Software Requirements Specification (SRS) Modelling Requirements

4.1 Use Case Diagram

When interacting with the game DigiSafe the user can select an avatar, select a lesson and put in an answer. Depending on the type of the question, the user can select a Multiple-Choice Answer or type in a short text answer.

During a lesson, the game poses a question for each PSG provided. The answer that the user puts in is then verified by the game.

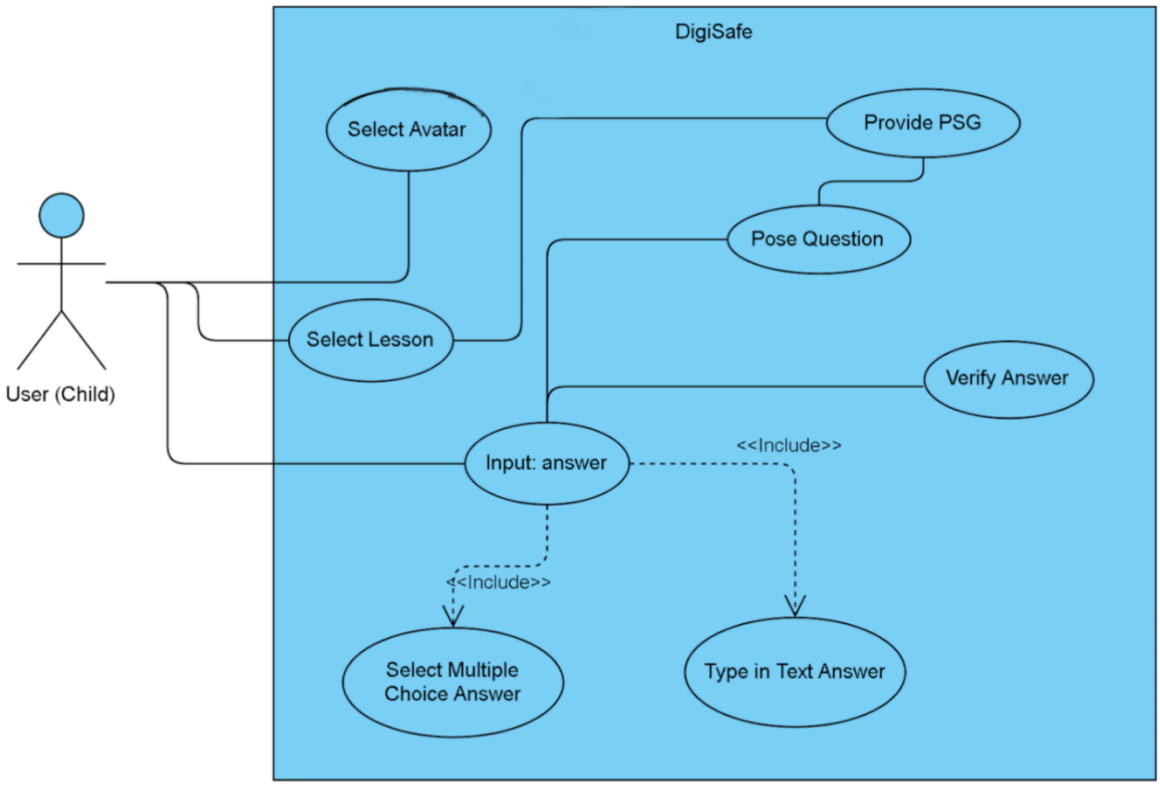


Fig. 2: DigiSafe Use Case Diagram

Use Case Name:	Select Avatar
Actors:	User (Child)
Description:	The user can select an avatar of their choice.
Type:	Primary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A
Uses cases:	N/A

Use Case Name:	Select Lesson
Actors:	User (Child)
Description:	The user can select one lesson of their choice.
Type:	Primary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A
Uses cases:	N/A

