IIT
University of Dhaka

Software Requirements Specification and Analysis
Software Project Lab - II [SE-505]

Submitted By

Nusrat Jahan Lia (**BSSE - 1306**)

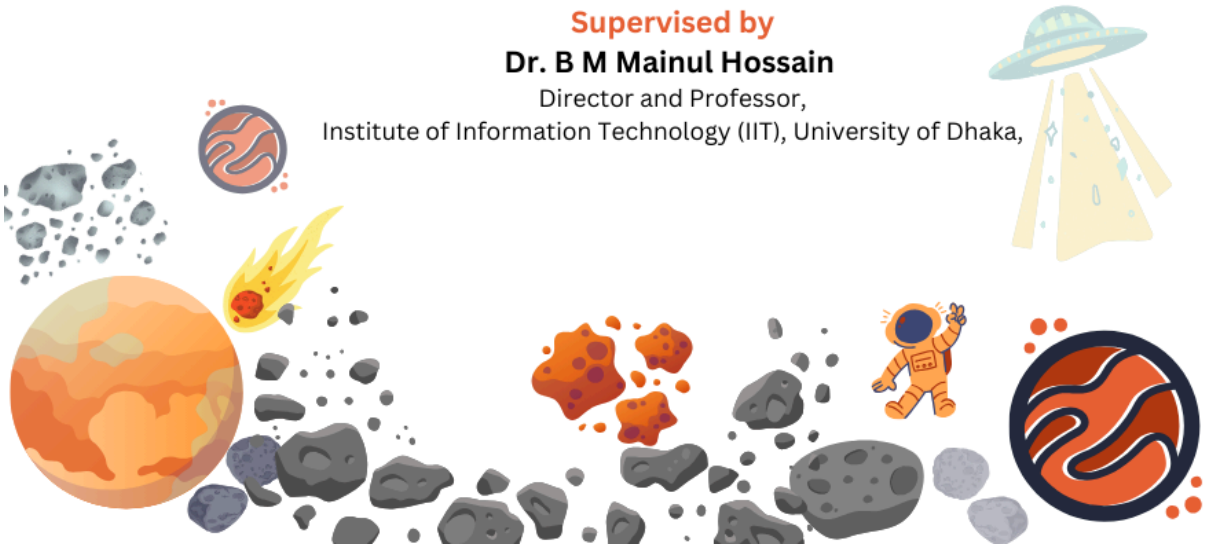
Imamul Hossain Rafi (**BSSE - 1323**)

Supervised by

Dr. B M Mainul Hossain

Director and Professor,

Institute of Information Technology (IIT), University of Dhaka,



1. INTRODUCTION

Welcome to MEG - the Mars Exploration Game, an innovative project to redefine gaming experiences by offering a journey into the realm of Martian exploration and learning about Mars. MEG blends entertainment with education, providing players with a platform to discover the mysteries of Mars while enjoying interactive gameplay.

1.1. Purpose

MEG serves a multifaceted purpose. Primarily, MEG aims to impart knowledge about Mars' geology, geography, and scientific endeavors, fostering a deeper understanding of Mars. Additionally, it aims to entertain and engage players by offering an enriching gaming experience centered around Martian exploration. In order to provide real-time experience MEG interacts with data provided by Curiosity Rover and displays real-time Martian weather Data to the players.

1.2. Intended Audience

MEG is designed to appeal to a diverse audience, including gamers of all ages (Primarily students of age 10 -16 years), space enthusiasts, educators, and individuals with an interest in scientific exploration.

1.3. Conclusion

MEG aims to inspire curiosity, spark interest in space exploration, and provide an enriching experience for players worldwide

2. INCEPTION OF MEG

The inception of MEG marks the initial phase of our endeavor, where we define the scope and nature of the project. To ensure an effective inception, we followed a systematic approach:

2.1. Establishing a Basic Understanding of the Problem

In the modern educational landscape, traditional methods of learning can often fall short of engaging students effectively. Particularly in subjects like astronomy, students struggle with visualizing complex structures and understanding space science merely through textual resources. Recognizing this challenge, our team envisioned a solution that transcends conventional learning approaches. We aimed to develop an application that transforms the learning experience into an interactive and engaging journey, akin to playing a game. By providing visualizations and interactive features of martian landscape exploration, we sought to bridge the gap between theoretical knowledge and practical understanding.

2.2. Identifying the Stakeholders of the Solution

Stakeholders play a crucial role in shaping the direction and success of our project. In our case, stakeholders encompass a diverse range of individuals and groups, including end-users, educators, educational institutions, and developers. Of particular significance are the individuals who will utilize the application to facilitate their understanding of mars.

3. ELICITATION OF MEG

After discussing the Inception phase, we need to focus on the Elicitation phase. So this chapter specifies the Elicitation phase.

3.1. Introduction

Requirements Elicitation is a part of requirements engineering that is the practice of gathering requirements from the stakeholders. We have faced many difficulties, like understanding the problems, making questions for the stakeholders, limited communication with stakeholders due to shortage of time and volatility of the stakeholders. Though it is not easy to gather requirements within a very short time, we have surpassed these problems in an organized and systematic manner.

3.2. Elicitation Requirements

In order to encourage a collaborative, team-oriented approach to requirements gathering, stakeholders work together to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements. In our elicitation phase, we completed the following tasks –

- Collaborative Requirements Gathering
- Quality Function Deployment

3.2.1. Collaborative Requirements Gathering

We have conducted meetings with many students who are preparing for NASA-SPACE-APP-CHALLAGE along with their supervisor . These meetings helped us to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements.

3.2.2. Quality Function Deployment

Quality Function Deployment (QFD) is a structured approach of defining customer needs or requirements and translating them into specific plans to produce products to meet those needs. The “voice of the customer” is the term to describe these stated and unstated customer needs or requirements.

3.2.2.1 Normal Requirements:

1. **Educational Content**: Provides accurate and educational information about Mars, its geography, atmosphere, and potential for exploration.
2. **Mission Objectives**: A series of mission objectives that guide the player through various aspects of Martian exploration.Objectives includes - finding martian rocks, finding water source, studying geological and rock formations, navigating to different martian regions.
3. **Realistic Simulation**: A realistic simulation of Martian environments, including terrain types, weather patterns, and day-night cycles.
4. **Progression System**: A progression system that rewards players for completing missions and achieving milestones, unlocking new tools, equipment, and areas to explore.

3.2.2.2 Expected Requirements:

1. **Intuitive Interface**: A user-friendly interface that is easy to navigate, with clear instructions and tooltips to guide them through the game mechanics.
2. **Compatibility**: The game should be compatible with users' preferred gaming platform, whether it's PC, console, or mobile.
3. **Physics Simulation**: A realistic physics simulation that accurately models the behavior of objects in the game world, including gravity, inertia, and collision dynamics.
4. **Visuals and Sound**: Immersive visuals and sound design that enhance the player's experience of exploring Mars, including realistic landscapes, atmospheric effects, and ambient sounds.
5. **Dynamic Weather Simulation**: Encounter dynamic weather patterns on Mars generated by advanced fluid dynamics simulations, including realistic dust storms, atmospheric disturbances that impact gameplay and exploration strategies.

3.2.2.3 Exciting Requirements:

1. **Virtual Reality (VR) Support**: Implement VR support to allow players to experience Martian exploration in an even more immersive way.
2. **Challenging Terrain**: Introduce challenging terrain features such as canyons, mountains, and caves that require creative problem-solving and navigation skills to overcome.

3. **Educational Quests and Challenges**: Engage in educational quests and challenges that teach them about Mars exploration, science, and technology. Players will solve puzzles, complete missions.
4. **Geological Survey Missions**: Geological survey tools to study Martian rock formations, mineral deposits, and geothermal features.
5. **Sampling**: Players collect and analyze the samples using onboard instruments and receive detailed information about objects' composition, age, and geological significance.

4. REQUIREMENTS MODELING

The requirements model, actually a set of models—is the first technical representation of a system. Requirements modeling uses a combination of text and diagrammatic forms to depict requirements in a way that is relatively easy to understand, and more important, straightforward to review for correctness, completeness, and consistency.

4.1. SCENARIO BASED MODELING

This chapter describes the Scenario Based Model for “MEG”.

4.1.1. User Story

Game Onboarding:

When players begin their journey into the game, they'll be prompted to input their name right on the homepage. On the next screen, there will be 4 options:

- **Resume Game :** Pick up where they left off.
- **New Game :** Start a fresh adventure.
- **High Scores :** Check out top scores.
- **About :** Learn more about the gameplay.

- **Credit** **:** View the game's creators.
- **Exit** **:** Leave the game.

On clicking the new game option, a map will unfold, revealing the levels available for exploration. Initially, only the first level will be accessible, with subsequent levels unlocking as players progress and accumulate points. Once unlocked, players have the freedom to choose which level they want to tackle next. On clicking the resume game option, previously unlocked levels will be also seen unlocked in the directed map.

Level Welcome:

Upon selecting a level, players will be greeted with a warm Martian-themed welcome screen. This screen sets the stage for the adventure ahead and features a 'Start' button to initiate the level. The game comprises five distinct levels. Each level transports players to a different region of the Martian landscape. On the starting of each level, the screen will show the level's region name followed by the amount of time they must complete the exploration within.

Real-time Martian Weather:

Throughout gameplay, players will receive real-time updates on Martian weather conditions directly on their screen. This includes essential information such as maximum and minimum

temperatures, UV index, and atmospheric pressure. For example, if a player engages with the game on May 1st, 2024, they'll see the actual weather data from Mars on that specific date.

Game Play and Puzzles:

.

- Puzzles within each level are crafted to reflect the unique geological features and landmarks of the region.
- Upon discovering objects within a level, players utilize shovels to collect samples of the objects and extract valuable information. On hitting the object with shovel, a pop up will show -

Name of the object

Sample Collected!

Analyze Button

- On clicking the analyze button, respective analyzing text will show up for each object.
- Extracted information serves as critical clues to aid players in deciphering puzzles.
- Each object from different levels has a different extraction time.
- The more upgraded shovels the player uses, the less time will be needed to extract information. Shovels and different suits can be bought from the store in exchange for currency. Currency for this game is “Basalt” samples. Every level has a designated

number of basalt rocks which players can collect and use as currency. Beside a currency icon, level-screen will show the number of basalts (currency) a player has.

- Player suits will have different activity features to aid the player in quest and reduce game play time.
- Accessible via puzzle icons on the gaming screen, players can view and attempt both unsolved puzzles at any time during gameplay. Number of puzzle icons depends on the number of puzzles required to solve at each level.
- Level screen will also have a store icon. Store icon will direct players to shovel and suit store. In the store, items will be listed as following :

Item icon
Item feature
Item price
Buy / Use Button

If an item is already bought, players can use it. If not bought yet, players have to buy it first.

- Level screen will also have an inventory icon. Upon clicking on this icon, players will be directed to an inventory where they can see their collected samples. Inventory will be

updated as they progress to collect samples throughout the game. In the inventory, samples will the listed as following :

Sample icon

Sample Name

- Each level imposes a specific time limit for completion. If players can not finis on time, no points will be gained.
- Players can gain points by solving puzzles correctly. Points of a level = 1.5 * seconds saved before the level's timespan ends.
- On clicking any option, system will check whether clicked option is collected in inventory or not, if not collected yet ,

  POP UP - “You didn’t collect this sample of this element yet ! ”

  If the sample is collected, for the right answer - “Correct!!” , for the wrong answer - “Try again ! ”

  On clicking the right answer of any puzzle, that puzzle icon will be disabled.

- Players can attempt puzzles multiple times until solved.
- Players will encounter dust storms and meteoroids falling, where they have to take shelter or face failure. If failed to escape, the player has to start the level from the initial point again.
- Each level will have a hideout where they start their journey from and come back in order to escape calamities.

- Players can level up only if the player has completed all the puzzles or challenges within the time limit.

Level scenario :

Level	Region	Discoveries	Analysis Text
1	**Southern Highlands**	- Basalt (Number of Basalt for this level is - 15)	• Formed through volcanic activity • Primarily consists of Olivine, Feldspars, and Pyroxenes.
		- Basaltic Shergottites	**Hurray! Found a Meteorite**! • Mineralogy comprises mostly pigeonite and augite. • Holds evidence of interaction with Martian water.
		- Lherzolitic Shergottites	**It's a Meteorite**

			• Ultramafic rocks of plutonic origin • Shares mineralogical and chemical features with basaltic shergottites • Silicon-poor
2	**Syrtis Major**	- Dacite	• Volcanic rock • High in silica • Low in alkali metal oxides • Composed predominantly of plagioclase feldspar and quartz • Has a fine-grained (aphanitic) to porphyritic texture.
		- Granitoids	• Coarse-grained igneous rocks • Predominantly consists of quartz, plagioclase,

			and alkali feldspar.
		- Basalt (Number of Basalt for this level is - 25)	● Already explored in Southern Highlands ● Consists primarily of Olivine, Feldspars and Pyroxenes
		- Nili Fossae (different type of terrain texture)	● A group of large, concentric grabens on Mars. ● Contains carbonate minerals ● One of seven finalists for the MSL(Mars Science Laboratory) landing sites.
		- Carbonate Rocks (In Nili fossae terrain)	● Formed through hydrothermal precipitation.

			• Evidence of living organisms could have been preserved here.
3	**Northern Lowlands**	- Andesite	• More evolved forms of magma • Highly volatile • Constitutes the majority of the crust
		- Dacite	• Volcanic rock • High in silica • Low in alkali metal oxides • Composed predominantly of plagioclase feldspar and quartz
		- Basalt (Number of Basalt for this level is - 35)	It's your currency to use ! Basalt !
		- Granitoids	Do you remember this high silica rock? You've already

			discovered it in Syrtis Major. This volcanic rock is composed predominantly of quartz, plagioclase, and alkali feldspar. Guess the name and win extra points! ☐ Basalt ☐ Dacite ☑ ~~Granitoids~~ ☐ Nili Fossae
		- Sedimentary Rocks	• Makes up the majority of the Northern lowlands deposits • May have formed from sea/lake deposits • Holds chances of finding fossilized life • Shows cross-bedding in their layers
		- Carbonate Rocks	Carbonate Rocks.Do you remember where you've seen

			them ?Guess the name and win extra points! ☑ ~~Nili Fossae~~ ☐ Southern Highlands ☐ Main region on Syrtis Major
		- Nakhlite	**Meteorite Again!** ● Contains carbonate and sulfate salts ● Suggests residence in an environment rich with liquid seawater ● Crystallized 1.3 to 1.4 billion years ago
4	**Meridiani Planum**	- Basalt (Number of Basalt for this level is - 40)	Can you guess the rock? You've already seen it in all the previous regions. ☑ ~~Basalt~~ ☐ Dacite
		- Jarosite	● Indicative of past

			aqueous activity ● Often found in association with hematite ● Formed in acidic, sulfate-rich environments
		- Hematite Outcrops	● Iron oxide mineral ● Abundant on Mars ● Associated with past aqueous environments ● Regarded as martian blueberries but they are not blue
		- Sulfate Salt Deposits	● Indicative of past aqueous activity ● an preserve evidence of ancient environments
		- Chlorine and Bromine Deposits	● Halogens associated with Martian soils and

			sediments ● can provide insights into the chemical composition of the Martian surface
		- Bounce Rock	**Meteorite!!** ● It's composition identical to that of Shergottites found on Earth ● Believed to have been ejected by impact of large asteroids or comets ● Composed mainly of the volcanic mineral, pyroxene ● Unlike any rock or volcanic deposit onMars."

5	Gusev Crater	- Humphrey	• A dark volcanic rock • About 60 centimeters (2 feet) tall • Shows bright material in interior crevices and cracks that looks like minerals crystallized out of water
		- Basalt (Number of Basalt for this level is - 50)	The more you collect me, the more you earn !
		- Carbonate rocks	It is formed through hydrothermal precipitation. Guess it and win extra points ! ☑ ~~Carbonates~~ ☐ Hematite ☐ Basalts

		- Sulfate Salt Deposits	● More sulfate salts than found anywhere else so far on Mars ● Indicative of past aqueous activity ● Can preserve evidence of ancient environments on Mars.
		- Goethite	Remember the jarosite from Meridiani Planum ? ● Over time, the water turns jarosite into another mineral called goethite. ● In dry conditions, such as on Mars, goethite dehydrates into yet another iron-rich mineral called hematite, which is

			what gives Mars its rust-red color. ● Goethite forms only in the presence of water, so its discovery is the first direct evidence of past water in the Columbia Hills's rocks.
		- Hematite (Iron-rich mineral that gives Mars its rust-red color)	This is what makes Mars rusty-red . Do you know its name ? Guess and Gain extra points!” ☐ Basalt ☑ ~~Hematite~~ ☐ Jarosite

Level Puzzles :

Level	Puzzles	Challenges
1. **Southern Highlands**	1. I am born from the depths of Lava reservoirs.Yet, I am the silent singer of silicon-poor compositions ☐ Basalt ☐ Basaltic Shergottites ☑ ~~Lherzolitic Shergottites~~	• Must solve two of the puzzles. • **Dust storm** for 10 seconds. • Dust storm appears right after 2 minute game play. • Must finish the Level in 4 minutes.
	2.In the cosmic dance of time, I emerge from ancient flames. For eons, I've journeyed through the void, bearing tales untold, Each crystal a witness to the touch of Liquid Crystals, bold ☐ Basalt ☑ ~~Basaltic Shergottites~~ ☐ Lherzolitic Shergottites	
2. **Syrtis Major**	1.Born from lava's swift freeze, with quartz and feldspar it gleams; scarce alkali, well-grained sheen, in Martian landscapes, it's a dream	• Must solve three of the puzzles.

	☐ Basalt ☑ ~~Dacite~~ ☐ Granitoids ☐ Nili Fossae	• **Meteoroids falling** for 15 seconds. • Meteoroids fall right after 3 minute game play. • Must finish the Level in 6 minutes.
	2.Through hydrothermal embrace, life's essence may gleam, within these silent depths, a preserved story, unseen ☐ Basalt ☐ Dacite ☐ Granitoids ☑ ~~Carbonated Rocks~~	
	3.A treasure trove they found, with carbonate minerals, their mysteries unwound ☐ Basalt ☐ Dacite ☐ Granitoids ☑ ~~Nili Fossae~~	
3. **Northern Lowlands**	1.In realms where waters once whispered tales, Beneath the ancient gaze of Martian skies, I lie, a testament to time's gentle hand, Where secrets of life may yet arise.	• Must solve two of the puzzles. • **Dust storm** for 20

	☐ Andesite ☐ Sedimentary Rocks ☑ ~~Carbonate Rocks~~ ☐ Nakhlite	seconds. ● Dust storm comes right after 3 minute game play. ● Must finish the Level in 8 minutes.
	2.I form the bedrock, in layers, stories unfold. From ancient waters or windswept sands, they say, Within my embrace, life's whispers may sway. ☐ Dacite ☑ ~~Sedimentary Rocks~~ ☐ Carbonate Rocks ☐ Nakhlite	
4.**Meridiani Planum**	1.Aqueous whispers in silent stone, where Martian skies once did roam. Minerals aligned in secret tales, within this ancient Martian veil. ☑ ~~Jarosite~~ ☐ Sulfates ☐ Carbonates ☐ Hematite	● Must solve three of the puzzles. ● **Dust storm** for 20 seconds. ● Dust storm comes right after 3 minute game play.
	2.In rusted hues, I stand with my friends, a testament to time's unending drone.	

	☐ Jarosite ☐ Sulfates ☑ ~~Hematite Outcrops~~	● **Meteoroids falling** for 15 seconds. ● Meteoroids fall right after 6 minute game play. ● Must finish the Level in 10 minutes.
	3. A wanderer from Earth's twilight. Ejected by force, from impact's might, A tale of cosmic collision, in the starry night. ☑ ~~Bounce Rock~~ ☐ Shergottite ☐ Martian Meteorite ☐ Pyroxene	
5. **Gusev Crater**	No Puzzle	● Must solve three of the puzzles. ● **Dust storm** for 10 seconds. ● Dust storm comes right after 3 and 10 minute game play.

		● **Meteoroids falling** for 15 seconds. ● Meteoroids fall right after 7 minute game play. ● Must finish the Level in 12 minutes. ● On clicking the puzzle/challenge icon -” No puzzles anymore ! Collect every element and get back to me !” ● After the player collects every other element on this level(After or within 12 minutes), on clicking the puzzle/challenge icon, they will be

		directed to inventory and asked to put every element on its respective region. If the player puts everything correctly ,They win !.

