Architecture 2:

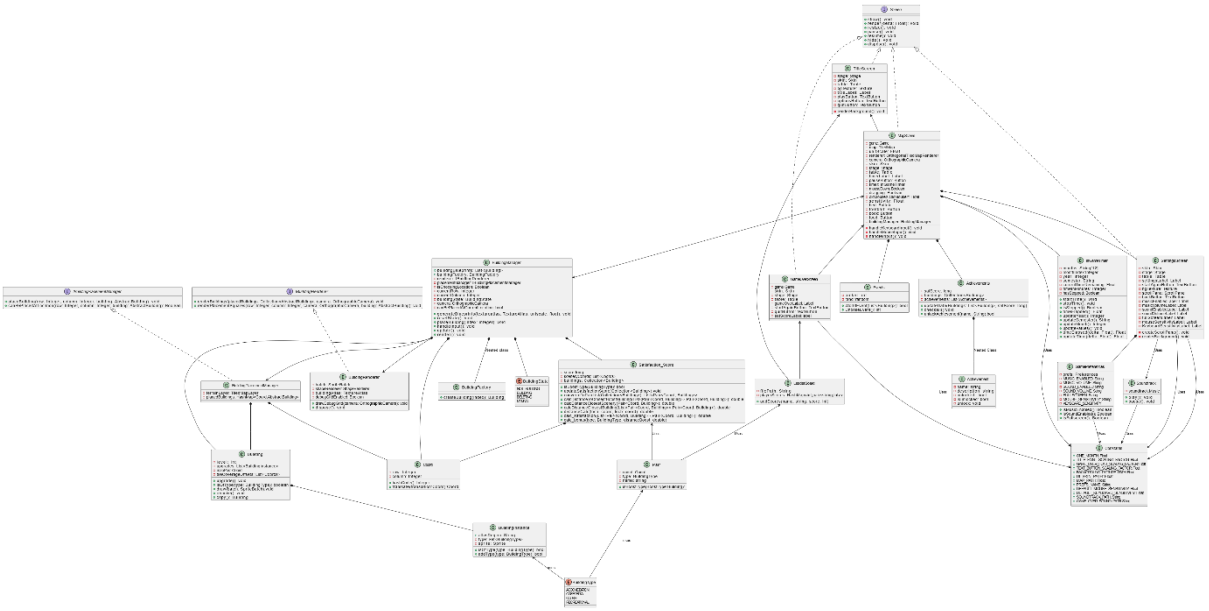

Figure 1: Overview UML Class diagram

Here we have our new class diagram adapted from the previous developers finalised class diagram, we opted to remove their abstract classes and create our own system alongside removing the JSON dependency in order to reduce complexity and to insert our own “Composite building” aspect of the game. Below you can see the new classes we have implemented for the replacement of the abstract classes and JSON dependency. This can be seen as we can now easily track what classes have associations with one another or specifically through our new system it is very clear what features in our game will have visual/front end effects on gameplay and what class will just work in the background. Here we can see that the “BuildingManager” Class has new functionality and is how the “SatisfactionScore” and “Achievements” classes are used, allowing quick tests and checks. Alongside being in line with the product brief and requirements “UR_ACHIEVEMENTS” , “FR_SATISFACTION_UPDATE” and “UR_STUDENT_SATISFACTION”.

