Software Requirements Specification

Document

**The Dungeoneer’s Assistant**

**Version 6**

12/05/19

Wizards of the Midwest TM

© Copyright 2019

Submitted in partial fulfillment of the requirements of

IT 326 Software Engineering

Contents

[Revision History iii](#_Toc26479613)

[Document Approval iii](#_Toc26479614)

[1. Introduction 1](#_Toc26479615)

[1.1 Purpose 1](#_Toc26479616)

[1.2 Scope 1](#_Toc26479617)

[1.3 Definitions, Acronyms, and Abbreviations 1](#_Toc26479618)

[1.4 References 3](#_Toc26479619)

[1.5 Overview 3](#_Toc26479620)

[2. General Description 3](#_Toc26479621)

[2.1 Product Perspective 3](#_Toc26479622)

[2.2 User Characteristics 4](#_Toc26479623)

[2.3 System Environment 4](#_Toc26479624)

[2.4 General Constraints 4](#_Toc26479625)

[2.5 Assumptions and Dependencies 4](#_Toc26479626)

[3. Specific Requirements 4](#_Toc26479627)

[3.1 Functional Requirements 4](#_Toc26479628)

[3.1.1 Add Multiple Enemies to Encounter 7](#_Toc26479629)

[3.1.2 Generate an Encounter 7](#_Toc26479630)

[3.1.3 Import Encounter from Share Code 7](#_Toc26479631)

[3.1.4 Generate Unique Share Code Based on Encounter 7](#_Toc26479632)

[3.1.5 Calculate XP Based on Encounter Parameters 8](#_Toc26479633)

[3.1.6 Remove Monster from Encounter 8](#_Toc26479634)

[3.1.7 Create Events on the Timeline 8](#_Toc26479635)

[3.1.8 Delete Timeline Event 9](#_Toc26479636)

[3.1.9 Edit Timeline Event 9](#_Toc26479637)

[3.1.10 Track Day/Night and Weather 9](#_Toc26479638)

[3.1.12 Create Character 10](#_Toc26479639)

[3.1.13 Edit Character 10](#_Toc26479640)

[3.1.14 Delete Character 10](#_Toc26479641)

[3.1.16 Generate a Character Share Code 11](#_Toc26479642)

[3.1.17 Import Character from Share Code 11](#_Toc26479643)

[3.1.23 Create Account 13](#_Toc26479644)

[3.1.24 Login/Logout 13](#_Toc26479645)

[3.3 Non-Functional Requirements 14](#_Toc26479646)

[3.3.1 Performance 14](#_Toc26479647)

[3.3.2 Reliability 14](#_Toc26479648)

[3.3.3 Availability 15](#_Toc26479649)

[3.3.4 Security 15](#_Toc26479650)

[3.3.5 Maintainability 15](#_Toc26479651)

[3.3.6 Portability 15](#_Toc26479652)

[3.4 Design Constraints 15](#_Toc26479653)

[4. Design & Development 15](#_Toc26479654)

[4.1 Software Process Model 15](#_Toc26479655)

[4.1.1 Introduction 15](#_Toc26479656)

[4.1.2 Process Model 15](#_Toc26479657)

[4.2 Diagrams 16](#_Toc26479658)

[4.2.1 Description 16](#_Toc26479659)

[4.2.3 Design 16](#_Toc26479660)

[4.3 Testing 61](#_Toc26479661)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 9/18/19 | Version 1 | Chloe Glass, William Paton,  Seth Tummillo, Ryan Busch, Austin Tran | Details for the project were established |
| 9/19/19 | Version 1.1 | ^ | Requirements adjusted, UML diagram added, grammar |
| 10/10/19 | Version 2 | ^ | Added some diagrams and started section 4 |
| 11/05/19 | Version 3 | ^ | Sequence and activity diagrams added, details updated, section 4 completed |
| 11/19/19 | Version 4 | ^ | Sequence and activity diagrams finished |
| 12/02/19 | Version 5 | ^ | Final Version |
| 12/05/19 | Version 6 | ^ | Final Final Version |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
| WP | William Paton | Conjuration Wizard | 12/5/2019 |
| RB | Ryan Busch | Illusion Wizard | 12/5/2019 |
| ST | Seth Tummillo | Head Wizard | 12/5/2019 |
| AT | Austin Tran | Other Wizard | 12/5/2019 |
| CG | Chloe Glass | Evocation Wizard | 12/5/2019 |

# 1. Introduction

This document is designed to clearly portray the features, constraints, and technicalities involved with implementing the “Dungeoneer’s Assistant” software suite. It has been organized to convey the purpose of the software and the guidelines that the developers should follow to construct and test the functional requirements. It also includes clarifications for terms, and considerations for those unfamiliar with the subject of D&D. This SRS serves as a blueprint for the “Wizards of the Midwest” team to design the project and ensure it adheres to the desired designs to the best of their ability.

## 1.1 Purpose

The SRS will describe the nature of the final product, list and describe the requirements necessary to build the project, and layout the dependencies, constraints and potential conflicts for the benefit of the Wizards of the Midwest and Dr. Rishi Saripalle.

## 1.2 Scope

1. Product
   1. The Dungeoneer’s Toolkit
      1. Encounter Generator
      2. Room Description Generator
      3. Campaign Editor
      4. Character Creator
      5. Timeline
2. Functionality
   1. It will simplify tasks typically performed by hand on pencil and paper by both players and the DM. Examples of tasks are:
      1. Character Creation
      2. Creating encounters
      3. Keeping track of events
   2. It will save resources created for several campaigns on a user account basis
   3. It will not be used as a tool to run an entire game digitally; this is a tool to aid an “in-person” game of Dungeons and Dragons
3. Goal
   1. The primary goal of this software is for it to be used before, during, or after a game session. The software should make running a game easier for both the user running the game and the other users who are playing it.

## 1.3 Definitions, Acronyms, and Abbreviations

Ability Score: Each PC, NPC, and Monster has 6 Ability Scores that make the foundation of what they are capable of. These scores are:

Strength: measure of physical power

Dexterity: measure of agility and speed

Constitution: measure of damage resistance and endurance

Intelligence: measure of intellect and logic

Wisdom: measure of perception and common sense

Charisma: measure of personality and persuasion

These scores range between 3 and 18, as they’re the sum of rolling 3 six-sided die.

Campaign: Overarching story, experienced over multiple sessions

Character Class: Each PC or NPC has a Character Class that determines how they interact with their possessions, skills, and world elements. These could include features that only that class has access to, or limitations on how that class may interact with in-game elements. An example being a Cleric having the unique ability to learn spells that can regenerate health but is restricted on using heavy weapons.

D&D(Dungeons and Dragons): Fantasy tabletop roleplaying game consisting of a single DM and up to several players.

DC(Difficulty Class): The difficulty of an action that a player is trying to accomplish.

DM(Dungeon Master): A role in a game of D&D who narrates the story, accepts player requests, generates challenges, enforces game mechanics, and generally controls all aspects of the game’s direction and pace.

DMG(Dungeon Master’s Guide): A document written by WotC with a focus on DM roles. These roles include rules specific to the DM, an expanded list of items and monsters, and general DM advice.

Encounter: An in-game event where the players interact with the world or engage in combat.

Item: Any physical possession that is owned by a PC or NPC.

Monster: Works similarly to an NPC but is not any of the race options for players and is usually involved in a combat encounter.

NPC(Non-Player Character): Any in-game character controlled by the DM, and not by any player.

One-Shot: Campaign designed to last one session.

PC(Player Character): Fictional character created by a player so the player can interact with the game world, other PCs, NPCs, and Monsters.

PHB(Player’s Handbook): A document written by WotC with a focus on character creation, player rules, classes, races, spells, and a selection of items, monsters and general player advice.

Player: Any non-DM participant in the game, who controls a PC and determines their actions within the game.

Race: Each PC or NPC has a Race that determines physical characteristics as well as modifications to their abilities. For example, a Half-Orc would have unique physical characteristics based on their depiction in the game, as well as benefits to game stats such as Strength and Constitution.

Session: Stretch of time that players gather.

“Splitting the Party” - The PCs separate to explore different areas.

WotC(Wizards of the Coast): The current publisher of D&D 5th Edition.

XP(Experience Points): Points that are earned by the players at the discretion of the DM in order to level up and improve their player characters.

## 1.4 References

Wizards of the Coast. Players Handbook. 5ed. Hasbro, 2014.

Wizards of the Coast. Dungeon Master’s Guide. 5ed. Hasbro, 2014.

Sources can be found online as pdf’s or physical copies can be purchased or rented from retailers.

## 1.5 Overview

The rest of the SRS is broken into two sections: General Description and System Requirements. The General Description section is made up of five parts that address outside influences on the creation of the product.

The System Requirements Section contains requirements that the group will fulfill by the end of the project, and design constraints that come up as ‘company’ restraints.

# 2. General Description

The goal with our application is to make managing and creating campaigns easier while also being intuitive for users to interact with. The app will provide tools to aid campaign creation such as an encounter generator and Map Generator, as well as providing tools to aid campaign tracking such as timelines and character sheets. Many of these terms will not sound familiar to those unacquainted with the game or the genre of role-playing games (RPG) in general.

## 2.1 Product Perspective

Many tools exist to help track the vast amount of information that goes into a game of D&D, but our product is aware of what the competition does; and our product will implement features that exist more uniquely while being more user-friendly, or features that have yet to be implemented into a single tool. The specifics can be viewed in the ‘Specific Requirements’ section below.

## 2.2 User Characteristics

Users will have an education level of Junior High or above. They will have minimum technical skills and will already have a fundamental understanding of how a game of D&D runs. Users must also be familiar with applications and generally know how to navigate through one based on various widgets, such as buttons and scrolling.

## 2.3 System Environment

Our product will be a .jar file and will be implemented/programmed in Java Swing. Our database of choice is local storage. We will save objects to a JSON file. It will be designed for a windows environment with the latest version of Java installed.

## 2.4 General Constraints

Our product will essentially be a collection of tools to aid a user. There will be no interactions between users. User information will be stored locally. Modules will operate independently of each other.

## 2.5 Assumptions and Dependencies

We assume that the user is using Windows operating system and has the latest version of Java installed

# 3. Specific Requirements

## 3.1 Functional Requirements

The function requirements fall into six categories: Encounter Generator, Timeline Generator, Character Generator, Room Generator, Campaign Management and Account Management.

The Encounter Generator allows the user to add a monster to an encounter, remove a monster from the encounter, generate a random encounter based on difficulty, calculate encounter xp, generate a sharecode, and load an encounter from a sharecode.

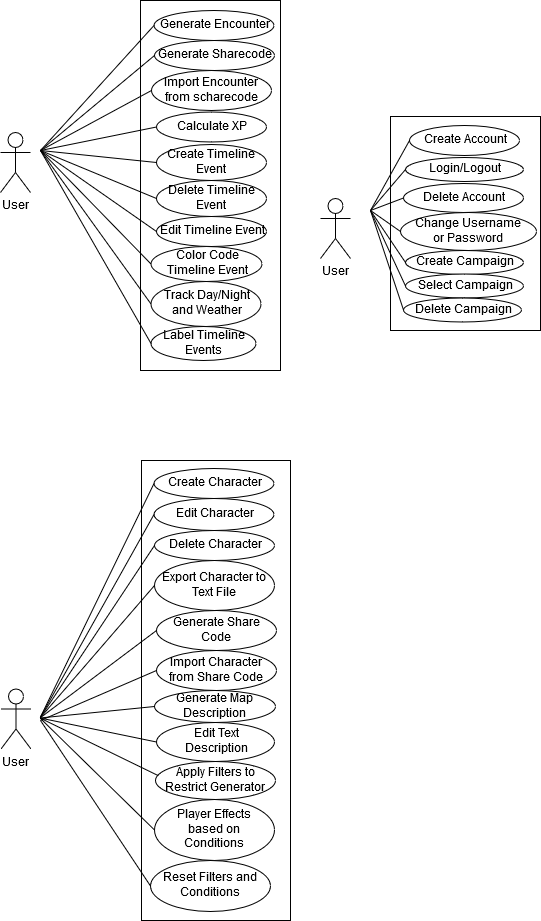
The Timeline Generator allows you to add, edit, and remove events from your campaign’s timeline. These events include a session number, label, date, in-game weather, in-game time, and description of the event.

The Character Generator allows the user to create a character, edit a pre-existing character, export the character sheet to a text file for printing, and generate a share code so other users can generate that same character more quickly. Characters include attributes such as their physical appearance, statistics, and abilities.

The Room Generator randomly generates descriptions of the games environment. These results can be filtered to include a variety of environments, and also include status ailments for the players based on environmental hazards.

Campaign Management is controlled from the main screen after login. This includes the tools listed above, but from a birds-eye view. You can track several campaigns at once, and select them from a list if the DM is running several games.

In Account Management the user can change their username, password, logout or delete their account.



### 3.1.1 Add Multiple Enemies to Encounter

Category - Encounter Generator

###### Description – The user will click a button that will display a menu for selecting an enemy. Enemy list will be in a dropdown menu. After the enemy is selected, the user clicks the “Add Enemy” button to add the enemy to the encounter

###### Actor(s) – The user

###### Trigger – The user will click a button to add a new enemy

###### Conditions

###### Pre – The user must be in the encounter generator module and have a monster selected in the monster menu.

###### Post – The user must submit or cancel the enemy interface to either add it to the pool or cancel it. If they hit “Add Enemy”, an enemy will be added. If they exit, the menu will close and nothing will happen.

### 3.1.2 Generate an Encounter

Category - Encounter Generator

###### Description – The user will generate a random encounter based on the chosen difficulty level. After the requirement for the encounter is chosen, the program will generate an encounter to fit that requirement.

###### Actor(s) – The user

###### Trigger – The user choses a difficulty and then clicks a button to generate the encounter

###### Conditions

###### Pre – Difficulty variable is selected.

###### Post – The user finalizes the encounter or clicks out of the menu

### 3.1.3 Import Encounter from Share Code

Category – Encounter Generator

###### Description – Using a share code the user can generate an encounter with certain parameters

###### Actor(s) – The user

###### Trigger – The user clicks a button after inserting a valid share code

###### Conditions

###### Pre – The user must have a valid share code

###### Post – An encounter will be generated

### 3.1.4 Generate Unique Share Code Based on Encounter

Category - Encounter Generator

###### Description – We want to be able to export possible encounters using a share code. Essentially the system will generate a share code based on parameters in the encounter

###### Actor(s) –The user

###### Trigger – The user will click a generate share code button

###### Conditions

###### Pre – The user must be in the encounter generator module

###### Post – The user must click the chosen difficulty or close out of the encounter generator

### 3.1.5 Calculate XP Based on Encounter Parameters

Category - Encounter Generator

###### Description – Calculate the XP that is to be rewarded to the party for the successful completion of the encounter

###### Actor(s) – The user clicks a button to generate XP based on the encounter that has been generated

###### Trigger – The user clicks a button to generate XP

###### Conditions

###### Pre –The encounter must be generated by the user

###### Post – The user must save the encounter or close out of the encounter generator

### 3.1.6 Remove Monster from Encounter

Category - Encounter Generator

###### Description – The user can remove a monster from an encounter

###### Actor(s) – The user

###### Trigger – After the user clicks on the “Encounter” module, and enters monsters to the encounter, either manually or by generating randomly. To remove the monster from the encounter, the user will click an ‘X’ button next to the monster text field

Conditions

###### Pre – There must be at least one monster in the encounter

###### Post – N/A

### 3.1.7 Create Events on the Timeline

Category - Timeline Generator

###### Description – The user will click a button to create an event, then a form will pop up. Fields cover information such as the session number, the session date, and a session description

###### Actor(s) – The user

###### Trigger – The user will click a button to create a new event

###### Conditions

###### Pre – The user must be on the Campaign page

###### Post – The user must click a ‘Create’ or ‘Cancel’ button at the bottom of the form

### 3.1.8 Delete Timeline Event

Category - Timeline Generator

###### Description – This will allow the user to delete an event from the list of events.

###### Actor(s) – The user

###### Trigger – The user will select an event, and hit a ‘Delete Event’ button

###### Conditions

###### Pre – The timeline event must already exist

###### Post – The event will be deleted forever after the user clicks the ‘Delete’ button

### 3.1.9 Edit Timeline Event

Category - Timeline Generator

###### Description – The user will click a button on the timeline event that will bring up a form like that of the ‘Create Event’ menu. The user will be able to alter the information of the fields. When they are done, they can save or cancel their changes

###### Actor(s) – The user

###### Trigger – The user will select a timeline event and click ‘Edit’ button

###### Conditions

###### Pre – The timeline event must exist or be in the process of being created

###### Post – The user must click a ‘Save’ or ‘Cancel’ button at the bottom of the form

### 3.1.10 Track Day/Night and Weather

Category - Timeline Generator

###### Description – There will be fields that allows the user to enter the in-game weather and in-game time

###### Actor(s) – The user

###### Trigger – The user will have the option to fill out the fields when editing or creating an event

Conditions

###### Pre – The timeline event must exist or be in the process of being created

###### Post – The user must click a ‘Save’ or ‘Cancel’ button at the bottom of the form

###### 3.1.11 Label Events

Category - Timeline Generator

###### Description – There will be a field that allows the user to give the session a label. The label will appear in the list of events, along with the session number

###### Actor(s) – The user

###### Trigger – The user must click a button on a timeline event

###### Conditions

###### Pre – The timeline event must exist or be in the process of being created

Post – The user must click a ‘Save’ or ‘Cancel’ button at the bottom of the form

### 3.1.12 Create Character

Category - Character Generator

###### Description – Creating a character to be used in a game of D&D

###### Actor(s) – The user

###### Trigger – User creates a new campaign, the Character Generator module will pop up

###### Conditions

###### Pre – The user must create a new campaign

###### Post – The user must click the ‘Save and Quit’ button to navigate back to the main menu

### 3.1.13 Edit Character

Category - Character Generator

###### Description – After a character has been created, they can be edited with the ‘Edit Character’ button on the main menu. Any attribute that was set when the character was created can be changed.

###### Actor(s) – The user

###### Trigger – User clicks on the ‘Edit Character’ button on the main menu

###### Conditions

###### Pre – The user must have already built a character and saved it to their account

###### Post – Any changes the user has made must be saved to their character

### 3.1.14 Delete Character

Category - Character Generator

###### Description – After a character has been created, the user is able to delete them from their account

###### Actor(s) – The user

###### Trigger – After the user clicks on the “Character Creator” module, they are given a list of characters/monster (if any) that they’ve previously created. These can be selected for deletion

###### Conditions

###### Pre - The user must have already built a character and saved it to their account

###### Post – N/A

###### 3.1.15 Export Character to Text File

Category - Character Generator

###### Description – After a character has been created, the user is able to export their character to a text file that they can print and use in their game

###### Actor(s) – The user

###### Trigger – The user clicks ‘Edit Character’ button from the main menu, then clicks the ‘Export to File’ button

###### Conditions

###### Pre – The user must be on the Character Generator screen with all the fields of the character filled out

###### Post – A text file will be generated and stored on the user’s desktop

### 3.1.16 Generate a Character Share Code

Category - Character Generator

###### Description – Generate a string that can be used by the Character Generator to instantly create a predeveloped Character to aid with sharing creations

###### Actor(s) – The user

###### Trigger – The user clicks ‘Edit Character’ button from the main menu, then clicks the ‘Generate Share Code’ button

###### Conditions

Pre – The user must be on the Character Generator screen with all the fields of the character filled out

###### Post – The user must copy this code to their clipboard to either share it or store it in a text file

### 3.1.17 Import Character from Share Code

Category – Character Generator

###### Description – Using a share code the user can generate a preexisting character

###### Actor(s) – The user

###### Trigger – The user clicks ‘Edit Character’ button from the main menu, then clicks the ‘Load Share Code’ button

###### Conditions

Pre – The user must be on the Character Generator screen with all the fields of the character filled out, and with a valid sharecode in the text field

###### Post – The character will be updated with the values contained in the sharecode

**3.1.18 Generate a Map Description**

Category - Map Generator

Description – Generate a list of key details for a room or dungeon to assist players with their immersion. There will be three randomized sentences, and there will be a total of 1000 possible combinations of the strings.

Actor(s) – The user

Trigger – In the Map Generator tool, there will be a “Generate Description” button the user will click

Conditions

Pre – The user must have the module open

Post – None

**3.1.19 Edit Text Description**

Category - Map Generator

Description – Edit the description generated by Map Description, biome filter, or conditions.

Actor(s) – The user

Trigger – The text within the application will be editable at any time, whether filters or conditions or a map description has already been applied/generated.

Conditions

Pre – The user must be in the map description generator module

Post – None, the edits are directly within the text field of the module

**3.1.20 Apply Filters to Restrict Generator**

Category - Map Generator

Description – The user can select a biome, which will affect what the generator displays for the text description

Actor(s) – The user

Trigger – There will be a menu with checkbox fields on it.

Conditions

Pre – The user must be on the Map Generator module

Post – The user must click ‘Save’

**3.1.21 Player Effects Based on Conditions**

Category - Map Generator

Description – If certain conditions are selected, then there will be effects on the player

Actor(s) – The user

Trigger – Based on filters that the user has selected, the text description will include effects on charactersand possible stat increase/decreases that could be permanent or temporary.

Conditions

Pre – The user must be on the Map Generator module

Post – The user must click ‘Save’

**3.1.22 Reset Filters and Conditions**

Category - Map Generator

Description – A button that will deselect any filter and/or conditions that have been previously applied

Actor(s) – The user

Trigger – There will be a button named “Reset”

Pre – The user must be on the Map Generator module

Post – When a user generates a new map description, no biome or conditions will be applied

### 3.1.23 Create Account

Category - Account

Description - This requirement will allow a user to register an account.

###### Actor(s) – The user

###### Trigger – Clicking button to create account

###### Conditions

###### Pre - None

###### Post – The user will be able to log in

### 3.1.24 Login/Logout

Category - Account

Description - The user should be able to login to gain access to features and then be able to log out to close their session

###### Actor(s) – The user

###### Trigger – The user clicks on a login button

###### Conditions

###### Pre – The user must have an existing account

###### Post – The user will be able to access features

###### 3.1.25 Delete Account

Category - Account

Description - The user should be able to delete their account if they wish. This should also require confirmation stating what they are about to do.

###### Actor(s) – The user

###### Trigger – Press ‘Delete Account’ button

###### Conditions

###### Pre – The user must be logged in and have an existing account

###### Post – The user will no longer have an account. All information related to their account will be permanently deleted.

###### 3.1.26 Change Password or Username

Category - Account

Description – Allow user to change their password or username

###### Actor(s) – The user

###### Trigger – The user clicks change password or change username button

###### Conditions

###### Pre – The user must have a verified account

###### Post – The user’s password or username should be different

###### 3.1.27 Create Campaign

Category – Campaign

Description – Allow user to create a campaign

###### Actor(s) – The user

###### Trigger – The user clicks create campaign button

###### Conditions

###### Pre – The user must be logged in

###### Post – The user will have access to the features. Different campaigns will have different content stored in some of the modules

###### 3.1.28 Select Campaign

Category – Campaign

Description – Allow user to select and view or edit a campaign

###### Actor(s) – The user

###### Trigger – The user clicks a campaign from a list of campaigns

###### Conditions

###### Pre – The user must be logged in and have previously created a campaign

###### Post – The user will have access to the features. Different campaigns will have different content stored in some of the modules, which they will be able to change

###### 3.1.29 Delete Campaign

Category – Campaign

Description – Allow user to delete a campaign

###### Actor(s) – The user

###### Trigger – The user selects a campaign from a list of campaigns and clicks a button to delete the campaign

###### Conditions

###### Pre – The user must be logged in and have previously created a campaign

###### Post – The campaign will be permanently deleted

## 3.3 Non-Functional Requirements

### 3.3.1 Performance

Each module should load in a reasonable time without interfering with the end-user's other processes.