Object Extraction time (without any shovel used):

Objects	
Name	**Extraction Time**
Basalt	8s
Basaltic Shergottites*	10s
Lherzolitic Shergottites*	10s
Dacite*	15s
Granitoids	12s
Nili Fossae*	15s
Carbonate Rocks*	15s
Andesite	14s

Sedimentary Rocks*	20s
Nakhlites	14s
Jarosite Deposits*	22s
Hematite Outcrops*	22s
Sulfate Salt Deposits	16s
Chlorine and Bromine Deposits	16s
Bounce Rock*	22s
Humphrey	20s
Sulfate Salt Deposits	20s
Goethite	20s

*denotes that this object is answer of any puzzle

Mining tools (Shovel) details : (Can be collected from store)

Shovels			
No.	**Color**	**Speed**	**Price**
Shovel-1	Wood	2X	5
Shovel-2	Bronze	4X	10
Shovel-3	Silver	6X	20
Shovel-4	Golden	7X	25
Shovel-5	Sky Blue	8X	30

Suits Details :

The store will have different astronaut suits as well. Different suits have different abilities like running faster, jumping higher. These suits will help the player to explore and to hide faster

Suits				
No.	**Color**	**Speed**	**Jump force**	**Price (Basalts)**
Suit-1	White	5X	0	10
Suit-2	Blue	10X	3	15
Suit-3	Dark-Yellow	15X	6	20
Suit-4	Dark-Green	20X	9	30

Visualization Reference :

Southern Highlands :

Source : https://www.universetoday.com/tag/mars-southern-highlands/

Basaltic Shergottites:

Source : http://www.meteorites.tv/284-nwa-2975-martian-meteorite.html

Basalt :

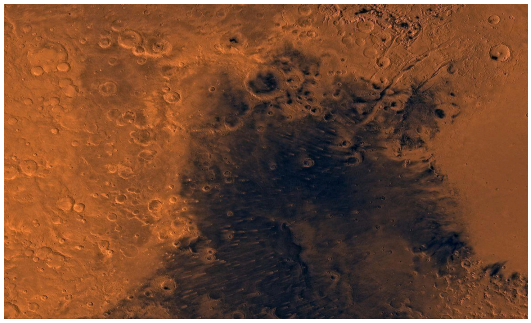
Source : https://en.wikipedia.org/wiki/Basalt

LherzolticShergotittes :

Source : www.arizonaskiesmeteorites.com/AZ_Skies_Links/Martian/NWA_195

0/\

Syrtis Major :



Source : https://en.wikipedia.org/wiki/Syrtis_Major_Planum

Dacites :

Source: https://en.wikipedia.org/wiki/Dacite

Granitoid :

Source : https://en.wikipedia.org/wiki/Granitoid

Nili Fossae :

Source : https://en.wikipedia.org/wiki/Nili_Fossae

Carbonate Rocks :

Source : https://news.cornell.edu/stories/2017/02/scientists-are-puzzled-over-lack-carbonate-mars

Northern Lowlands :

Source :

https://reasons.org/explore/publications/articles/martian-lowlands-are-old

Andesite :

Source :

https://rocksminerals.flexiblelearning.auckland.ac.nz/rocks/images/andesite.jpg

Nakhlites :

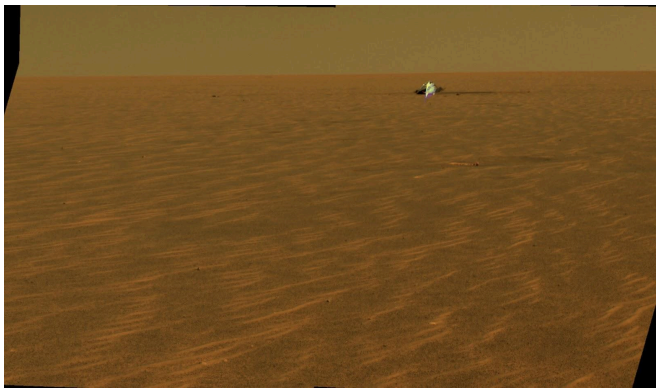
Source:https://upload.wikimedia.org/wikipedia/commons/thumb/f/fe/NWA_998_meteorite%2C_nakhlite.jpg/1024px-NWA_998_meteorite%2C_nakhlite.jpg

Sedimentary Rocks :

Source :

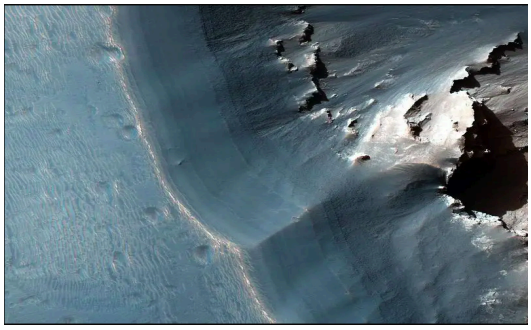
https://geology.com/stories/13/rocks-on-mars/mudstones.jpg

Meridiani Planum :



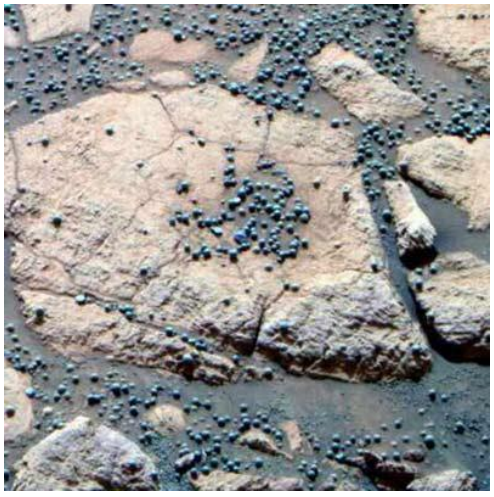
Source : https://upload.wikimedia.org/wikipedia/commons/6/6b/Sol322B.Smooth.Sheet.bedforms.close.to.heat.shield.crp.jpg

Jarosite Deposits :



Source : https://www.nasa.gov/wp-content/uploads/2023/03/pia20371.jpg

Hematite Outcrops:

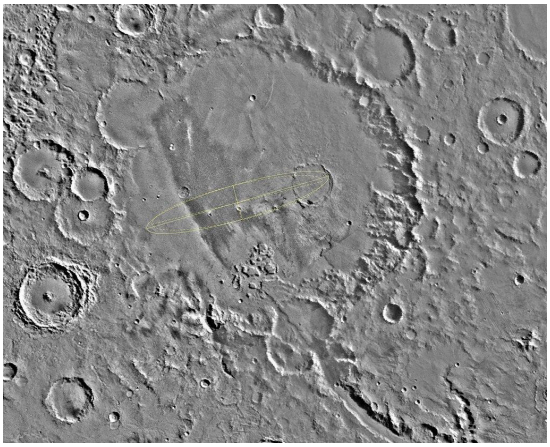


Source : https://www.chinadaily.com.cn/english/doc/2004-03/20/xinsrc_060ba2574b7b42c4acb15e60b01145b1_blueberry.jpg (They are not blue, the picture shows false-color composite image)

Bounce Rock :

Source : https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/58606main_image_feature_167_jwfull.jpg/800px-58606main_image_feature_167_jwfull.jpg

Gusev Crater :



Source : https://upload.wikimedia.org/wikipedia/commons/thumb/6/62/

Gusev_crater_Spirit_landing_ellipse.jpg/1024px-Gusev_crater_

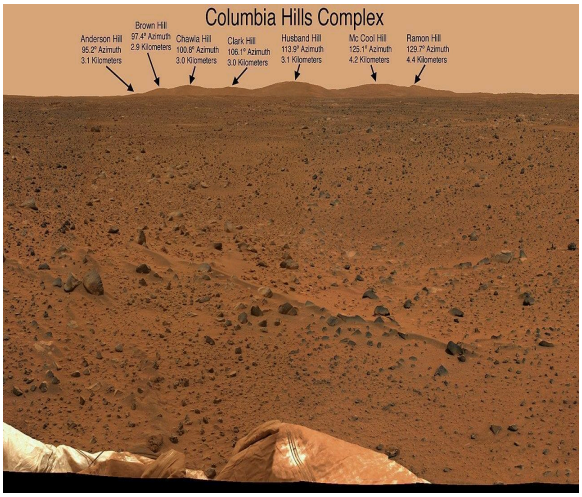
Spirit_landing_ellipse.jpg

Humphrey :

Source :

https://mars.nasa.gov/mer/gallery/press/spirit/20040305a/05-ra-03-humphrey-A061R1_br.jpg

Columbia Hills :



Source :

https://upload.wikimedia.org/wikipedia/commons/thumb/a/ae/Columbia_Hills_from_MER-A_landing_site_PIA05200_br2.jpg/1280px-Columbia_Hills_from_MER-A_landing_site_PIA05200_br2.jpg

4.1.2. Use Case Diagram

A use case is a list of actions or event steps typically defining the interactions between a role (actor) and a system to achieve a goal. The actor can be a human or other external systems. In this modeling, use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. Use case diagrams are a blueprint for the system. Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Use Case ID: 0

Name: Martian Exploration Game

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage.

[The {MAAS} API is an open source REST API built to help make it easier and more efficient to build interactive applications that want to utilize the wealth of weather data being transmitted by the Curiosity Rover on Mars. Our API is built upon the REMS (Rover Environmental Monitoring Station) data provided by the Centro de Astrobiologia (CSIC-INTA).]

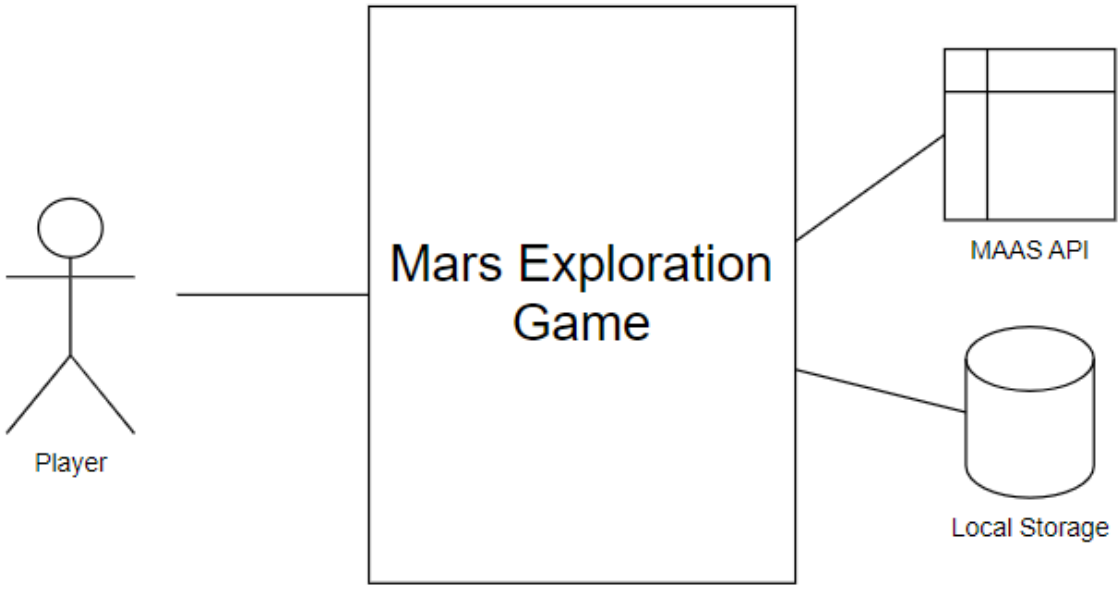


Figure 0 : MEG

Description: This “Mars Exploration Game” Allows players to interact with a virtually created Mars environment and learn about Mars through gameplay.

Player:

Action: The player enters name, age and press start.

Replay: Gameplay starts.

Action: Player roam through different levels

Replay: Each level gives some rewards

Use Case ID: 1

Name: Mars Exploration Game Details

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage.

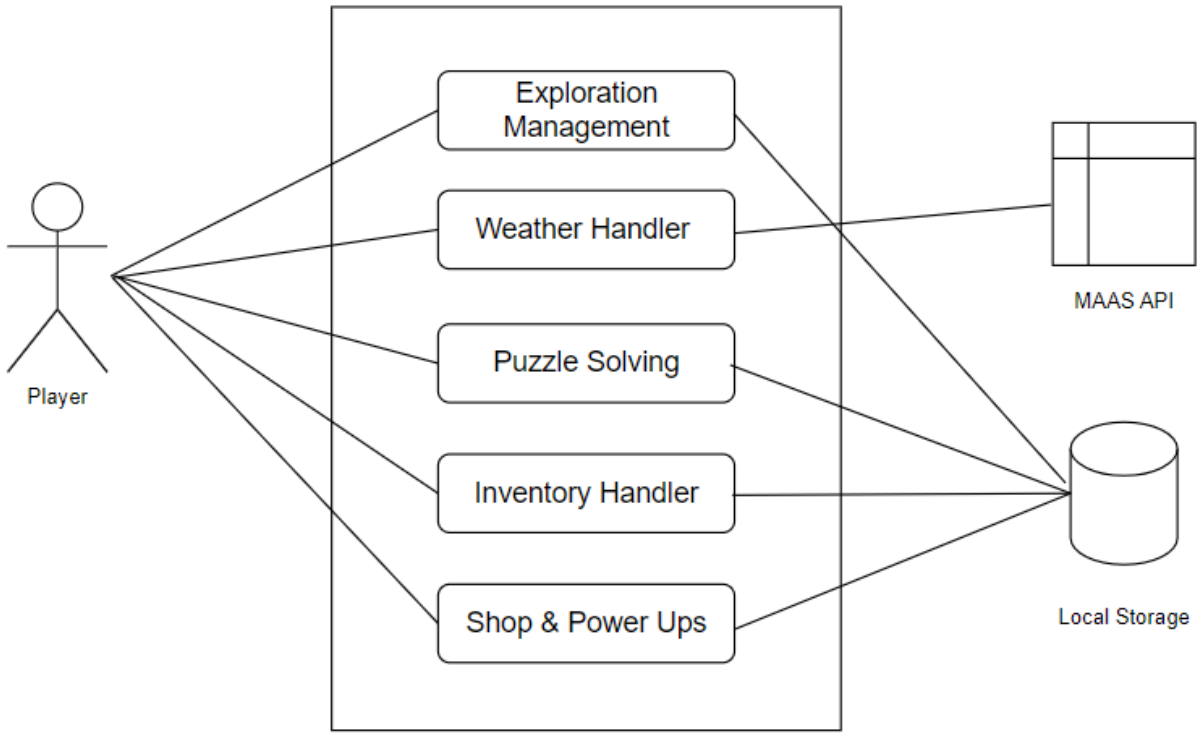


Figure 1 : MEG Details

Description:

Entry: When a player starts playing the game for the first time, the entry system will ask for the player's name, age in the Starting UI. Then there will be a map pop up with a level roadmap. Initially the Map will only give access to level 1. But the game will save player data in local storage and when a second time or later player enters the game, all the unlocked levels will also be shown in the Map. Players can choose their level to play at will. After pressing start, players can begin playing.

Use Case ID: 1.1

Name: Exploration System

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage,

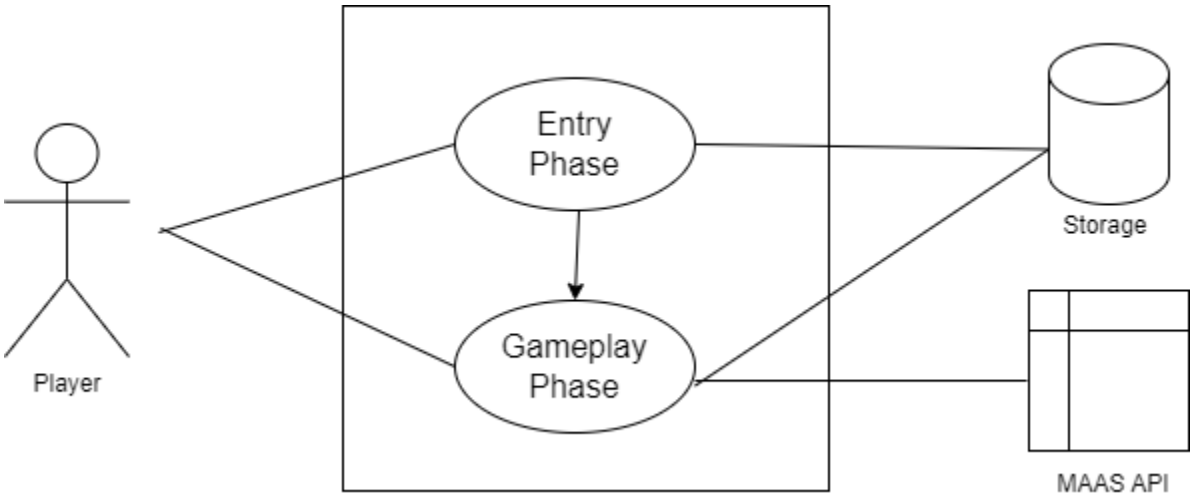


Figure 1.1 : Exploration System

Description:

Entry Phase: When players begin their journey into the game, they'll be prompted to input their name right on the homepage. On the next screen, they will choose between a resume game or a new game. On clicking the new game option, a map will unfold, revealing the levels available for exploration. Initially, only the first level will be accessible, with subsequent levels unlocking as players progress and accumulate points. Once unlocked, players have the freedom to choose which level they want to tackle next. On clicking the resume game option, previously unlocked levels will be also seen unlocked in the directed map.

Gameplay Phase: In the roadmap section, the player will select any of the unlocked levels. Following, players will spawn in level wise regions. Every region has collectibles, the player's main goal is to collect these objects in order to solve the puzzles to pass the level. On the final level, there will be no puzzles, but the player will be asked to organize the objects according to their region.

Use Case ID: 1.1.1

Name: Entry Phase

Primary Actor: Player,

Secondary Actor: Local storage

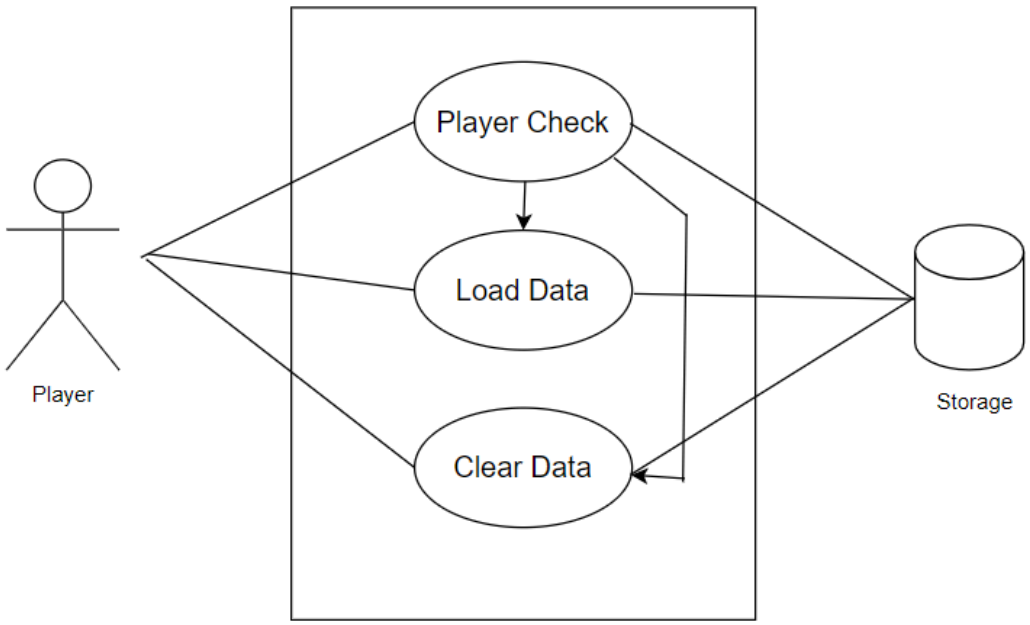


Figure 1.1.1 : Entry System

Action and Reply:

Action: Player launch game

Reply: Game will be launched. If it is the very first time, the system will ask for the player's name and age. If not, the player will be directed to the home screen.

Action: Player select Resume game or New game.

Reply: Resume game redirect player to roadmap with previously unlocked levels, New game makes a fresh start.