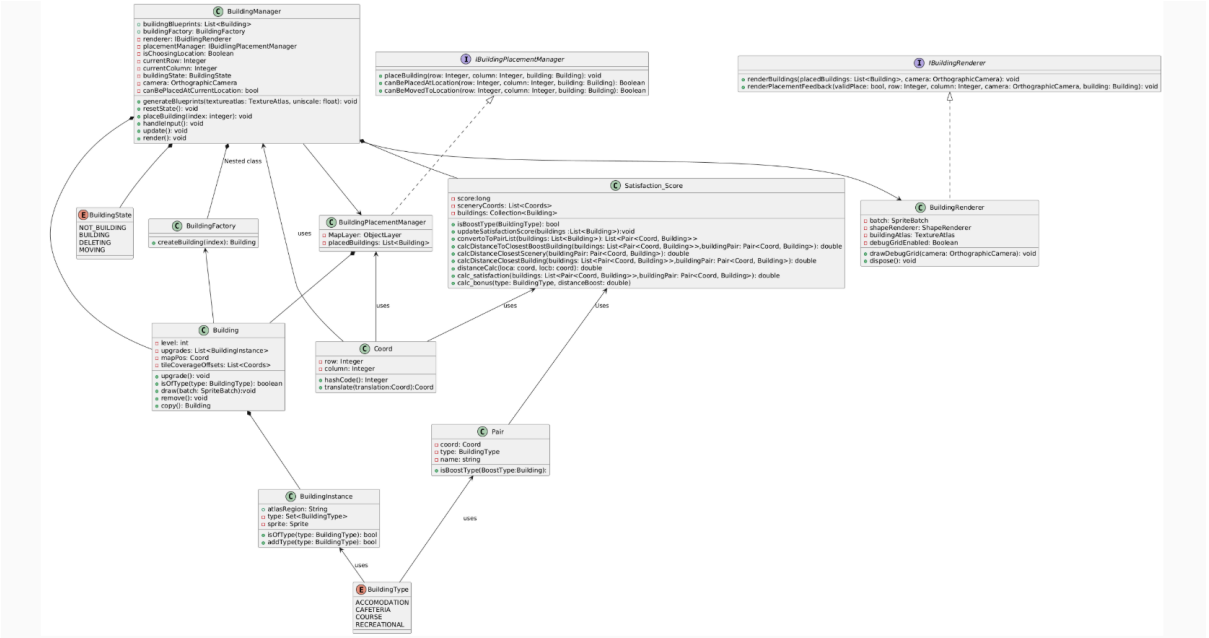

Figure 2: Planned implementation for satisfaction score feature seeing how it interlinks with the removal of the Json dependency and building manager refactorization.

Here we can see a better look of how the features are planned to be implemented. We can see that they are ran through the BuildingManager class. This makes partially them compliant with the requirements “FR_SATISFACTION_UPDATE”. We chose to run the instances through BuildingManager as it reduced time between checks happening and time between events/updates in planning since we were no longer running the checks every frame but every time a building was manipulated. Below you will see that we removed the JSON dependency and what we replaced it with in better detail

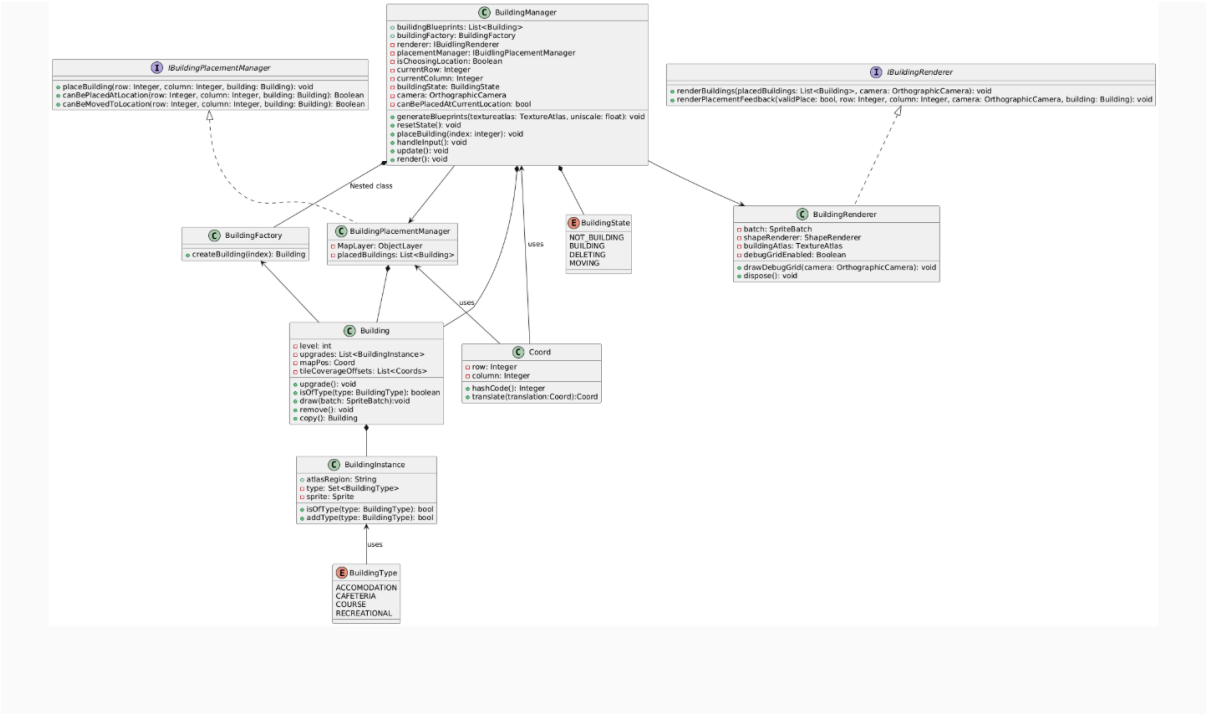

Figure 3: The new classes created to replace the abstract class and remove redundant classes (First Strand of development)

We have stripped the program of reduntant classes such a class for each building type, these have been replaced by the enum “BuilidingType” you can see in the above diagram. This lead to both easier traceability and for us to establish the groundwork for requirements “FR_RECREATON_MULTIPLICITY” and “UR_LOCATION_UPGRADE” to be completed.