Use Case Name:	Provide PSG
Actors:	N/A
Description:	During the selected lesson, the PSG will provide necessary information in order to answer the question. In a passage of 8-10 sentences the situation is explained to the user and they learn how to behave.
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A
Uses cases:	N/A

Use Case Name:	Pose Question
Actors:	N/A
Description:	After the situation was explained by the PSG, the system poses a question in how to behave in this
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A

Uses cases:	Provide PSG

Use Case Name:	Input: answer
Actors:	User (Child)
Description:	User answers the question posed by the game.
Type:	Secondary, Essential
Includes:	Select Multiple Choice Answer, Type in Text Answer
Extends:	N/A
Cross-refs:	N/A
Uses cases:	Select Multiple Choice Answer, Type in Text Answer

Use Case Name:	Select Multiple Choice Answer
Actors:	User (Child)
Description:	The user picks their answer from a multiple choice catalogue.
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Input: answer
Uses cases:	N/A

Use Case Name:	Type in text answer
Actors:	User (Child)
Description:	The user types in their text answer to the question posed.
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Input: Answer
Uses cases:	N/A

Use Case Name:	Verify answer
Actors:	N/A
Description:	System checks whether the answer put in is correct or incorrect.
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A
Uses cases:	Input: answer

4.2 Sequence Diagram

This diagram shows different scenarios that will occur while using Digisafe. The first scenario will bring the user directly to choose an Avatar. The user will pick one from the selection of 5 different avatars. There is only one outcome that can occur in the scenarios which is to go forward to pick a lesson.

The second scenario will be to pick a lesson. There will be 8 lessons for the user to pick from. This scenario will also have only one outcome. Users will pick one out of the 8 lessons and go forward to take quiz.

The third and the last scenario will be where the user can watch PSG and take a quiz consisting of 10 questions. There will be two possible outcomes that can occur in this scenario. The user can either go back to pick another lesson or go back to change the avatar and keep playing the game.