Use Case ID: 1.1.2

Name: Gameplay Phase

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage,

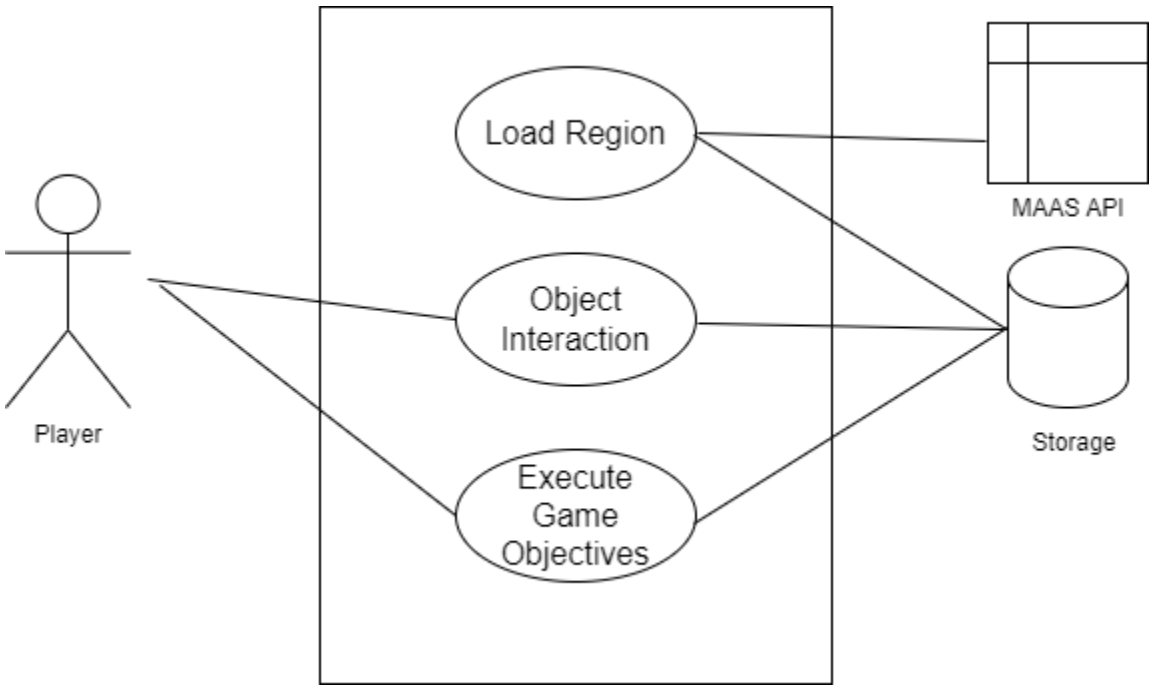


Figure 1.1.2 : GamePlay

Action and Reply:

Action: Player select level from the map.

Reply: Player will be spawned in level wise regions and region data(Player collection, weather, environmental objects) will be loaded.

Action: Player interacts with objects with hand or shovel.

Reply: Object's information will be shown in an UI and samples will be collected.

Action: Player choose puzzle/inventory/store icon

Reply: puzzle/inventory/store system execute.

Use Case ID: 1.2

Name: Weather Handler

Primary Actor: Player,

Secondary Actor: MAAS API

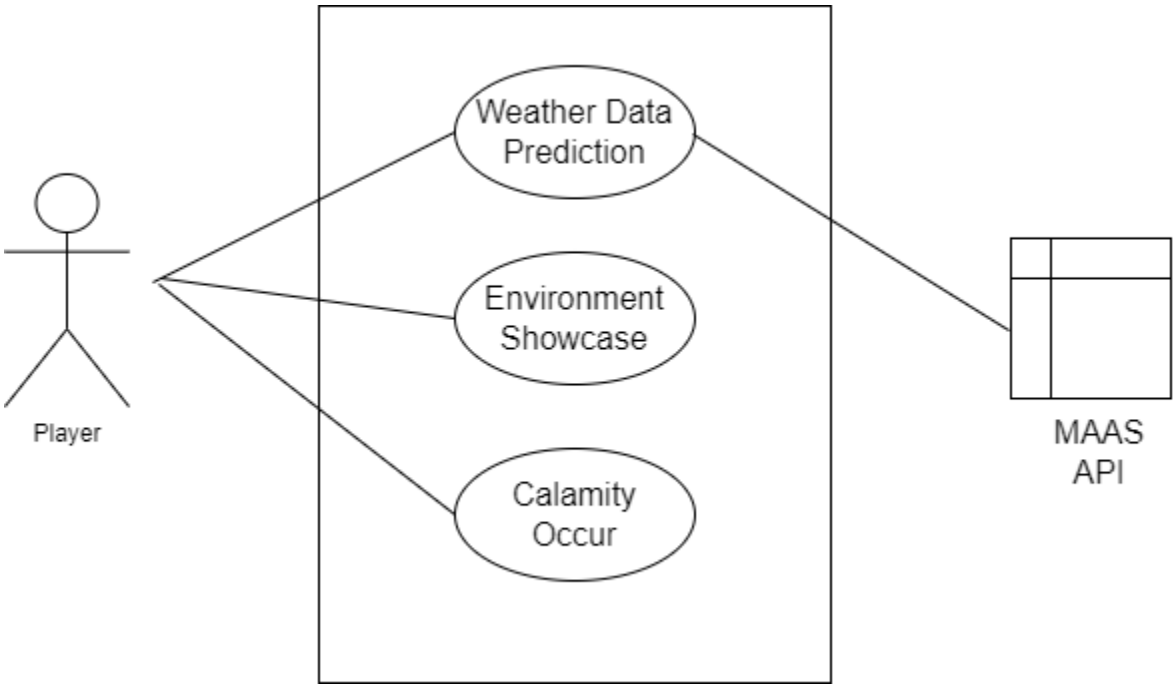


Figure 1.2: Weather Handling

Description:

Weather Data Prediction: After a player enters a level, the live weather data will be fetched from MAAS API. As those data are basically 4 days prior to actual weather of Mars. So, an Algorithm is there to predict 4 days of posterior weather data.

Environment showcase: The predicted data will be shown as the current weather in the game.

Calamity Occurrence: There will be in-game dust storms and asteroid falls. During this time, the player will have to go hide out in order to survive.

Action and Reply:

Action: Player enters a region

Reply: System calls MAAPS Api and predicted real time weather data will be fetched and shown as an environment.

Action: Player plays a level for a designated time period

Reply: Dust storm or Asteroid falls happen.

Use Case ID: 1.3

Name: Puzzle Solving

Primary Actor: Player,

Secondary Actor: Database

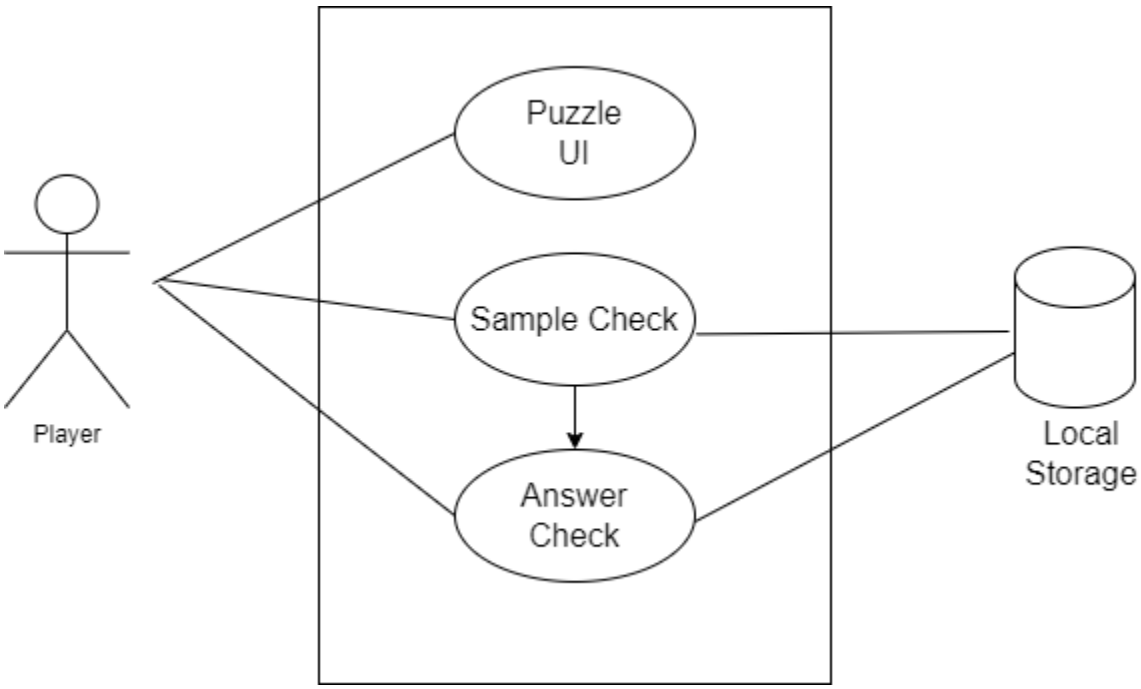


Figure 1.3 : Puzzle Solving

Description:

Puzzle UI: There will be an in Level Puzzle button, if players press it, puzzle data will be fetched from the database and shown in a Puzzle Box. Puzzles will be multiple choice questions.

Sample Check: Before clicking any options for an object in a puzzle, it will be checked if the item is present in inventory. If not, the player can't choose that answer.

Answer Check: The write answer will be checked, and after completing all the puzzle, level ends.

Action and Reply:

Action: Player clicks puzzle button

Reply: Puzzle UI Shown, and ask for puzzles answer.

Action: Players select answers based on their object analysis.

Reply: System will check if the player has collected the sample of the answer. If yes, the answer will be evaluated. If not, UI will ask for the sample to collect.

Action: Player solved all the puzzles

Reply: Level up.

Use Case ID: 1.4

Name: Inventory Handling

Primary Actor: Player,

Secondary Actor: Database

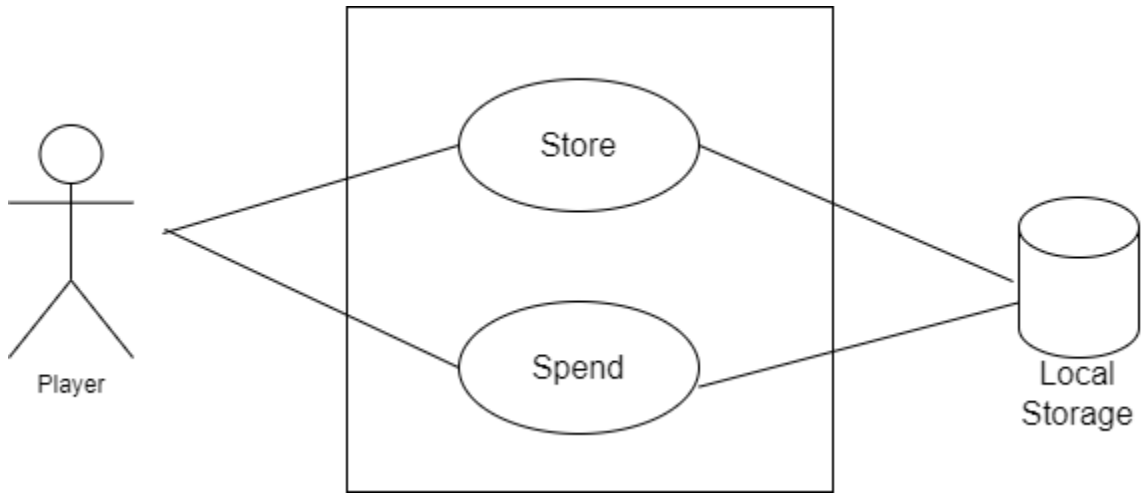


Figure 1.4 : Inventory Handling.

Description:

Store: When a player collects an object, the object will be stored in the inventory and show in inventory ui.

Send: In store, players can spend collected Basalts to purchase items. The corresponding amount will be deducted from the total Basalts amount.

Action and Reply:

Action: Player hits an object to collect the object.

Reply: The object will be stored in the inventory with the same type of objects as a total amount.

Action: Players purchase an item from the store.

Reply: Equivalent amount of Basalts will be deducted from the inventory for purchase.

Use Case ID: 1.5

Name: Store & PowerUps

Primary Actor: Player,

Secondary Actor: Database

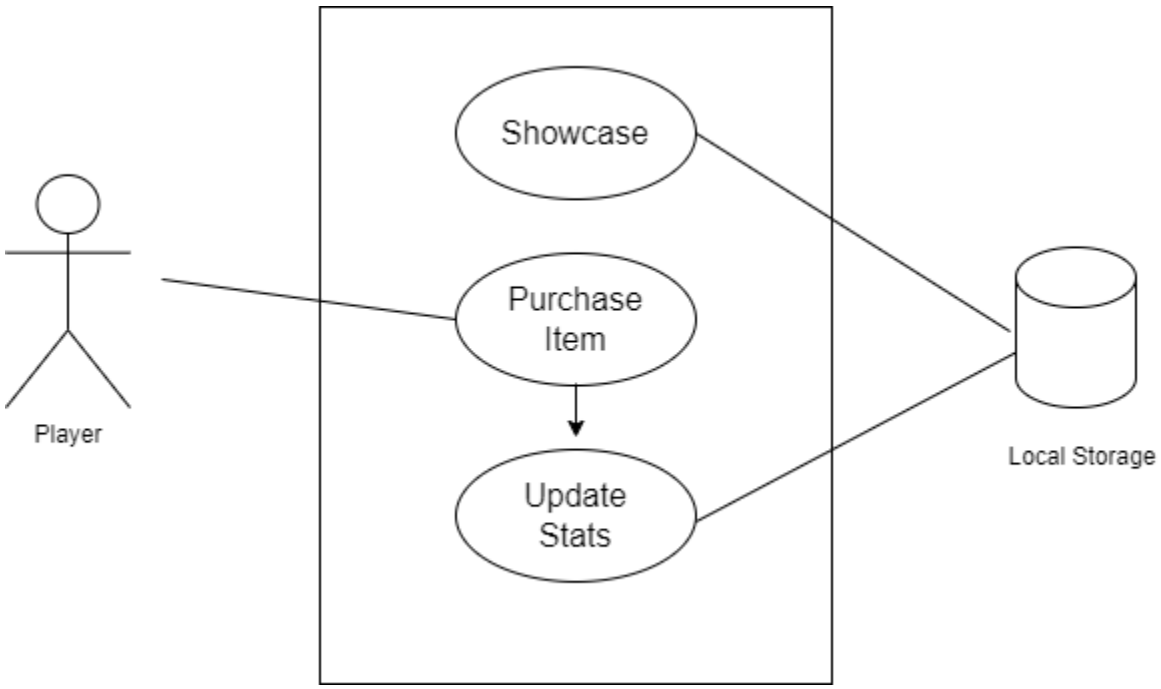


Figure 1.5 : Storage and Power System

Description:

Showcase: When a player clicks the store icon, a store UI will be shown. Store will have two categories, shovels and suits. Each item will have an icon, feature, price.

Purchase Item: Player can make a purchase of any item of the store with required Basalts as price.

Update Stats: After making a ;purchase of an item from the store, the item will be stored in the player's inventory. Players can also equip these items directly from the store. Once equipped, player ability will be updated, according to the feature of the item.

Action and Reply:

Action: Player clicks store icon on gaming screen.

Reply: Store UI will open with two categories of items, suits and shovels.

Action: Players purchase an item from the store.

Reply: Items get stored in the player's inventory with an option to equip immediately.

Action: Players click the equip button of an item.

Reply: Player status will be updated according to item features.