### 3.3.2 Reliability

Any creations that users have produced must be reproducible by the tools whenever the source file is uploaded to its corresponding tool. Our product should be usable so long as the end user’s computer is working properly.

### 3.3.3 Availability

Since our product is offline, it will be usable whenever.

### 3.3.4 Security

Account access shall be password restricted to prevent users from altering the accounts of others. Passwords should meet requirements.

### 3.3.5 Maintainability

The system must include a reference document for each module to document how the code operates.

### 3.3.6 Portability

The product will only be able to run on computers with the most recent version of java. For the purposes of programming, so our code will be published in collaboration on GitHub which is accessible from any computer so long as one of the members provides their credentials.

## 3.4 Design Constraints

This toolkit must be used to ease the use of WotC documents, but not replace them.

The navigation of the application must not result in fail state where the user is unable to continue editing their creations.

# 4. Design & Development

This section is to outline how our team plans to execute the design and development of the “Dungeoneer’s Assistant” application. It will cover the software process model that we chose to follow and justify it. It will also show a diagram of this model that will provide details to our processes such as our sprint schedule and meeting frequency.

Our team plans to execute our application through the Scrum model and its multiple implementations of weekly and/or bi-weekly sprints. Our group is most familiar with this model from past experiences and thought it would be best fit out of the options. More specific details below.

## 4.1 Software Process Model

### 4.1.1 Introduction

This section will briefly describes the process model chosen and any background information required to understand the rest of this section.

### 4.1.2 Process Model

The process model we have chosen for the development of our application is the Scrum Model. Scrum allows us to pick a feature to be expanded upon in sprints which will last 1-2 weeks. Currently we hold meetings on Wednesdays at Milner at 3:30 PM which can last anywhere from thirty minutes to two hours, depending upon what needs to be discussed and accomplished as a group.

#### 4.1.3 Plan Description

Our focus for the first couple of sprints is the planning and designing of our application, while the following sprints will focus more heavily on the actual code integration and the setup of a collaborative environment with GitHub to share our code repositories respectively. We expect to hold virtual/online meetings during later sprint stages since we all have our own components to work on and are all comfortable with the online communication space platform.

#### 4.1.4 Plan Diagram

|  |  |  |
| --- | --- | --- |
| Sprint | Time Table | Goal |
| 1 | 9/20/19 - 10/10/19 | Add all requirements to SRS. Have diagrams started. |
| 2 | 10/11/19 - 11/5/19 | Complete class diagram and sequence diagrams. Partway through activity diagrams. Revisions to SRS have been made to make any unclear ambiguities more specific and concise. |
| 3 | 11/6/19 - 11/19/19 | Polish up anything we need to on diagrams, finish activity diagrams. Databases set up. |
| 4 | 11/20/19 - 12/5/19 | Have a working product that implements at least 25 requirements. |

## 4.2 Diagrams

### 4.2.1 Description

This section will be used to explain our diagrams and our design choices. We have created a class diagram, activity diagrams, sequence diagrams, and an application navigation graph.

**4.2.2 Satisfying Requirements**

The focus of our first sprint is to start the class, sequence, activity, and state diagrams to help visualize our application and see if any necessary changes became apparent as we finalize our models. We plan on revising our diagrams after the first sprint. The first versions will be rough, they will be improved as we continue to figure out what content will be needed to achieve our requirements. Due to the sheer number of diagrams, not all of them will be finished by the end of the first sprint, so the second sprint will be used for making more progress. By sprint three, we expect all the diagrams to be finished and can be improved as we get further into coding.

## 4.2.3 Design

#### 4.2.3.1 Description

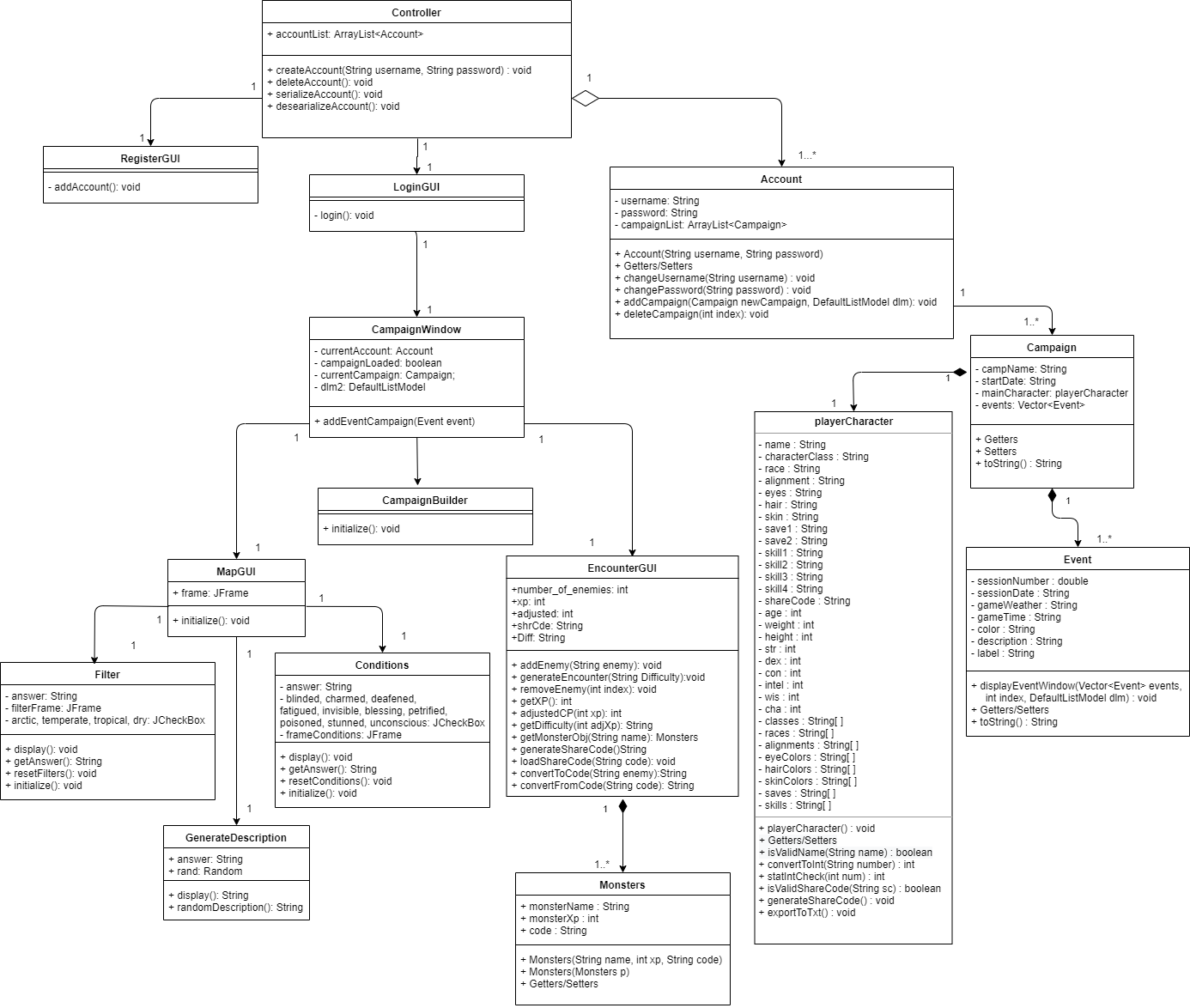
We focused on a simple design for the first sprint, as we felt the features we were implementing would be much further expanded on and integrated down the line in order to work with the rest of our use cases.

#### 4.2.3.2 Standards, Frameworks and Tools

Tools we used in the creation of the design of the sprint were modeling tools from Microsoft Visio, and a free modeling website, gliffy.com. Standards we followed in the design of features of the sprint was UML.