History of these charts can be found on our website alongside how we planned the Events, Satisfaction_Score and Achievements class.

Reasoning and Justification for “UPGRADE BUILDING” functionality;

- In line with the Briefs request of different building types, field research of actual university campuses and requirements “UR_LOCATION_UPGRADE” we decided that this function would add more realism to the game. To feel more like a real campus
- Originating from the client meeting where the client showed an interest in having the ability to upgrade the buildings

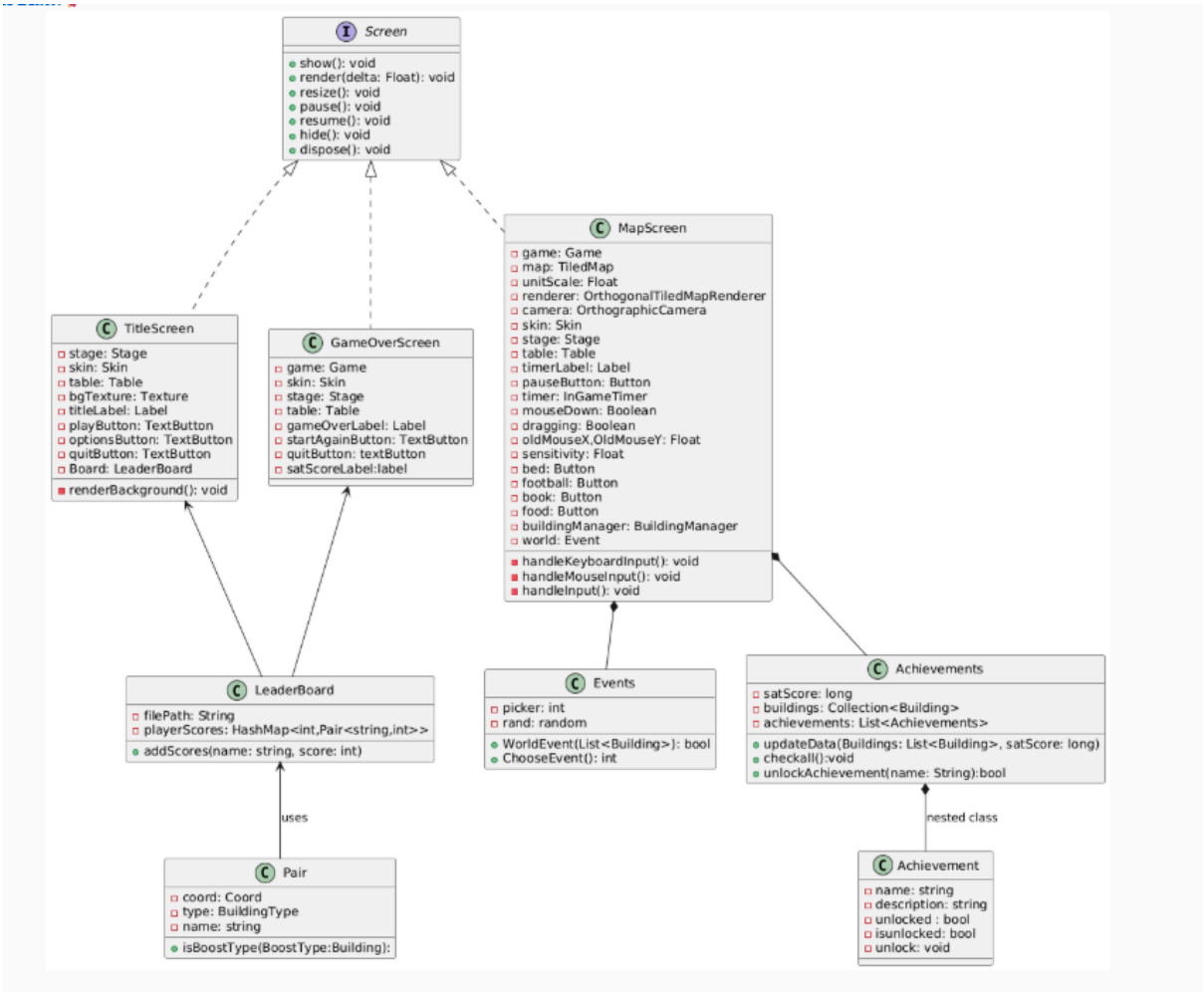

Figure 4: The graph depicting where the new predominantly visual elements fit In the game (second strand of development)

Above we can see that graph depicting how the predominantly visual elements are meant to work within the game. The Leaderboard and events living within appropriate screens for where they are accessible. This fulfils the requirements “UR_EVENTS” and “UR_LEADERBOARD” and also completes the requirements “FR_SATISFACTION _UPDATE” as events interact with satisfaction through map screen -> an instance of BuildingManager as can be seen in figure 1 and above in figure 4.

