Alaa Malabeh
IN4MATX 122
January 15, 2026

Assignment 1-1: UML Design I

Draft of UML Classes/Attributes:

```
class User {
  - userId: int
  - username: String
  + createCampaign(): Campaign
  + removeCampaign(): void
  + renameCampaign(): void
  + archiveCampaign(): void
  + setCampaignVisibility(): void
  + addCharacter(: void
  + shareCampaign(): CampaignShare
  + shareEvent(): EventShare
}

class Campaign {
  - campaignId: int
  - name: String
  - archived: boolean
  - visibility: Visibility
  + addEvent(): void
  + removeEvent(): void
  + updateEvent(): void
  + filterEvents(): List<QuestEvent>
  + rename(): void
  + archive(): void
  + unarchive(): void
}

class QuestEvent {
  - eventId: int
  - title: String
  - startTime: GlobalTime
  - endTime: GlobalTime
  + updateTitle(): void
  + updateStartTime(): void
  + updateEndTime(): void
```

Alaa Malabeh
IN4MATX 122
January 15, 2026

```
  + clearEndTime(): void
  + changeRealm(): void
  + addParticipant(): void
  + removeParticipant(): void
  + addItem(): void
  + removeItem(): void
}

class WorldClock {
  - currentTime: GlobalTime
  + now(): GlobalTime
  + advance(): void
  + resetTime(): void
}

class GlobalTime {
  - totalMinutes: int
  + compareTo(): int
  + plus(): GlobalTime
  + minus(): Duration
  + toDays(): int
  + toHours(): int
  + toMinutes(): int
  + toString(): String
}

class Duration {
  - days: int
  - hours: int
  - minutes: int
  + toMinutes(): int
  + fromMinutes(): Duration
}

class Realm {
  - realmId: int
  - name: String
  - mapId: String
  - coordinates: String
```

Alaa Malabeh
IN4MATX 122
January 15, 2026

```
  - description: String
  + convertToLocalTime(): LocalTime
  + convertToWorldTime(): GlobalTime
}

class LocalTime {
  - days: int
  - hours: int
  - minutes: int
  + toString(): String
}

class Settings {
  - currentRealm: Realm
  - theme: Theme
  - timeDisplayPreference: TimeDisplayPreference
  + updateCurrentRealm(): void
  + updateTheme(): void
  + updateTimeDisplayPreference(): void
}

class TimelineView {
  - campaign: List<Campaign>
  + eventsDay(): List<QuestEvent>
  + eventsWeek(): List<QuestEvent>
  + eventsMonth(): List<QuestEvent>
  + eventsYear(): List<QuestEvent>
}

class Character {
  - characterId: int
  - name: String
  - className: String
  - level: int
  + updateName(): void
  + updateClassName(): void
  + updateLevel(): void
  + levelUp(): void
}
```

Alaa Malabeh
IN4MATX 122
January 15, 2026

```
class ItemEntry {
  - entryID: int
  - quantity: int
  - item: Item
  + addQuantity(): void
  + minusQuantity(): void
}

class Item {
  - itemId: int
  - name: String
  - type: String
  - rarity: String
  - description: String
}

class CampaignShare {
  - shareId: int
  - permission: Permissions
  - createdAt: GlobalTime
  + canView(): boolean
  + canEdit(): boolean
}

class EventShare {
  - shareId: int
  - permission: Permissions
  - createdAt: GlobalTime
  + canView(): boolean
  + canEdit(): boolean
}


Visibility- enum {
  Public
  Private
}
```