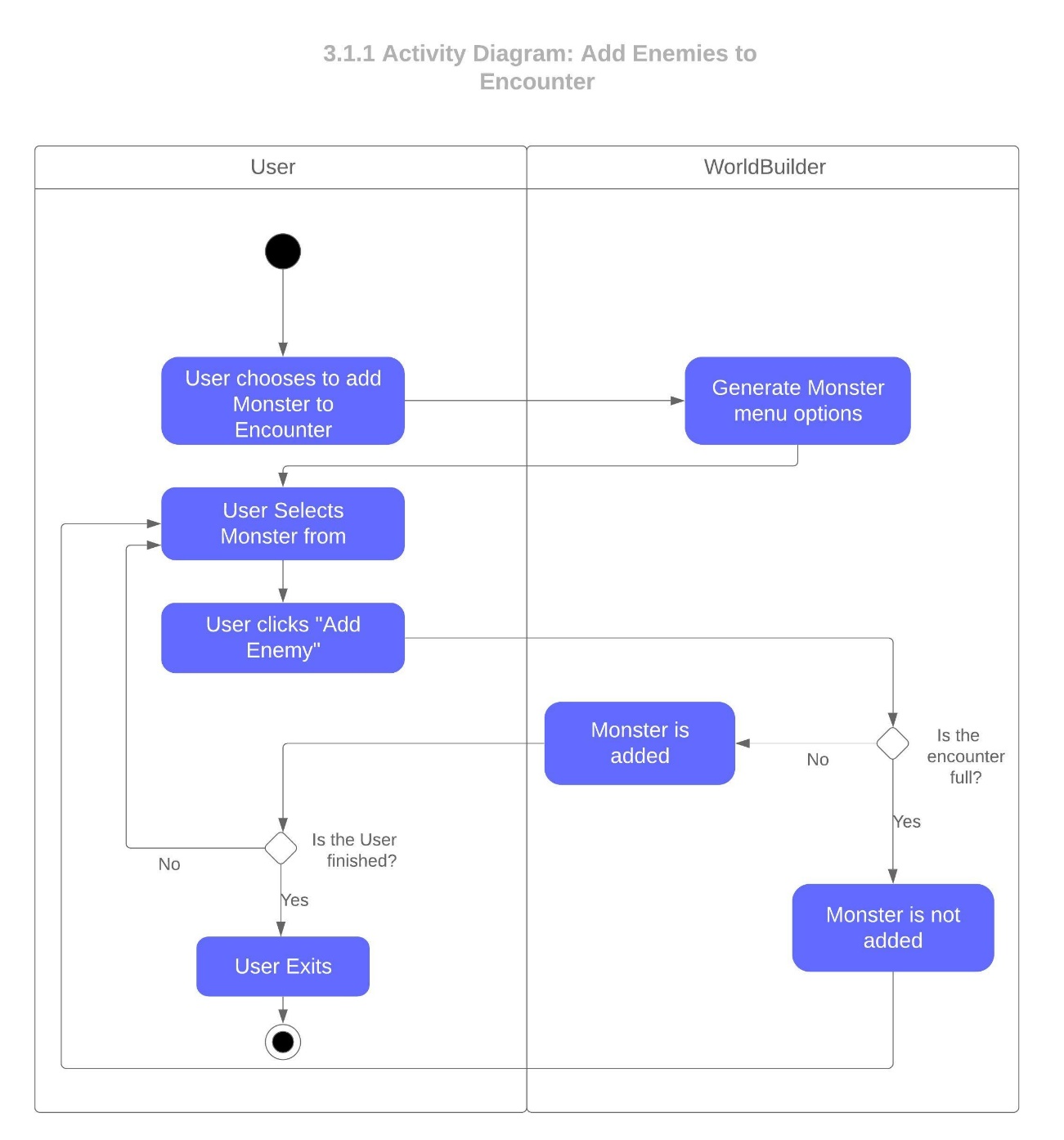
#### 4.2.3.3 Class Diagram

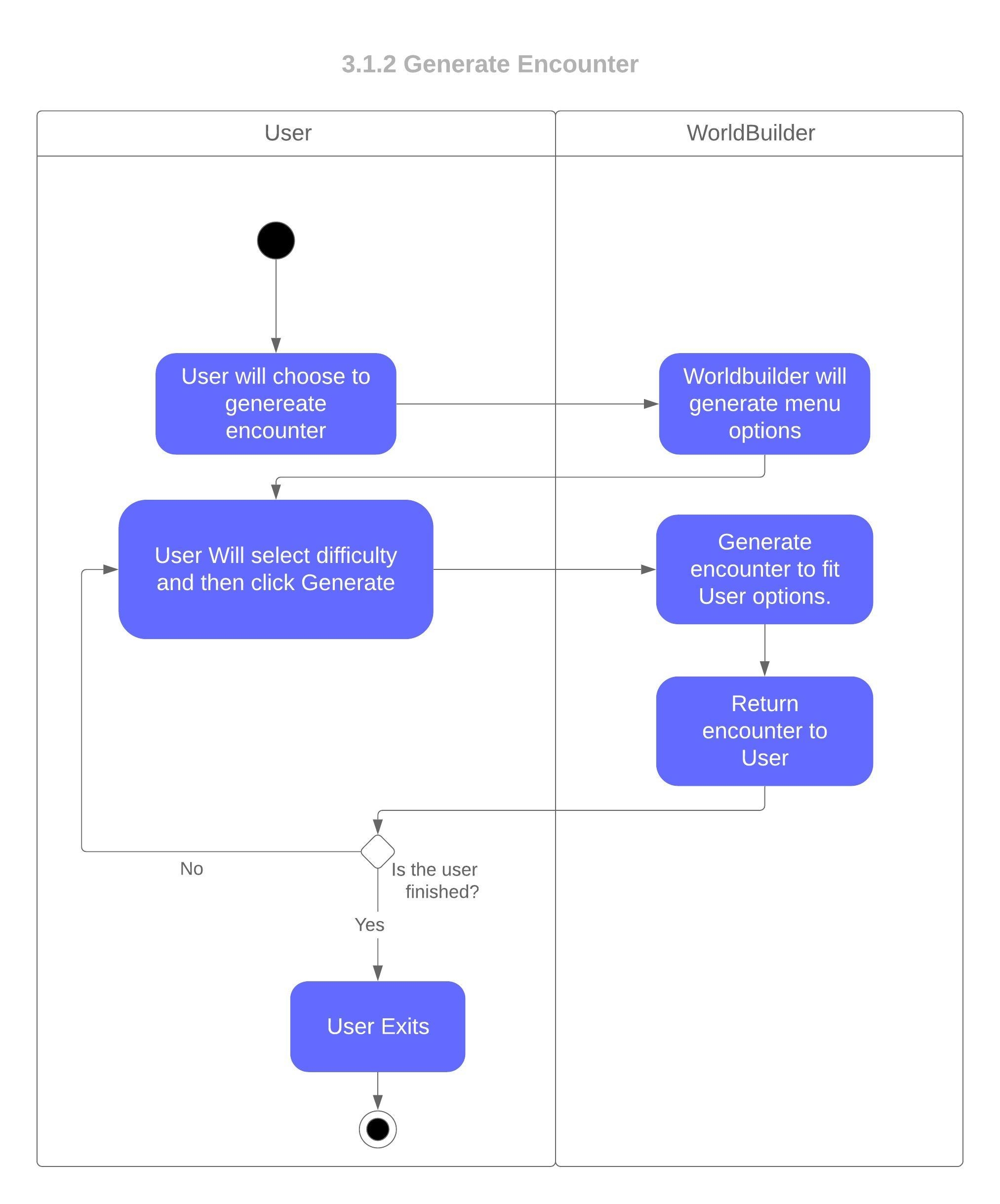
**Figure 1: Class Diagram**

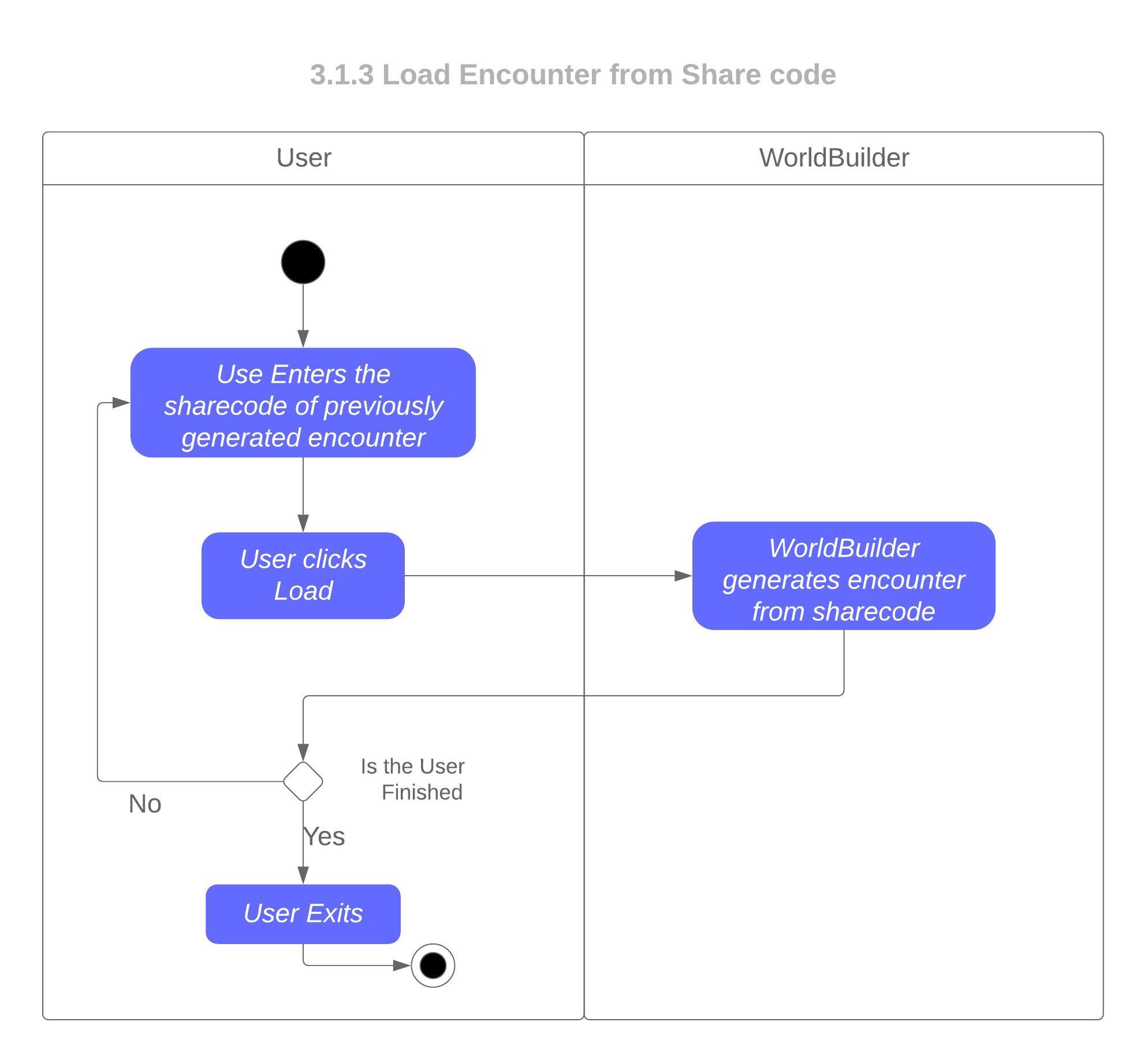


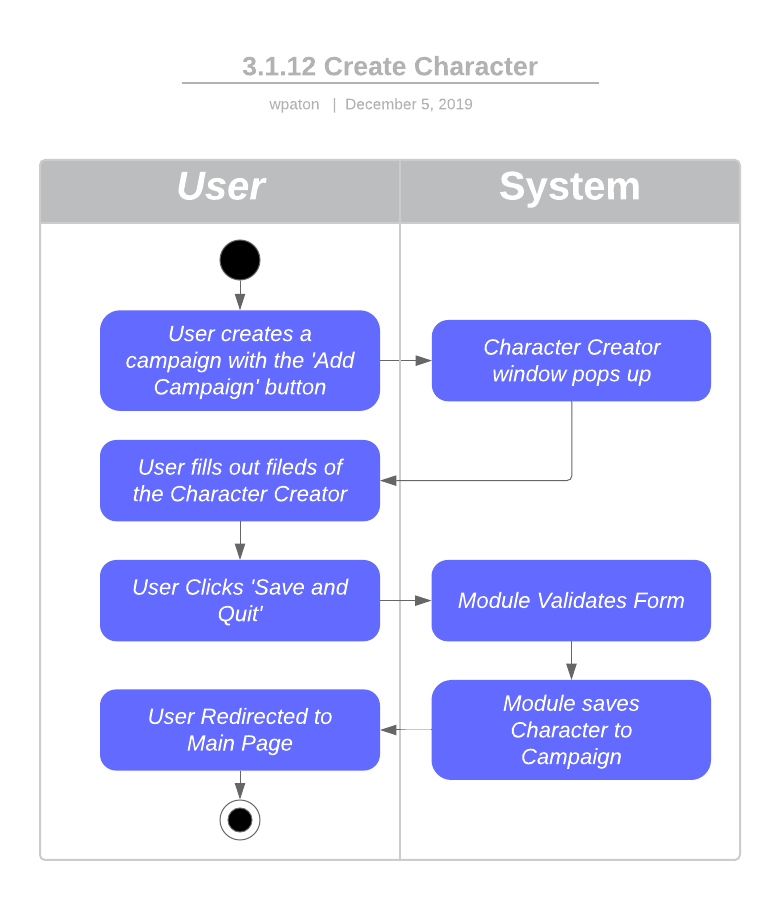
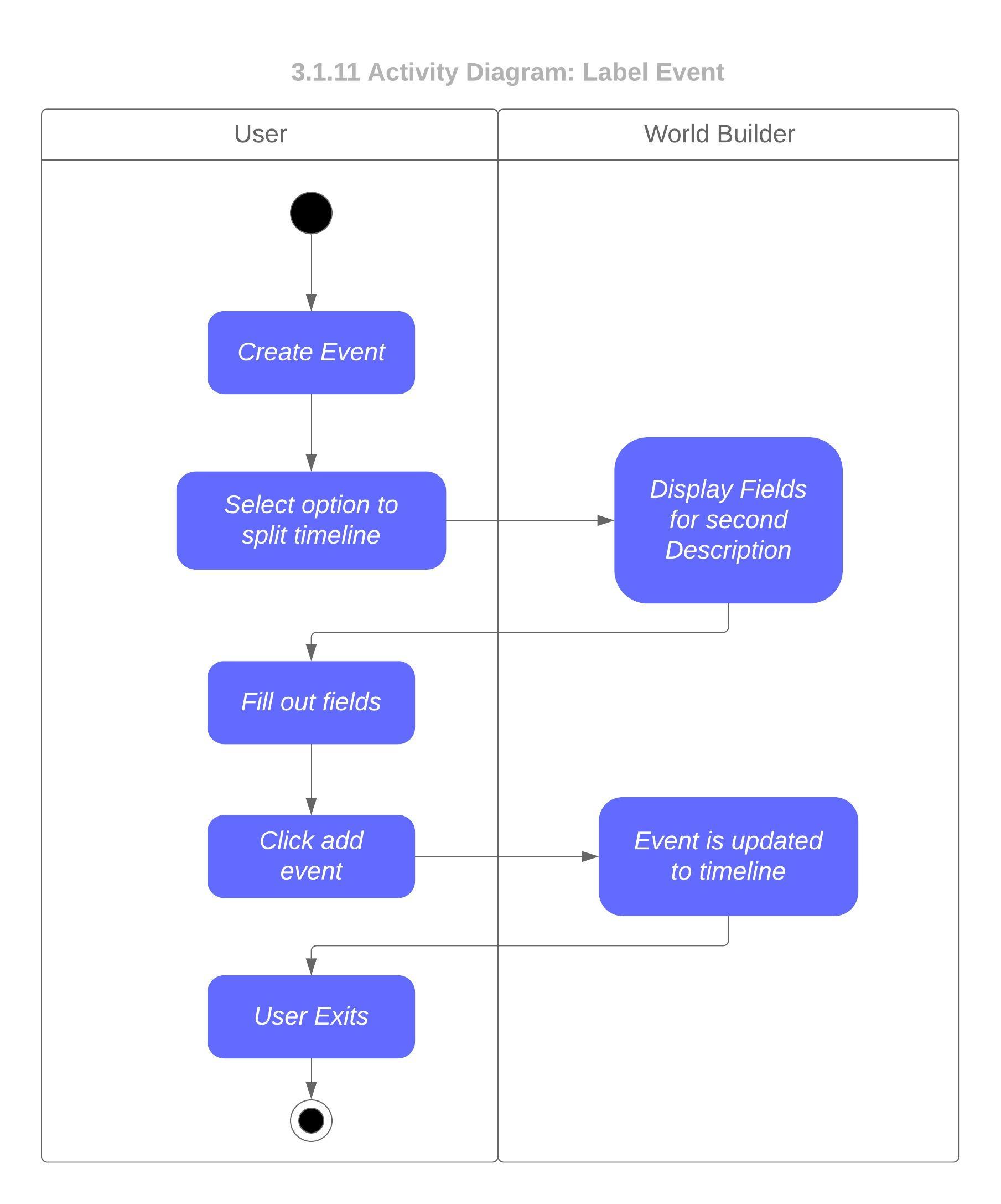