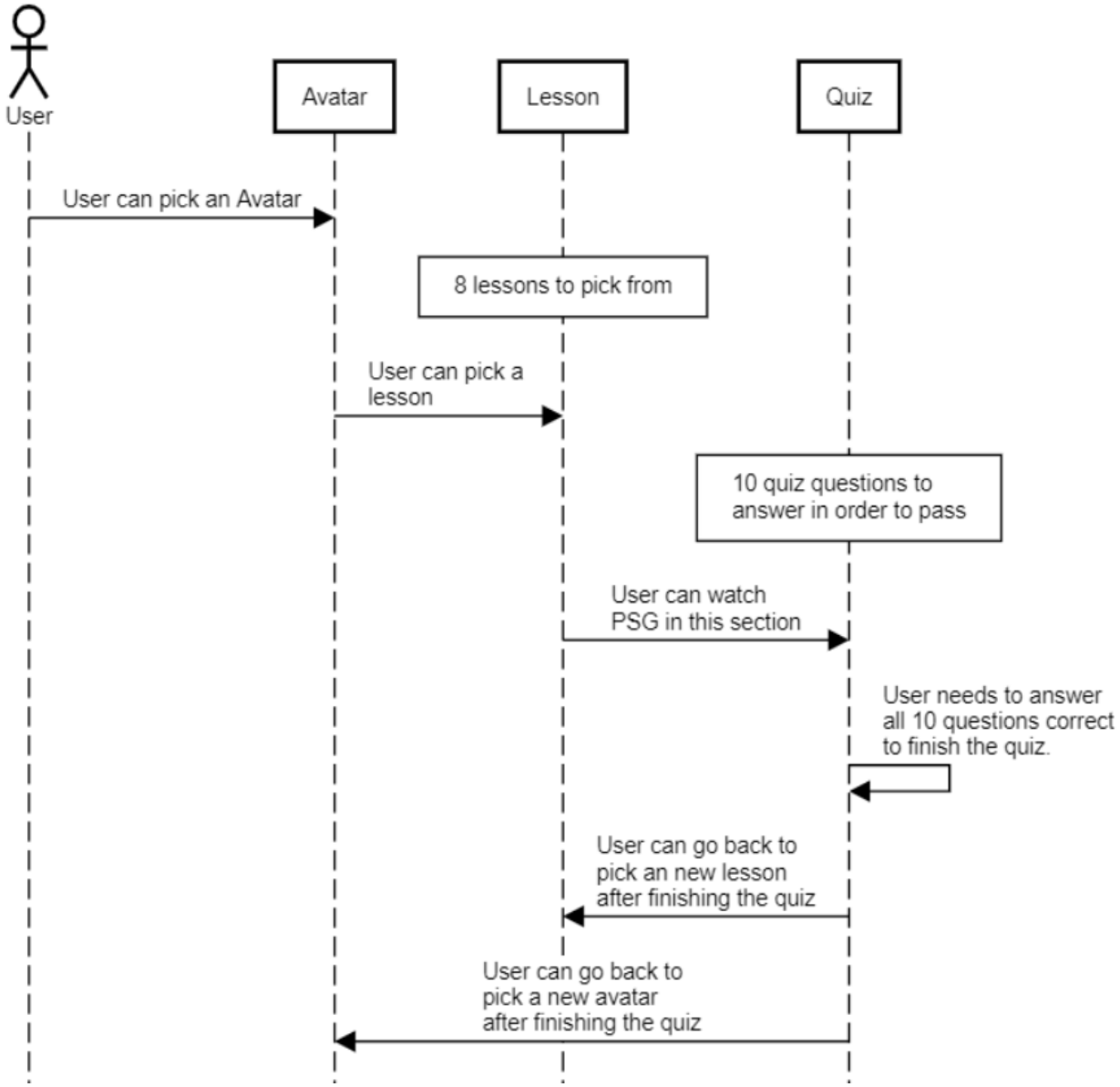


Fig. 3: Sequence Diagram

4.3 State Diagram

Below is the state diagram for DigiSafe. Each state is represented by a circle and the transactions made from each state are represented by arrows. All the states are either waiting for user input or idle. At first, the computer will be idle in the “Game(idle)” state. This is the homepage where the user will need to select an avatar for the game to proceed to the ”Avatar Selected” state. The user will then be able to select a lesson, which brings the game to the “Lesson Selected - PSG provided - Questions provided” state. After entering the state, the PSGs will be provided without any user input and sequentially the questions will be posed to the user. The user will watch all PSGs and answer all questions which brings the game to the “Take Quiz” state. In this state the user will have to answer 10 questions in either multiple choice or text format. After taking the quiz the user will proceed to an idle state - “Pass the Quiz(idle)”. Here the user input will be verified by the game and the game can only transition to another state once the user has passed the quiz for the current lesson. From this state the user can go back to select another avatar in the “Game(idle)” state. Alternatively, another lesson can be selected by going back to the “Avatar Selected” state.