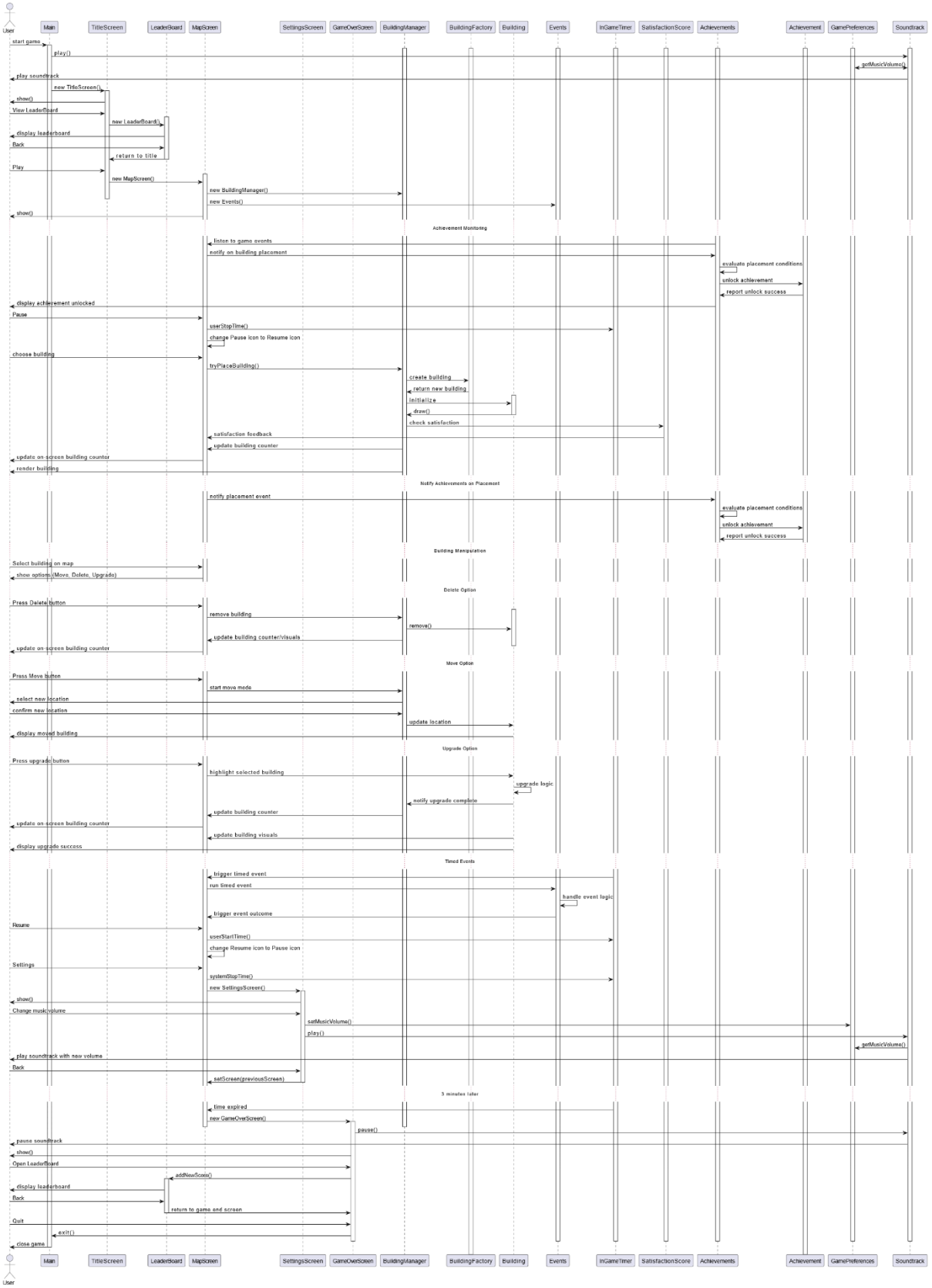

Figure 5: Sequence diagram to see how the game flows

The Flow of the game is predominantly the same the only real difference is the part where the user has options to manipulate the buildings labelled where the user

selecting a building results in a variety of options for changing the building such as deletion, moving or upgrade. The this all in line with functionality “UR_LOCATIONS_MOVE, UR_LOCATIONS_UPGRADE, ”

Following this we have also implemented the game flow for all new features such as the Leaderboard, Events, Satisfaction score and Achievements which are labelled and can be seen in all the above charts.

Non-Diagrammatic Notes:

- In figures 3 and 4 you may note that there is a mention of “Strands” these strands were planned as pieces of development that could be worked on simultaneously. This is mentioned in other documents as well as it was a part of our CI pipelines