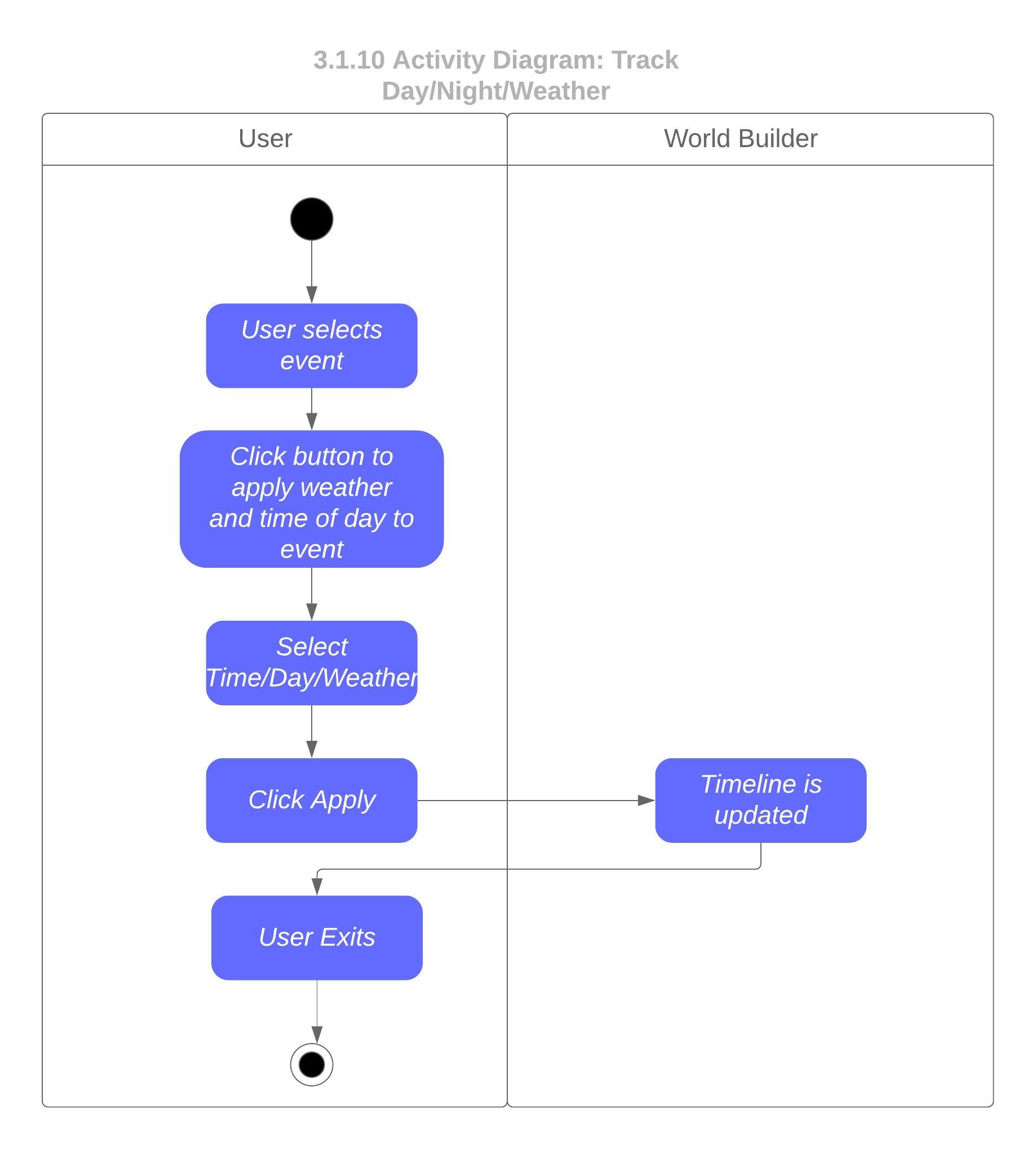
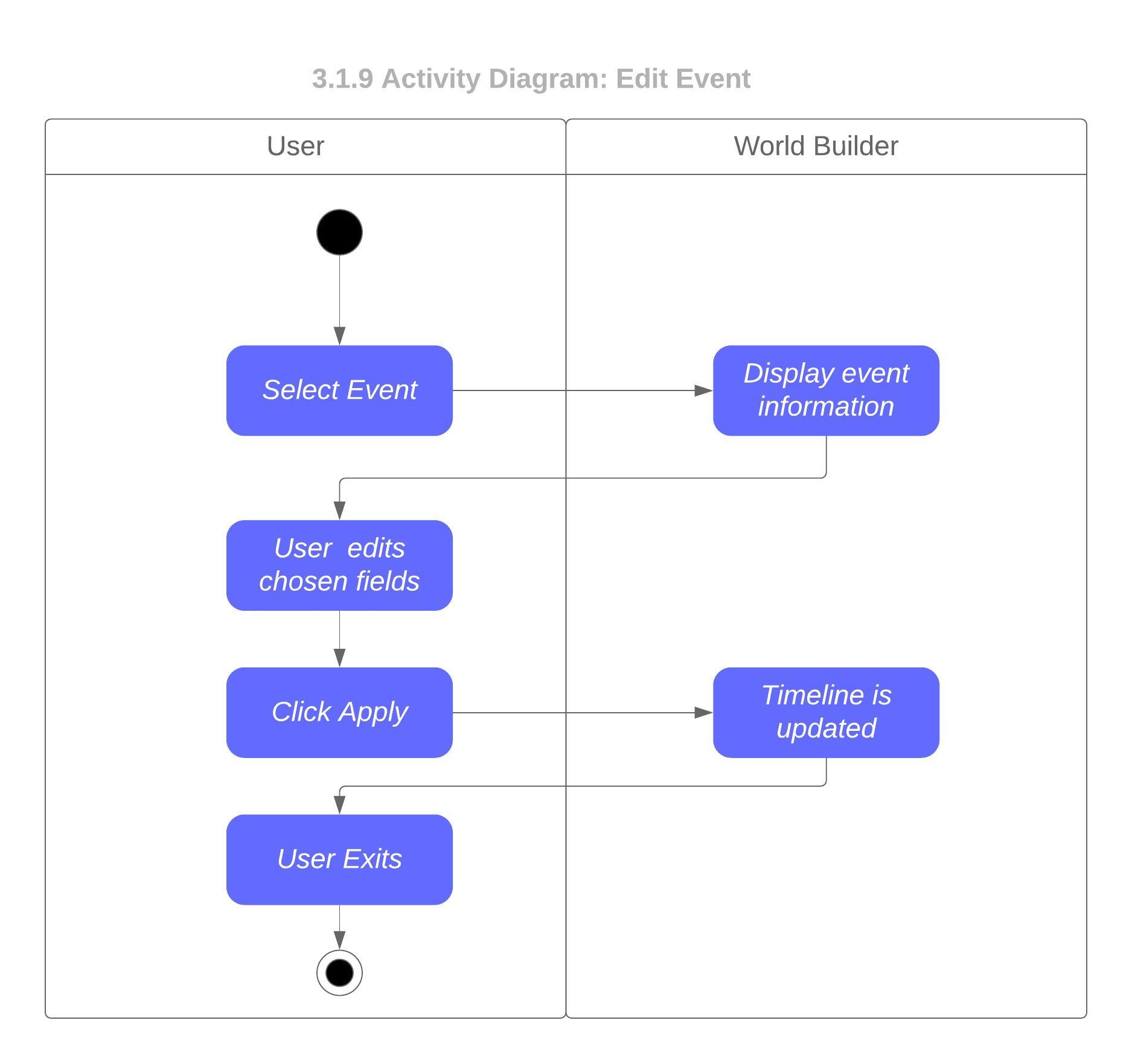
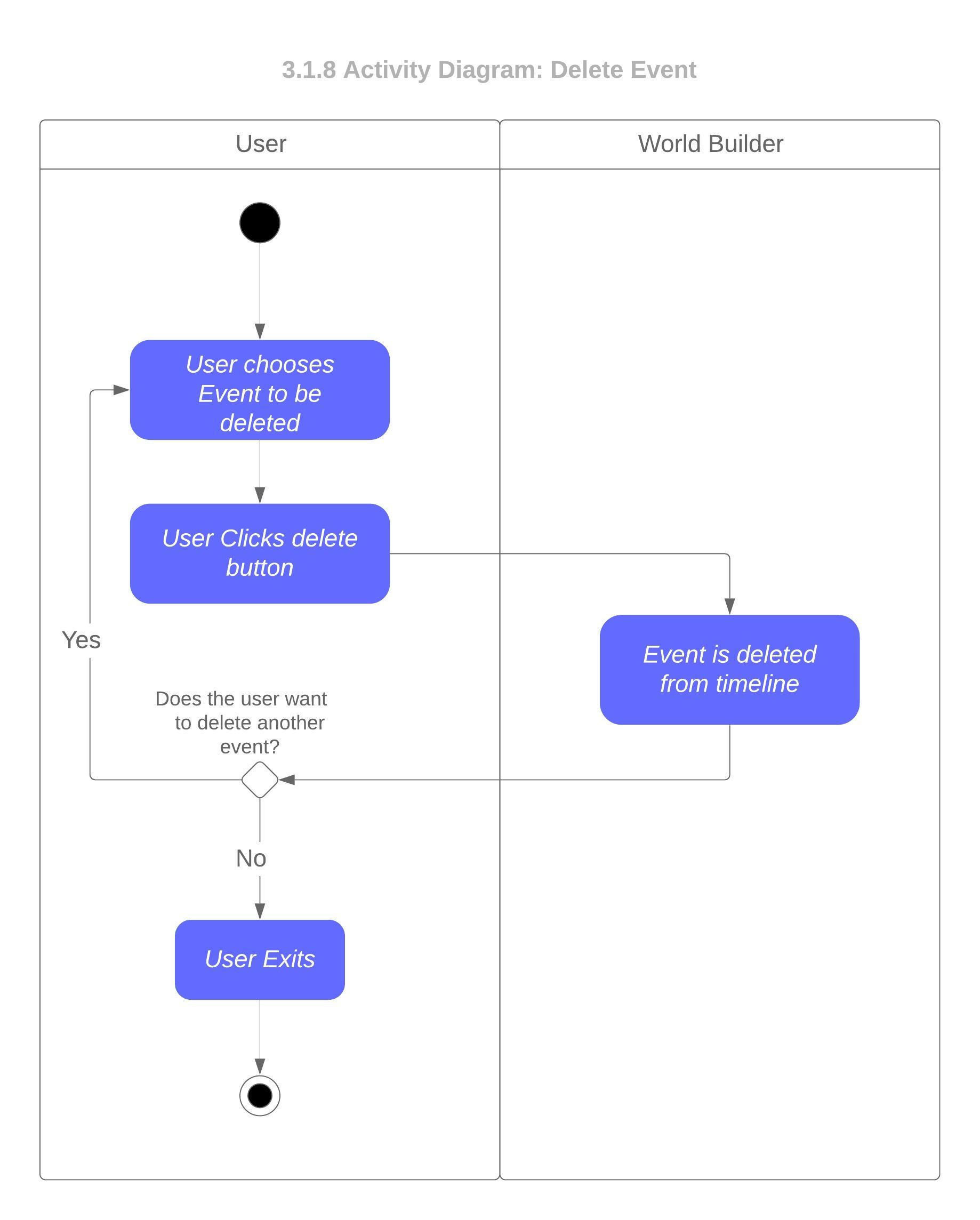
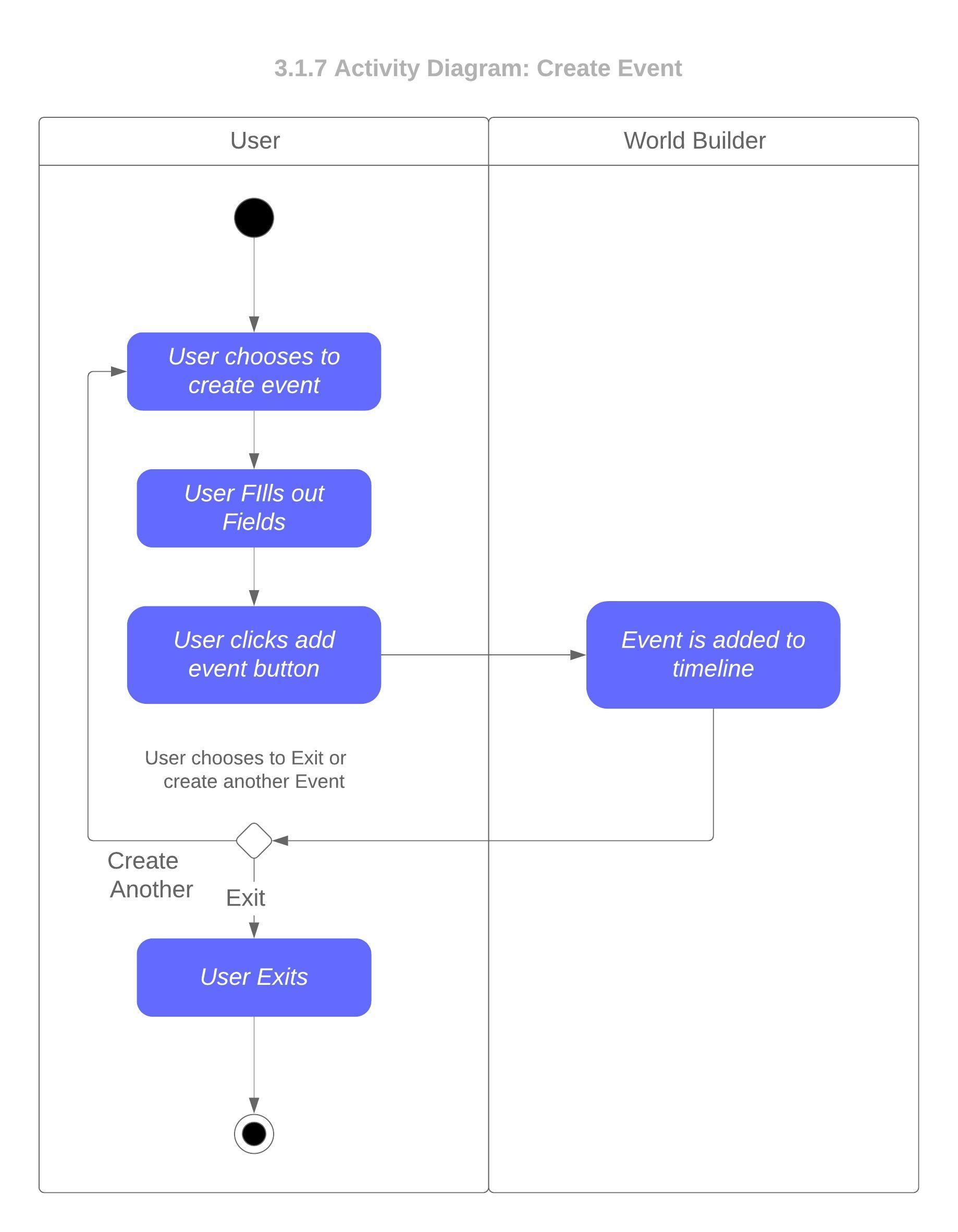
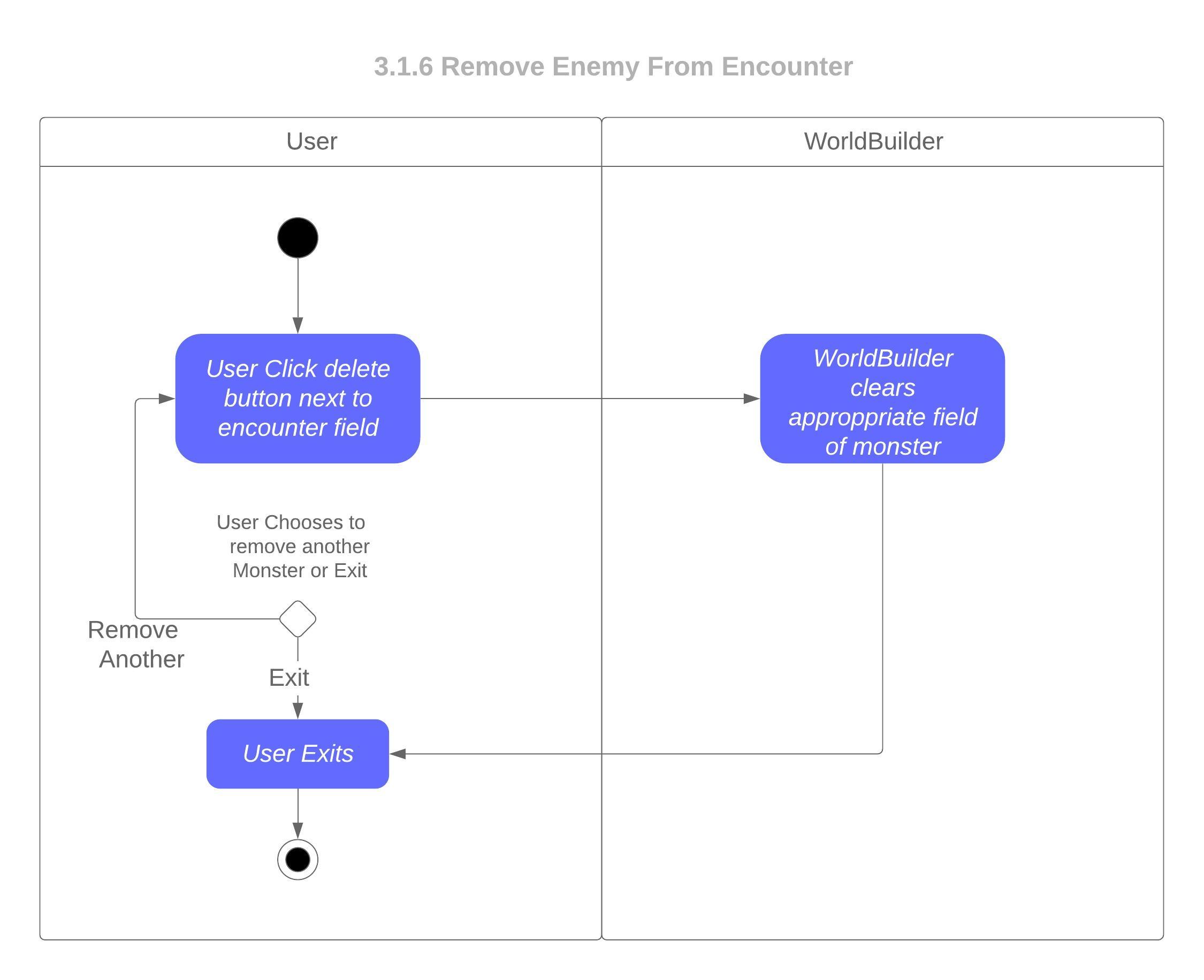
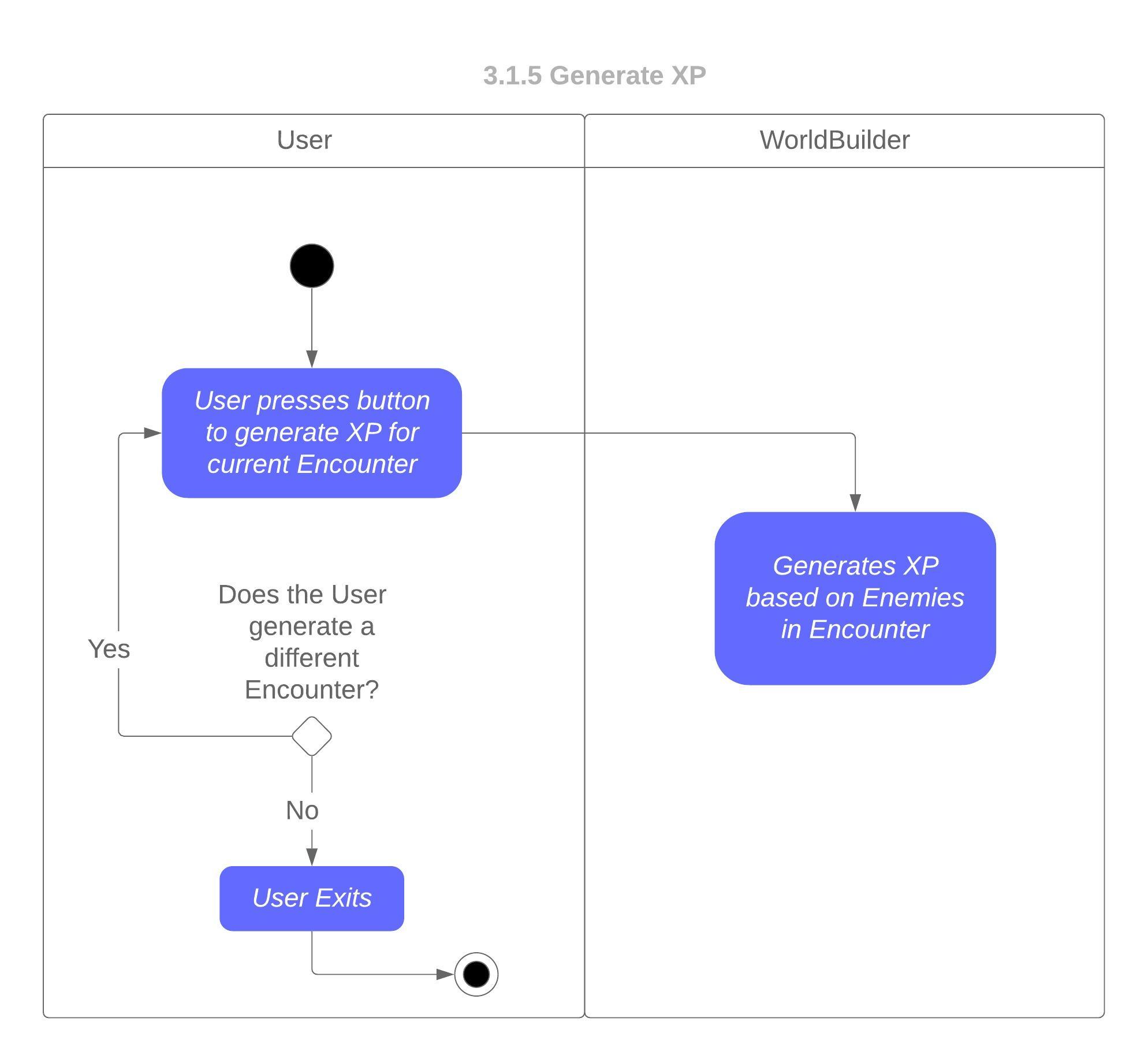
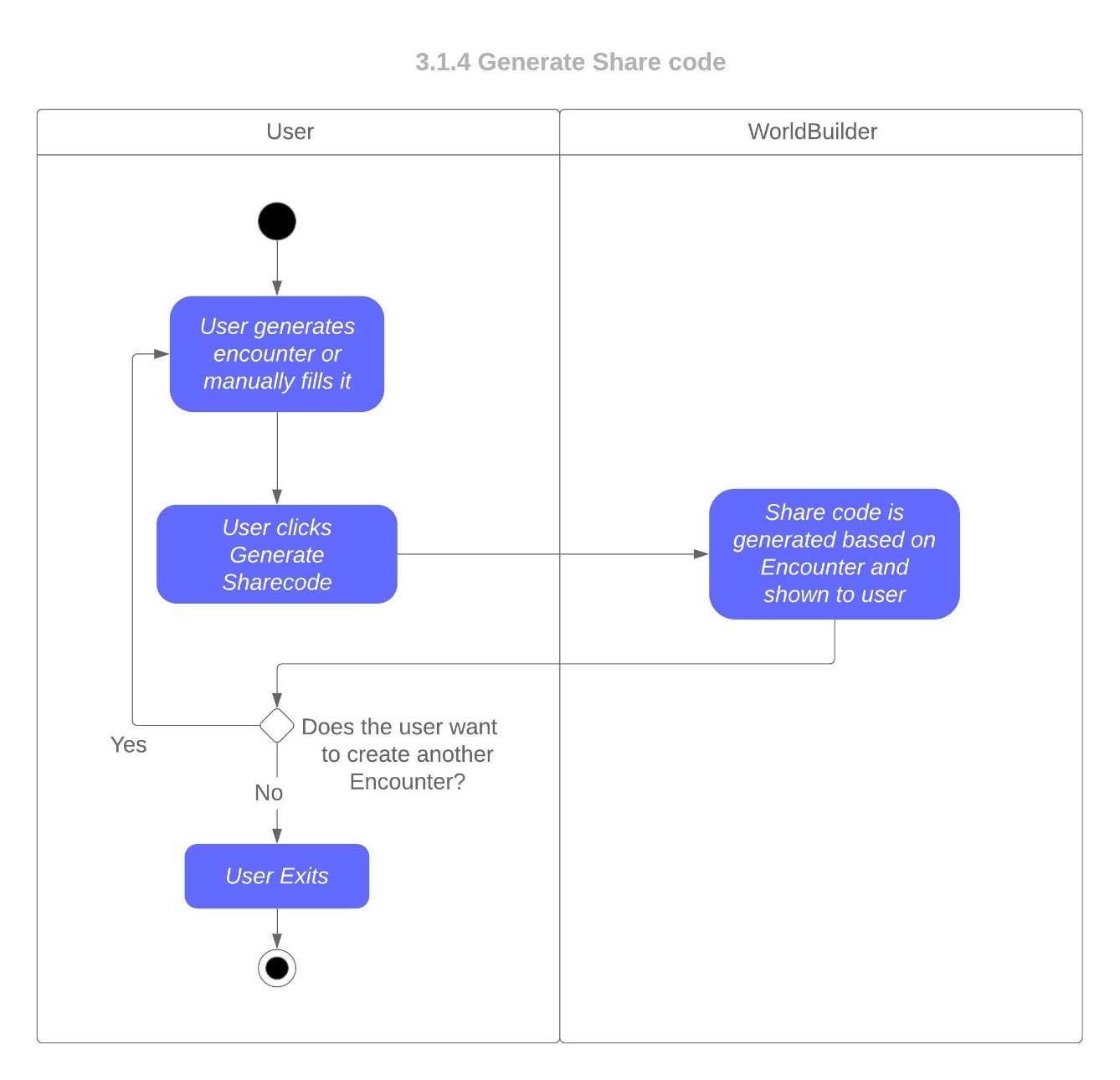
#### 

