Task 3 – Creating a Class diagram and design pattern selection (30 marks)

Create a simple Class diagram which should consists of the Classes that might be used to represent the system and the association between them. You don't have to declare the attributes and operations for this activity. You do have to explain the class responsibility of each class declared. You can use software like StarUML to complete this activity.

Output – A class diagram containing classes and associations. In Word format, uploaded to GitHub.

Consider the problem and select a suitable design pattern that can be implemented on the problem. Give justification on why the design pattern was chosen. Draw the UML diagram representing your class diagram as a design pattern UML. Include all the abstract class/interface, concrete class and inheritance (if any) used to represent the problem.

Output – UML diagram representing the design pattern. In Word format, uploaded to GitHub.

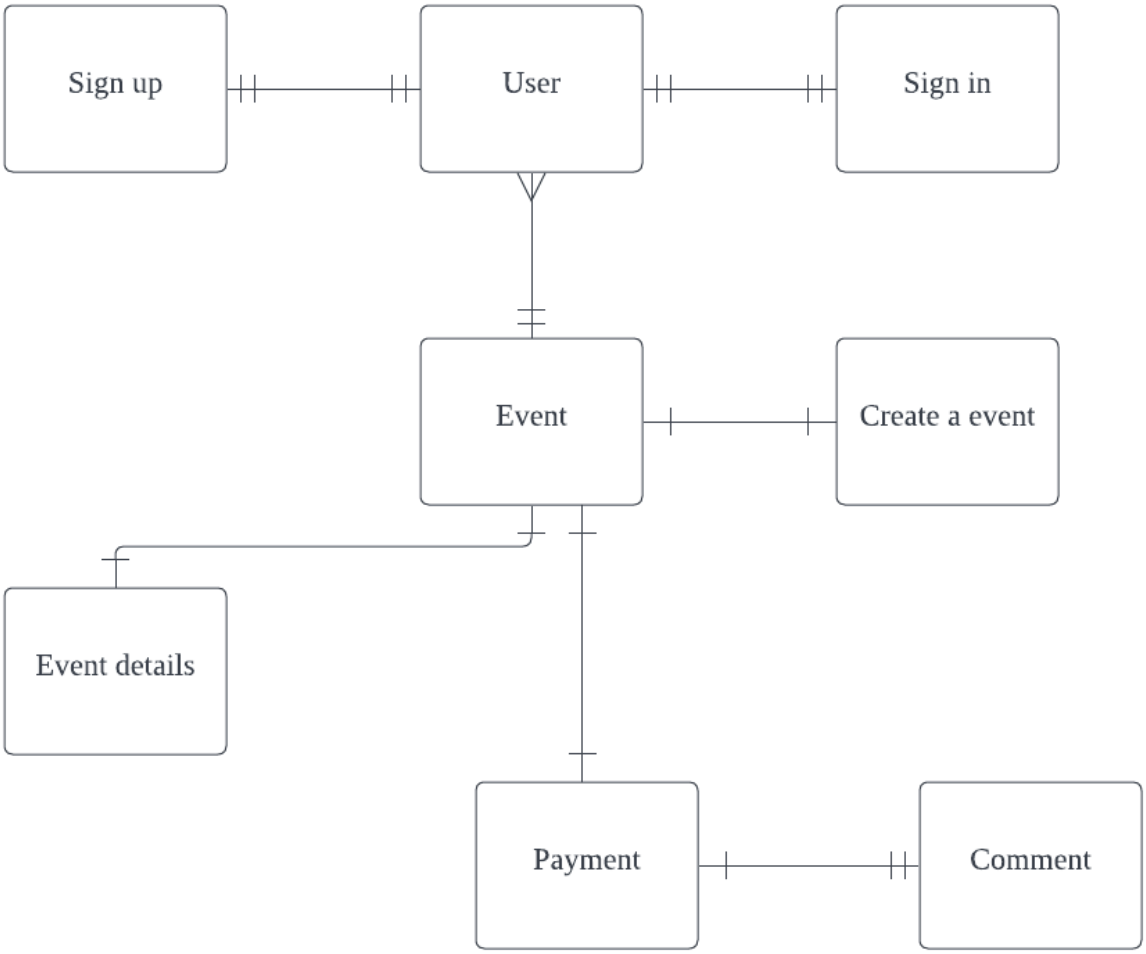

Figure 1 : Class diagram of the system

Design pattern

For this project, I have chosen observer design pattern. This one-to-many relationship between the subject and all of the observers waiting for data so they can be updated is the aim of the observer design pattern. Therefore, all of the observers will be informed and updated promptly whenever the subject's state changes.

Some instances where I use this pattern include handling user account, updating, and delivering user notifications.

Consider a single page application with three feature dropdown lists that depend on the choice of a category from a higher level dropdown. This is typical on numerous retail websites, including Home Depot. On the page, a number of filters that depend on the value of a top-level filter are included.

The problem we faced now is trouble with notification spamming. Consider that you enjoy viewing Netflix episodes or videos on YouTube. On YouTube, there might be one channel that attracted your eye. Or perhaps you can't wait to watch the upcoming season of your favourite TV show. What would you do to watch the upcoming movie or TV season?

Opening the app or website repeatedly to see if fresh information has been posted is an easy fix. But the user would find this difficult. The consumer would waste a lot of time repeatedly visiting the website for updates.

To get around this, the website might notify every other user through email each time new content is posted. On the website, however, thousands of fresh movies or shows are posted every day. Additionally, a lot of consumers might not be interested in being notified about every other performance. This would eventually cause mayhem by flooding the user's inbox with spam.

Sending a notification to only interested clients will fix the aforementioned issue. The "Remind me" feature on Netflix is available. You will receive an email from Netflix as soon as new content is added. You can take the reminder out if you decide otherwise. On YouTube, you can subscribe to a channel by clicking the Bell icon. The subscribers are all notified when a new video is added. So, referring to this, we can also setup a ‘Im interested’ for our event notification for user to overcome this issue.

The Observer design pattern is based on this. Fundamentally, we employ an observer whenever a list of entities needs to be informed of a state change. The entities may want to subscribe to particular state changes or unsubscribe from them. In the parts that follow, we'll delve into this pattern in great detail.