Architecture

Group 12

Team 12

James Leney
Daniel Agar
Charles Thompson
Leyi Ye
Vanessa Ndiangang
Matt Smith

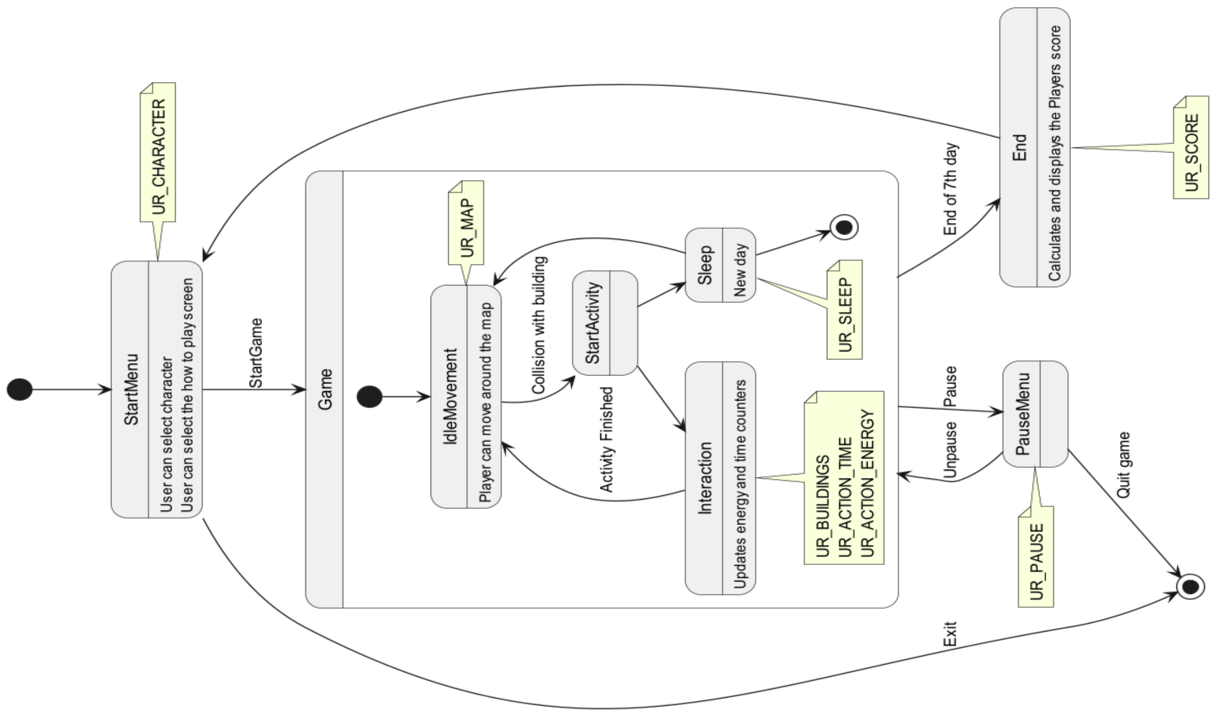

Figure 1: State Diagram

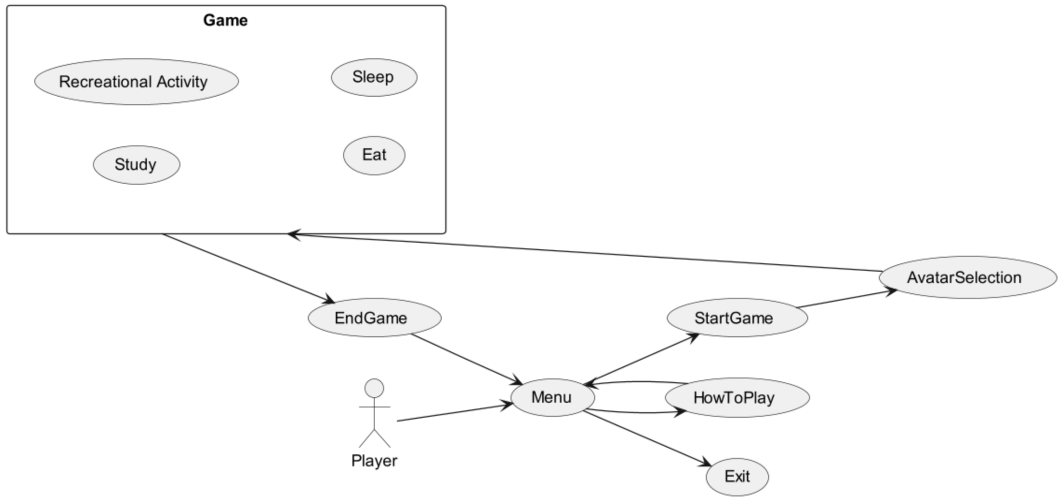

Figure 2: Use Case Diagram

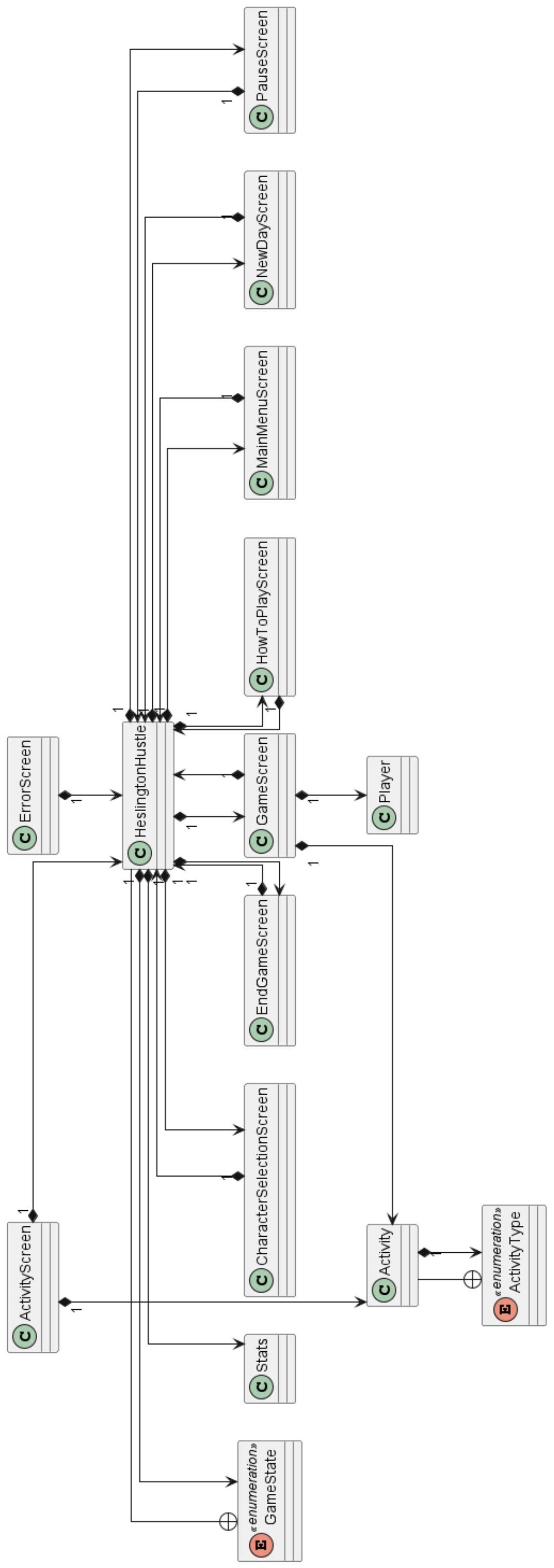

Figure 3: Overview of Class Diagram

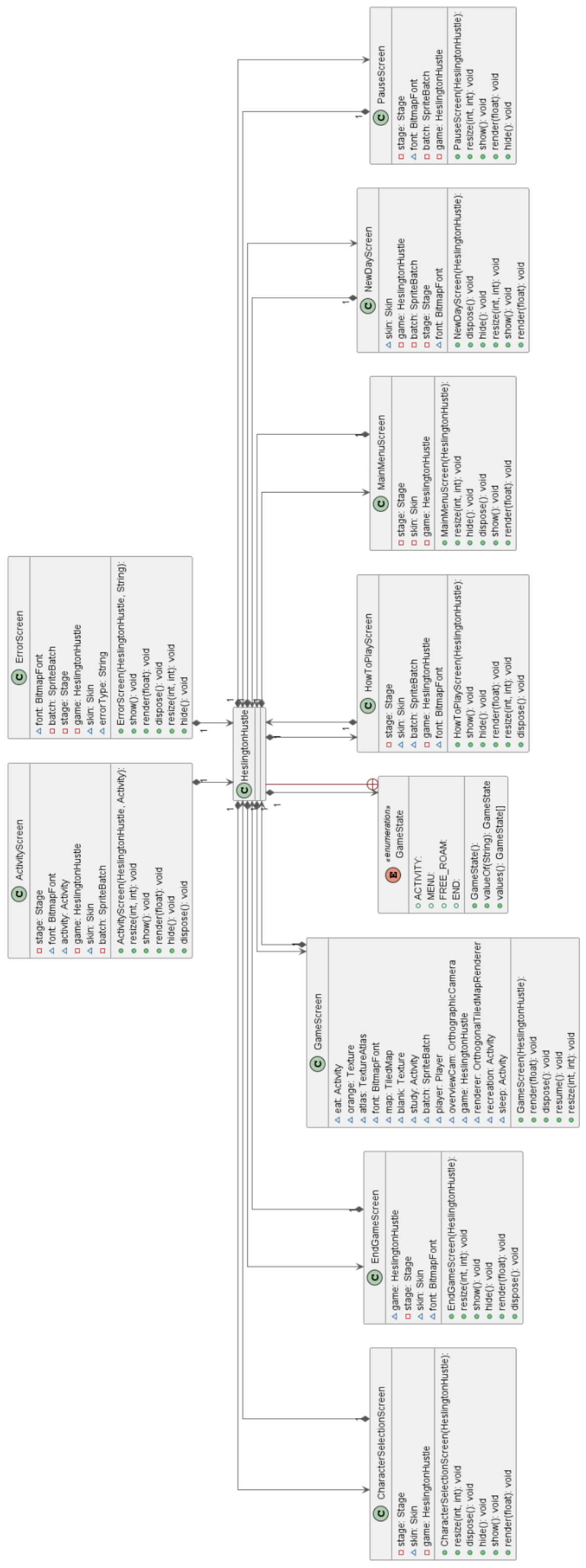

Figure 4: Screen-Centred Class Diagram

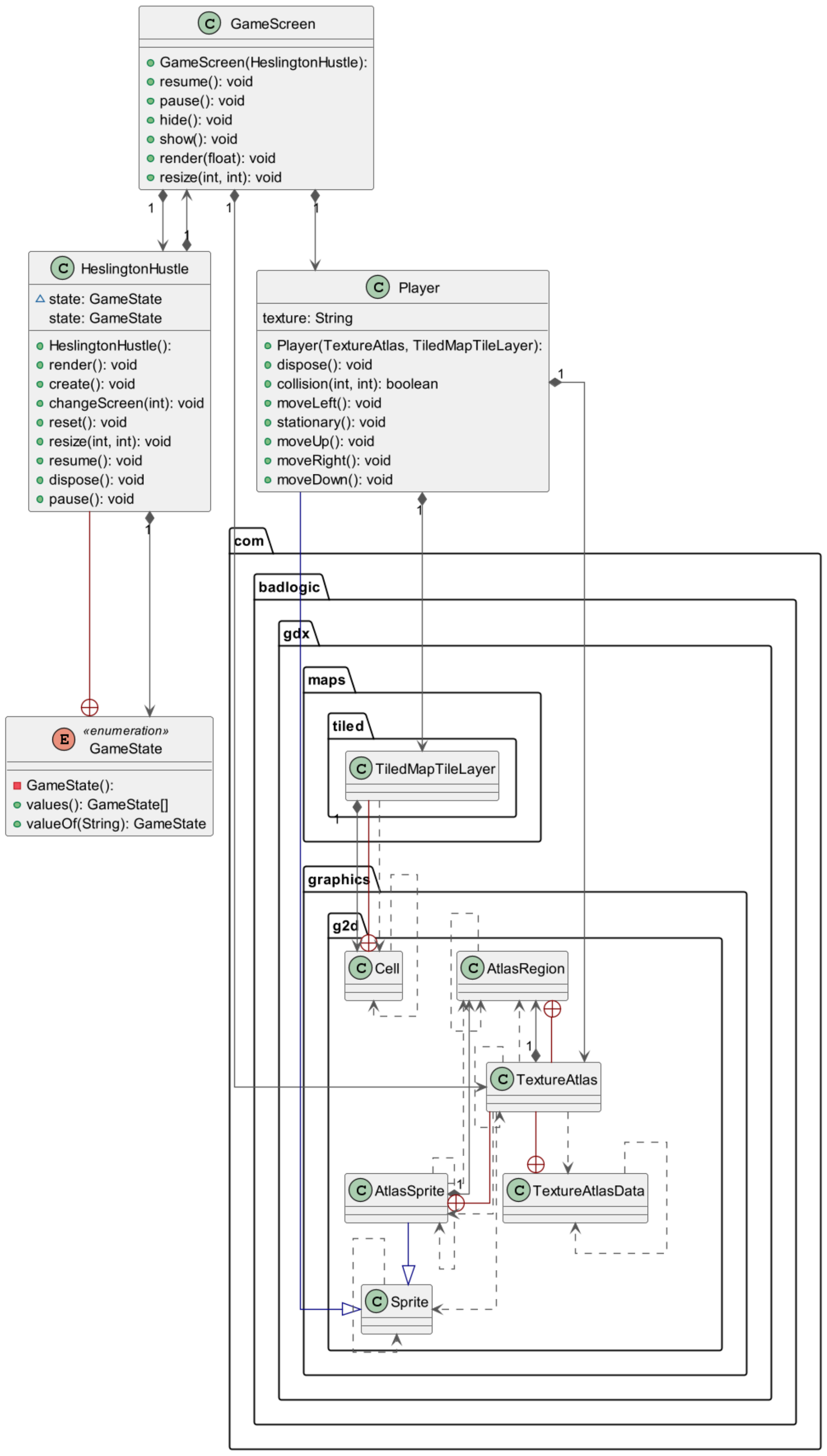

Figure 5: Player-Centred Class Diagram

To begin the design process we iterated through the RDD process and built a wider overview of the data structures needed in the product. We then made behavioural (use case and state) diagrams and structural