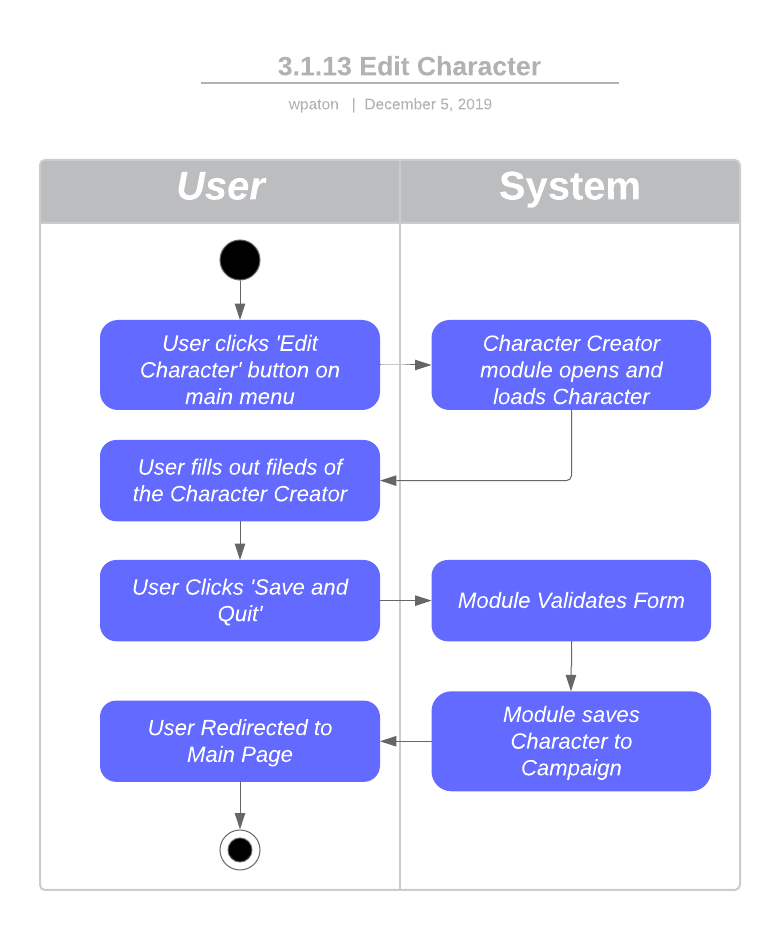
#### 4.2.3.4 Activity Diagrams

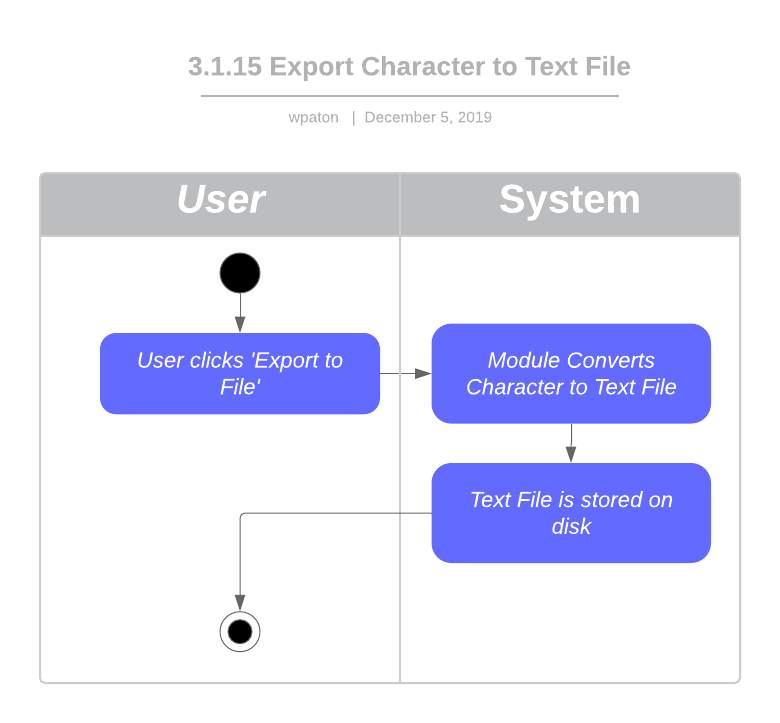


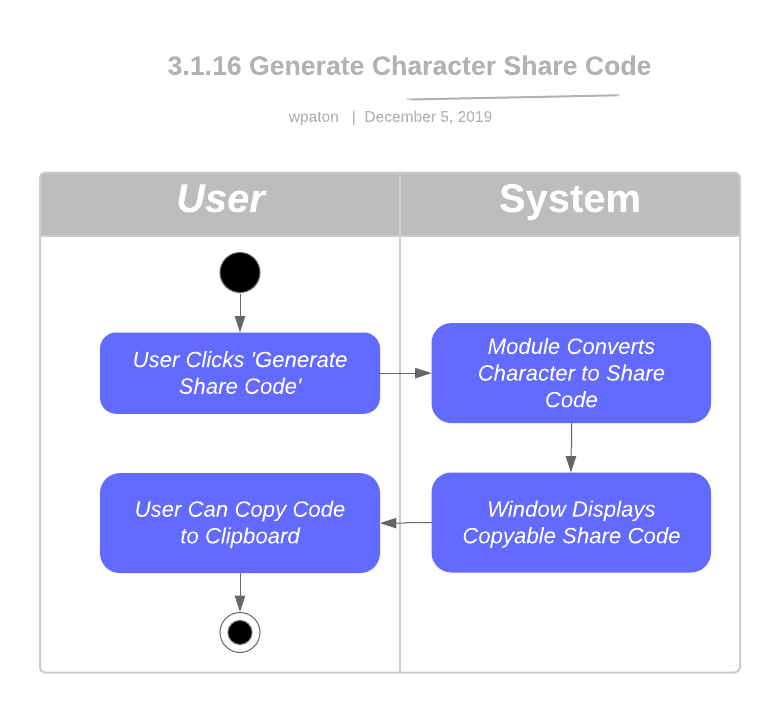


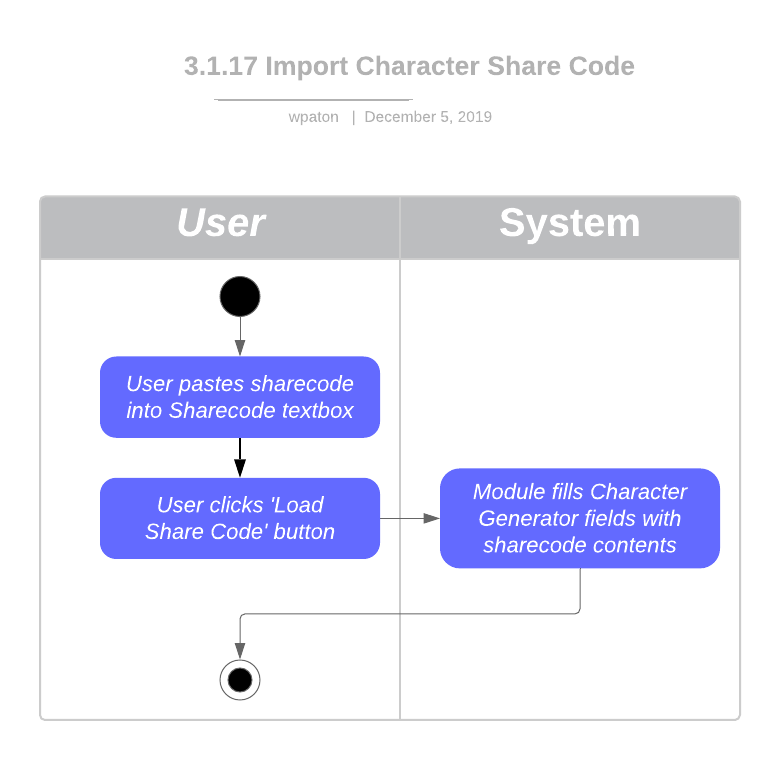


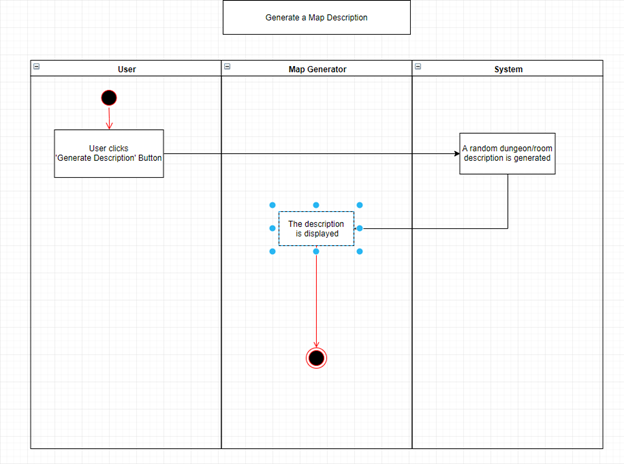


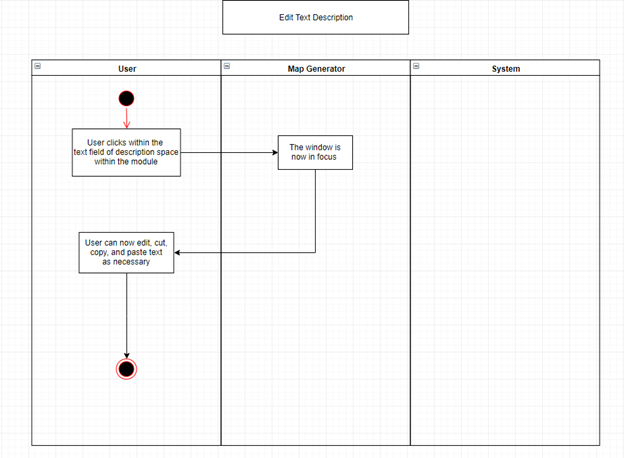


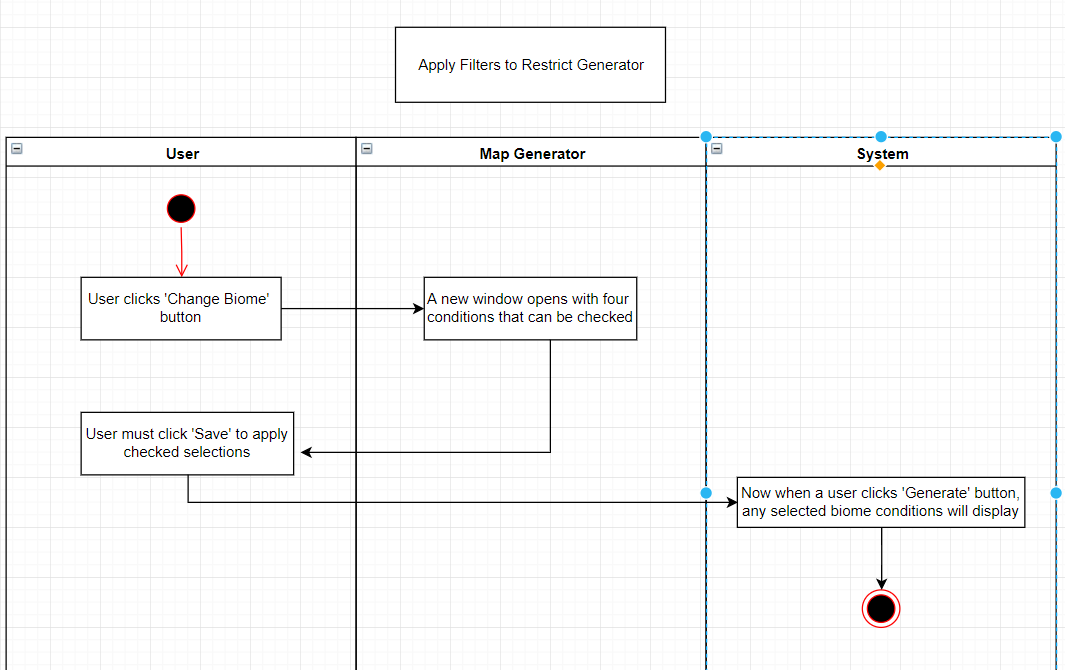


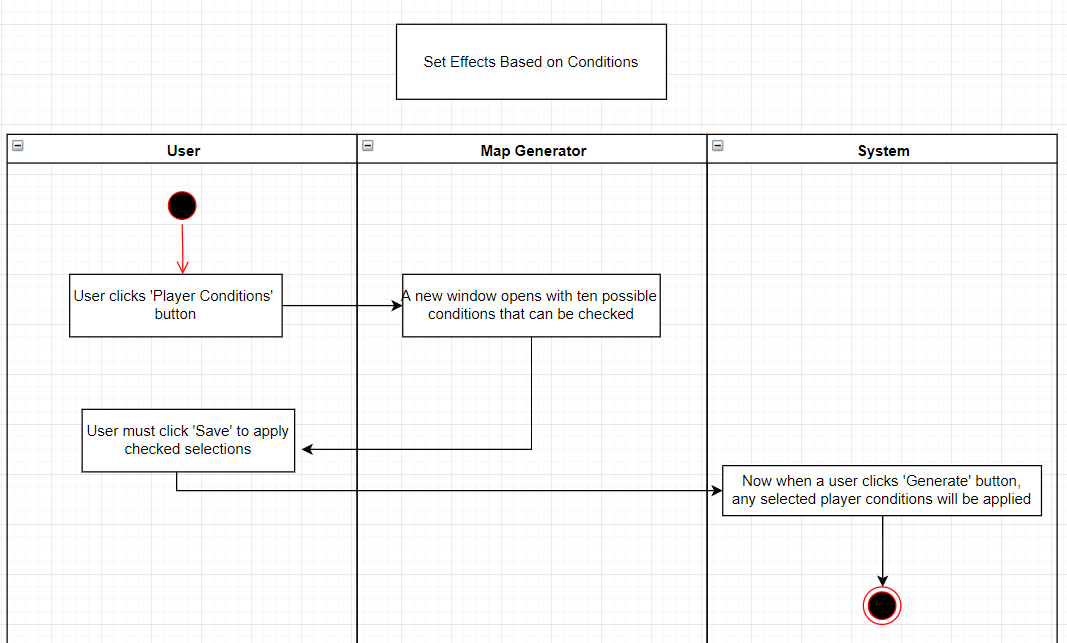


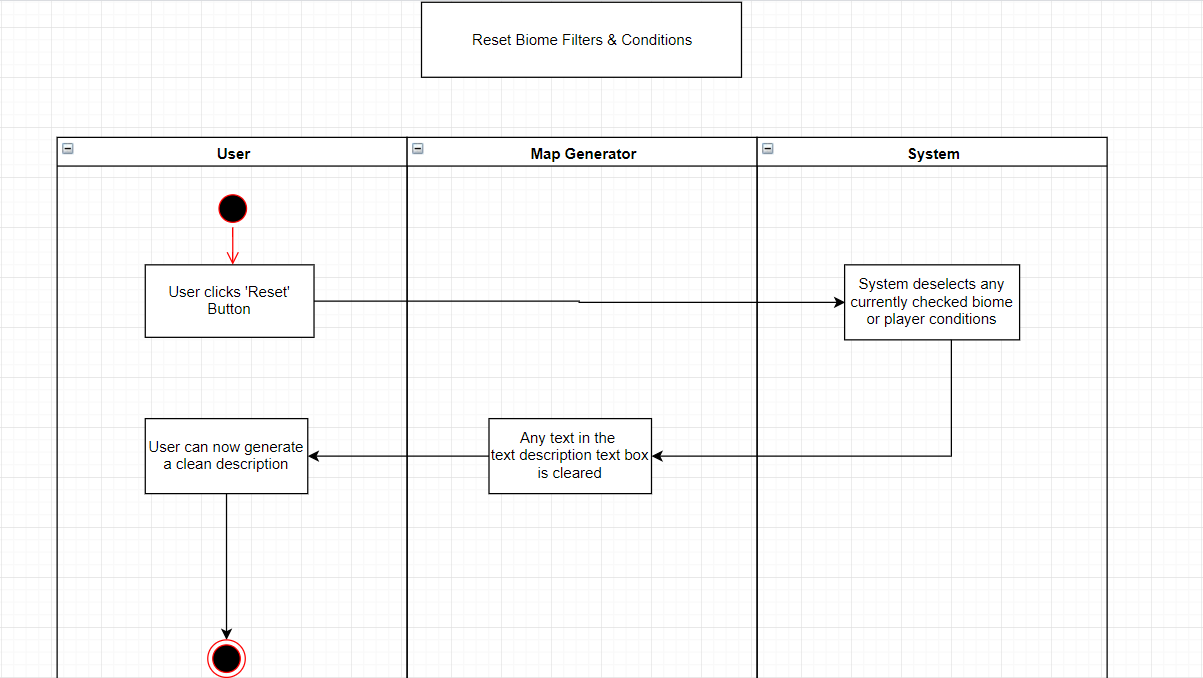


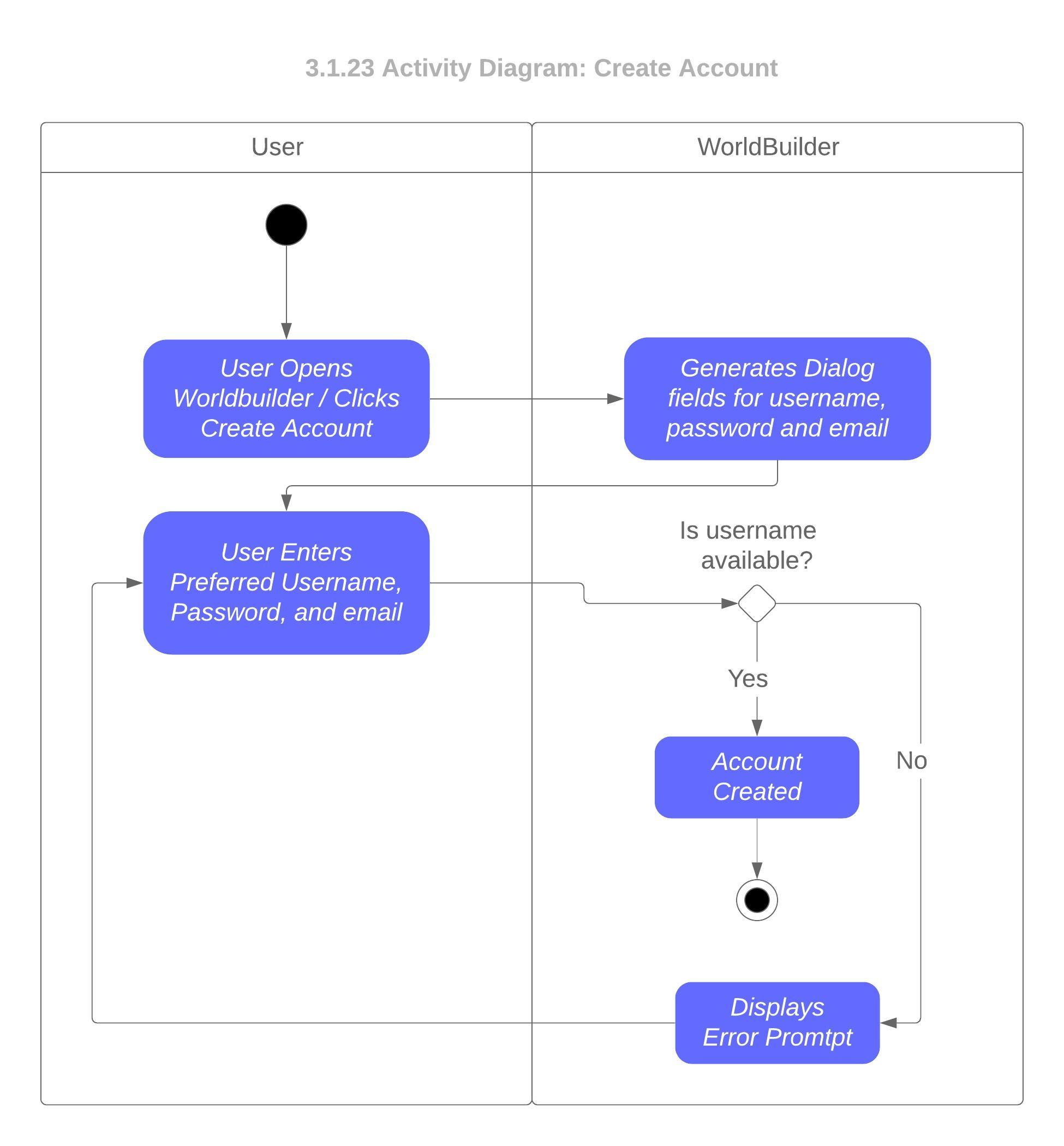


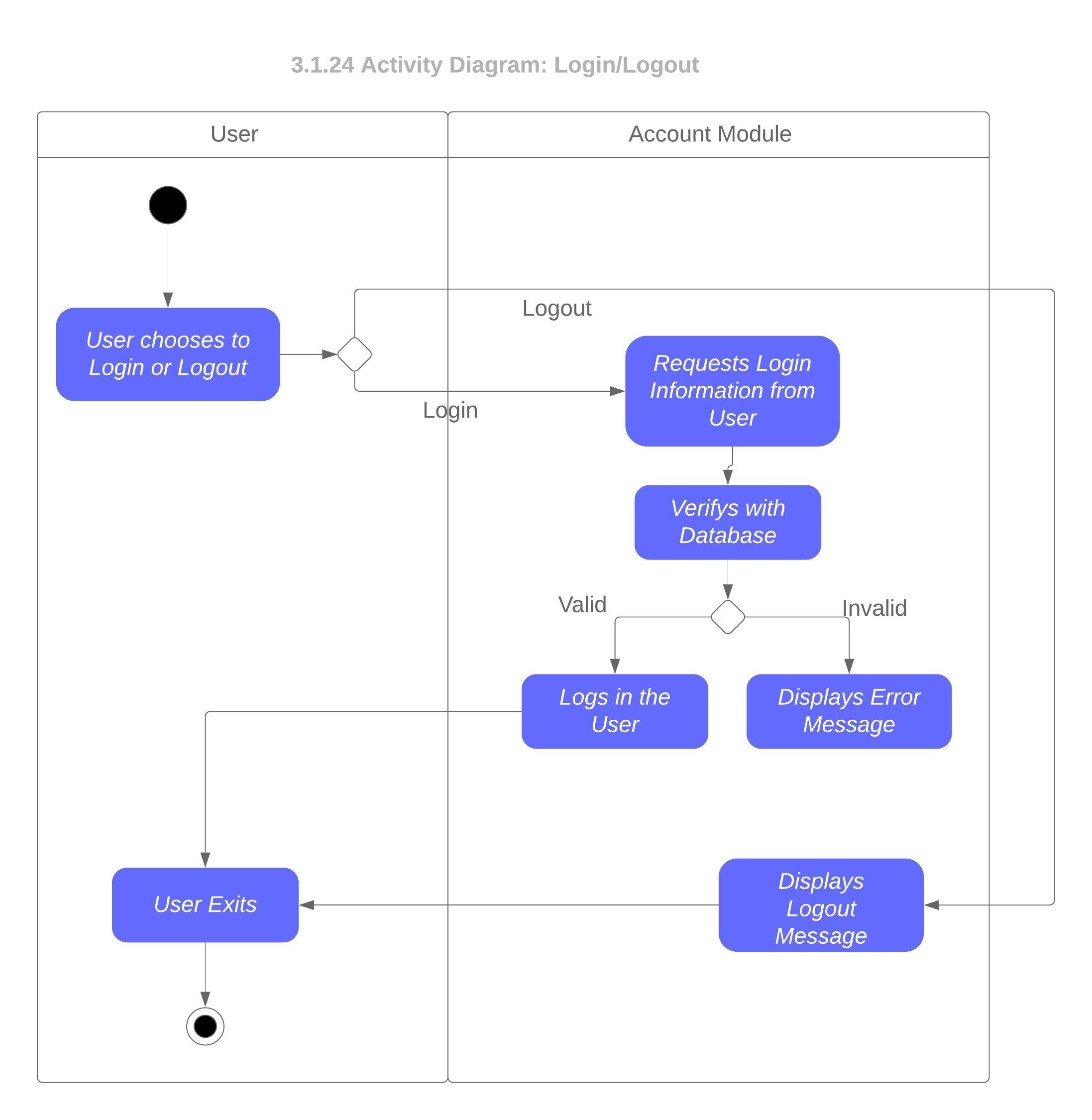


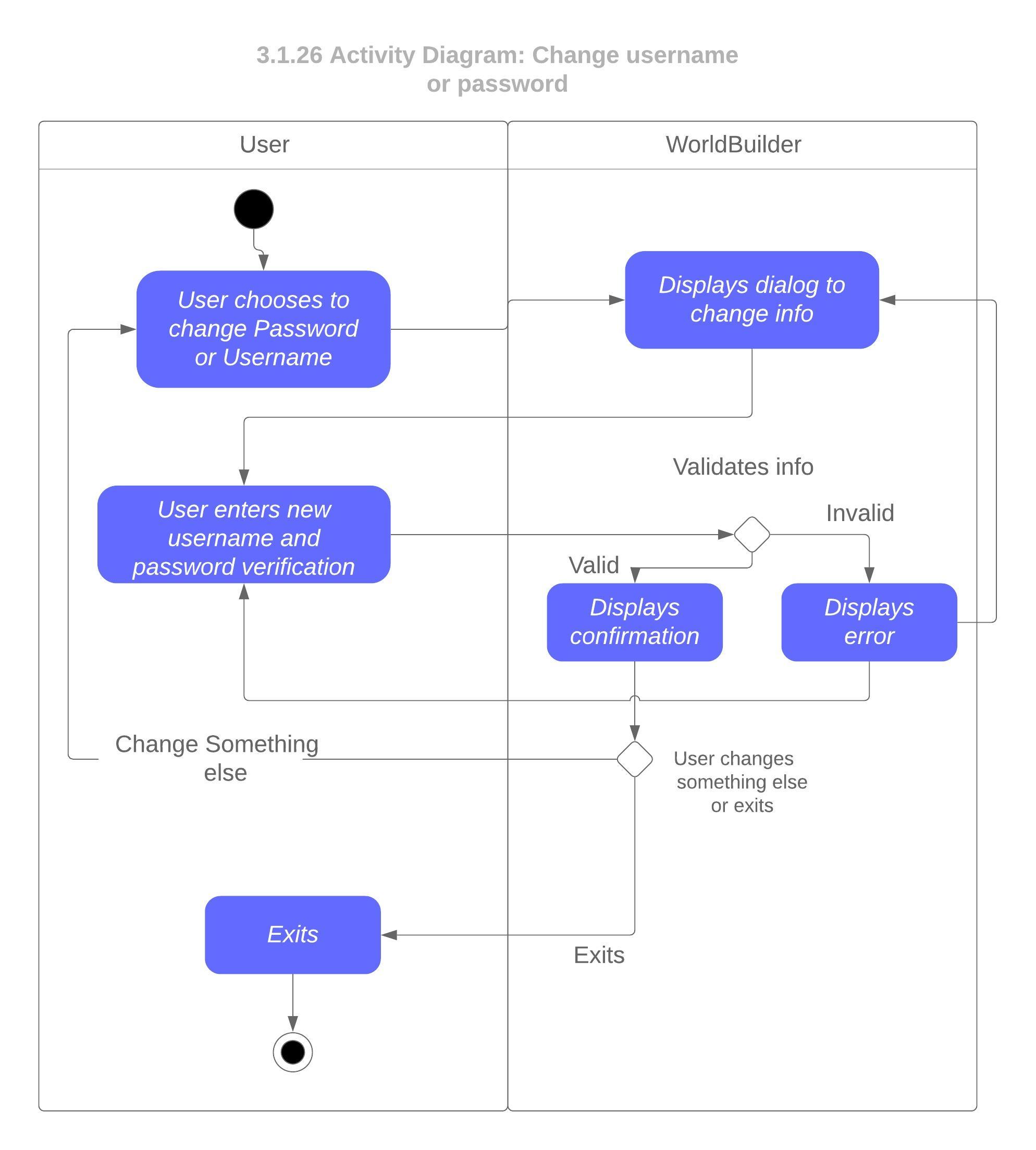
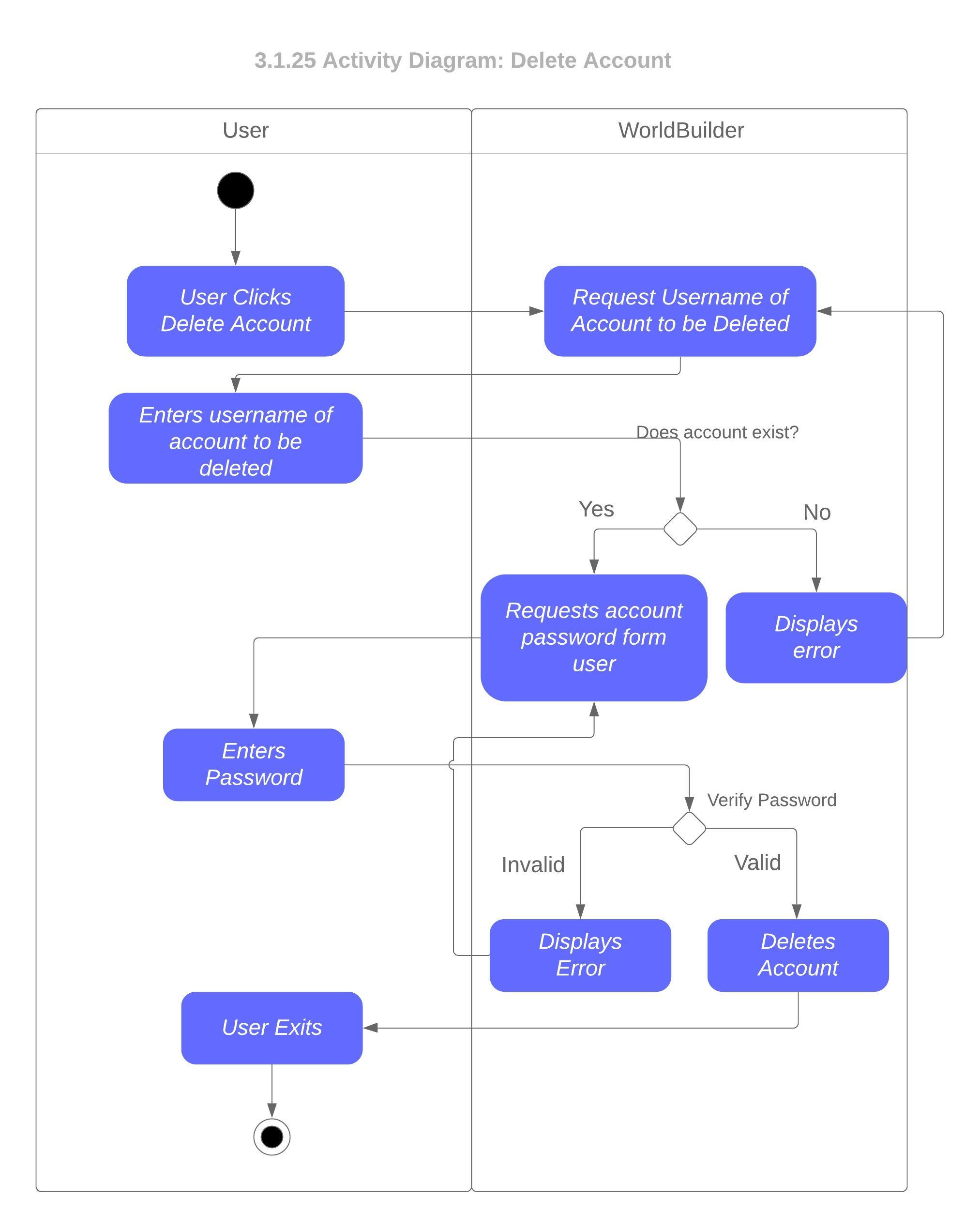


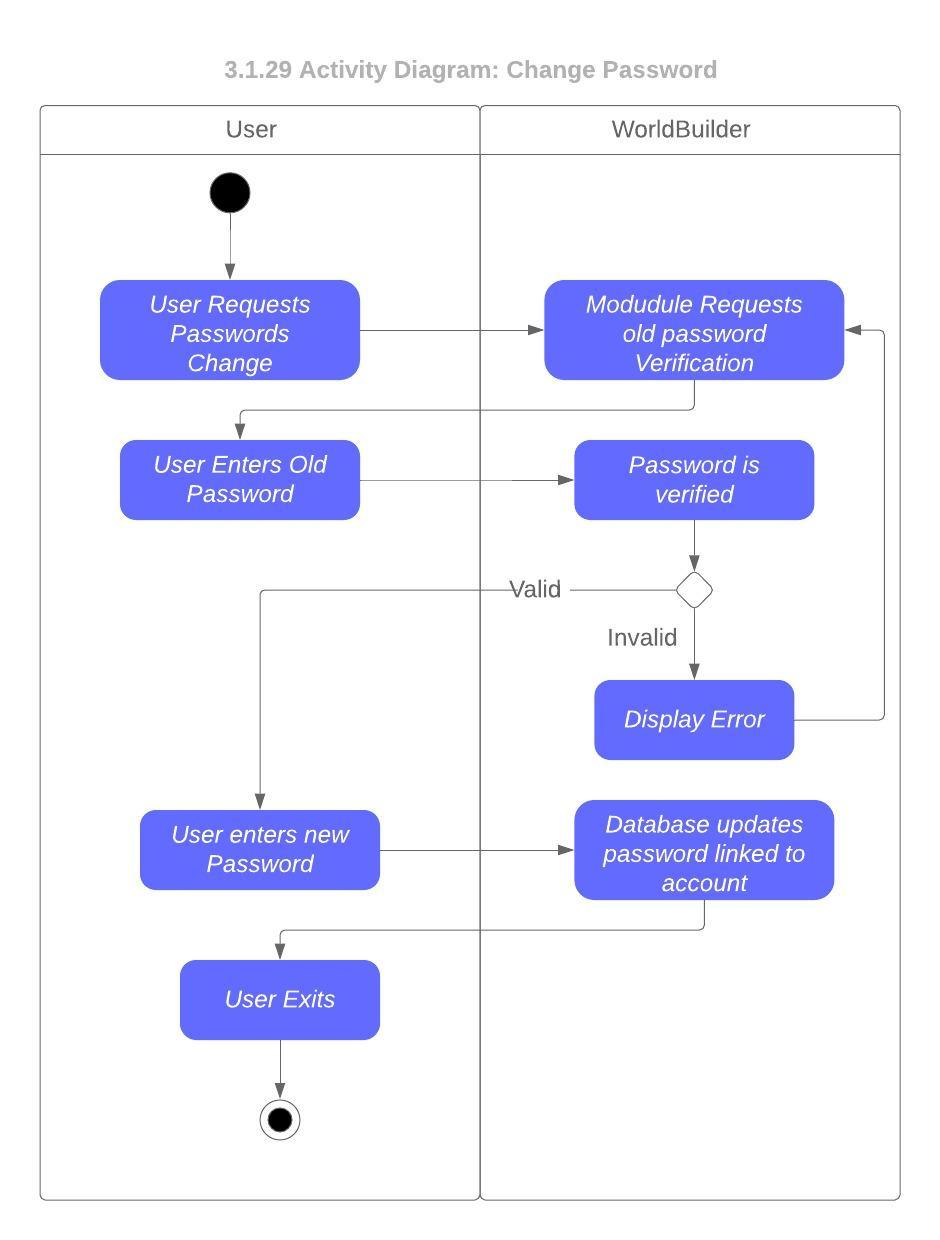
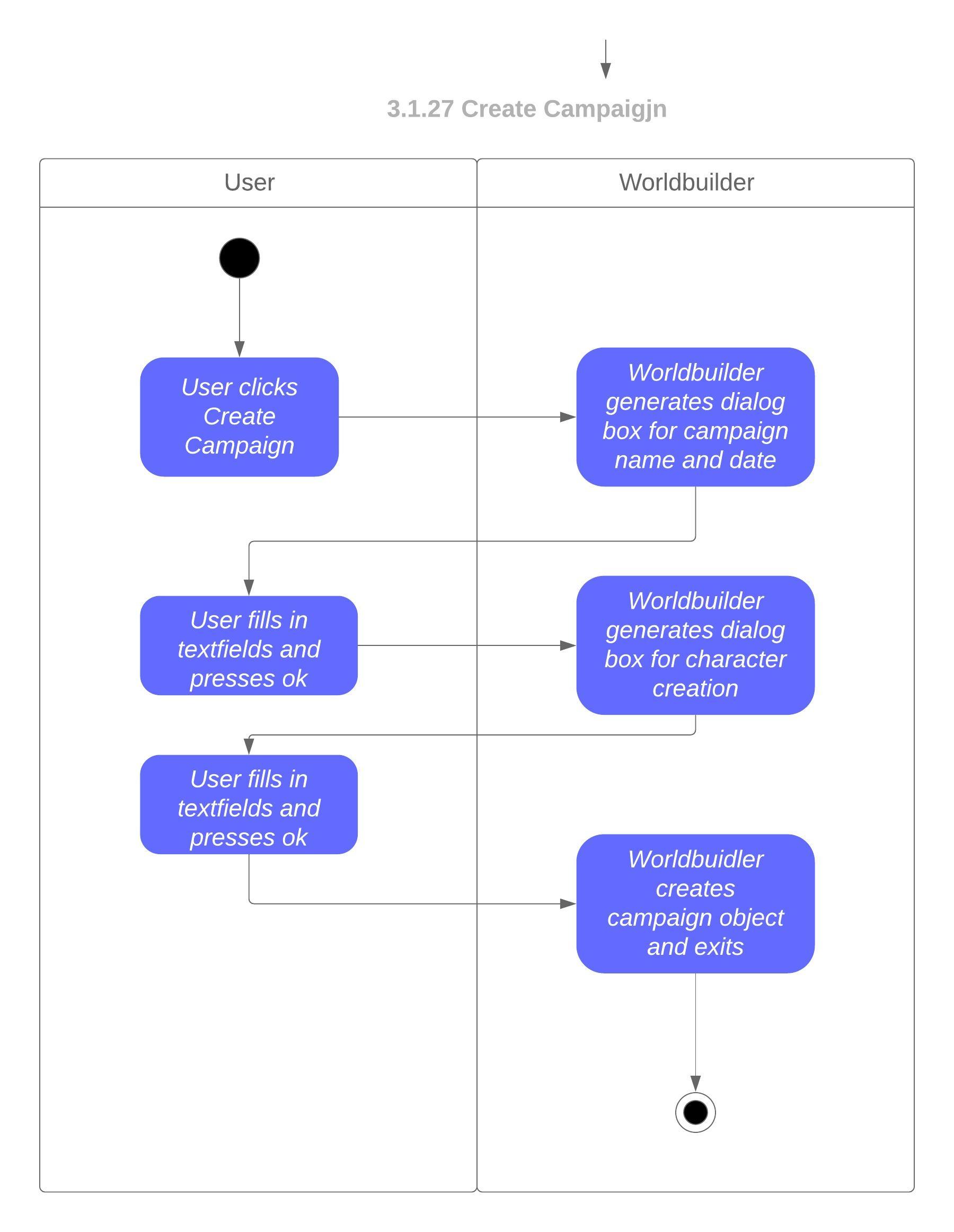