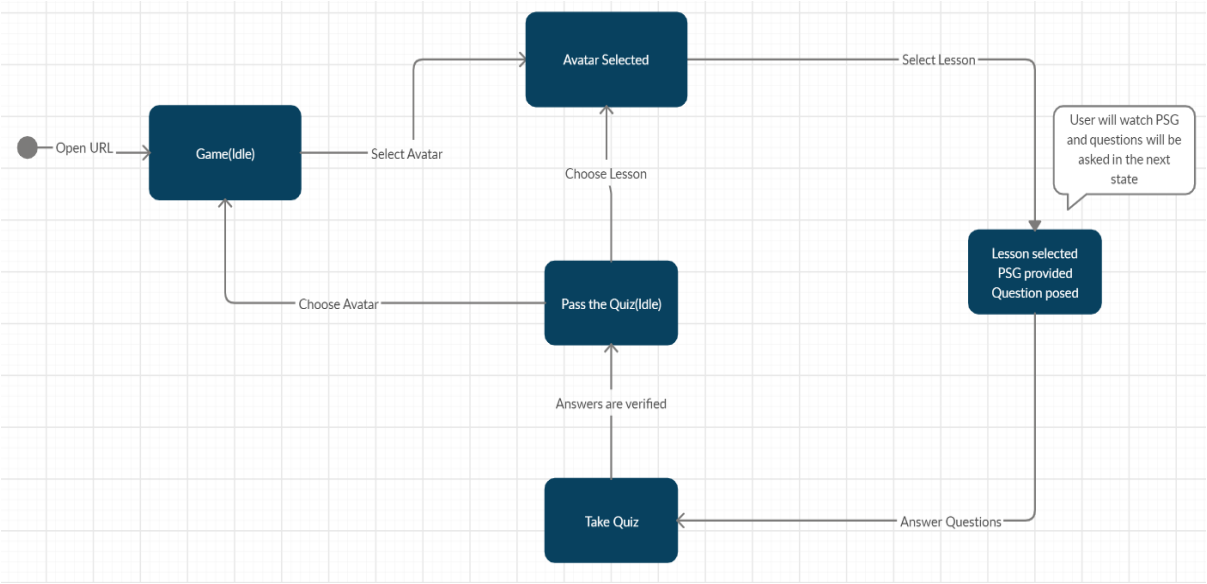
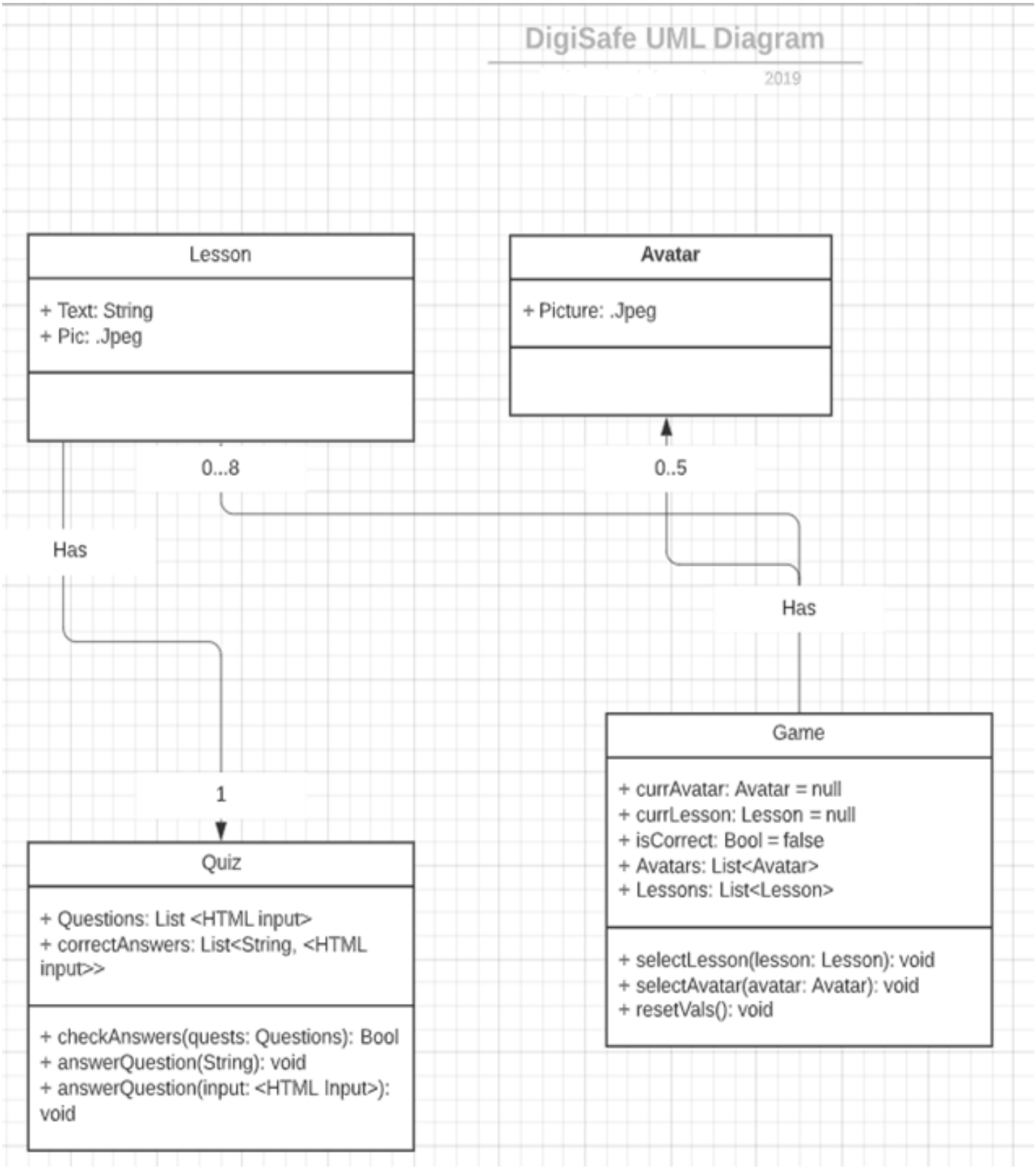