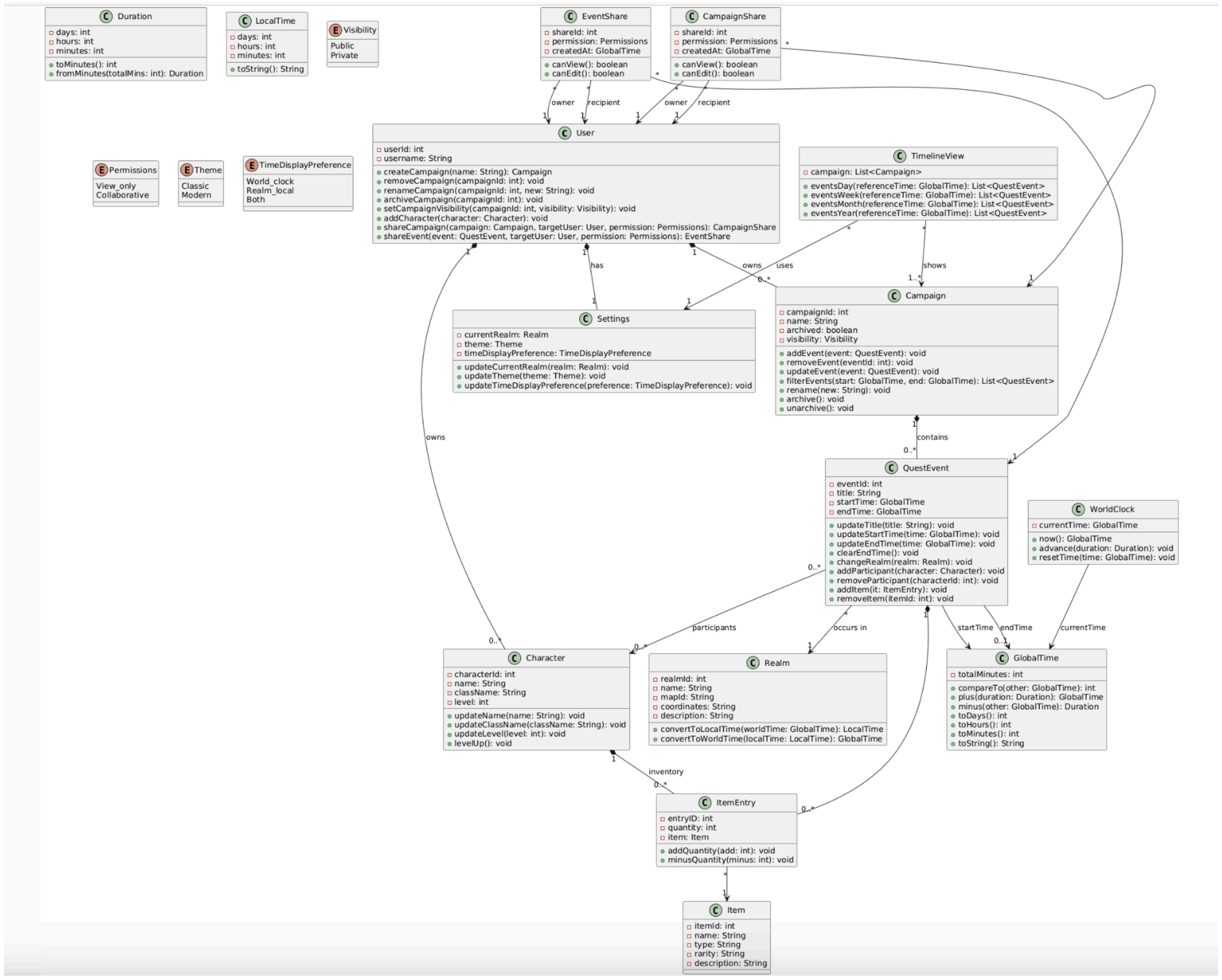
Alaa Malabeh
IN4MATX 122
January 15, 2026

```
Permissions- enum {
  View_only
  Collaborative
}

Theme- enum {
  Classic
  Modern
}

TimeDisplayPreference- enum {
  World_clock
  Realm_local
  Both
}
```

Alaa Malabeh
IN4MATX 122
January 15, 2026

UML Diagram:

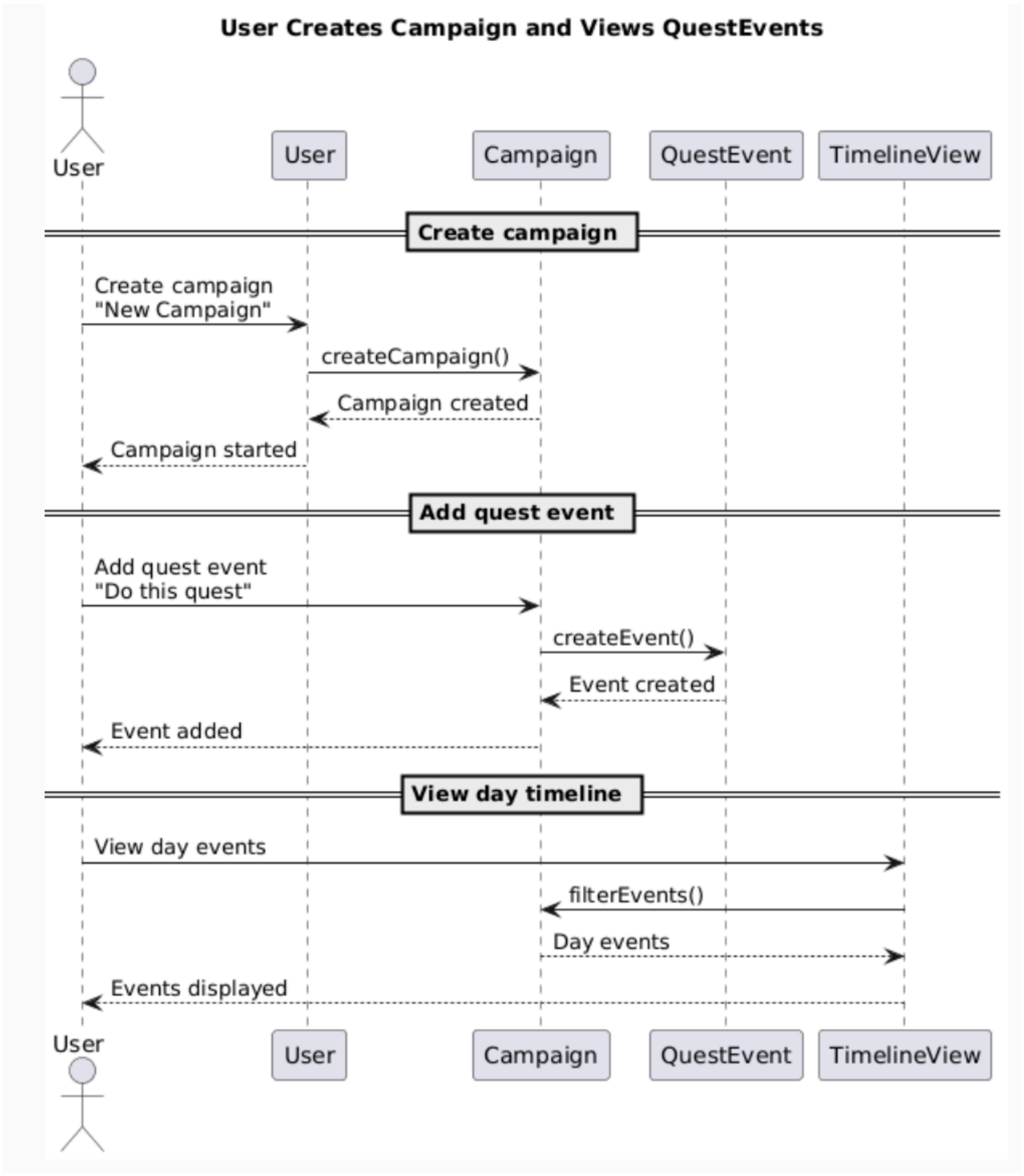
Link to PlantUML: https://drive.google.com/file/d/16jNIYj5GaekuTOfToY1L4akHoUAXFXHd/view?usp=drive_link