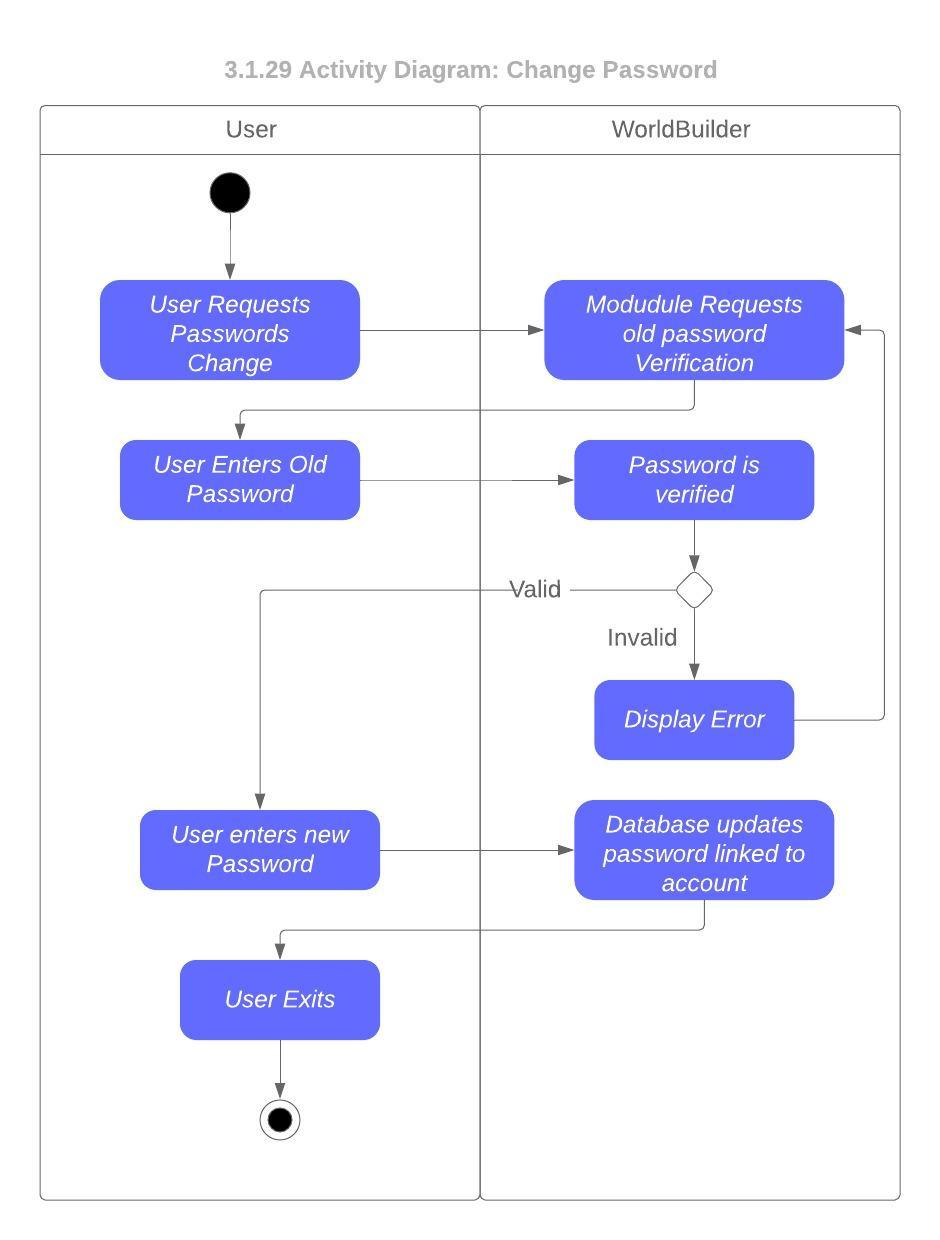




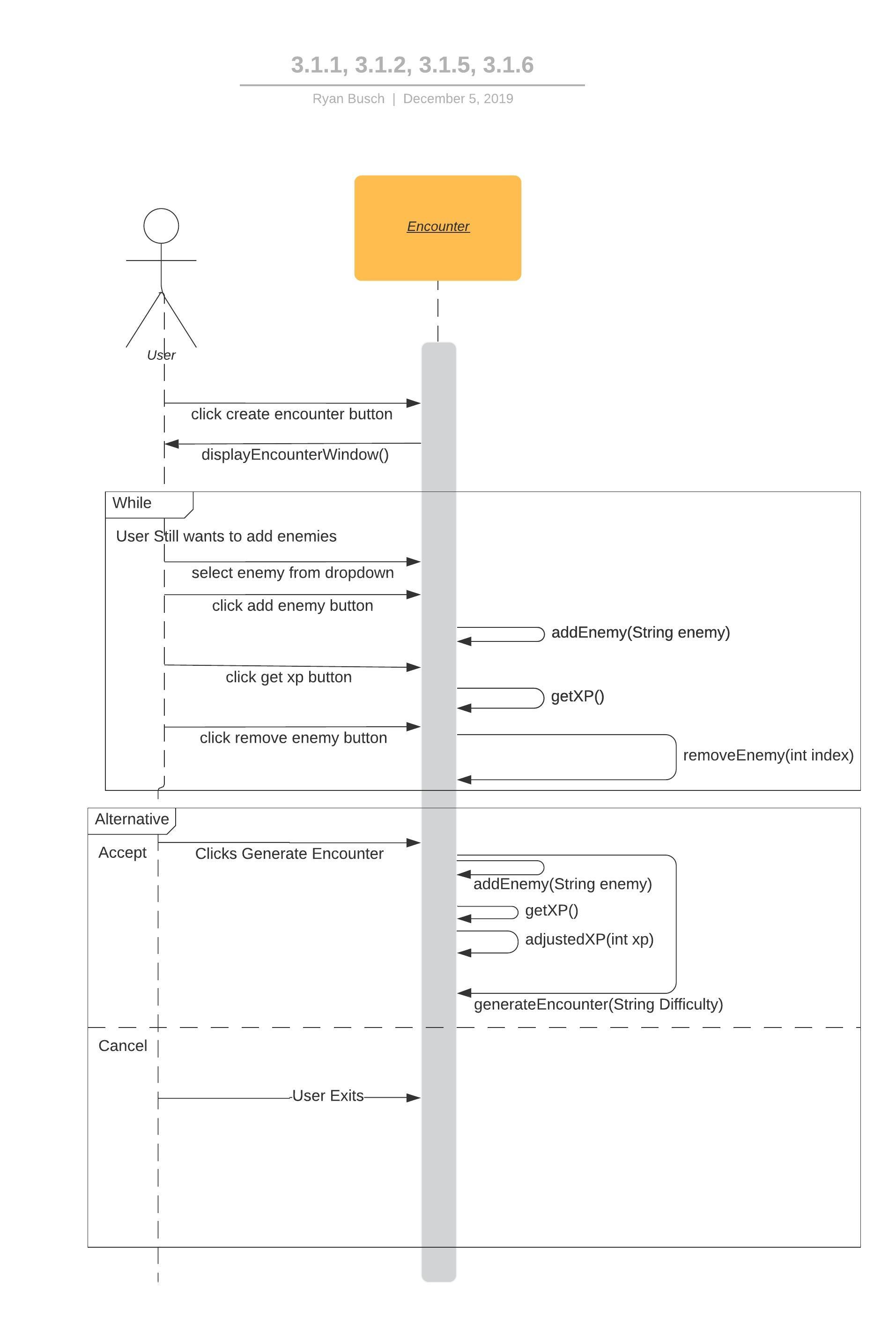


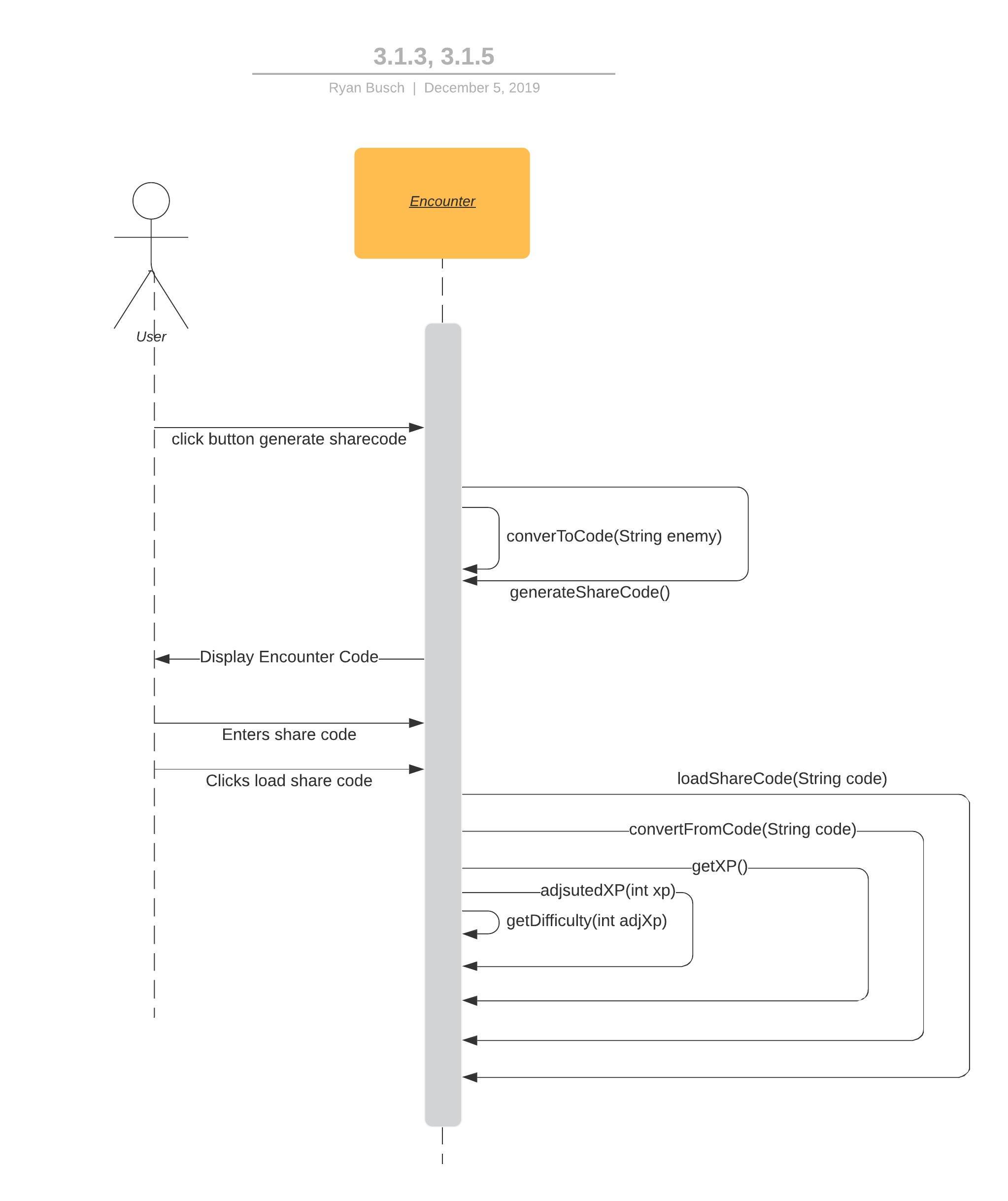


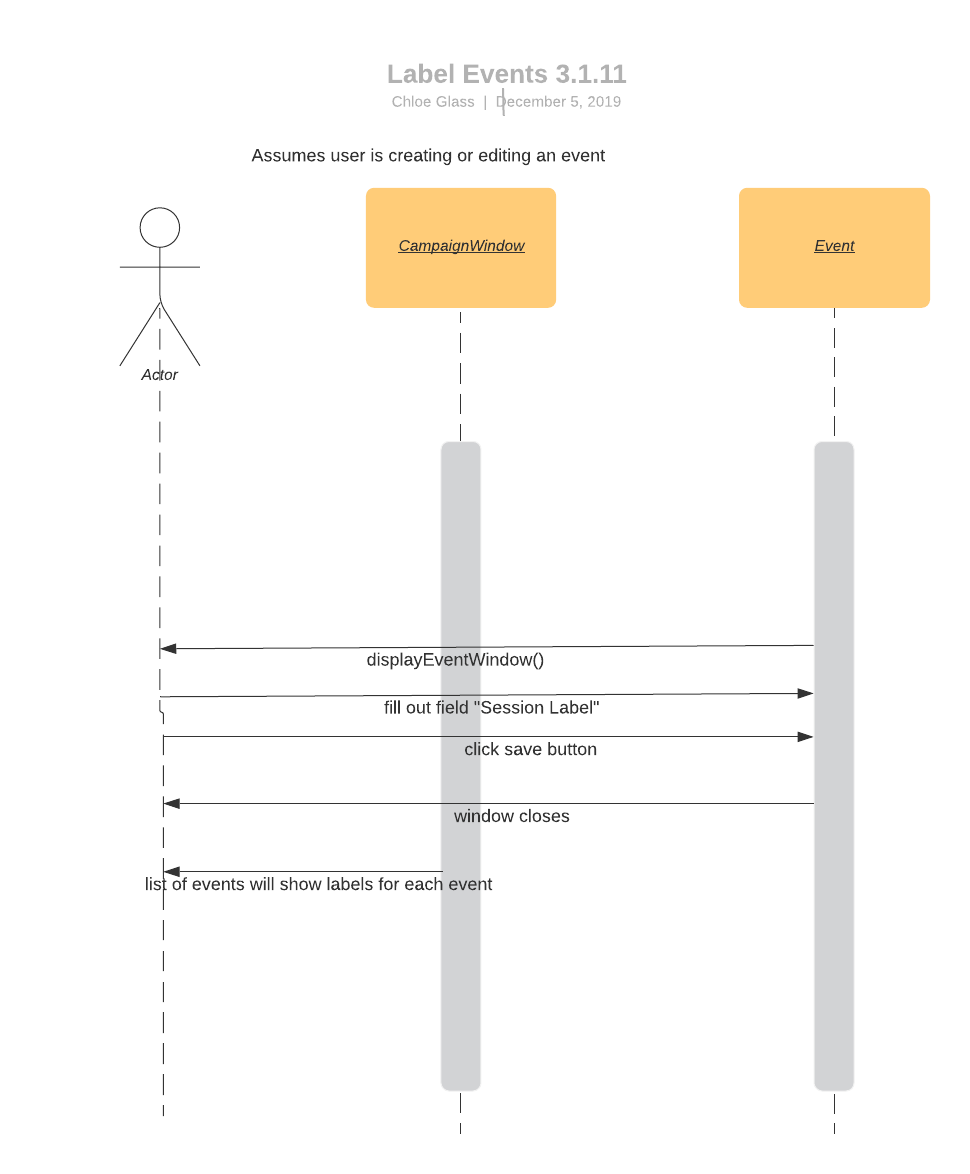
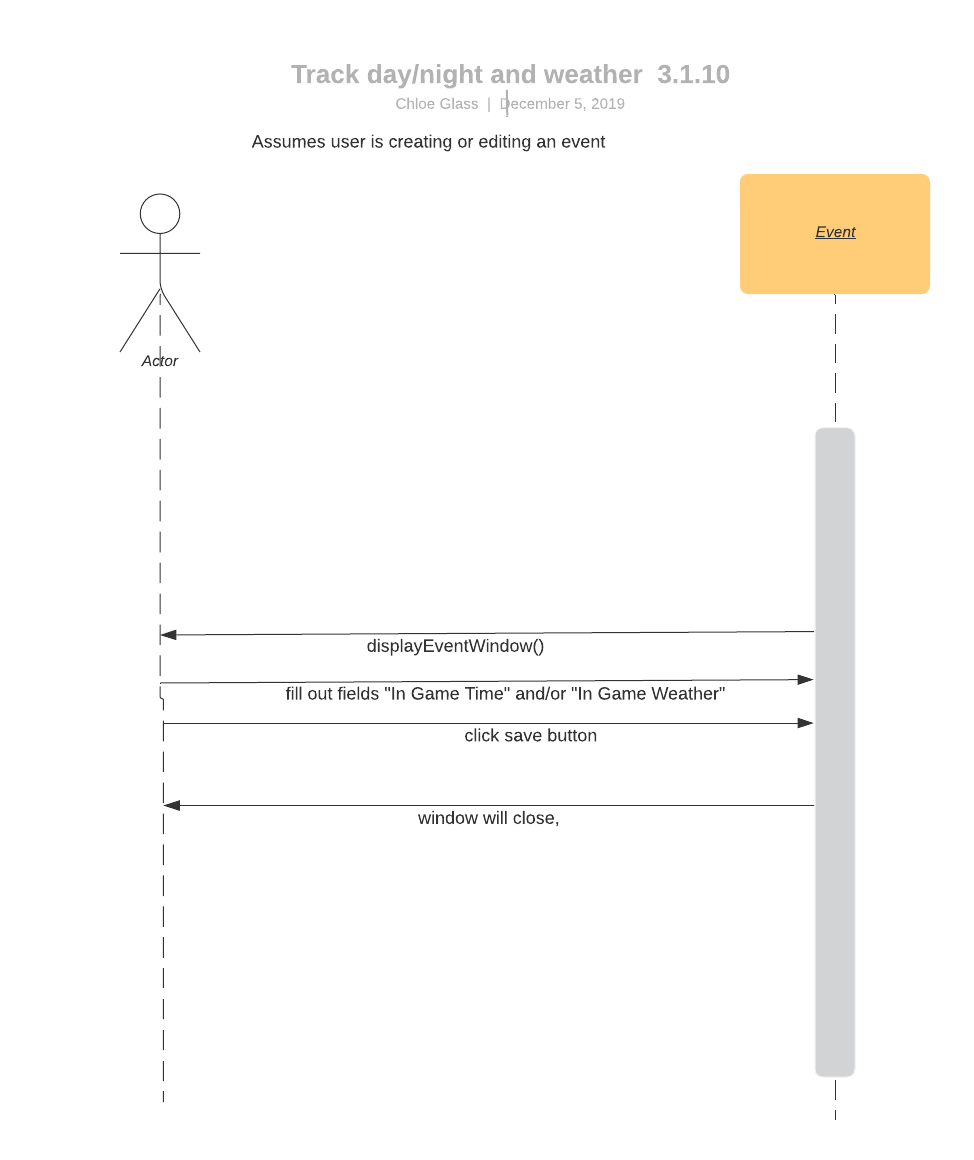
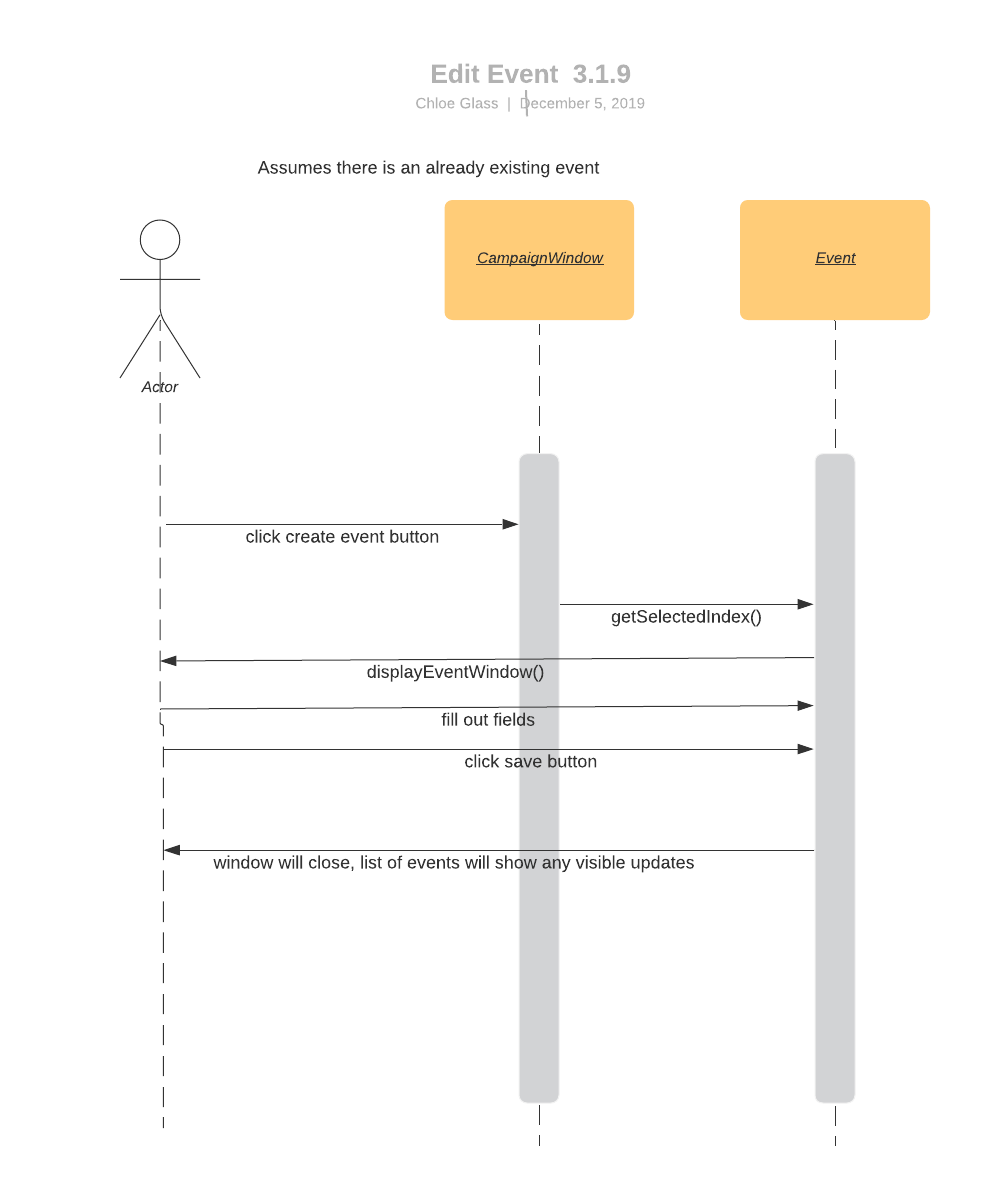
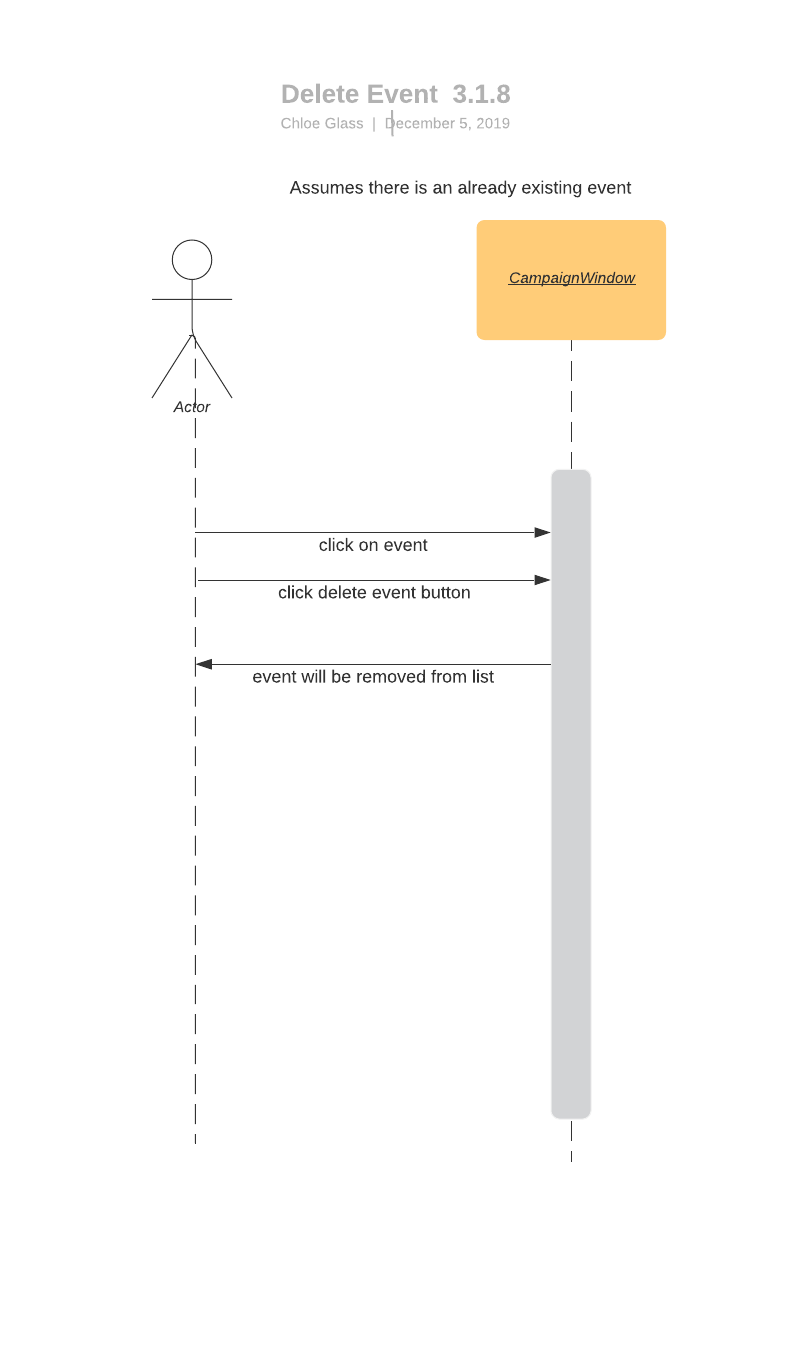
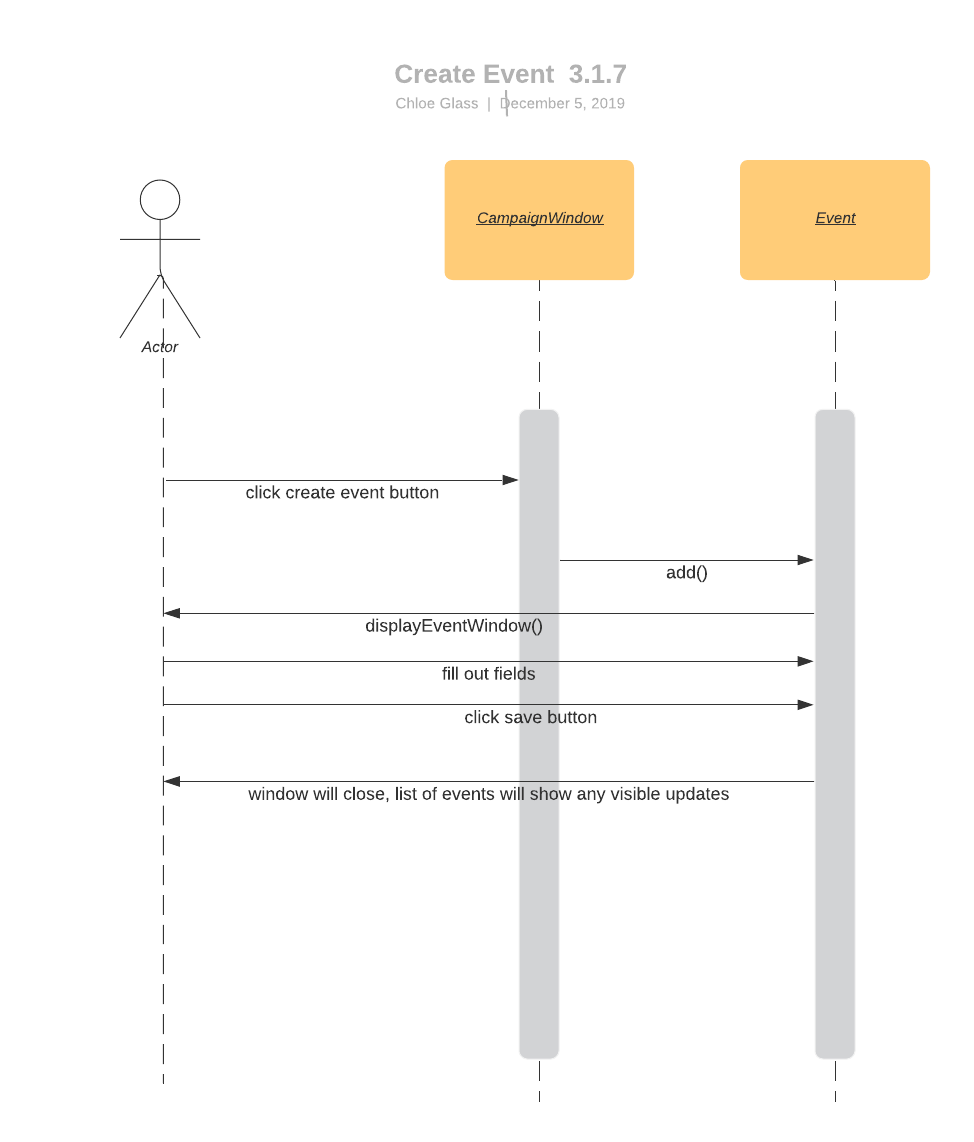


