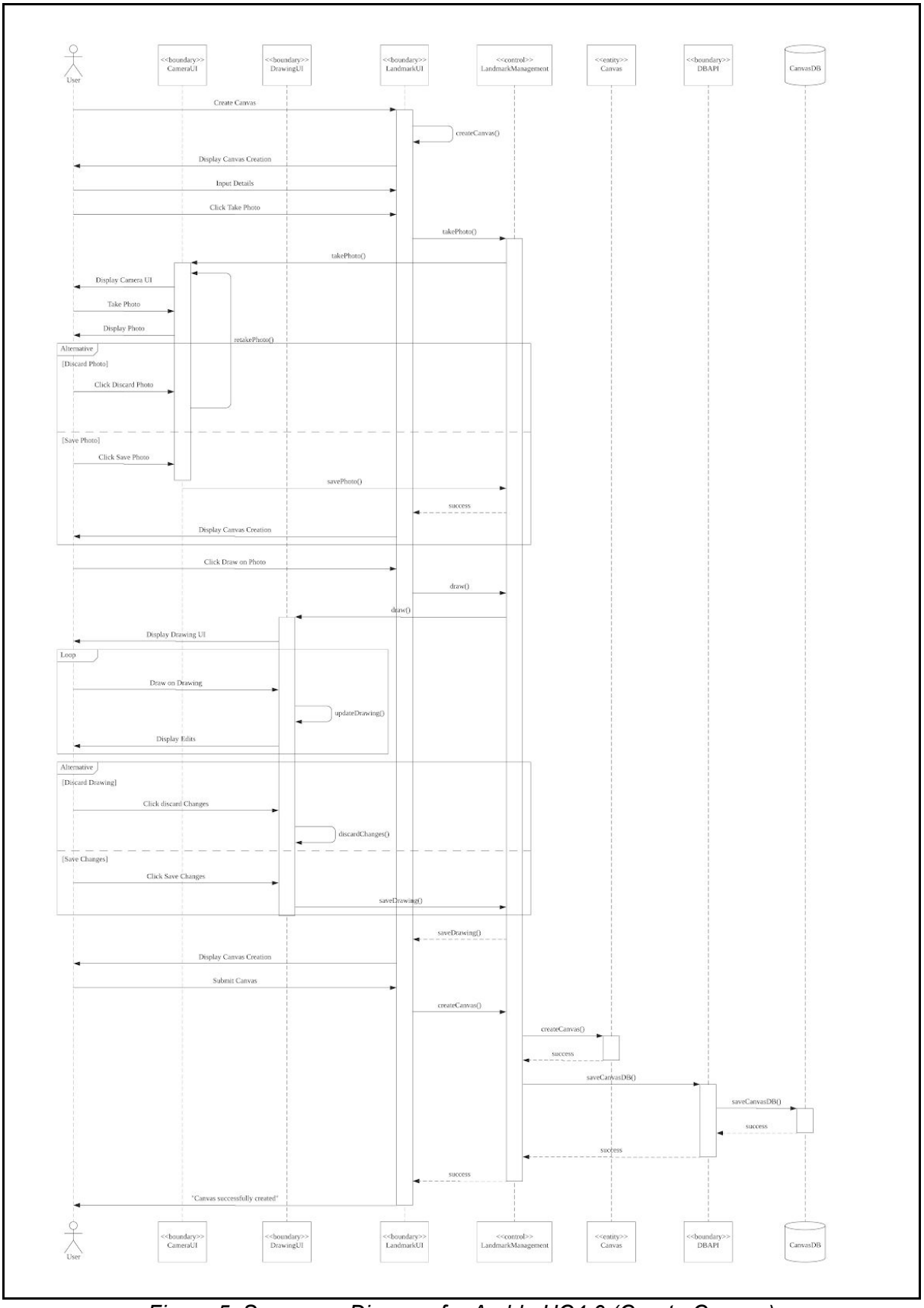

Figure 5. Sequence Diagram for Amble UC4.0 (Create Canvas)

The User starts by creating a Canvas when the User is at the particular Landmark through the Landmark interface from the Landmark UI. Through the process of creating a Canvas, the User has to enter details about the canvas and additionally, take a photo through the camera UI. After taking the photo, if the user decides to discard the photo, the photo will be discarded and the user will take another photo through the camera UI. Upon saving the photo, the user will then start to draw on the photo. At any point, if the user decides to discard the edited photo, the drawings on the photo will be discarded and user will be able to view the original photo and start drawing again. The Canvas entity will be created through the database API and stored in the CanvasDB when the user submits the Canvas with the necessary details.