Fig. 4: State Diagram

4.4.1 Class Diagram

The class diagram depicts the software structure of DigiSafe. The main class is “Game” which has 1 to 5 avatars, represented by the “Avatar” class. Also, “Game” has 1 to 8 lessons which are represented by the “Lesson” class. Each lesson consists of 1 quiz. More detailed descriptions about the attributes and functions that each class offers can be found in the data dictionary below.



4.4.2 Data Dictionary

Element Name		**Description**
Avatar		User chooses an avatar
Attributes	+ Picture: .Jpeg	Selectable pictures of avatar
Operations	N/A	N/A
Relationships	Each Game has an associated Avatar	
UML Extensions	N/A	

Element Name		**Description**
Lesson		Our Lesson class only comprises of two attributes. For each Lesson, there exists associated text and one or more pictures to describe the lesson.
Attributes	+ Text: String	Associated text of a lesson
	+ Pics: List<.Jpeg>	A list of one or more pictures to describe the lesson.
Operations	N/A	N/A
Relationships	Each Lesson has an associated Quiz.	
UML Extensions	N/A	

Element Name		**Description**
Quiz		Users are to take a quiz after lessons, quiz contains multiple choice questions where users can input their answer by typing in text or select a choice.
Attributes	+ Questions: List<HTML input>	Questions are a list of HTML input elements.
	+ correctAnswers: List<Regex, HTML Input>	A list containing all correct multiple choice answers and Regex for verifying short-answer questions.
Operations	+ checkAnswers(quests: List<HTML input>): Bool	Checks the user-inputted answers for completion and correctness.
	+ answerQuestion(String): void	User inputs a string to answer a question
	+ answerQuestion(input: <HTML Input>): void	User selects an HTML input to answer a question.
Relationships	N/A	
UML Extensions	N/A	

Element Name	**Description**

Game		Element that holds both operations to avatar and lesson.
Attributes	+ currAvatar: Avatar = null	The current avatar selected.
	+ currLesson: Lesson = null	The current lesson selected.
	+ isCorrect: Bool = false	isCorrect is set to true when checkAnswers() returns true.
	+ Avatars: List<Avatar>	List of available Avatars.
	+ Lessons: List<Lesson>	List of available Lessons.
Operations	+ selectLesson(lesson: Lesson): void	Select a Lesson from the list of Lessons
	+ selectAvatar(avatar: Avatar): void	Select an Avatar from the list of Avatars.
	+resetVals(): void	Resets currAvatar and currLesson to null when user is finished with lesson or exits game.
Relationships	Each Game has a current Lesson.	
UML Extensions	N/A	