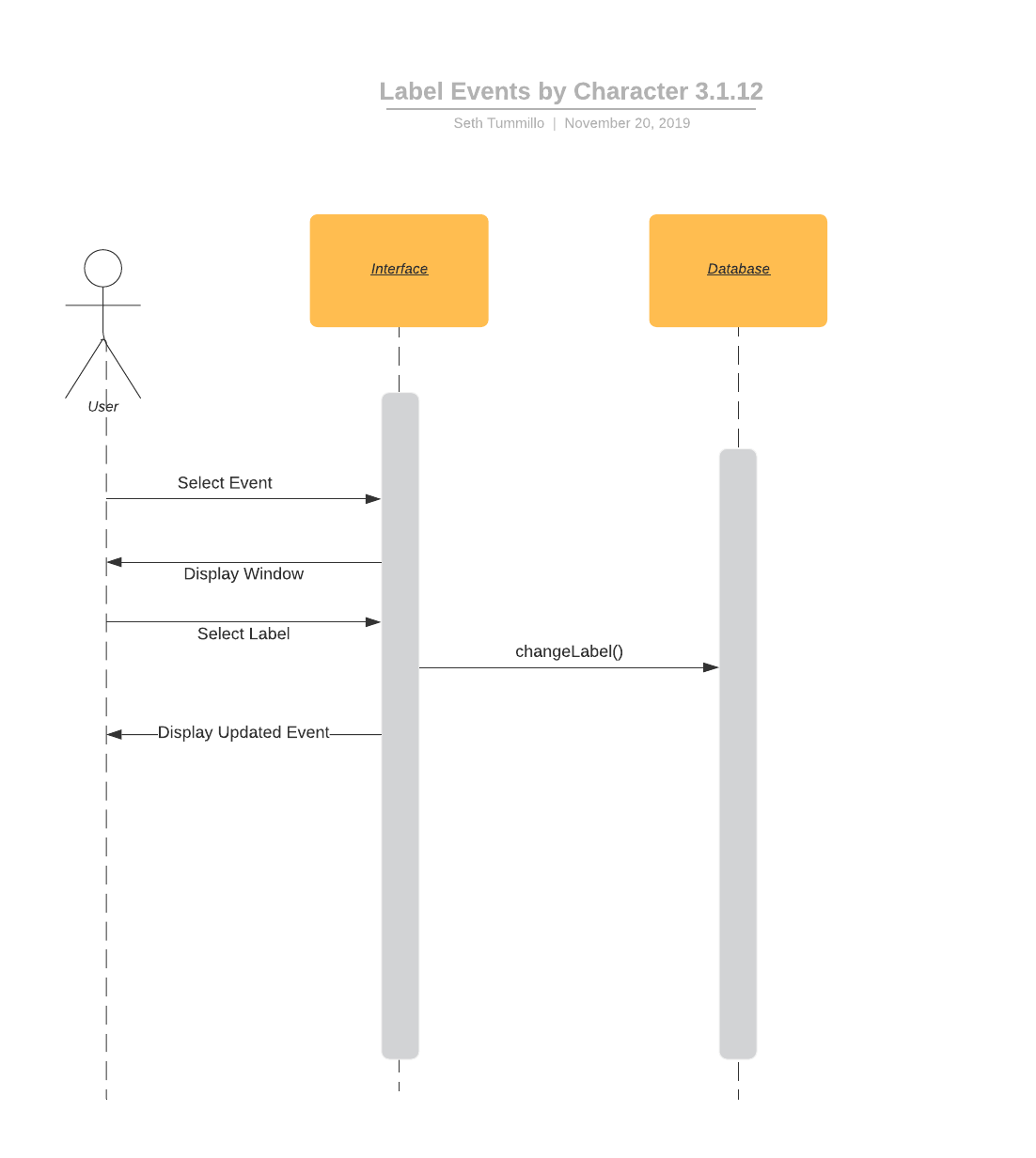


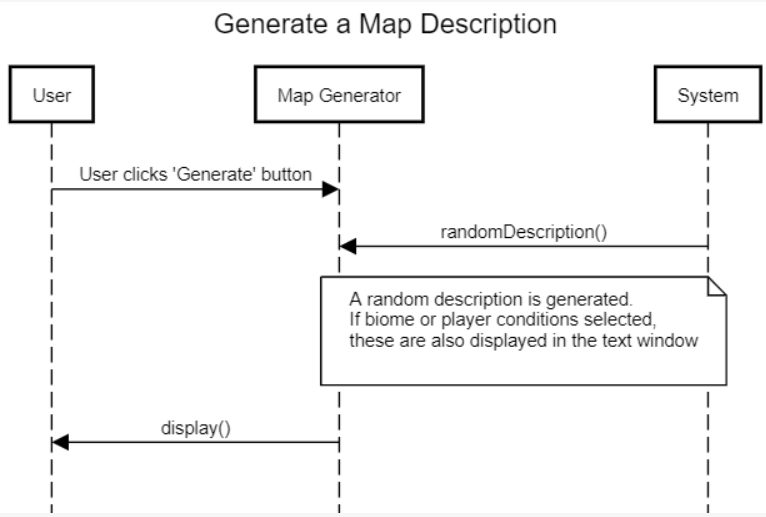
#### 4.2.3.5 Sequence Diagrams

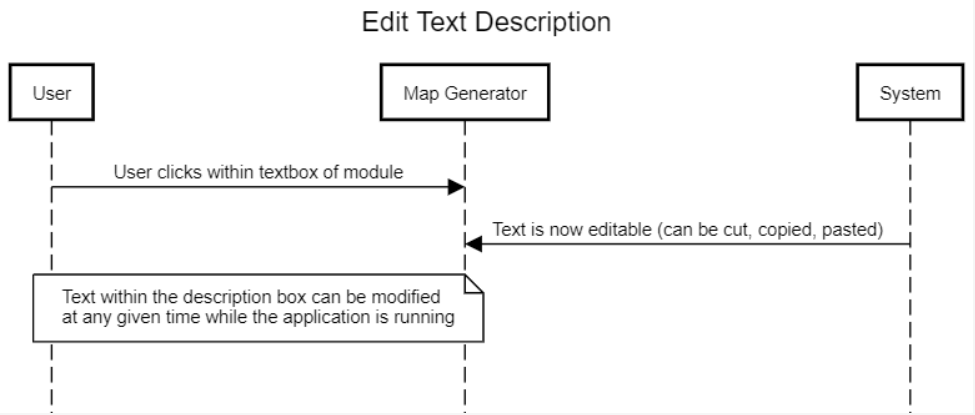


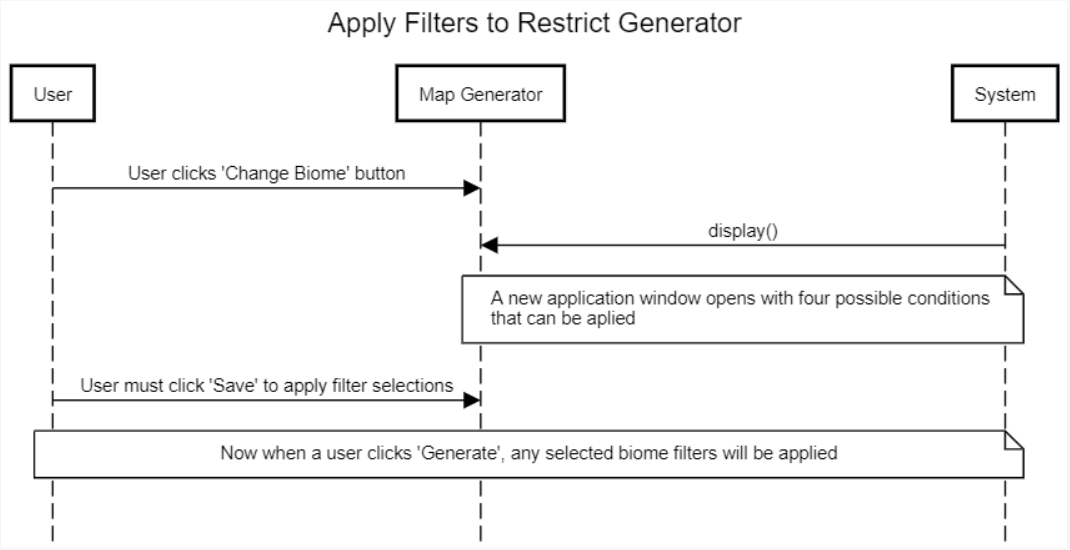


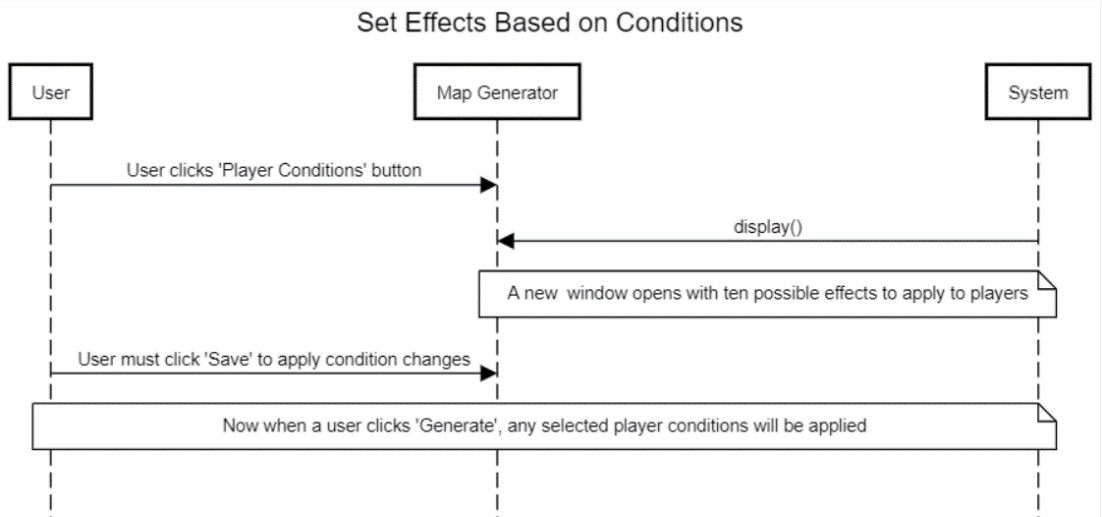


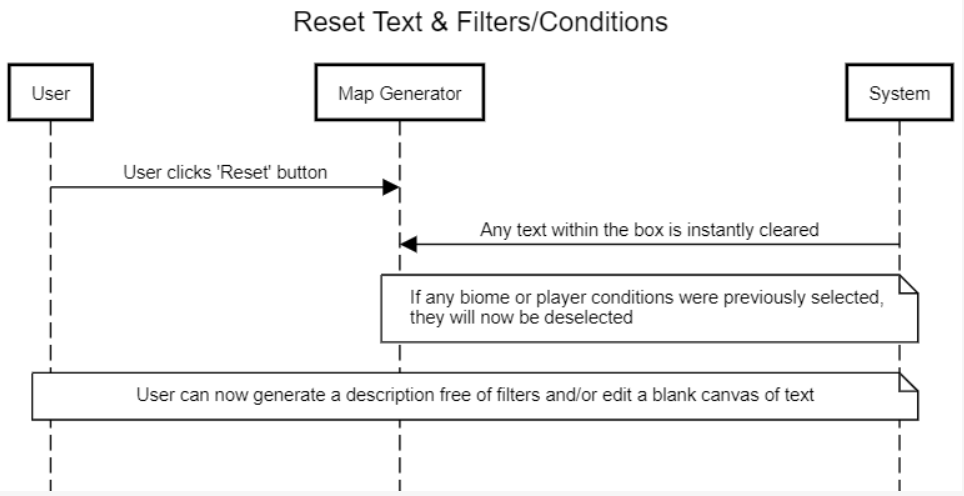


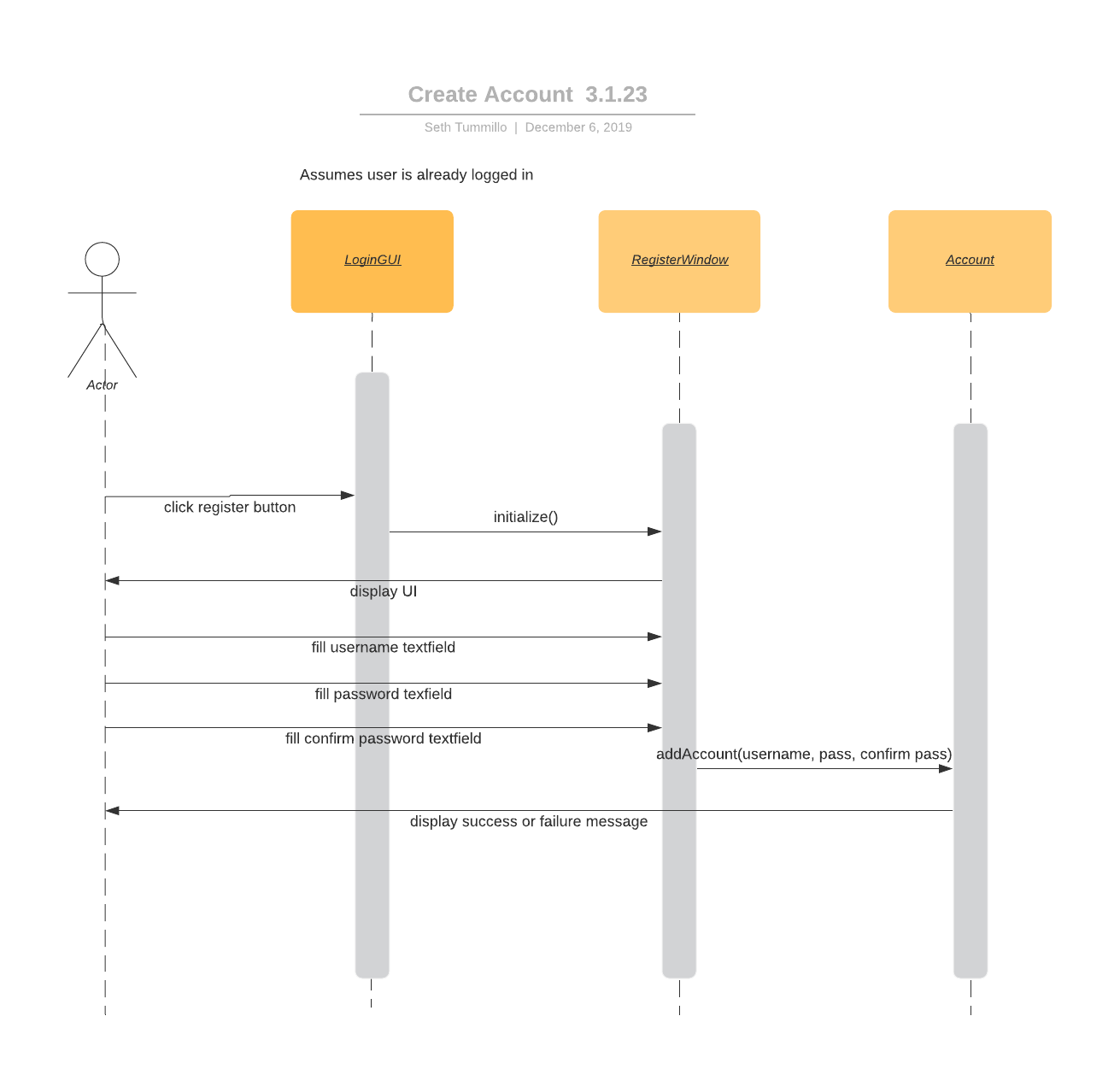


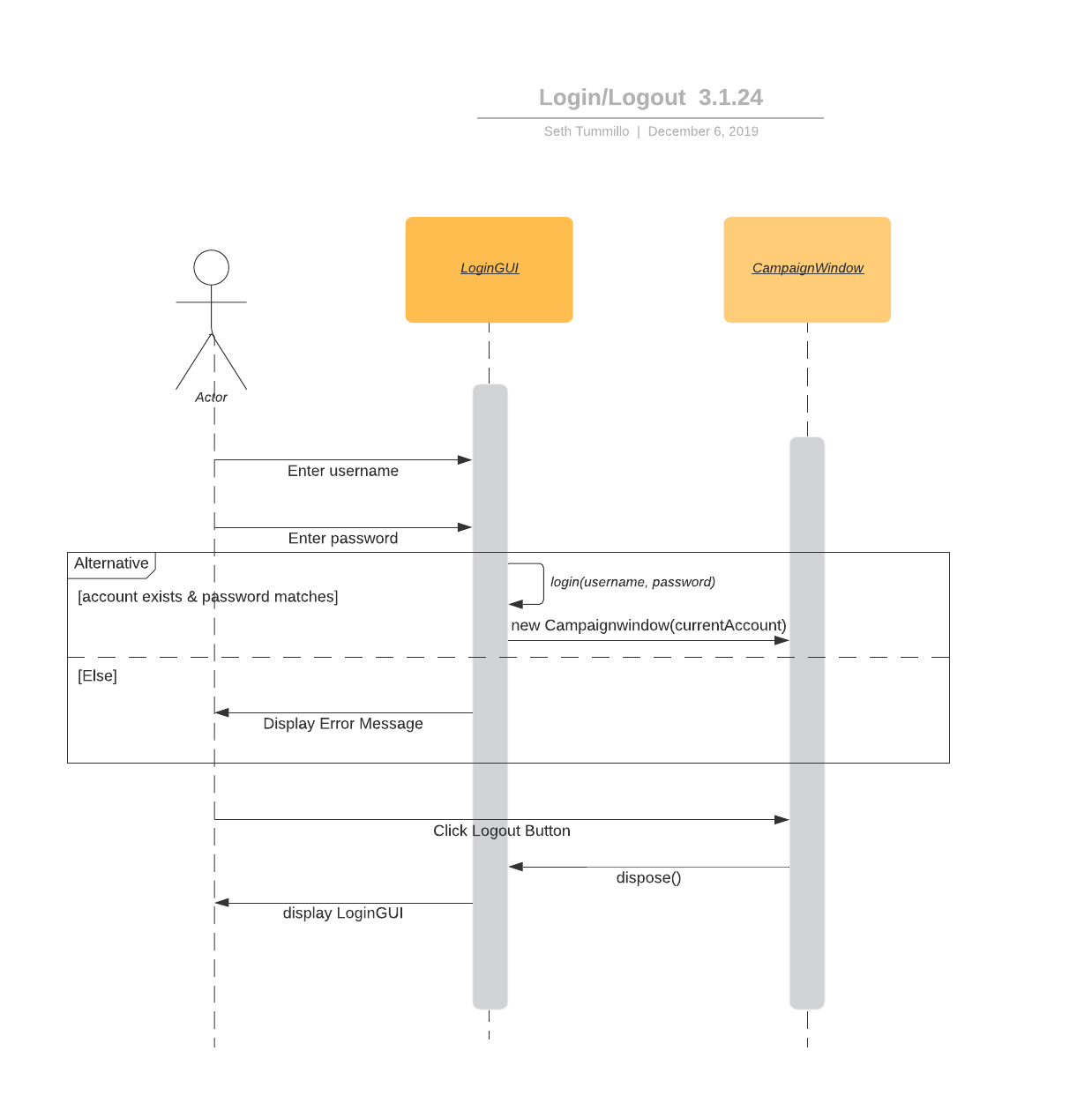


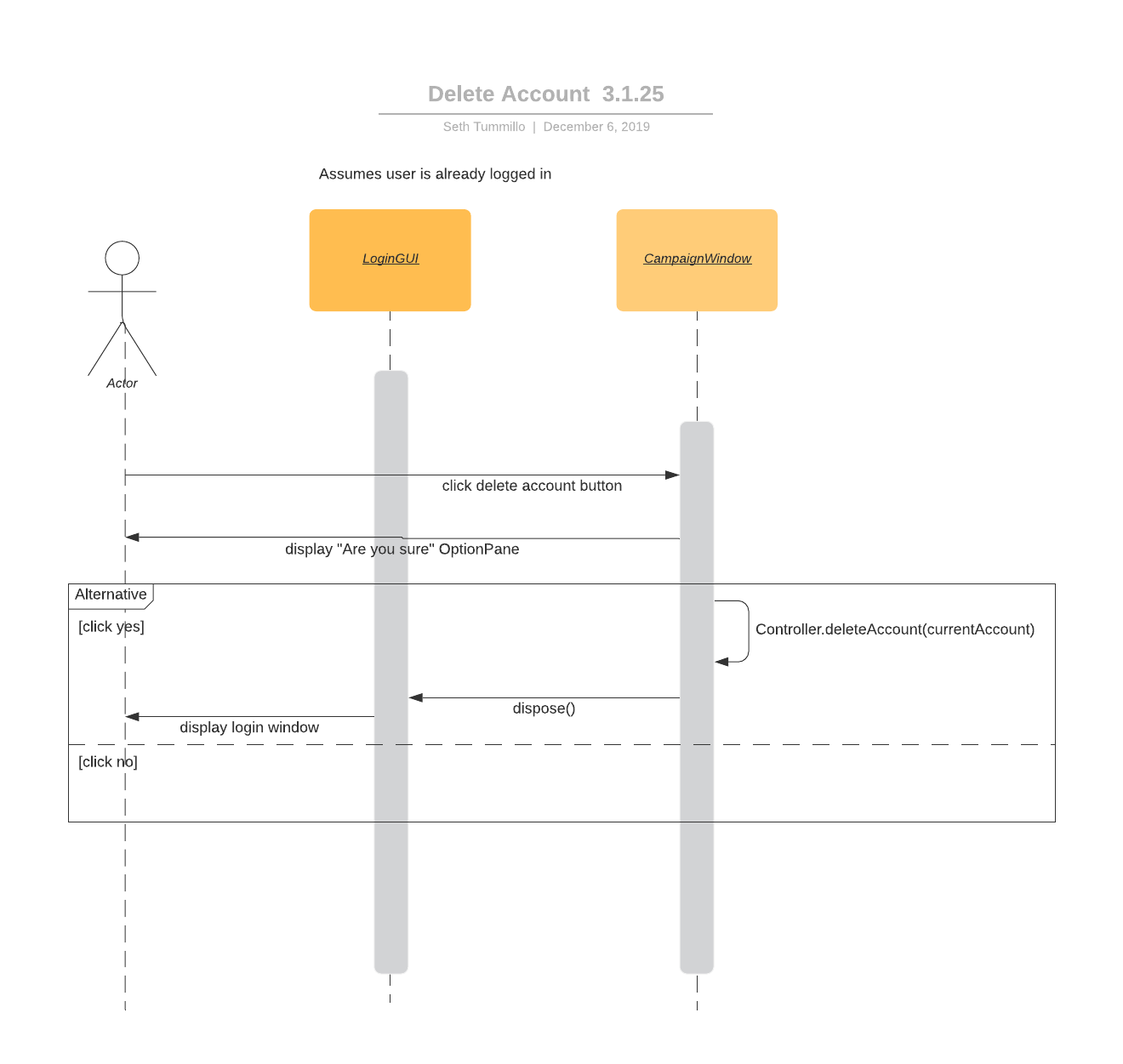


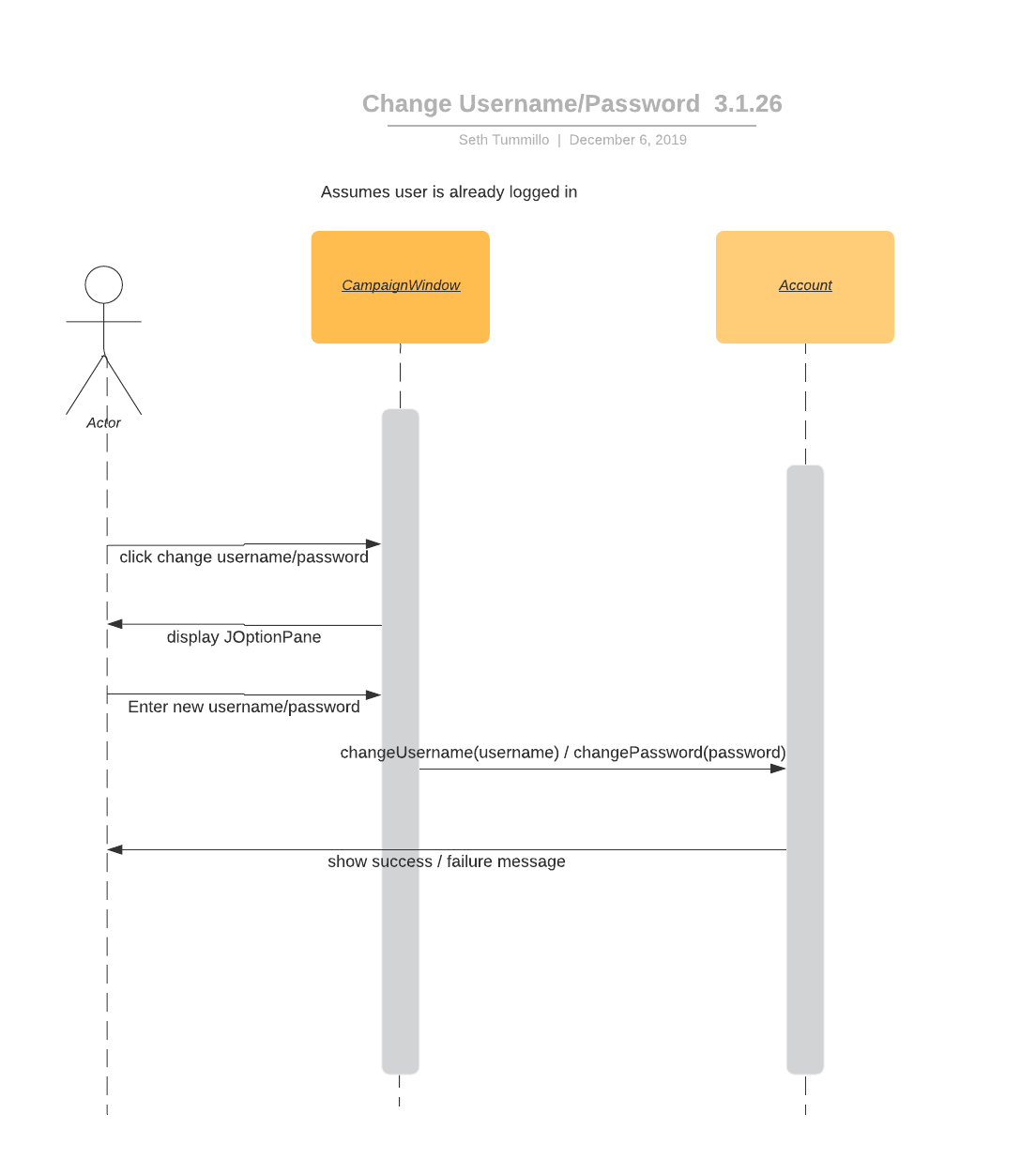


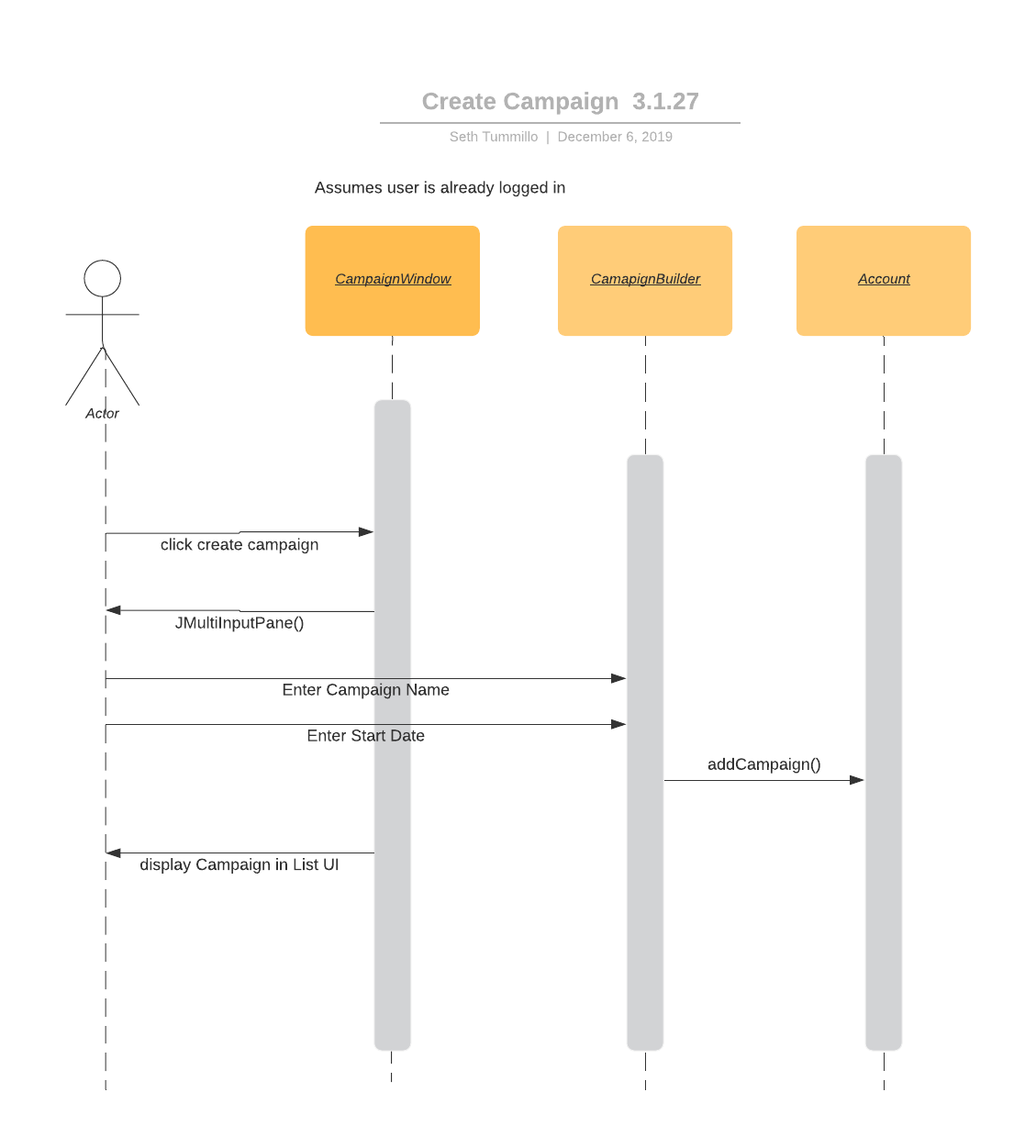


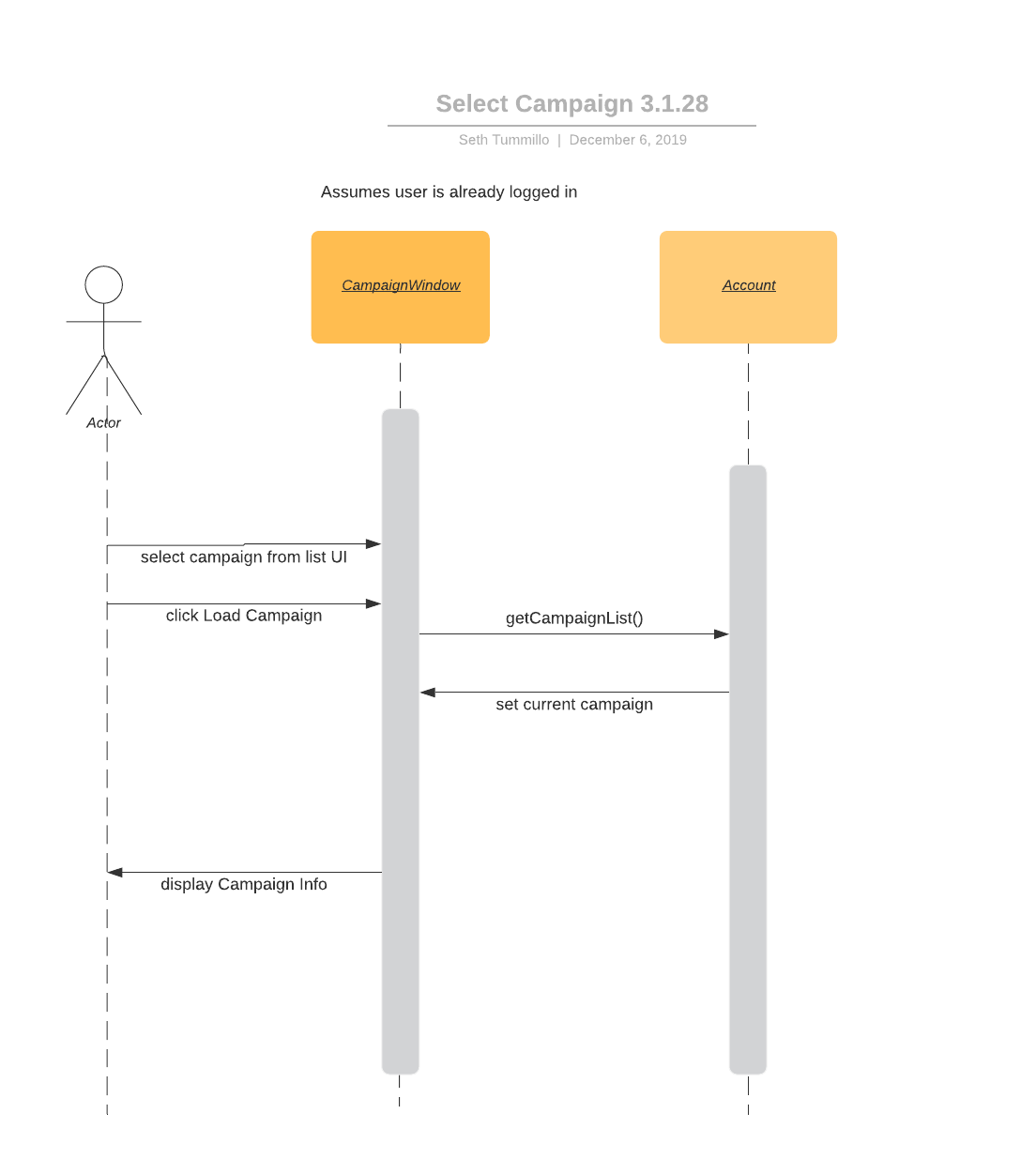


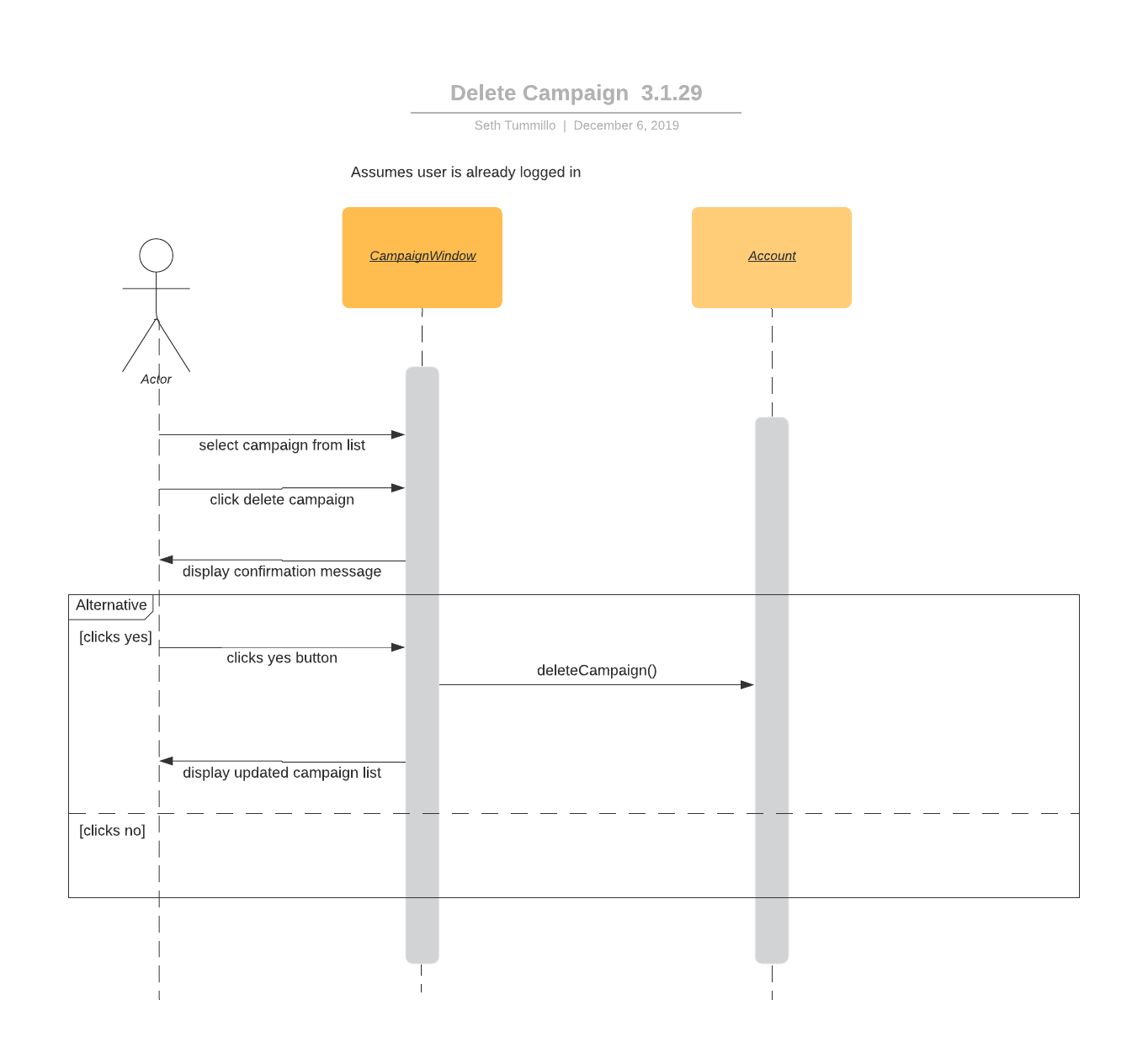


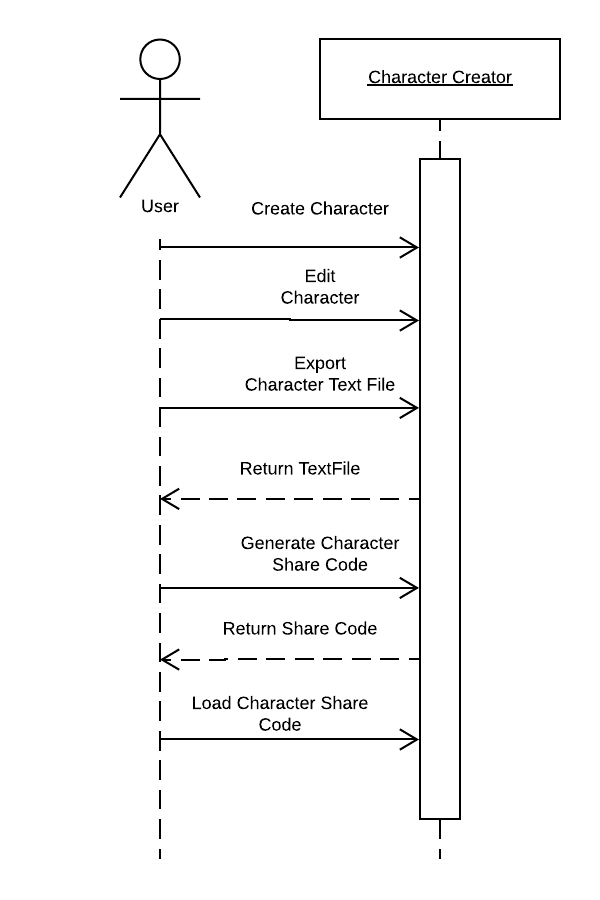












## 4.3 Testing

Our project was tested via smoke-testing since it was highly efficient within the scope of both our project and our experience. Our smoke tests are given in the following format:

Assertion: The declaration of code behavior based on our documentation

Action: The action performed to test the assertion

Outcome: The result of our action, which should match our assertion

4.3.1 Logging in with the wrong password

Assertion: With an existing account, any attempts to login with a different password will not allow logging in

Action: The password “bananabread” is used, when the actual password is “christmas”

Outcome: A popup alerts that the password is incorrect and logging in is prevented

4.3.2 Attempting to create a character with ‘0‘Charisma

Assertion: D&D does not allow ability scores to be less than 3, so any attempts to make a score lower than 3 should not be allowed

Action: Charisma is set to 0 by the user

Outcome: Upon saving or generating the sharecode, Charisma defaults to 3

4.3.3 Attempting to create a character that weighs “one hundred” kilograms

Assertion: Weight is stored as an integer, so any String that does not represent an integer should not be accepted

Action: Weight is set to “one hundred”

Outcome: Upon saving or generating the sharecode, weight defaults to 0

4.3.4 Attempting to load an invalid sharecode in the Encounter Generator

Assertion: Sharecodes should be parsed before they are accepted, in order to ensure their validity

Action: Sharecode is set to “vanilla beans”

Outcome: The Encounter Generator does not update

4.3.5 Generating a random room description

Assertion: User is within the room generating module application of the tool, they have not selected a biome condition or player conditions to apply

Action: User clicks ‘Generate’ button

Outcome: The text field area will display the randomly generated room description consisting of three lines, there are 1000 combinations possible

4.3.6 Edit the text area field of a room description

Assertion: User has the room generator module in focus

Action: User clicks anywhere within the text area field

Outcome: The text (if any is there) can be modified (deleted, copied, cut, etc.)

4.3.7 Apply a biome to the room description generator

Assertion: User has selected one of the four biomes to be applied through the ‘Change Biome’ button

Action: User clicks ‘Generate’ button

Outcome: A randomly generated text of 3 lines is displayed, along with one additional string text containing the biome atmosphere.

4.3.8 Select player conditions to hinder/strengthen players

Assertion: User has selected one to ten of the player conditions through the ‘Player Conditions’ button

Action: User clicks ‘Generate’ button