Duration
days: int
hours: int
minutes: int
toMinutes(): int
fromMinutes(totalMins: int): Duration
LocalTime
days: int
hours: int
minutes: int
toString(): String
Visibility
Public
Private
EventShare
shareId: int
permission: Permissions
createdAt: GlobalTime
canView(): boolean
canEdit(): boolean
CampaignShare
shareId: int
permission: Permissions
createdAt: GlobalTime
canView(): boolean
canEdit(): boolean
owner
recipient
owner
recipient
Permissions
View_only
Collaborative
Theme
Classic
Modern
TimeDisplayPreference
World_clock
Realm_local
Both
User
userId: int
username: String
createCampaign(name: String): Campaign
removeCampaign(campaignId: int): void
renameCampaign(campaignId: int, new: String): void
archiveCampaign(campaignId: int): void
setCampaignVisibility(campaignId: int, visibility: Visibility): void
addCharacter(character: Character): void
shareCampaign(campaign: Campaign, targetUser: User, permission: Permissions): CampaignShare
shareEvent(event: QuestEvent, targetUser: User, permission: Permissions): EventShare
TimelineView
campaign: List<Campaign>
eventsDay(referenceTime: GlobalTime): List<QuestEvent>
eventsWeek(referenceTime: GlobalTime): List<QuestEvent>
eventsMonth(referenceTime: GlobalTime): List<QuestEvent>
eventsYear(referenceTime: GlobalTime): List<QuestEvent>
has
owns
uses
shows
Settings
currentRealm: Realm
theme: Theme
timeDisplayPreference: TimeDisplayPreference
updateCurrentRealm(realm: Realm): void
updateTheme(theme: Theme): void
updateTimeDisplayPreference(preference: TimeDisplayPreference): void
Campaign
campaignId: int
name: String
archived: boolean
visibility: Visibility
addEvent(event: QuestEvent): void
removeEvent(eventId: int): void
updateEvent(event: QuestEvent): void
filterEvents(start: GlobalTime, end: GlobalTime): List<QuestEvent>
rename(new: String): void
archive(): void
unarchive(): void
owns
contains
QuestEvent
eventId: int
title: String
startTime: GlobalTime
endTime: GlobalTime
updateTitle(title: String): void
updateStartTime(time: GlobalTime): void
updateEndTime(time: GlobalTime): void
clearEndTime(): void
changeRealm(realm: Realm): void
addParticipant(character: Character): void
removeParticipant(characterId: int): void
addItem(it: ItemEntry): void
removeItem(itemId: int): void
WorldClock
currentTime: GlobalTime
now(): GlobalTime
advance(duration: Duration): void
resetTime(time: GlobalTime): void
participants
occurs in
startTime
endTime
currentTime
Character
characterId: int
name: String
className: String
level: int
updateName(name: String): void
updateClassName(className: String): void
updateLevel(level: int): void
levelUp(): void
Realm
realmId: int
name: String
mapId: String
coordinates: String
description: String
convertToLocalTime(worldTime: GlobalTime): LocalTime
convertToWorldTime(localTime: LocalTime): GlobalTime
GlobalTime
totalMinutes: int
compareTo(other: GlobalTime): int
plus(duration: Duration): GlobalTime
minus(other: GlobalTime): Duration
toDays(): int
toHours(): int
toMinutes(): int
toString(): String
inventory
ItemEntry
entryID: int
quantity: int
item: Item
addQuantity(add: int): void
minusQuantity(minus: int): void
Item
itemId: int
name: String
type: String
rarity: String
description: String

Alaa Malabeh
IN4MATX 122
January 15, 2026

Sequence Diagram:

Use case: User can create a campaign and view the quest events
Link to sequence diagram (via PlantUML):
https://drive.google.com/file/d/1r3zilTQauXHGjAzSvTVrlgw4drLrHWC1/view?usp=sharing



Alaa Malabeh
IN4MATX 122
January 15, 2026

GuildQuest Documentation

System Overview

The system being designed in this document is a game called GuildQuest, which is a textbased RPG campaign manager to help users with organizing their campaigns, quest events, inventories, and real world times of the campaigns they are currently running. This specific system is designed around a world clock which accurately records the time in the campaign which allows events to be stored, ordered, and shown to the users, depending on their system preferences.

Core Classes/Responsibilities

The class User represents the actual individual that is using the system. The user will own the campaigns, manage the characters, configure settings, and control which other users they are sharing with. The methods that were added to this class give the ability to support the creation and managing of campaigns, adding characters, and sharing campaigns/quests with other users, depending on which permission levels they are given.