diagrams to represent the product and adjusted the diagrams as the product developed. PlantUML and UML were used to make the various diagrams as our design evolved.

Behavioural Diagrams:
State Diagram: Firstly, the player is taken to the Menu page, where they can choose between StartGame, HowToPlay (which is a brief explanation of the game), and Exit. If the player decides to start the game, he/she must choose an avatar to play with. Once selected, the game will start. Here we use a rectangle to represent all the actions that can be taken during the seven days before the exam (sleeping, eating, studying, and doing recreational activities). When the seven days are up, the game ends and the player is taken back to the main menu page again.

Use Case Diagram: This behavioural diagram was created using the PlantUML extension with Google Docs. It reflects the different actions a player can take within the game.

Structural Diagrams:
The architecture of the project was designed according to the user, functional and non-functional requirements defined in the product brief and elicited during the customer meeting. The following table shows the relationship between the classes created in the game and the user and functional requirements that were taken into consideration:

Class	User Requirement	Functional Requirement
Activity	UR_BUILDINGS UR_SLEEP UR_STUDY_LOC UR_SLEEP_LOC UR_RECREATIONAL_LOC UR_EAT_LOC	FR_INTERACTIONS FR_ACTION_TIME FR_ACTION_ENERGY
HeslingtonHustle	UR_PLAYABLE	FR_TIME_IN_DAY
Player	UR_SINGLE_PLAYER	FR_MOVEMENT
Stats	UR_SCORE	FR_SCORE FR_INTERACTIONS
ActivityScreen	UR_PLAYABLE	
CharacterSelectionHustle	UR_CHARACTER	

EndGameScreen	UR_PLAYABLE	
ErrorScreen	UR_PLAYABLE	
GameScreen	UR_PLAYABLE	FR_TIME FR_CLOCK
HowToPlayScreen	UR_PLAYABLE	FR_TOOLTIPS
MainMenuScreen	UR_PLAYABLE	
NewDayScreen	UR_PLAYABLE	FR_TIME FR_SLEEP
PauseScreen	UR_PAUSE	

The non-functional requirements are not relatively reflected in the architecture of the design, but rather in the coding style of the game. We have written the code in such a way that it is easy to test, maintain and extend.