Outcome: A randomly generated text of 3 lines is displayed, along with an additional string text for each selected player condition selected. A total of 13 lines of text is possible if all conditions are checked.

4.3.9 Reset button clears text and any selected biome/player conditions

Assertion: There is text within the text area (whether it be generated, or typed by the user)

Action: User clicks ‘Reset’ button

Outcome: Any text within the text area field is cleared. Also, any previously selected biome or player conditions will be deselected.

4.3.10 Generating an encounter does not work if the share code isn’t valid.

Assertion: The user has entered a share code that does not meet the length requirements.

Action: User clicks ‘Generate Encounter’ button.

Outcome: No enemies will be added to the encounter, even if parts of the share code would be valid if parsed.

4.3.11 Selecting a difficulty ensures that only encounters that match that difficulty are presented.

Assertion: The user selected a chosen difficulty from the dropdown menu.

Action: User clicks ‘Generate Encounter’ button.

Outcome: Encounter is generated to match the difficulty. If an encounter of the wrong

difficulty is generated, it is discarded and a new encounter is generated.

4.3.12 Add enemy button fills in the highest available space and not the next available space.

Assertion: The user has generated an encounter and removed an enemy not occupying

the lowest space in the Encounter.

Action: User clicks ‘Add Enemy’ to add a new enemy to the encounter.

Outcome: The highest available encounter slot is filled and not the lowest.

4.3.13 Create Event button adds a new event

Assertion: The user has logged in

Action: User clicks ‘Create Event’, then clicks ‘Save’ in the window that pops up

Outcome: The list of events on the main page is updated to include the new event

4.3.14 Delete Event button removes an event

Assertion: The user has logged in and has previously created an event

Action: User clicks on an event, then clicks ‘Delete Event'

Outcome: The list of events on the main page is updated and the selected event is removed

4.3.15 Edit Event button brings up the event window and the event object variables are set accordingly

Assertion: The user has logged in and previously created an event

Action: User clicks on an event, then clicks 'Edit Event'. Once they edit the fields in the window that pops up, they click ‘Save’

Outcome: If the user changed any fields that show up in the event list, the event list will reflect those changes. If the user selects the same event and clicks ‘Edit Event’ again, the window that pops up will have the changes they made

4.3.16 Track day/night and weather is a functioning field

Assertion: The user has logged in and is creating or editing an event

Action: User fills out the field for “In Game Time” or “In Game Weather”

Outcome: The field does not clear out as soon as the user clicks away or if the user saves the event and re-opens it later

4.3.17 The session label is shown in the list of session events

Assertion: The user has logged in and has created or edited an event

Action: User fills out the field “Session Label” and saves the event

Outcome: The list of events on the main page is updated and the event in the list of events will show a label in between the session number and the session date

In addition to this we did add a few test cases where they were possible.