4.1.3 Activity Diagram

Activity ID: 1

Name: Mars Exploration Game Details

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage,

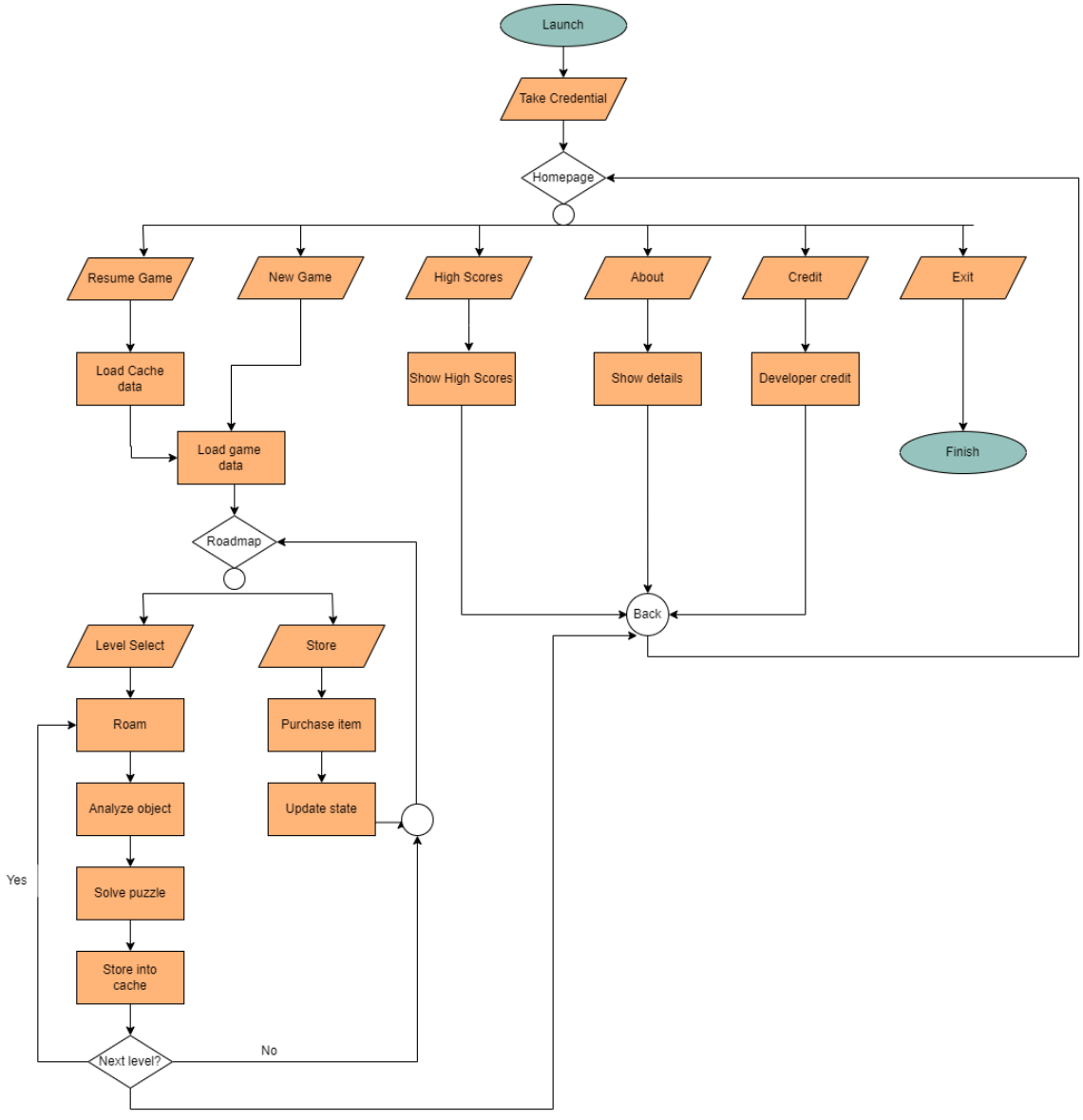


Figure 2 : Mars Exploration Game Details

Activity ID: 1.1.1

Name: Entry phase

Primary Actor: Player,

Secondary Actor: MAAS API, Local storage,

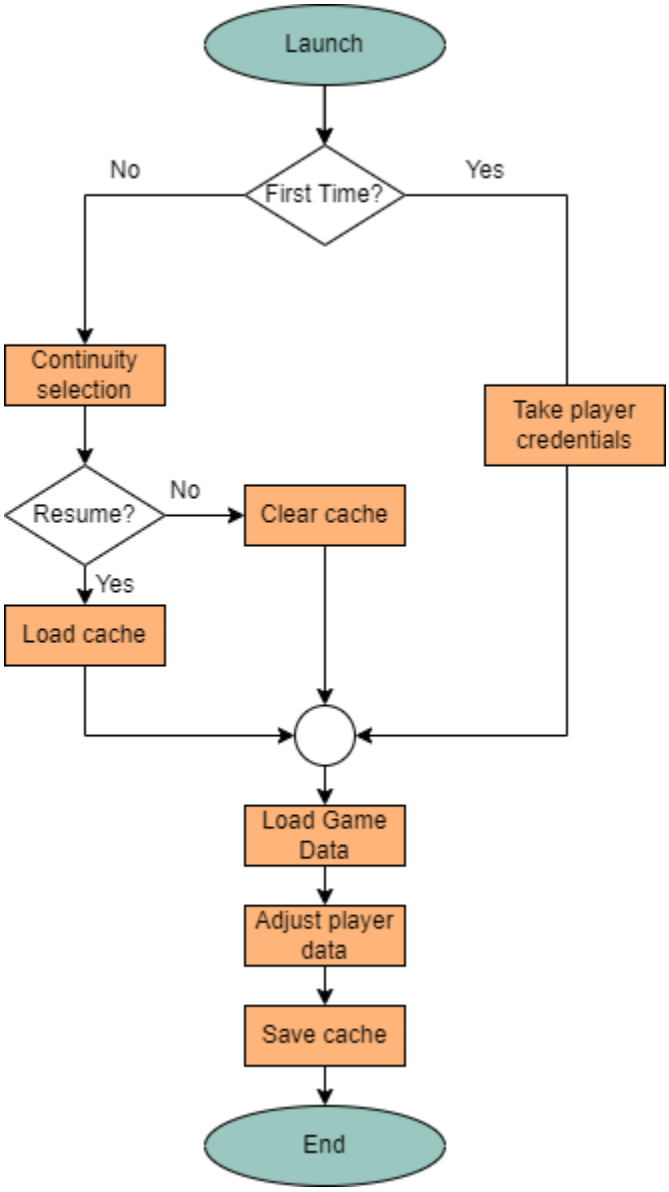


Figure 2.1.1 : Entry phase

Activity ID: 1.1.2

Name: Gameplay phase

Primary Actor: Player,

Secondary Actor: Local storage,

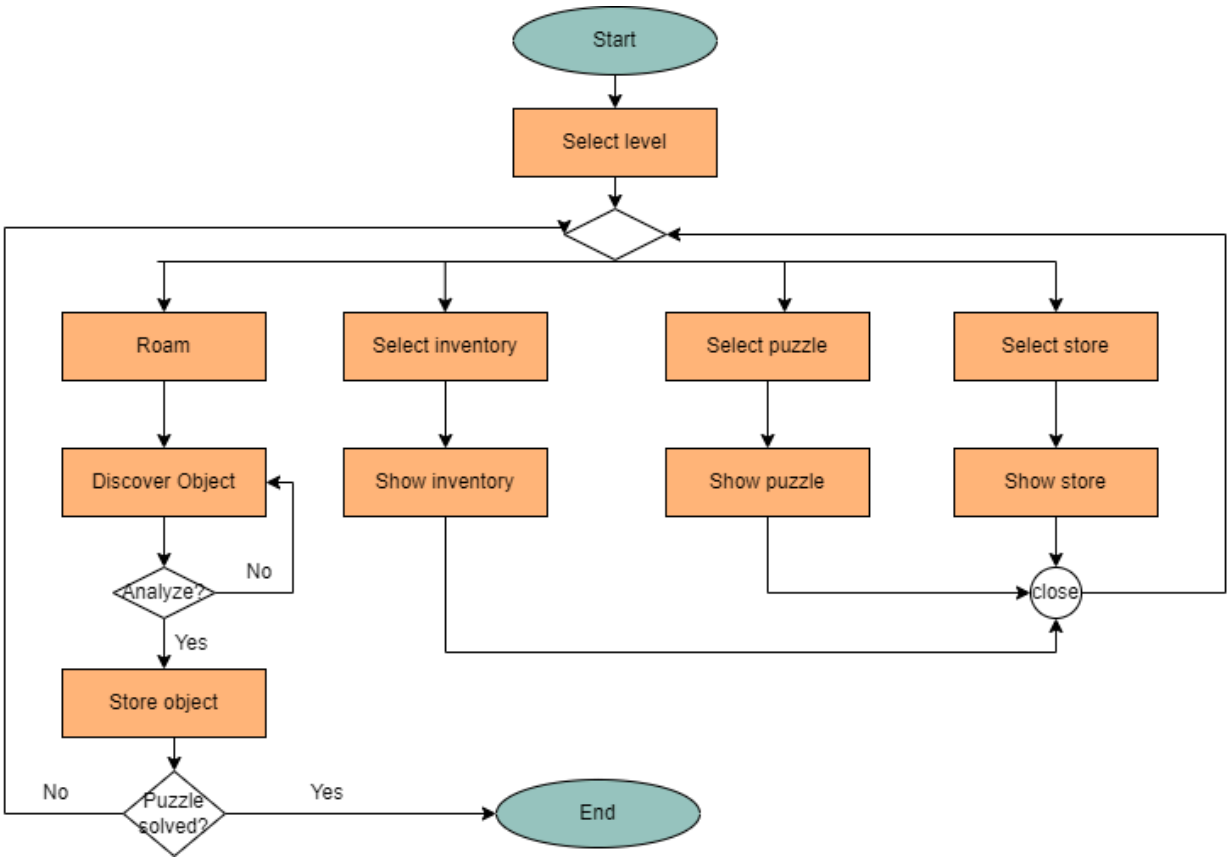


Figure 2.1.2 : GamePlay

Activity ID: 1.2

Name: Weather Handler

Primary Actor: Player,

Secondary Actor: Local storage, MAAS API

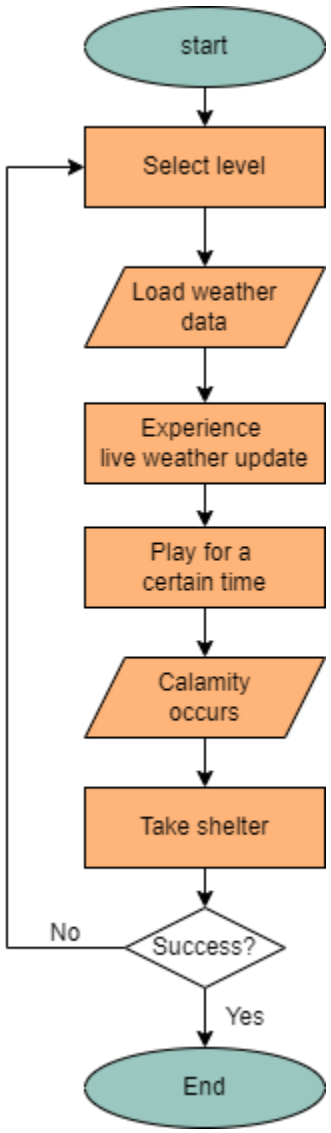


Figure 2.2: Weather Handling

Activity ID: 1.3

Name: Puzzle Solving

Primary Actor: Player, system

Secondary Actor: Local storage

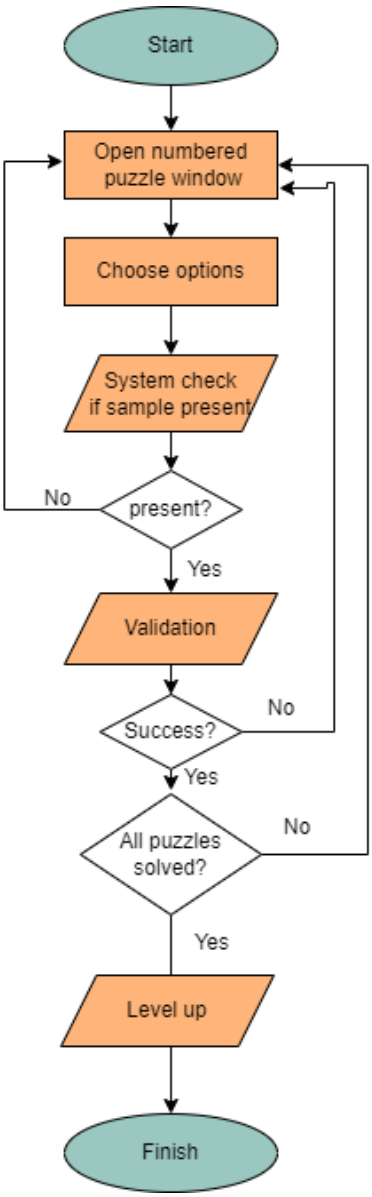


Figure 2.3: Puzzle Solving

Activity ID: 1.4

Name: Inventory Management

Primary Actor: Player, system

Secondary Actor: Local storage

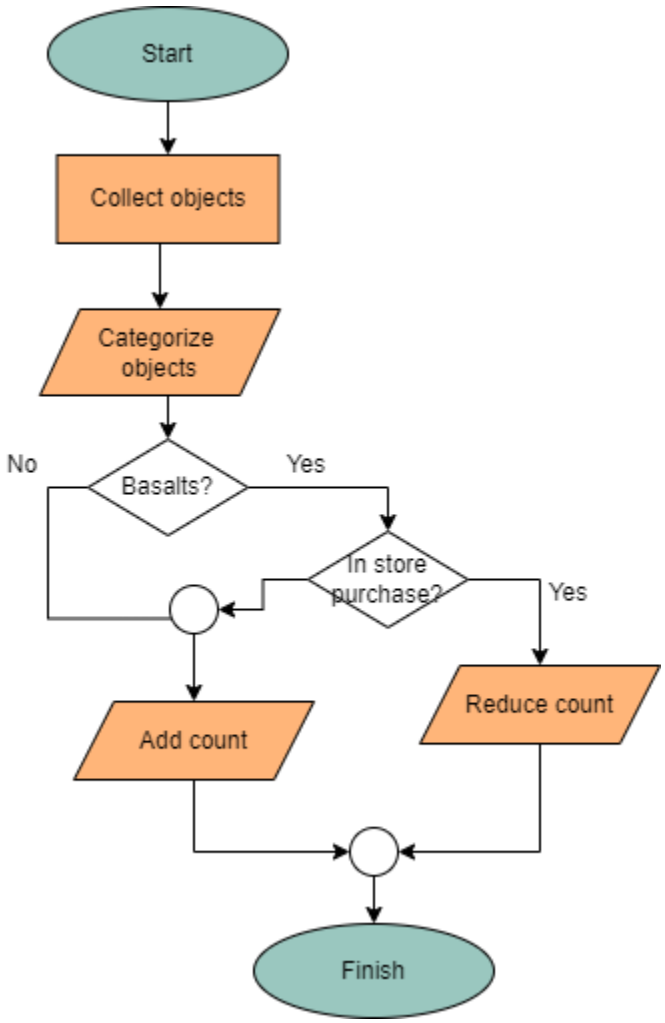


Figure 2.4: Inventory Management

Activity ID: 1.5

Name: Shop & Power Ups

Primary Actor: Player, system

Secondary Actor: Local storage

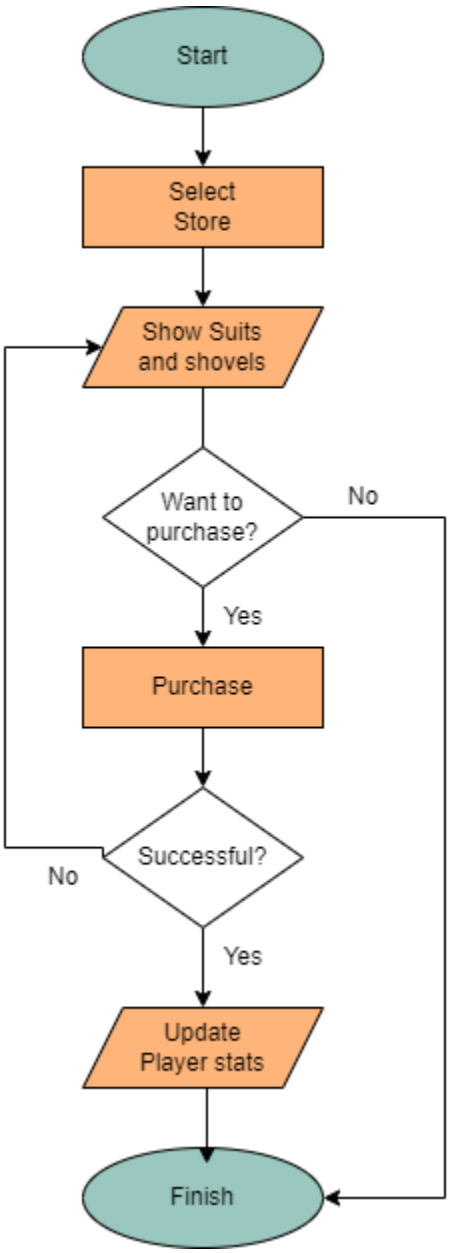


Figure 2.5 : Store and PowerUp

4.1.4 SwimLane Diagram

Swimlane ID: 1.1.1

Name: Entry Phase

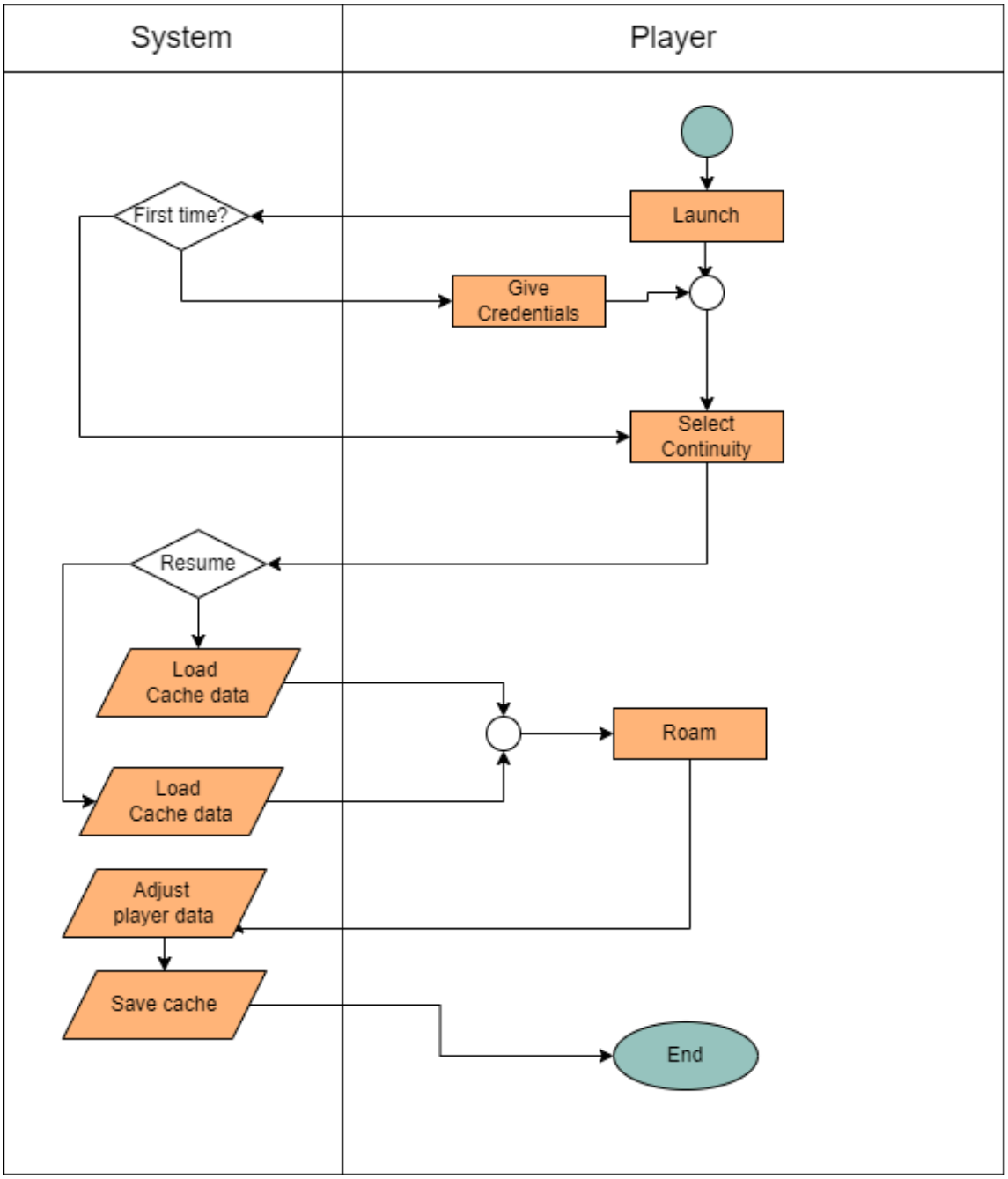


Figure 3.1.1 : Entry System

Swimlane ID: 1.1.2

Name: Gameplay Phase

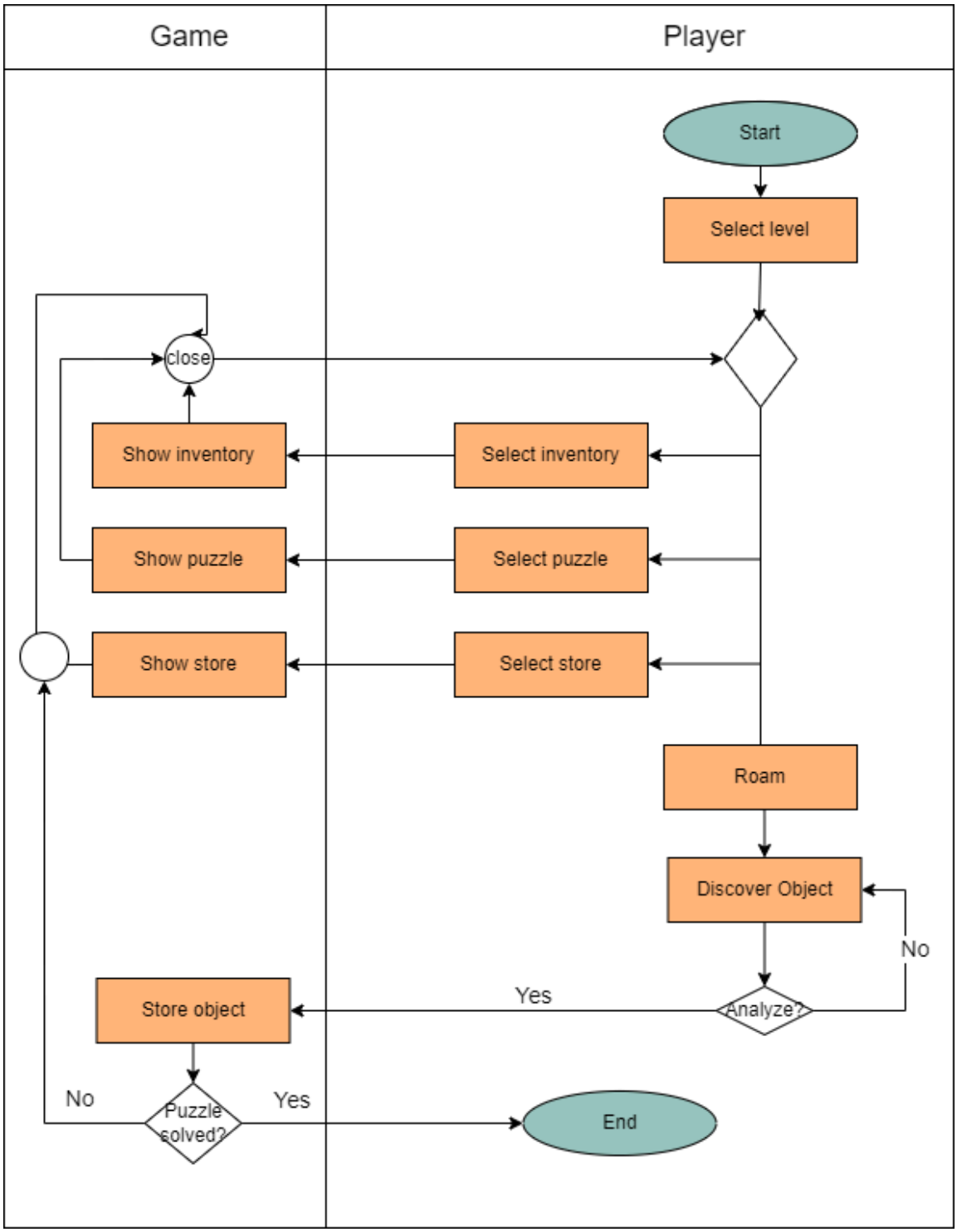


Figure 3.1.1 : Gameplay System

Swimlane ID: 1.2

Name: Weather Handler

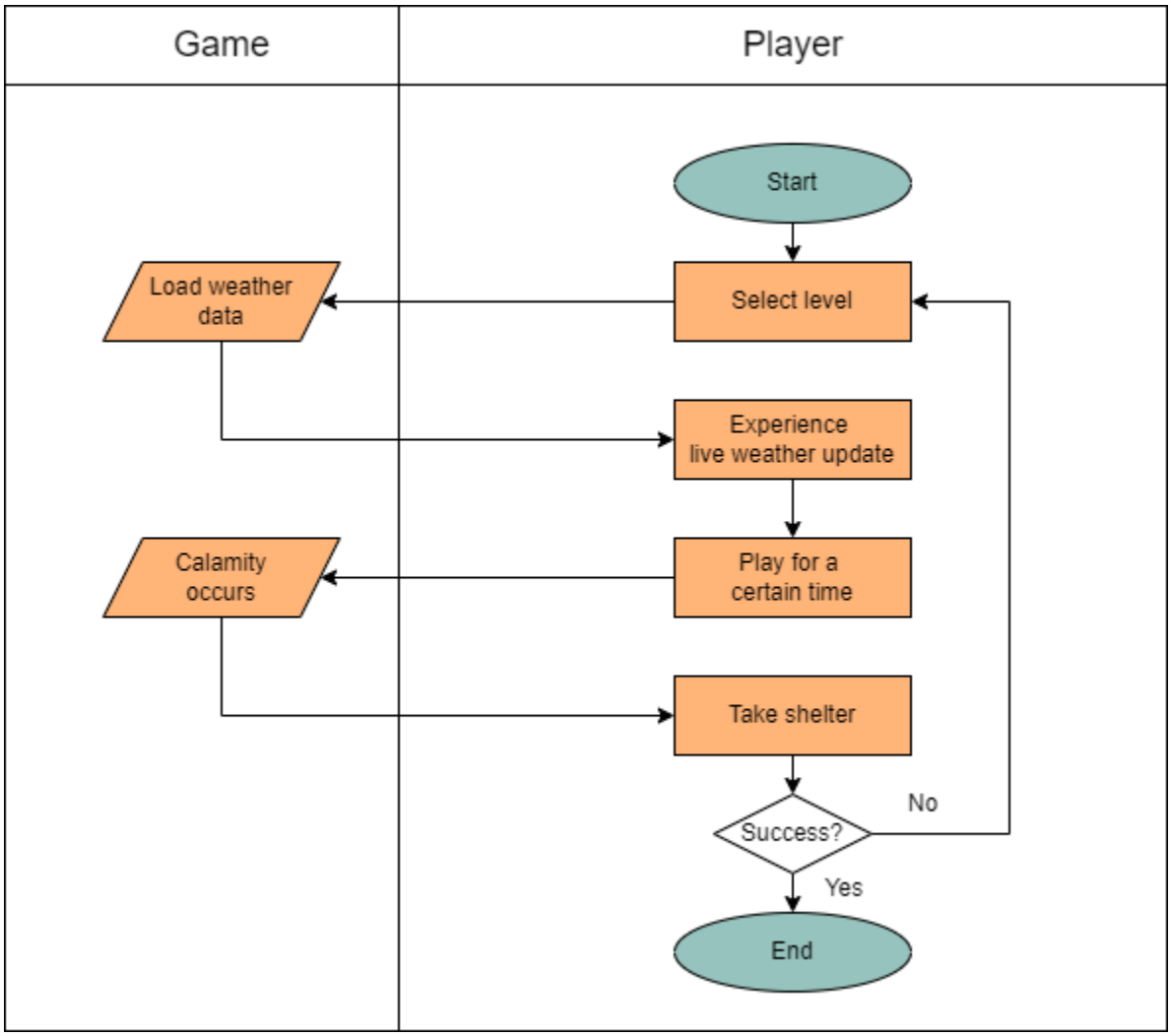


Figure 3.2 : Weather Handler

Swimlane ID: 1.3

Name:PuzzleSolving

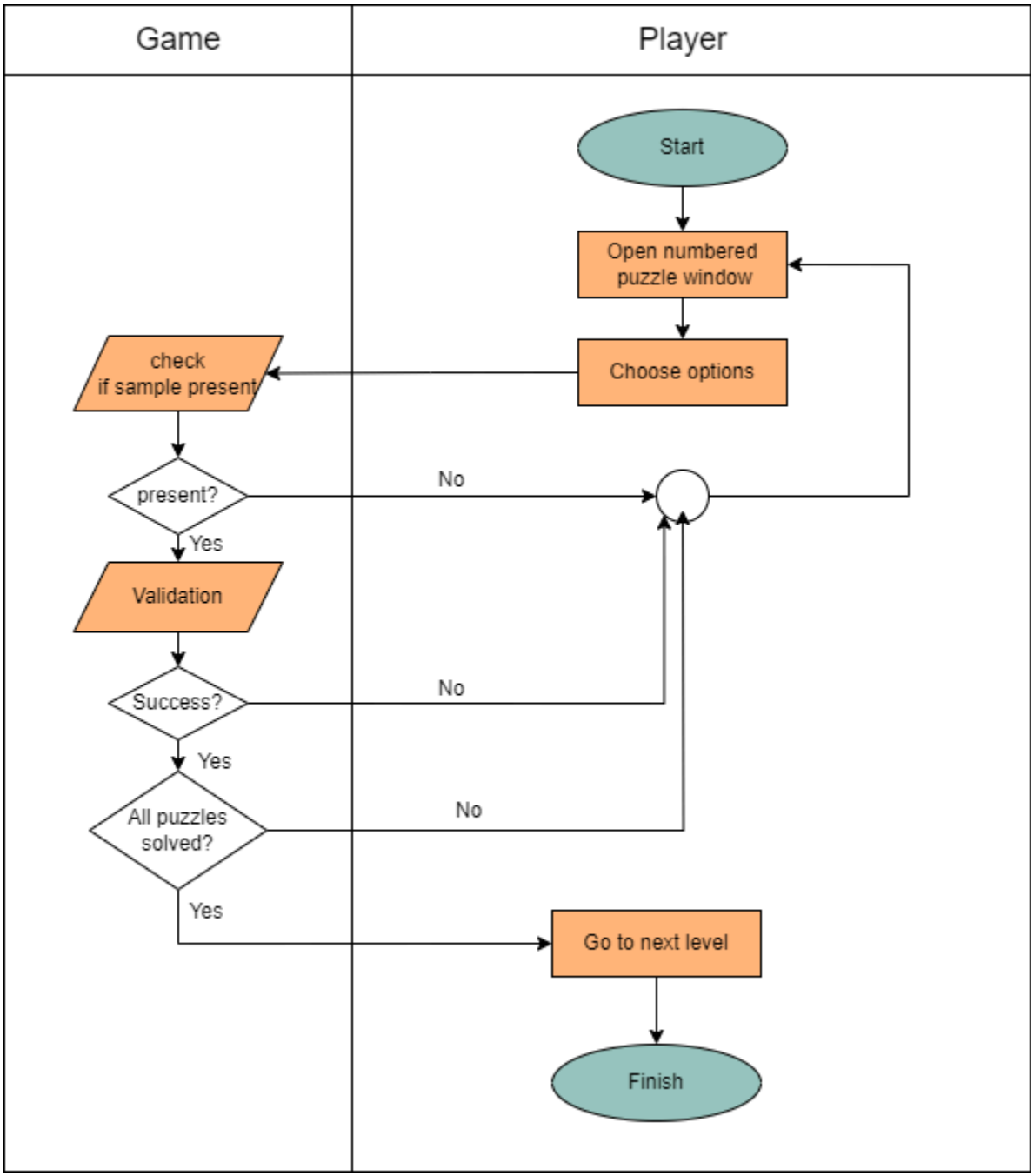


Figure 3.3 : Puzzle Solving

Swimlane ID: 1.4

Name: Inventory Management

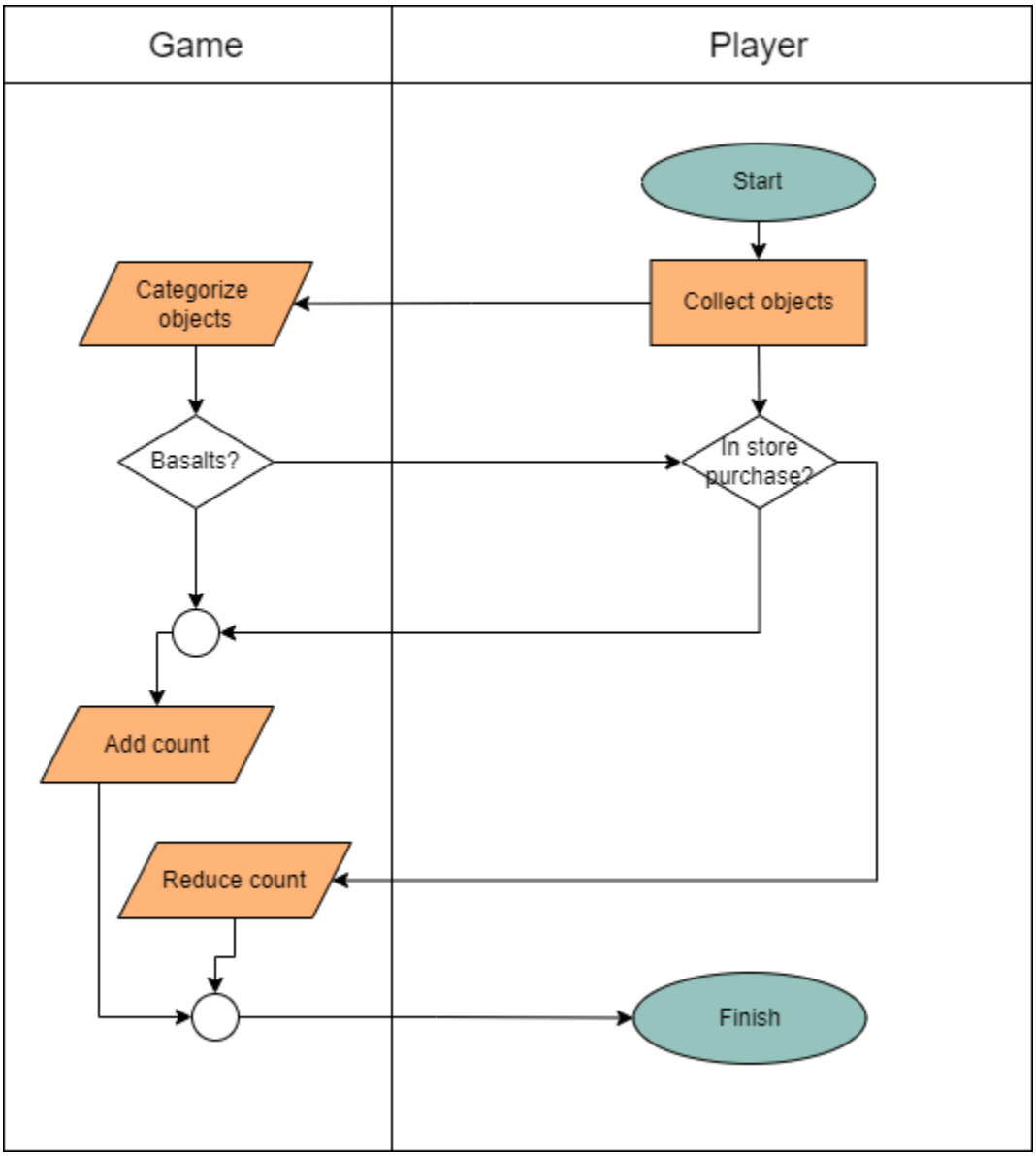


Figure 3.4 : Inventory Management

Swimlane ID: 1.5

Name: Shop & Power Ups

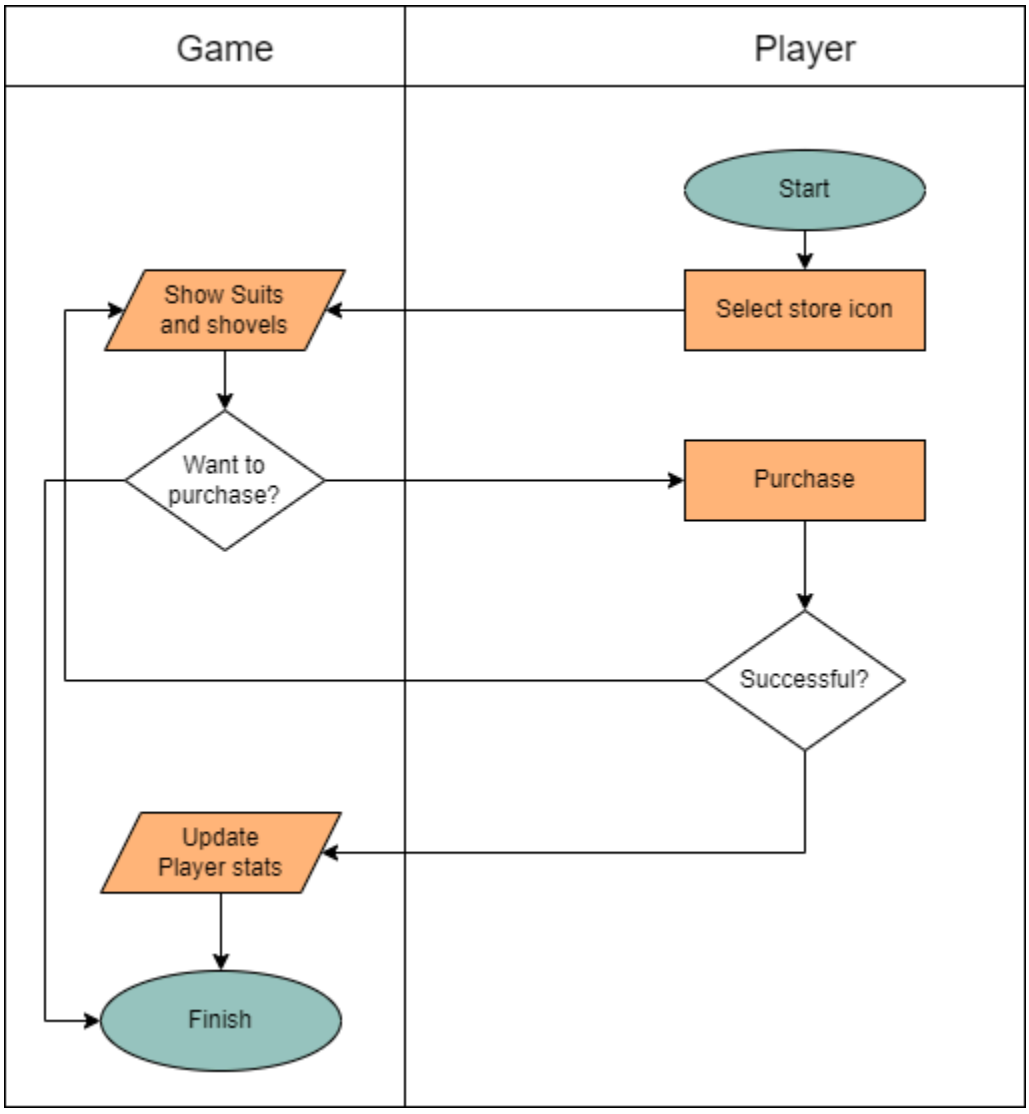


Figure 3.5 : Store and PowerUp

4.2 DATA BASED MODELING

This chapter describes the Data Based Model for “MEG”.

4.2.1. Introduction

Data modeling, sometimes also called information modeling, is the process of visually representing what data the application or system will use and how it will flow. The resulting diagram or other visual representation is meant to be designed in a way that is as easy to understand as possible. The fundamental elements that a data model needs to include and describe are the data objects, more frequently called "entities"; the attributes of those objects or entities; and the relationships between the objects or entities.

4.2.2. Data Object

A data object is a region of storage that contains a value or group of values. Each value can be accessed using its identifier or a more complex expression that refers to the object. In addition, each object has a unique data type. The data type of an object determines the storage allocation for that object and the interpretation of the values during subsequent access. It is also used in any type checking operations. Both the identifier and data type of an object are established in the object declaration.