UC4.1.3 - Edit Canvas Drawing

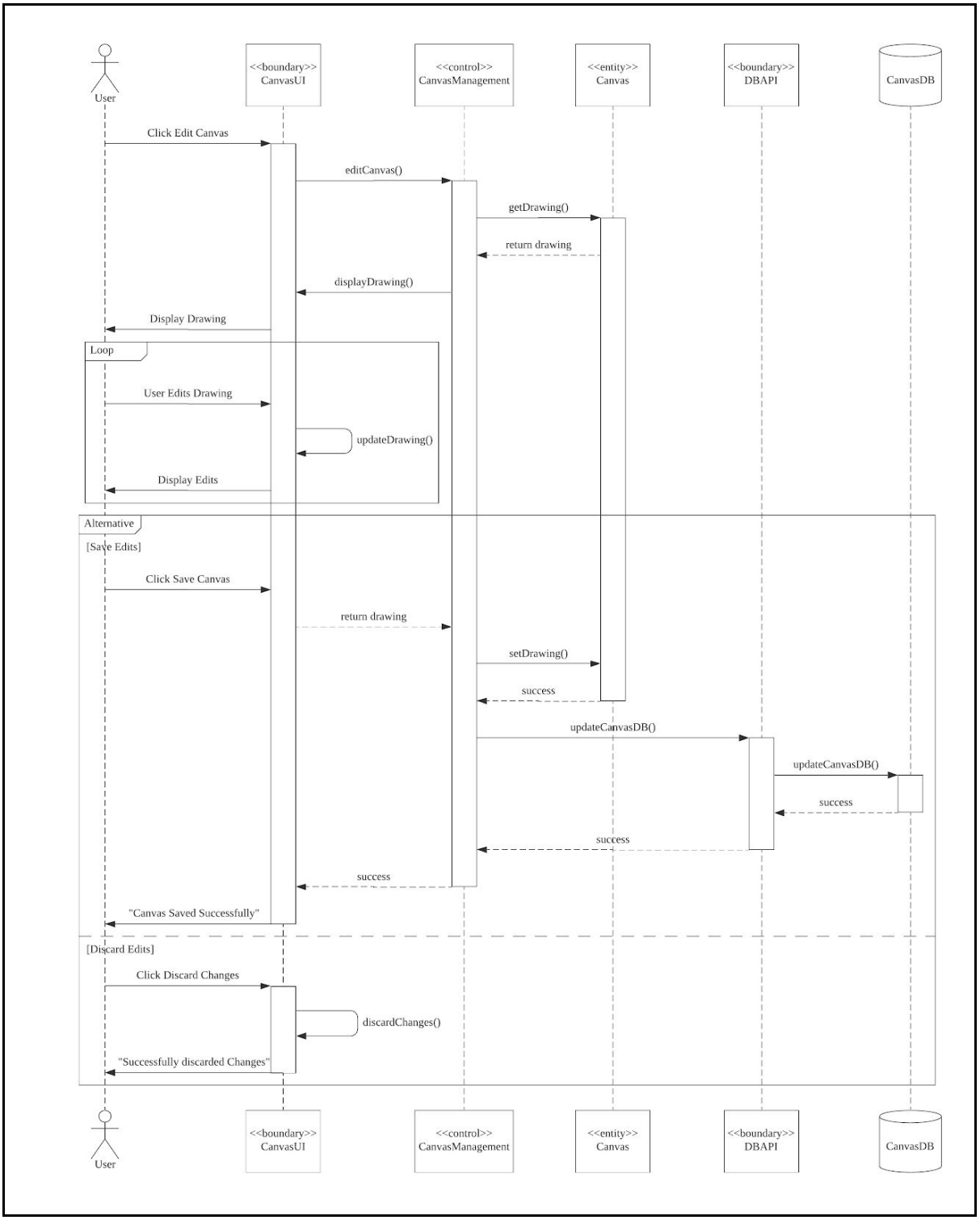

Figure 3. Sequence Diagram for Amble UC4.1.3 (Edit Canvas Drawing)

The User initiates the editing of the Canvas after which the System will display the Canvas Drawing to the User to edit. After editing, the User can click save and the system will save the Canvas Drawing to the CanvasDB or alternatively the user may discard their changes.

Glossary

Canvas
A canvas consists of a canvas drawing, with a title, description, location of the landmark at which the canvas was created.

Canvas Drawing
A canvas drawing consists of a picture taken at a Landmark, with drawings/scribbles done by the Users layered upon the picture.

First Mile/Last Mile (FMLM)
Imagine a commuter who usually gets to work by MRT every morning. After waking up and getting ready, the commuter typically has to either walk or cycle to the train station. This portion of the journey of getting from home to the MRT train station is an example of a First Mile (FM). Conversely, when the train arrives at the destination station, the commuter then needs to walk from the station to the office location, and this is the Last Mile (LM).

First Mile (FM)
The journey from the User's current location to the location where the User boards a public transport vehicle.

Last Mile (LM)
The journey from the location where the User alights from a public transport vehicle to their final destination.

Journey
The process of following a defined route from one point to another point on the map.

Landmark
Designated areas dotted around Singapore where users who are within a certain distance from the area can view canvas created in that landmark or create their own canvas.

Landmark rating score
Users are able to give a rating from 0 - 5 for any particular landmark. The landmark rating would be the combined average rating from all the users that ever rated that particular landmark.

Canvas rating score

Users will be able to rate a Canvas by upvoting or downvoting on it. The number of upvotes and downvotes will contribute to a final rating score attributed to the Canvas, making it the Canvas rating score.

Personal rating score

A score determined by the combined value of all the upvotes on all the canvas created by the user divided by the combined value of all the downvotes on all the canvas created by the user.

Route

A defined path for the User to follow between any number of points on the map.

System

This refers to the Amble application.