The class Campaign represents one RPG campaign that is owned by a user. These campaigns can have quest events, the visibility settings to choose whether it is public or private, and whether or not this campaign can be archived. This class supports the TimelineView, as they can gather the events needed for that campaign's time frame.

The class QuestEvent is an individual event that happens, which has a title, a start time, and optional end time, and is always paired with a Realm. We use the other class globalTime to store start and end times. These events also refer to other characters who have joined the quest and include any items that are added/removed during a quest.

WorldClock and globalTime are two separate classes to represent time in the RPG. WorldClock tracks the current global time and can go ahead in time, but globalTime is an immutable class, so that players can store and compare events.

The class Realm represents a physical place in the GuildQuest game, which is associated with a map identity and a localTimeRule that shows how the current world time can be converted to the realm local time. The local realm time is only shown when displaying events.

The class Settings will store the user's preferences such as which realm they are in, UI theme, and their time display preference.

Alaa Malabeh
IN4MATX 122
January 15, 2026

The class TimelineView shows the campaign events that happen in a day, week, month, or year depending on the tune. TimelineView will then gather any data that is in the system that relates to that certain event depending on the time.

The classes Character, Inventory, Item, and ItemEntry are all entities that are in charge of making sure that there is an inventory of items, and that they can increase and decrease depending on how the player goes through the quests. ItemEntry is used to show the inventory contents and any item changes depending on the event.

CampaignShare and EventShare show the ability that the game has for collaboration. These classes show the relationships that are possible in the game, including the owner, the target user, and permission levels, which allows the game to have both campaign and event sharing.

Support for At Least Three Future Changes

1) Additional realm mechanics (such as travel time or realm-specific calendars) can be added by extending the LocalTimeRule interface or adding new realm related classes without changing the existing event or campaign logic.
2) Networking/distribution and real time collaboration can be implemented by synchronizing the User, Campaign, and Share classes across devices or through a client-server architecture, as sharing and permissions are already clearly modeled.
3) Undo actions can be implemented by logging changes to campaigns, events, and inventories, since all modifications are updated in already defined methods and objects like GlobalTime and ItemEntry.
4) Saving, importing, and exporting campaigns can easily be implemented in the future because the core classes such as Campaign, QuestEvent, Character, and Item are all defined and already self contained. As a result, they can easily be imported/exported to a different structured format like JSON without many changes to their logic.

Alaa Malabeh
IN4MATX 122
January 15, 2026

Appendix

Link to ZotGPT chat:
https://drive.google.com/file/d/1gLywvyUoIdaURWxNYrg2RB4_PhpUp_76/view?usp=sharing

For this assignment, I used generative AI (ZotGPT) to help review what I had been writing to see if I was going on the right track as I completed the assignment. I used it throughout my design process to check for any redundancy or repetitiveness in my UML classes and to ensure that I was using object oriented design principles such as when to use a class versus an enum or an interface. I used it to get feedback and then made my own decisions based on its responses and the assignment requirements. Most of the prompts I asked were focused on improving my existing work instead of asking for a complete design. For example, I asked whether some of my UML classes were repetitive, or if certain classes or parts were unnecessary for the scope of the assignment. I also asked questions like whether something should be modeled as a class or an enumeration, and if some parts of the diagram were over designed for a first version of the system, as the instructions said it was just a first version of the system.

The AI responses in this case pointed out duplicate classes that I had, and suggested that I consolidated classes, and also suggested that I should use enums for some of the design choices, such as Visibility, Theme, and Permissions classes since they usually have fixed values and don't change. However, there were also times where I didn't take the suggestions of the AI. For one prompt, ZotGPT told me to consolidate my CampaignShare and EventShare classes since they had the same method and attributes, but I chose not to take this suggestion. Although they did have the same methods, Campaigns and Events can be shared separately, so I decided that it wasn't the best design choice to follow ZotGPT's suggestions. Additionally, I chose to keep TimelineView since the instructions explicitly required multiple time views for the day, week, month, and year.

Overall, the final UML diagram design reflects my own understanding of object oriented design, and I used generative AI as a reference to see if I was going about my ideas correctly instead of copying and pasting solutions. ZotGPT did help me think about the trade offs of certain design choices, but the final structure was from my own personal reasoning.