4.2.3 Analysis

1. **Player:**
    - **Attributes:** SessionID, Name
    - **Reasoning:** The Player entity represents individual players participating in the game. SessionID uniquely identifies each player session, while Name identifies the player by their chosen name. This entity is essential for tracking player progress and maintaining a personalized gaming experience.
2. **Level:**
    - **Attributes:** ID, Region, TimeLimit, DustStormSpan, MeteoriteFallSpan
    - **Reasoning:** The Level entity represents the different levels available for exploration in the game. Each level has unique attributes such as its ID, region setting, and various parameters like time limit, dust storm duration, and meteorite fall duration. These attributes are crucial for defining the characteristics and challenges of each level.
3. **RegionalObject:**
    - **Attributes:** ID, Name, LevelID, Analysis
    - **Reasoning:** RegionalObject represents the objects or elements that players can discover within each level. Each RegionalObject has a unique ID and Name, along with an associated LevelID indicating the level in which it is found. The Analysis attribute stores information about the object, which serves as clues for solving puzzles. This entity facilitates tracking of discovered objects and their properties.

4. **Puzzle:**
    - **Attributes:** PuzzleID, Question
    - **Reasoning:** Puzzle entity represents the challenges or puzzles that players encounter within each level. Each puzzle has a unique ID and a corresponding question that players must solve to progress. This entity is essential for managing and tracking the various puzzles present in the game.
5. **Store:**
    - **Attributes:** ItemID, ItemName, Color, Speed, JumpForce, Price
    - **Reasoning:** The Store entity represents the in-game store where players can purchase items such as shovels and astronaut suits. While shovels and suits could be modeled as separate entities, it's beneficial to include them within the Store entity for several reasons:
        - **Centralized Management:** Having shovels and suits within the Store entity allows for centralized management of all purchasable items in the game.
        - **Consistency in Pricing:** By including shovels and suits in the Store entity, pricing consistency can be maintained across all items available for purchase.
        - **Simplified Data Management:** Combining shovels and suits within the Store entity simplifies data management and reduces the complexity of the overall data model.

- **Ease of Expansion:** If additional items are introduced into the game in the future, they can be seamlessly integrated into the existing Store entity without requiring significant modifications to the data model.

Table	Attributes
Player	SessionID, Name
Level	ID, Region, TimeLimit, DustStomSpan, MeteoriteFallSpan
RegionalObject	ID, Name, LevelD, Analysis
Puzzle	PuzzleID, Question
Store	ItemID, ItemName, Color, Speed , JumpForce, Price

Figure 4.1: Final Data Objects

4.2.4 ER Diagram

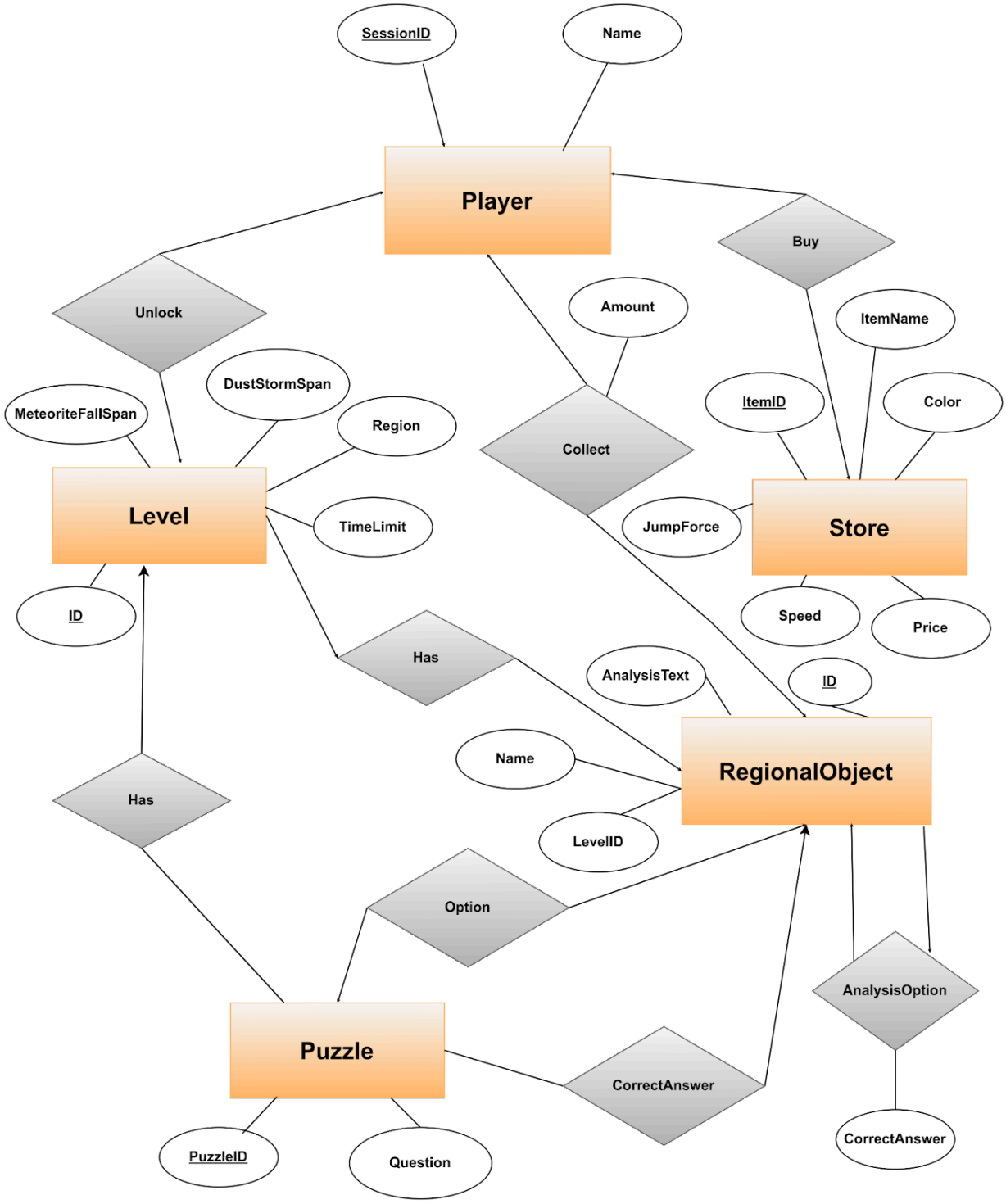


Figure 4.2 : ER diagram

4.2.5 Schema

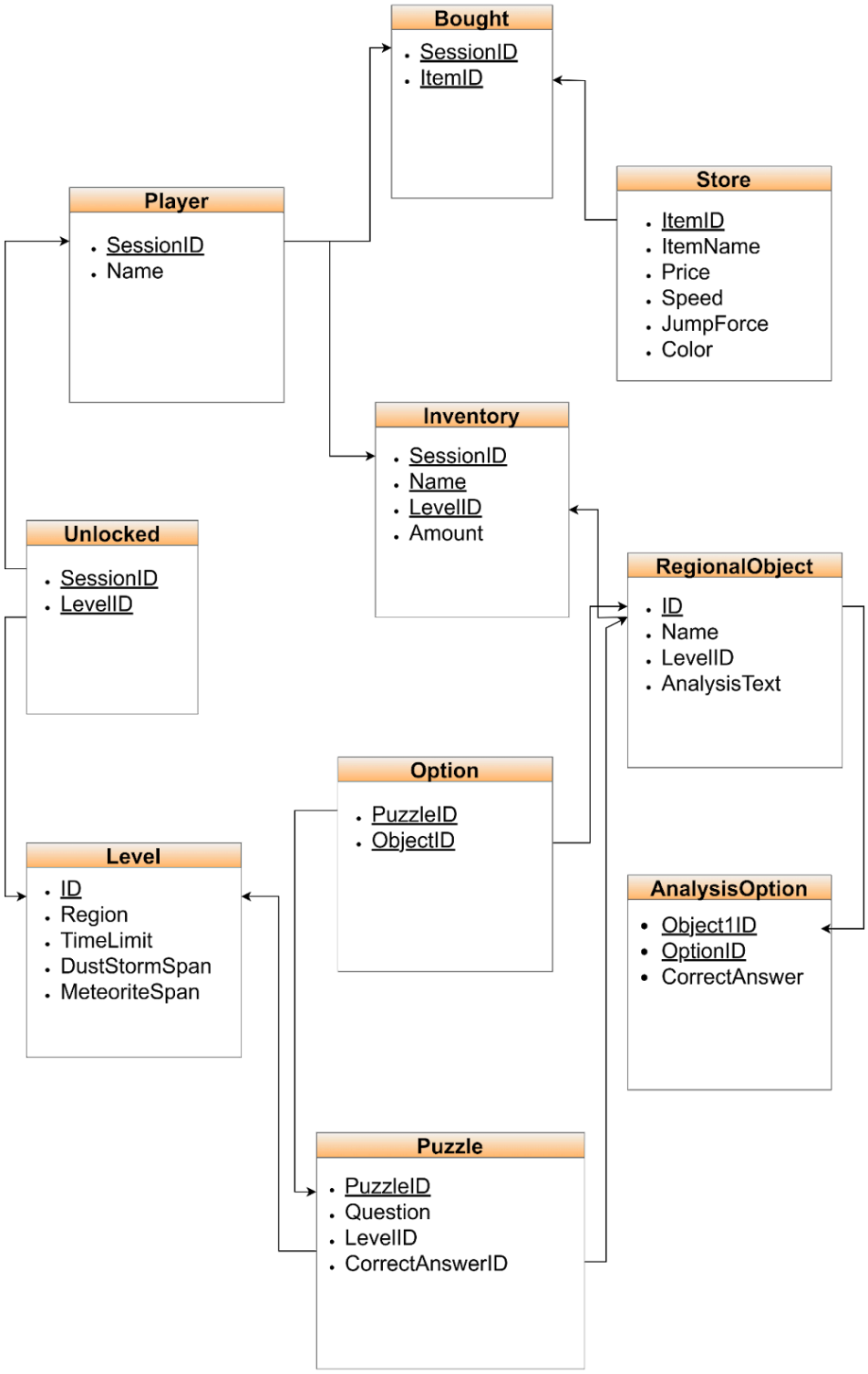


Figure 4.3 : Schema

4.3. CLASS BASED MODELING

This chapter describes the Class Based Model for “MEG”.

4.3.1. Introduction

Class-based modeling represents the objects that the system will manipulate, the operations (also called methods or services) that will be applied to the objects to effect the manipulation, relationships (some hierarchical) between the objects, and the collaborations that occur between the classes that are defined. The elements of a class-based model include classes and objects, attributes, operations, class-responsibility-collaborator (CRC)models, collaboration diagrams, and packages.

4.3.2 Class Analysis

In analyzing the Mars Explorer Game's requirements, its core activity is that players will embark on an adventure through simulated Martian terrain. They'll explore, collect resources, solve puzzles, and face challenges like dust storms and meteorite falls.

To make this happen, the game needs several key components. Firstly, there's the Player, who represents the user. This character keeps track of their progress, inventory, and interactions. Then, there's the RegionalObject class, which handles objects found in the game, like rocks and minerals.

Players will need tools like Shovels and Suits for exploration and resource gathering. These items can be bought from the Store, where players can spend their in-game currency. Puzzle challenges will be managed by a Puzzle class, with each level serving as a different environment to explore and solve puzzles in.

Dynamic events such as Dust Storms and Meteorite Falls will spice up gameplay. A WeatherForecast class could give players a heads-up about upcoming weather conditions. A Map will help players navigate levels, and a HomePage will act as the game's main hub.

These components form the backbone of the Mars Explorer Game. Following are further analyses of potential classes.

- The Player class encapsulates attributes and methods vital for managing the player's progress, inventory, interactions, and actions within the game. Its collaboration with various other classes such as Level, Inventory, Puzzle, Shovel, Suit, and Store ensures seamless gameplay and progression tracking.
- RegionalObject represents discoverable elements within levels and facilitates interactions between the player and the game environment. Its methods enable extraction, analysis, and management of collected samples, fostering engagement and exploration.
- The Shovel and Suit classes offer players equipment choices that influence gameplay mechanics. With methods for purchasing, equipping, and utilizing these items, players can customize their experience and overcome challenges effectively.
- Store acts as a centralized hub for purchasing items, including shovels and suits, streamlining the player's access to essential tools and enhancing immersion by mimicking real-world shopping experiences.

- Inventory provides a repository for collected objects, facilitating organization and management of player resources. Its collaboration with RegionalObject ensures seamless integration of collected samples into the player's arsenal.
- Puzzle and Level classes contribute to the game's narrative and progression by offering challenges and environments for exploration. Their methods enable interaction, completion tracking, and event handling, enriching the gameplay experience.
- DustStorm and MeteoriteFalling classes introduce dynamic environmental events, adding layers of challenge and realism to gameplay. Their methods manage event initiation, duration, and effects, enhancing immersion and strategic decision-making.
- WeatherForecast class leverages real-time data and machine learning to simulate Martian weather conditions, enriching the game world's authenticity and providing players with valuable information for planning and strategy.
- Map class serves as a visual interface for navigating the game world, allowing players to select levels and track their progress. Its methods enable level unlocking, selection, and display, enhancing player engagement and exploration.
- HomePage class serves as the entry point to the game, offering options for starting, resuming, or exiting gameplay, as well as accessing additional information such as high scores and credits. Its methods facilitate player interaction and navigation, ensuring a smooth and intuitive user experience.

4.3.3 Attribute and Method Identification

Class	Attribute	Method
Player	**+Name**: • **Description:** Stores the name of the player. • **Type:** String **+Progress**: • **Description:** Tracks the player's progress in the game, including unlocked levels. • **Type:** Dictionary or data structure to store level status. **+Inventory**: • **Description:** Stores the collected samples by the player. • **Type:** List or dictionary to manage collected items.	**+set_name(name: str) -> None**: • **Description:** Sets the name of the player. **+update_progress(level: int, status: str) -> None**: • **Description:** Updates the progress of the player in the game (unlocking levels). **+add_to_inventory(RegionalObject reference) -> None**: • **Description:** Adds a collected sample to the player's inventory. **+update_points(points: int)**

	+Points: ● **Description:** Tracks the points earned by the player. ● **Type:** Integer **+Currency**: ● **Description:** Tracks the amount of Basalt currency collected by the player. ● **Type:** Integer **+CurrentLevel**: ● **Description:** Keeps track of the current level the player is on. ● **Type:** Reference to level object. **+CurrentTime**: ● **Description:** Tracks the current time elapsed in the current level. ● **Type:** Integer or float (representing seconds or	**-> None**: ● **Description:** Updates the player's points based on certain actions/events. **+update_currency(amount: int) -> None**: ● **Description:** Updates the player's Basalt currency based on collection or transactions. **+start_level(level: reference) -> None**: ● **Description:** Initiates a new level for the player to play. **+complete_level(time_taken : int) -> None**: ● **Description:**

	minutes).	Handles actions when the player successfully completes a level within the time limit. **+encounter_failure() -> None**: ● **Description:** Handles actions when the player fails to complete a level due to dust storms or meteoroids. **+attempt_puzzle(puzzle:reference, answer: str) -> str**: ● **Description:** Handles the player's attempt to solve a puzzle and returns feedback (correct or incorrect). **+buy_item(item: reference)**

		-> None: ● **Description:** Allows the player to buy items from the store using Basalt currency. **+show_inventory() -> None**: ● **Description:** Displays the player's inventory with collected samples/items. **+handle_time_limit() -> None**: ● **Description:** Manages actions when the player exceeds the time limit for a level. **+handle_menu_option(option: str) -> None**:

		• **Description:** Handles the player's selection from the main menu. **+choose_shovel(shovel_type : str) -> None**: • Description: Allows the player to select a shovel from the inventory. **+use_shovel(object_name: reference) -> None** • **Description:** Initiates the extraction process for a given object using the selected shovel. **+choose_suit(suit_type: str) -> None:** • **Description:** Allows

		the player to select a suit from the inventory. **+use_suit(suit: reference) -> None:** ● **Description:** Utilizes the benefits of the currently equipped suit during gameplay.
RegionalObject	**+Name**: ● **Description:** Stores the name of the object. ● **Type:** String **+Level**: ● **Description:** Indicates the level in which the object is found. ● **Type:** Integer or reference	**+get_name() -> str**: ● **Description:** Retrieves the name of the object. **+get_level() -> int**: ● **Description:** Retrieves the level in which the object is found.

	to level object. **+ExtractionTime**: ● **Description:** Represents the time required to extract information from the object. ● **Type:** Integer or float (representing seconds or minutes). **+Type**: ● **Description:** Indicates the type/category of the object (e.g., Basalt, Shergottites, etc.). ● **Type:** String or enumeration representing object types. **+AmountCollected** ● **Description:** Number of the object collected by the player. ● **Type**: Integer	**+get_extraction_time() -> int**: ● **Description:** Retrieves the extraction time required for the object. **+get_type() -> str**: ● **Description:** Retrieves the type/category of the object. **+set_name(name: str) -> None**: ● **Description:** Sets the name of the object. **+set_level(level: int) -> None**: ● **Description:** Sets the level in which the object is found.

		+set_extraction_time(time: int) -> None: ● **Description:** Sets the extraction time required for the object. **+set_type(object_type: str) -> None**: ● **Description:** Sets the type/category of the object. **+getSample()->reference to itself:** ● **Description:** After extraction, returns reference to itself as a sample to be added to the inventory. **+analyze() -> str**: ● **Description:** Initiates the analysis

		process for the object and returns the analysis results for the level it is in. **+interact() -> None**: ● **Description:** Represents the interaction of the player with the object, triggering extraction and analysis processes. **+increase_count()->None** ● **Description:** Increase value of the amount of object collected by the player while interaction.
Shovel	**+Color**: ● **Description:** Represents the color of the shovel.	**+get_color() -> str**: ● **Description:** Retrieves the color of

	• **Type:** String **+Speed**: • **Description:** Indicates the extraction speed multiplier provided by the shovel. • **Type:** Integer or float **+Price**: • **Description:** Represents the price of the shovel in Basalt currency. • **Type:** Integer	the shovel. **+get_speed() -> int**: • **Description:** Retrieves the extraction speed multiplier provided by the shovel. **+get_price() -> int**: • **Description:** Retrieves the price of the shovel. **+set_color(color: str) -> None**: • **Description:** Sets the color of the shovel. **+set_speed(speed: int) -> None**: • **Description:** Sets the extraction speed multiplier provided

		by the shovel. **+set_price(price: int) -> None**: ● **Description:** Sets the price of the shovel. **+buy_shovel(player_currency: int) -> bool**: ● **Description:** Allows the player to purchase the shovel if they have enough Basalt currency. Returns True if purchase is successful, False otherwise. **+use() -> None**:

		• **Description:** Allows the player to equip and use the shovel during gameplay, affecting the extraction speed of objects. **+get_shovel()->reference:** • **Description :** Returns reference to itself to be added to the store.
Suit	**+Type**: • **Description:** Indicates the type or level of the suit. • **Type:** String or enumeration representing suit types. **+Color**: • **Description:** Represents	**+get_type() -> str**: • **Description:** Retrieves the type of the suit. **+get_color() -> str**: • **Description:** Retrieves the color of the suit.

	the color of the suit. • **Type:** String or enumeration representing suit colors. **+SpeedMultiplier**: • **Description:** Represents the speed multiplier for the player while wearing the suit. • **Type:** Integer or float representing speed multiplier. **+JumpForce**: • **Description:** Represents the force or height multiplier for the player's jumps while wearing the suit. • **Type:** Integer or float representing jump force multiplier. **+Price**:	**+get_speed_multiplier() -> int**: • **Description:** Retrieves the speed multiplier of the suit. **+get_jump_force() -> int**: • **Description:** Retrieves the jump force multiplier of the suit. **+get_price() -> int**: • **Description:** Retrieves the price of the suit. **+use_suit() -> None**: • **Description:** Equips the suit for the player to use during gameplay, enhancing their speed and jump abilities.

	• **Description:** Represents the price of the suit in Basalt currency. • **Type:** Integer	**+buy_suit(player_currency: int) -> bool**: • **Description:** Allows the player to purchase the suit if they have enough Basalt currency. Returns True if purchase is successful, False otherwise. **+get_suit()->reference:** • **Description :** Returns reference to itself to be added to the store.
Store	**+AvailableSuits**: • **Description:** A list or dictionary containing	**+add_suit(suit: Suit) -> None**: • **Description:** Adds a

	references to available suits in the store. • **Type:** List or dictionary of Suit objects. **+AvailableShovels**: • **Description:** A list or dictionary containing references to available shovels in the store. • **Type:** List or dictionary of Shovel objects.	new suit to the available suits in the store. **+add_shovel(shovel: Shovel) -> None**: • **Description:** Adds a new shovel to the available shovels in the store. **+display_available_suits() -> None**: • **Description:** Displays the available suits for purchase in the store, showing details such as type, color, speed multiplier, jump force, and price. **+display_available_shovels( ) -> None**: • **Description:**

		Displays the available shovels for purchase in the store, showing details such as type, color, speed multiplier, and price. **+buy_item(item_name: str, player_currency: int) -> bool**: ● **Description:** Allows the player to purchase an item from the store by providing the item's name and the player's available currency. Returns True if the purchase is successful, False otherwise. **+handle_menu_option(option: str) -> None**: ● **Description:**

		Handles the player's selection from the store menu, allowing them to view available items and make purchases.
Inventory	**+CollectedObjects**: ● **Description:** A list or dictionary containing references to the regional objects collected by the player. ● **Type:** List or dictionary of RegionalObject objects.	**+add_object(obj: RegionalObject) -> None**: ● **Description:** Adds a new regional object to the inventory. **+remove_object(obj: RegionalObject) -> None**: ● **Description:** Removes a regional object from the inventory. **+display_inventory() -> None**:

		Description: Displays the collected regional objects in the inventory.
Puzzle	**+Level**: ● **Description:** The level in which the puzzle is present. ● **Type:** Integer **+Region**: ● **Description:** The region where the puzzle is located. ● **Type:** String **+PuzzleText**: ● **Description:** The text describing the puzzle. ● **Type:** String **+PossibleAnswers**: ● **Description:** The possible answers to the puzzle. ● **Type:** List of strings	**+display_puzzle() -> None**: ● **Description:** Displays the puzzle text to the player. **+check_answer(player_answer: str) -> str**: ● **Description:** Checks if the player's answer matches. Returns a message indicating whether the answer is correct or not. **+check_inventory(inventory: Inventory) -> bool**: ● **Description:** Checks if the player has collected all the

	+CorrectAnswer: • **Description:** The correct answer to the puzzle. • **Type:** String **+Completed**: • **Description:** Indicates whether the puzzle has been solved. • **Type:** Boolean	required samples for solving the puzzle by accessing the inventory. **+is_completed() -> bool**: • **Description:** Checks if the puzzle has been solved.Returns true if the puzzle is solved, False otherwise. **+disable_puzzle() -> None**: • **Description:** Disables the puzzle icon once it's solved. **+get_level() -> int/reference**: • **Description:** Retrieves the level of the puzzle.

		+get_region() -> str: ● **Description:** Retrieves the region where the puzzle is located. **+get_puzzle()->reference:** ● **Description:** Returns reference of itself.
Level	**+LevelNumber**: ● **Description:** The number or identifier of the level. ● **Type:** Integer **+Region**: ● **Description:** The region where the level is located. ● **Type:** String **+TimeLimit**: ● **Description:** The time	**+unlock_level() -> None**: ● **Description:** Unlocks the level for the player. **+complete_level() -> None**: ● **Description:** Marks the level as completed. **+add_puzzle(puzzle: Puzzle) -> None**: ● **Description:** Adds a

	limit for completing the level. • **Type:** Integer (seconds) **+AvailablePuzzles**: • **Description:** List of puzzles available in the level. • **Type:** List of Puzzle objects **+Completed**: • **Description:** Indicates whether the level has been completed. • **Type:** Boolean **+Unlocked**: **Description:** Indicates whether the level is unlocked for the player. **Type:** Boolean **+HideoutLocation**: • **Description:** The location	puzzle to the level. **+remove_puzzle(puzzle: Puzzle) -> None**: • **Description:** Removes a puzzle from the level. **+start_dust_storm() -> None**: • **Description:** Initiates a dust storm event in the level. **+stop_dust_storm() -> None**: • **Description:** Stops the dust storm event in the level. **+start_meteoroids_falling()**

	where the player starts and can return to for safety. • **Type:** String **+MeteoroidsFalling**: **Description:** Indicates whether meteoroids are falling in the level and if falling then for how many seconds. **Type:** Integer(Seconds) **+DustStorm**: **Description:** Indicates whether a dust storm is occurring in the level and if occurring then for how many seconds. **Type:** Integer(Seconds)	**-> None**: • **Description:** Initiates meteoroids falling event in the level. **+stop_meteoroids_falling() -> None**: • **Description:** Stops the meteoroids falling event in the level. **+get_time_limit() -> int**: • **Description:** Retrieves the time limit for completing the level. **+is_completed() -> bool**: • **Description:** Checks if the level has been completed.

		+is_unlocked() -> bool: ● **Description:** Checks if the level is unlocked for the player. **+calculate_points(seconds_saved: int) -> int**: ● **Description:** Calculates the points earned by the player for completing the level. **+show_weather_forecast()->None:** ● **Description:** Shows realtime martian weather data on display.
DustStorm	**+Duration**: ● **Description:** The duration of the dust storm event.	**+start() -> None**: ● **Description:** Starts the dust storm event.

	• **Type:** Integer (seconds) +**Severity**: • **Description:** The severity level of the dust storm. • **Type:** String +**AffectedArea**: • **Description:** The area affected by the dust storm. • **Type:** String +**VisibilityReduction**: • **Description:** The extent to which visibility is reduced during the dust storm. • **Type:** Float (percentage)	+**end() -> None**: • **Description:** Ends the dust storm event. +**get_duration() -> int**: • **Description:** Retrieves the duration of the dust storm. +**get_severity() -> str**: • **Description:** Retrieves the severity level of the dust storm. +**get_affected_area() -> str**: • **Description**: Retrieves the area affected by the dust storm.

		+get_visibility_reduction() -> float: • **Description:** Retrieves the extent to which visibility is reduced during the dust storm.
MeteoriteFalling	**+Duration**: • **Description:** The duration of the meteorite falling event. • **Type:** Integer (seconds) **+Impact_intensity**: • **Description:** The intensity of the meteorite impacts. • **Type:** String **+Affected_area**: • **Description:** The area affected by the meteorite falling.	**+start() -> None**: **Description:** Starts the meteorite falling event. **+end() -> None**: **Description:** Ends the meteorite falling event. **+get_duration() -> int**: **Description:** Retrieves the duration of the meteorite falling event. **+get_impact_intensity() -> str**:

	• **Type:** String	**Description:** Retrieves the intensity of the meteorite impacts. +**get_affected_area() -> str**: **Description:** Retrieves the area affected by the meteorite falling.
WeatherForecast	+**Date**: • **Description:** The date for which the weather forecast is applicable. • **Type:** Date +**MaxTemperature**: • **Description:** The maximum temperature forecasted. • **Type:** Float	+**fetch_weather_data_from _api** (): • **Returns**: Dictionary containing weather data fetched from the MAAS API • **Description:** Fetches weather data from the MAAS API for a specific date.

	+MinTemperature: ● **Description:** The minimum temperature forecasted. ● **Type:** Float **+UvIndex**: ● **Description:** The UV index forecasted. ● **Type:** Integer **+AtmosphericPressure**: ● **Description:** The atmospheric pressure forecasted. ● **Type:** Float **+Model** ● **Description:**Represents the machine learning model used for weather prediction. **+MaasAPI** ● **Description :** API key of	+**get_weather_forecast(date)**: ● **Returns:** Dictionary containing weather forecast information ● **Description:** Retrieves weather forecast for a specific date by fetching data from the MAAS API and using the ML model to predict the weather. **+display_predicted_data** (PredictedData): ● **Description:** Displays the predicted weather data obtained from

	Curiosity rover to collect historical data of Martian Weather.	the ML model.
Map	**+Levels** ● **Description:** A list containing Level objects representing the different levels available for exploration on the map. ● **Type:** List of Level objects **+UnlockedLevels** ● **Description:** A list containing Level objects representing the levels that have been unlocked by the player. ● **Type:** List of Level objects	**+add_level(Level reference):** ● **Description:** Adds a new level to the map. **+unlock_level(Level reference):** ● **Description:** Unlocks a specific level on the map based on its index. **+display_map():** ● **Description:** Displays the map, highlighting the unlocked levels and indicating the locked ones. **+select_level(Level**

		reference): ● **Description:** Allows the player to select a level for exploration.
HomePage	**+Options** ● **Description:** Stores the available options on the homepage. ● **Type:** List of Strings **+Choice** ● **Description:** Chosen option ● **Type :** String	**+input_player_name() -> None:** ● **Description:** Prompts the player to input their name and stores it in the Player object's name attribute. **+chose_action(Choice)->** ● **Description :** Calls method according to the option chosen by the player. **+display_options() -> None:** ● **Description:** Displays the

		available options to the player on the homepage. **+resume_game() -> None:** ● **Description:** Allows the player to resume the game, directing them to the map screen where previously unlocked levels are visible. **+start_new_game() -> None:** ● **Description:** Initiates a new game, unfolding the map to reveal the levels available for exploration. Only the first level is initially accessible. **+view_high_scores() ->**

		None: ● **Description:** Allows the player to view the top scores achieved in the game. **+view_about() -> None:** ● **Description:** Displays information about the gameplay, providing details about the game's concept and mechanics. **+view_credits() -> None:** ● **Description:** Displays credits, acknowledging the creators of the game. **+exit_game() -> None:** ● **Description:** Allows the player to exit the

		game.

4.3.4 Class cards

Player		
Method	**Responsibility**	**Collaboration Class**
update_progress(level: int, status: str)	Updates the progress of the player in the game (unlocking levels).	Level
add_to_inventory(RegionalObject reference)	Adds collected samples to the player's inventory.	Inventory, RegionalObject
start_level(level: int)	Initiates a new level for the player to play.	Level, Map
complete_level(time_taken: int)	Handles actions when the player successfully completes a level within the time limit.	Level
encounter_failure()	Handles actions when the player fails to complete a level due to dust storms or meteoroids.	Level
attempt_puzzle(puzzle: reference, answer: str)	Handles the player's attempt to solve a puzzle and returns	Puzzle

	feedback (correct or incorrect).	
buy_item(item: reference)	Allows the player to buy items from the store using Basalt currency.	Store, Suit, Shovel
show_inventory()	Displays the player's inventory with collected samples/items.	Inventory
handle_time_limit()	Manages actions when the player exceeds the time limit for a level.	Level
handle_menu_option(option: str)	Handles the player's selection from the main menu.	HomePage
choose_shovel(shovel_type: str)	Allows the player to select a shovel from the inventory.	Shovel
use_shovel(shovel: reference)	Initiates the extraction process for a given object using the selected shovel.	Shovel
choose_suit(suit_type: str)	Allows the player to select a suit from the inventory.	Suit
use_suit(suit: reference)	Utilizes the benefits of the currently equipped suit during gameplay.	Suit

RegionalObject		
Method	**Responsibility**	**Collaboration Class**
getSample()	After extraction, it returns reference to itself as a sample to be added to the inventory.	Player, Inventory
interact()	Represents the interaction of the player with the object, triggering extraction and analysis processes.	Player
increase_count()	Increase the value of the amount of the object collected by the player	Player

Shovel		
Method	**Responsibility**	**Collaboration Class**
buy_shovel(player_currency: int)	Allows the player to purchase the shovel if they have enough Basalt currency. Returns True if purchase is successful, False otherwise.	Player, Store

use()	Allows the player to equip and use the shovel during gameplay, affecting the extraction speed of objects.	Player
get_shovel()	Returns reference to itself to be added to the store.	Store

Suit		
Method	**Responsibility**	**Collaboration Class**
use_suit()	Equips the suit for the player to use during gameplay, enhancing their speed and jump abilities.	Player
buy_suit(player_currency: int)	Allows the player to purchase the suit if they have enough Basalt currency. Returns True if purchase is successful, False otherwise.	Player, Store
get_suit()	Returns reference to itself to be added to the store.	Store

Store		
Method	**Responsibility**	**Collaboration Class**
buy_item(item_name: str, player_currency: int)	Allows the player to purchase an item from the store by providing the item's name and the player's available currency. Returns True if the purchase is successful, False otherwise.	Player
handle_menu_option(option: str)	Handles the player's selection from the store menu, allowing them to view available items and make purchases.	Player

Inventory		
Method	**Responsibility**	**Collaboration Class**
add_object(obj: RegionalObject)	Adds a new regional object to the inventory.	RegionalObject
remove_object(obj: RegionalObject)	Removes a regional object from the inventory.	RegionalObject
display_inventory()	Displays the collected regional objects in the inventory.	Player

Puzzle		
Method	**Responsibility**	**Collaboration Class**
check_answer(player_answer: str)	Checks if the player's answer matches. Returns a message indicating whether the answer is correct or not.	Player
check_inventory(inventory: Inventory)	Checks if the player has collected all the required samples for solving the puzzle by accessing the inventory.	Inventory
get_puzzle()	Returns reference to itself	Player, Level

Level		
Method	**Responsibility**	**Collaboration Class**
unlock_level()	Unlocks the level for the player.	Player, Map
complete_level()	Marks the level as completed.	Player, Map
add_puzzle(puzzle: Puzzle)	Adds a puzzle to the level.	Puzzle
remove_puzzle(puzzle: Puzzle)	Removes a puzzle from the level.	Puzzle

start_dust_storm()	Initiates a dust storm event in the level.	DustStorm
stop_dust_storm()	Stops the dust storm event in the level.	DustStorm
start_meteoroids_falling()	Initiates meteoroids falling event in the level.	MeteoroidsFalling
stop_meteoroids_falling()	Stops the meteoroids falling event in the level.	MeteoroidsFalling
calculate_points(seconds_saved: int)	Calculates the points earned by the player for completing the level.	Player
show_weather_forecast()	Displays real time martian weather data on screen	WeatherForecast

DustStorm		
Method	**Responsibility**	**Collaboration Class**
start()	Starts the dust storm event.	Level
end()	Ends the dust storm event.	Level

MeteoriteFalling		
Method	**Responsibility**	**Collaboration Class**
start()	Starts the meteorite falling event.	Level
end()	Ends the meteorite falling event.	Level

WeatherForecast		
Method	**Responsibility**	**Collaboration Class**
get_weather_forecast(date)	Retrieves weather forecast for a specific date by fetching data from the MAAS API and using the ML model to predict the weather.	Level

Map		
Method	**Responsibility**	**Collaboration Class**
select_level(Level reference)	Allows the player to select a level for exploration.	Player, Level

HomePage		
Method	**Responsibility**	**Collaboration Class**
input_player_name()	Prompts the player to input their name and stores it in the Player object's name attribute.	Player
chose_action(Choice)	Call method according to the option chosen by the player.	Player
resume_game()	Allows the player to resume the game, directing them to the map screen where previously unlocked levels are visible.	Player, Map
start_new_game()	Initiates a new game, unfolding the map to reveal the levels available for exploration. Only the first level is initially accessible.	Player, Level
view_high_scores()	Allows the player to view the top scores achieved in the game.	Player

4.3.5 CRC Modelling

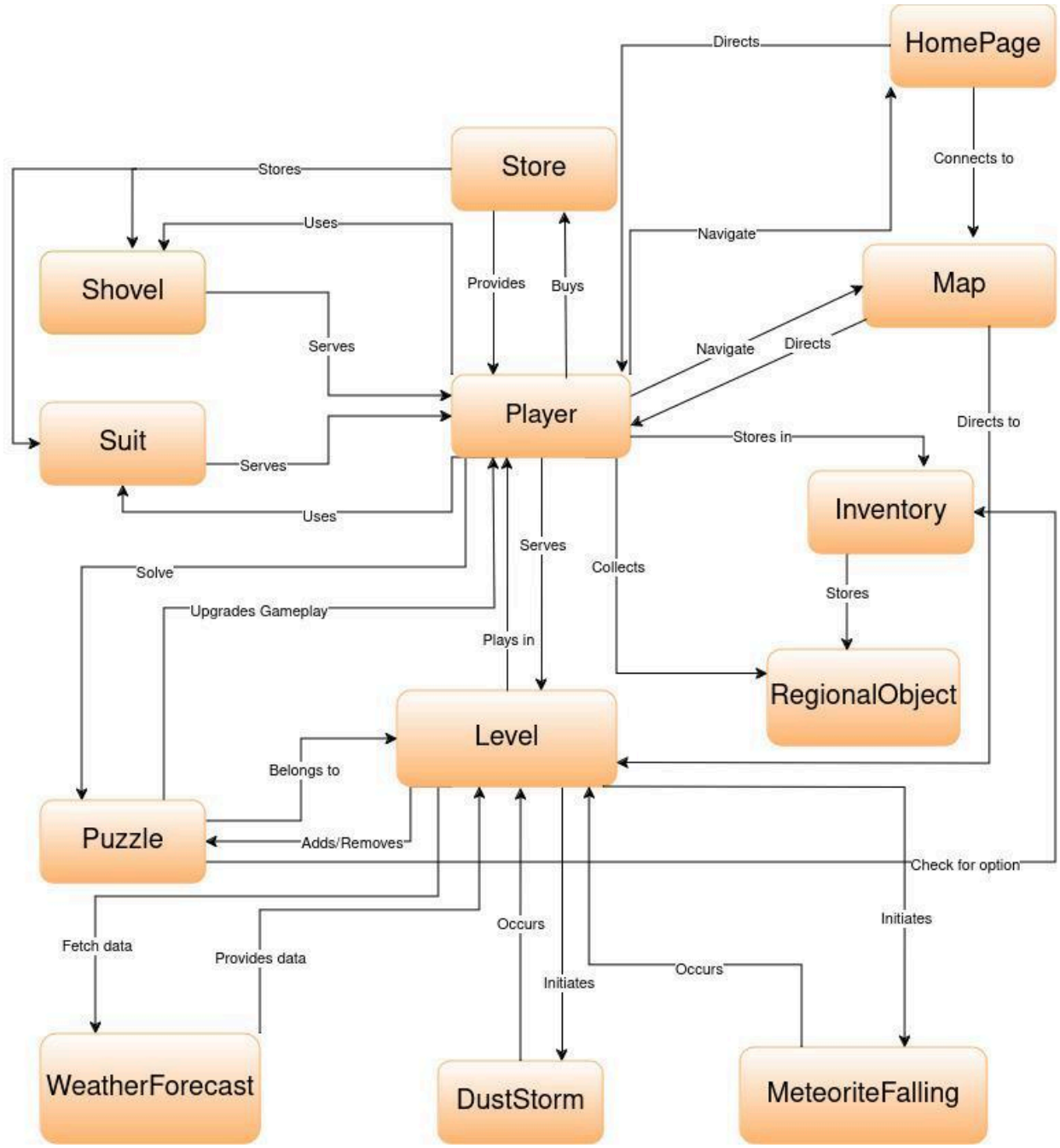


Figure 5: Class relationship diagram

4.4 BEHAVIORAL MODELING

This chapter describes the Behavioral Model for “MEG”.

4.4.1. Introduction

The behavioral model indicates how software will respond to external events or stimuli. In the context of behavioral modeling, two different characterizations of states must be considered: (1) the state of each class as the system performs its function and (2) the state of the system as observed from the outside as the system performs its function.

4.4.2. State Transition

One component of a behavioral model is a UML state diagram that represents active states for each class and the events (triggers) that cause changes between these active states.

Events Table :

Serial No.	Class	Event (Method)	Responsibility	Collaborator Class
1	Player	Player Name Set (set_name)	Sets the name of the player.	-
2	Player	Progress Updated (update_progress)	Updates the player's progress in the game (unlocking levels).	Level
3	Player	Regional Object Collected (add_to_inventory)	Adds collected samples to the player's inventory.	Inventory, RegionalObject
4	Player	Points Updated (update_points)	Updates the player's points based on certain actions/events.	-
5	Player	Currency Updated	Updates the player's Basalt	-

		(update_currency)	currency based on collection or transactions.	
6	Player	Level Started (start_level)	Initiates a new level for the player to play.	Level, Map
7	Player	Level Completed (complete_level)	Handles actions when the player successfully completes a level.	Level
8	Player	Failure Encountered (encounter_failure)	Handles actions when the player fails to complete a level due to external factors.	Level
9	Player	Puzzle Attempted (attempt_puzzle)	Handles the player's attempt to solve a puzzle and returns feedback.	Puzzle

10	Player	Item Bought (buy_item)	Allows the player to buy items from the store using Basalt currency.	Store, Suit, Shovel
11	Player	Inventory Viewed (show_inventory)	Displays the player's inventory with collected samples/items.	Inventory
12	Player	Time Limit Handled (handle_time_limit)	Manages actions when the player exceeds the time limit for a level.	Level
13	Player	Menu Option Handled (handle_menu_option)	Handles the player's selection from the main menu.	HomePage
14	Player	Shovel Chosen (choose_shovel)	Allows the player to select a shovel from the	Shovel

			inventory.	
15	Player	Shovel Used (use_shovel)	Initiates the extraction process for a given object using the selected shovel.	Shovel
16	Player	Suit Chosen (choose_suit)	Allows the player to select a suit from the inventory.	Suit
17	Player	Suit Used (use_suit)	Utilizes the benefits of the currently equipped suit during gameplay.	Suit
18	RegionalObject	Sample Extracted (interact)	Represents the interaction of the player with the object,	Player

			triggering extraction and analysis processes.	
19	RegionalObject	Sample Count Increased (increase_count)	Increases the amount of the object collected by the player during interaction.	Player
20	Shovel	Shovel Bought (buy_shovel)	Allows the player to purchase the shovel if they have enough Basalt currency.	Player, Store
21	Shovel	Shovel Equipped (use)	Allows the player to equip and use the shovel during gameplay.	Player

22	Shovel	Shovel Added to Store (get_shovel)	Returns reference to itself to be added to the store.	Store
23	Suit	Suit Bought (buy_suit)	Allows the player to purchase the suit if they have enough Basalt currency.	Player, Store
24	Suit	Suit Equipped (use_suit)	Equips the suit for the player to use during gameplay.	Player
25	Suit	Suit Added to Store (get_suit)	Returns reference to itself to be added to the store.	Store
26	Store	Item Bought (buy_item)	Allows the player to purchase an item	Player

			from the store.	
27	Store	Menu Option Handled (handle_menu_option)	Handles the player's selection from the store menu.	Player
28	Inventory	Object Added to Inventory (add_object)	Adds a new regional object to the inventory.	RegionalObject
29	Inventory	Object Removed from Inventory (remove_object)	Removes a regional object from the inventory.	RegionalObject
30	Inventory	Inventory Displayed (display_inventory)	Displays the collected regional objects in the inventory.	Player
31	Puzzle	Puzzle Solved (check_answer)	Checks if the player's answer matches. Returns feedback.	Player

32	Puzzle	Inventory Checked (check_inventory)	Checks if the player has collected all the required samples for solving the puzzle.	Inventory
33	Puzzle	Puzzle Disabled (disable_puzzle)	Disables the puzzle icon once it's solved.	-
34	Puzzle	Puzzle Level Retrieved (get_level)	Retrieves the level of the puzzle.	Level
35	Level	Level Unlocked (unlock_level)	Unlocks the level for the player.	Player, Map
36	Level	Level Completed (complete_level)	Marks the level as completed.	Player, Map
37	Level	Puzzle Added to Level (add_puzzle)	Adds a puzzle to the level.	Puzzle

38	Level	Puzzle Removed from Level (remove_puzzle)	Removes a puzzle from the level.	Puzzle
39	Level	Dust Storm Started (start_dust_storm)	Initiates a dust storm event in the level.	DustStorm
40	Level	Dust Storm Ended (stop_dust_storm)	Stops the dust storm event in the level.	DustStorm
41	Level	Meteoroids Falling Started (start_meteoroids_falling)	Initiates meteoroids falling event in the level.	MeteoriteFalling
42	Level	Meteoroids Falling Ended (stop_meteoroids_falling)	Stops the meteoroids falling event in the	MeteoriteFalling
43	Level	Points Calculated	Calculates the points earned by	Player

		(calculate_points)	the player for completing the level.	
44	Level	Weather Forecast Displayed (show_weather_forecast)	Displays real-time Martian weather data on screen.	WeatherForecast
45	DustStorm	Dust Storm Started (start)	Starts the dust storm event.	Level
46	DustStorm	Dust Storm Ended (end)	Ends the dust storm event.	Level
47	MeteoriteFalling	Meteorite Falling Started (start)	Starts the meteorite falling event.	Level
48	MeteoriteFalling	Meteorite Falling Ended (end)	Ends the meteorite falling event.	Level
49	WeatherForecast	Weather Data Fetched	Fetches weather data from the	Level

		(fetch_weather_data_from_api)	MAAS API for a specific date.	
50	WeatherForecast	Weather Forecast Retrieved (get_weather_forecast)	Retrieves weather forecast for a specific date.	Level
51	HomePage	Player Name Input Prompted (input_player_name)	Prompts the player to input their name and stores it.	Player
52	HomePage	Action Chosen (chose_action)	Calls method according to the option chosen by the player.	Player
53	HomePage	Game Resumed (resume_game)	Allows the player to resume the game.	Player, Map
54	HomePage	New Game Started (start_new_gam	Initiates a new game.	Player, Level

		e)		
55	HomePage	High Scores Viewed (view_high_scores)	Allows the player to view the top scores achieved in the game.	Player
56	HomePage	About Information Viewed (view_about)	Displays information about the gameplay.	Player
57	HomePage	Credits Viewed (view_credits)	Displays credits, acknowledging the creators of the game.	Player
58	HomePage	Game Exited (exit_game)	Allows the player to exit the game.	Player

State Transition Diagram:

Name: Home Page

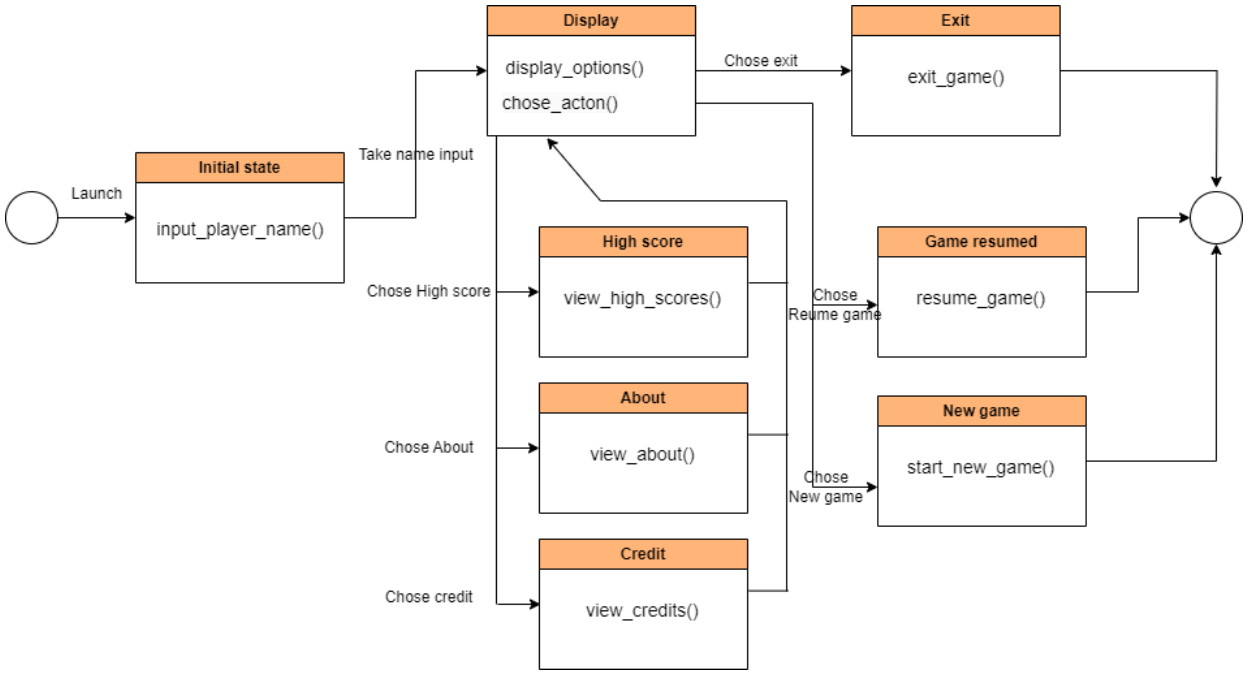


Figure 6.1 : HomePage State Transition

Name: Player

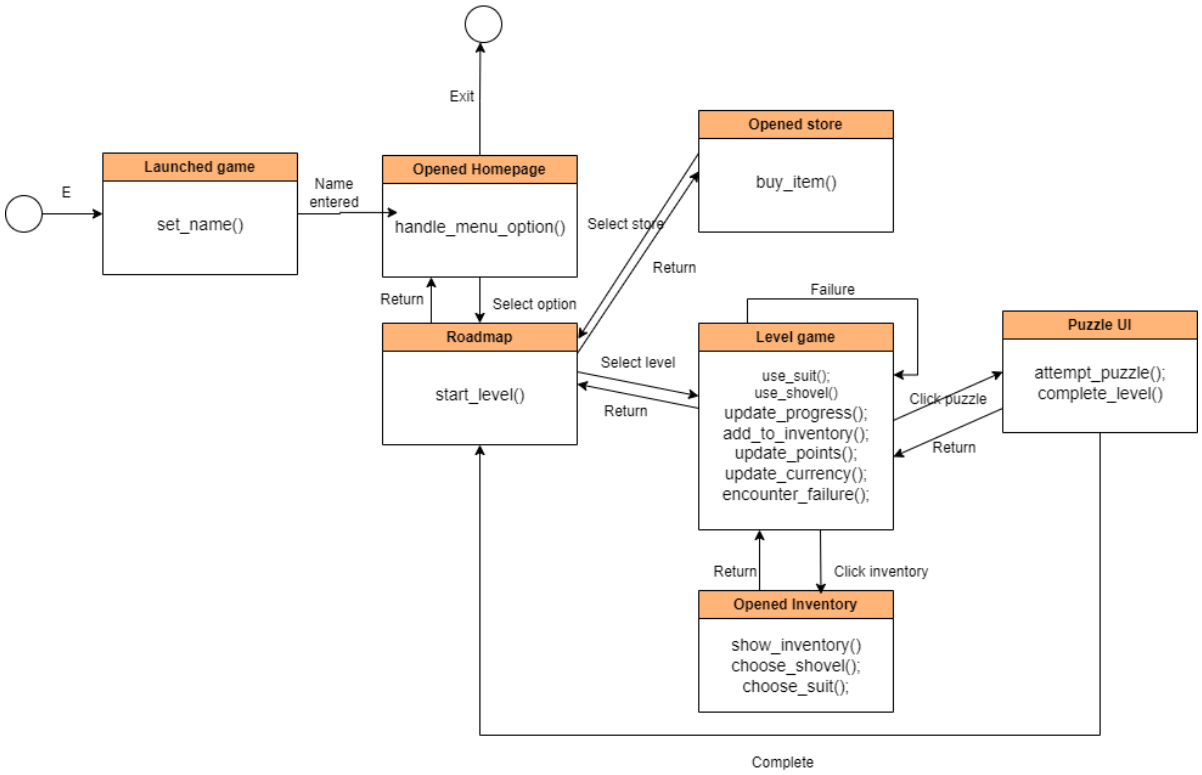


Figure 6.2 : Player state transition

Name: RegionalObject

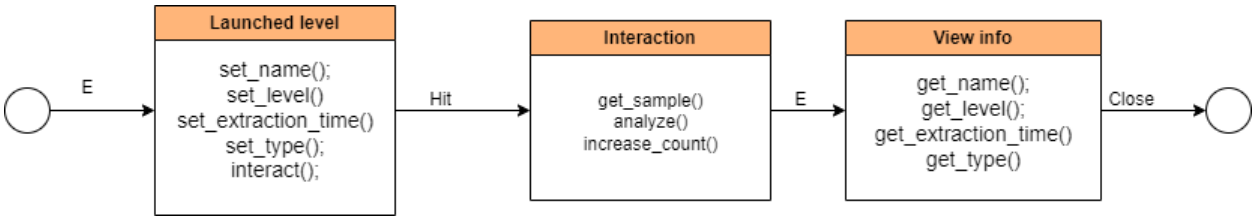


Figure 6.3 RegionalObject State Transition

Name: Shovel

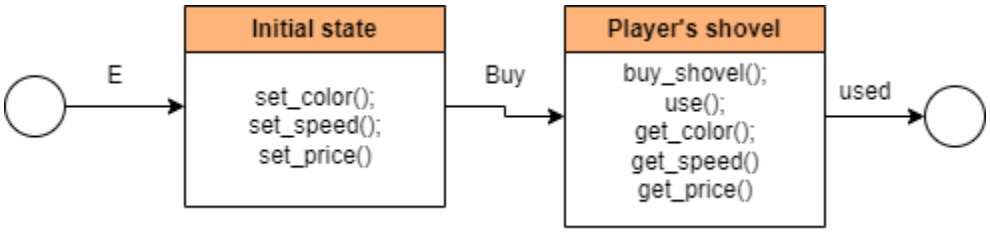


Figure 6.4 : Shovel State Transition

Name: Suit

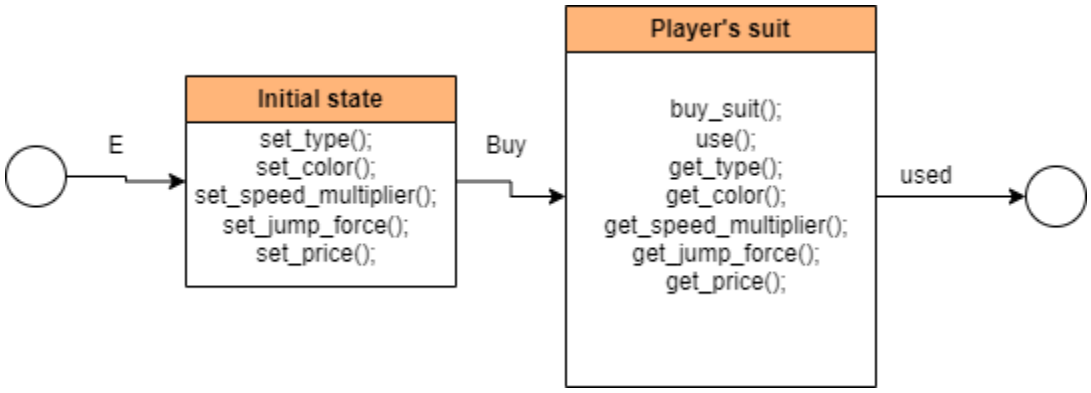


Figure 6.5 Suit state transition

Name: Store

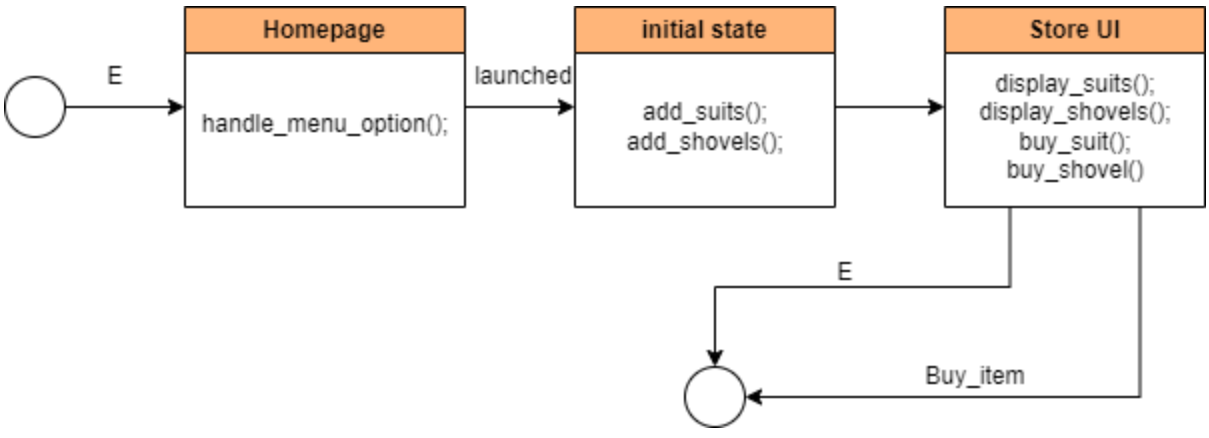


Figure 6.6 : Store State Transition

Name: Inventory

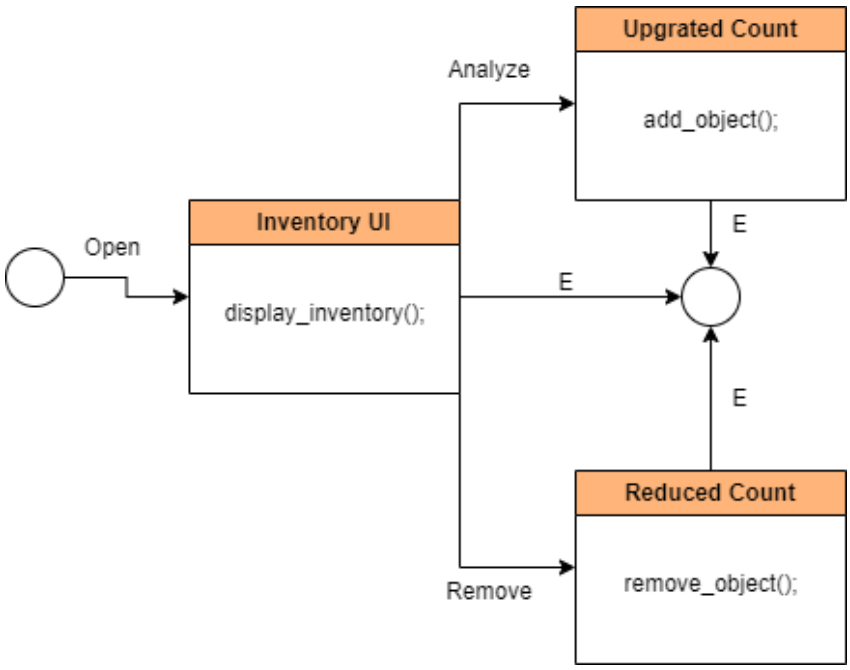


Figure 6.7 : Inventory State Transition

Name: Puzzle

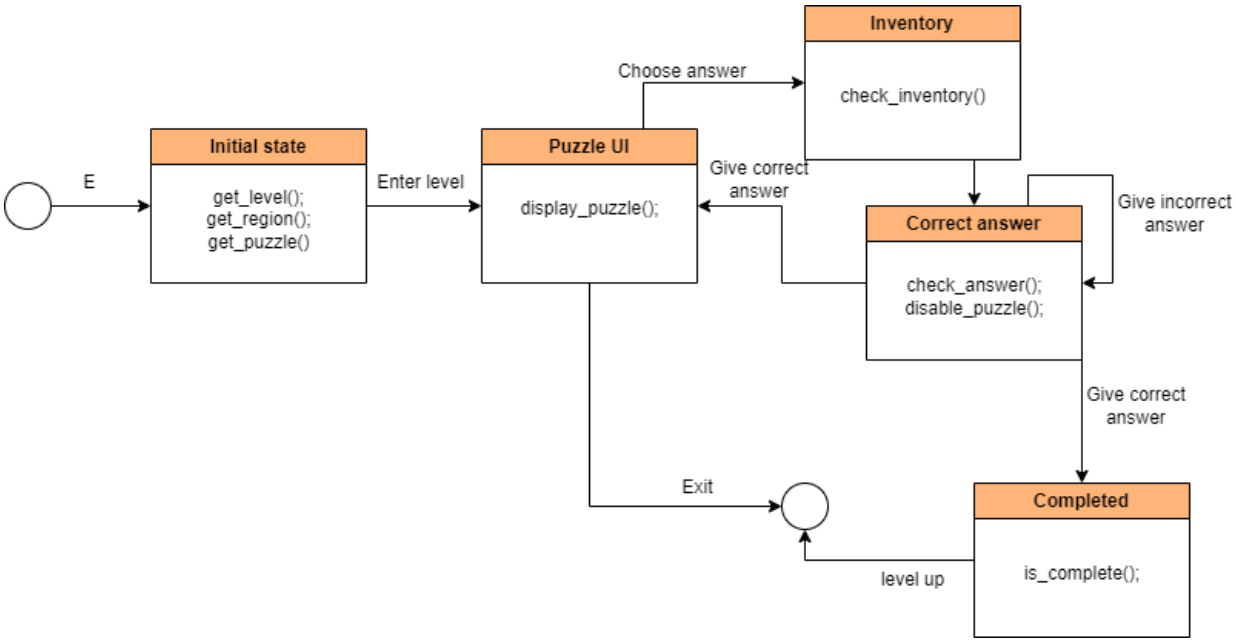


Figure 6.8 : Puzzle State Transition

Name: Map

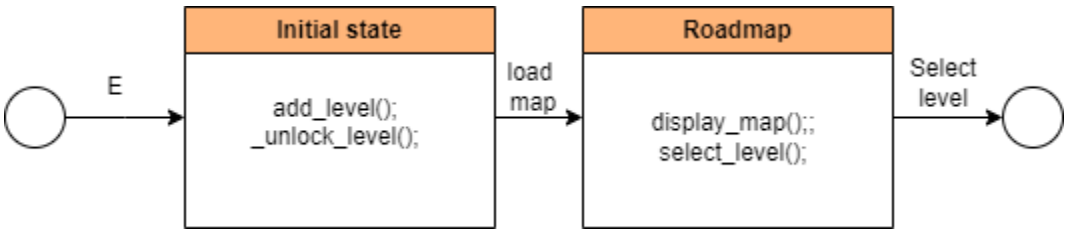


Figure 6.9 Map state transition

Name: Level

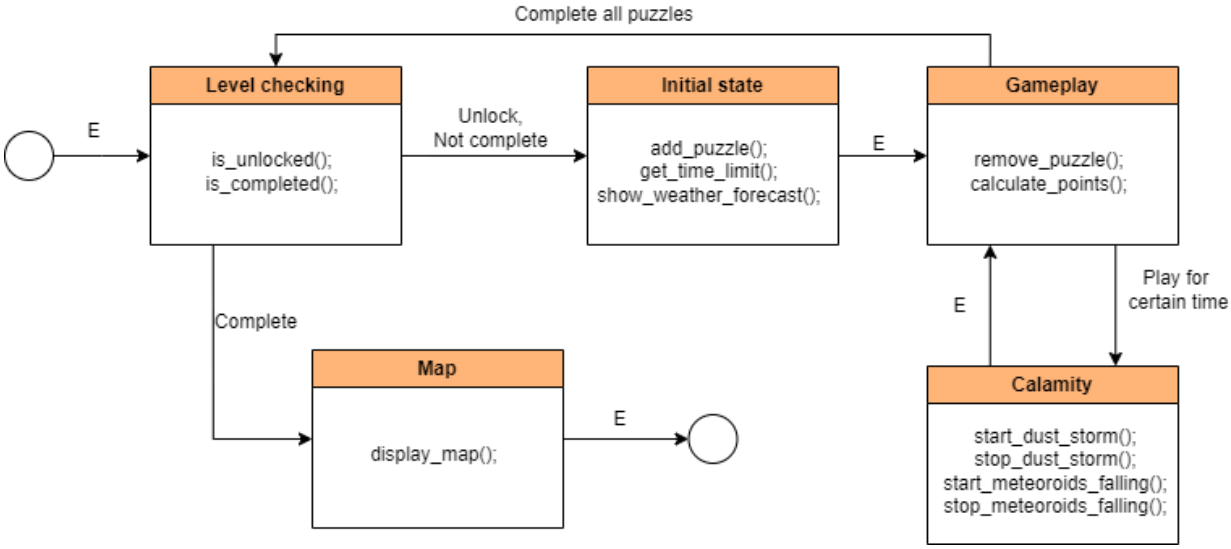


Figure 6.10 : Level state transition

Name: DustStorm

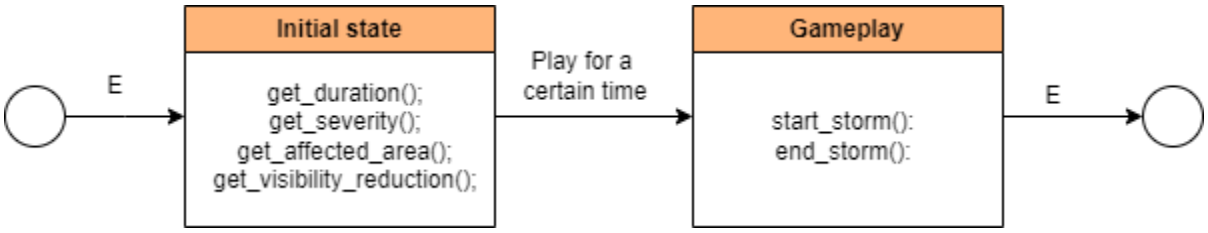


Figure 6.11 : DustStorm state transition

Name: MeteoriteFalling

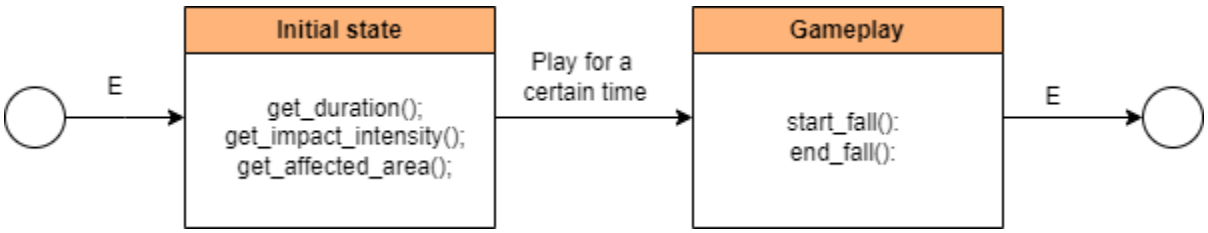
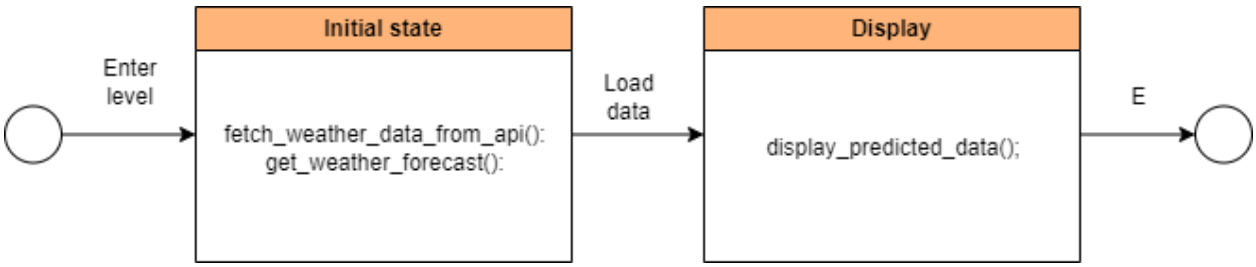


Figure 6.12 : MeteoriteFalling State transition

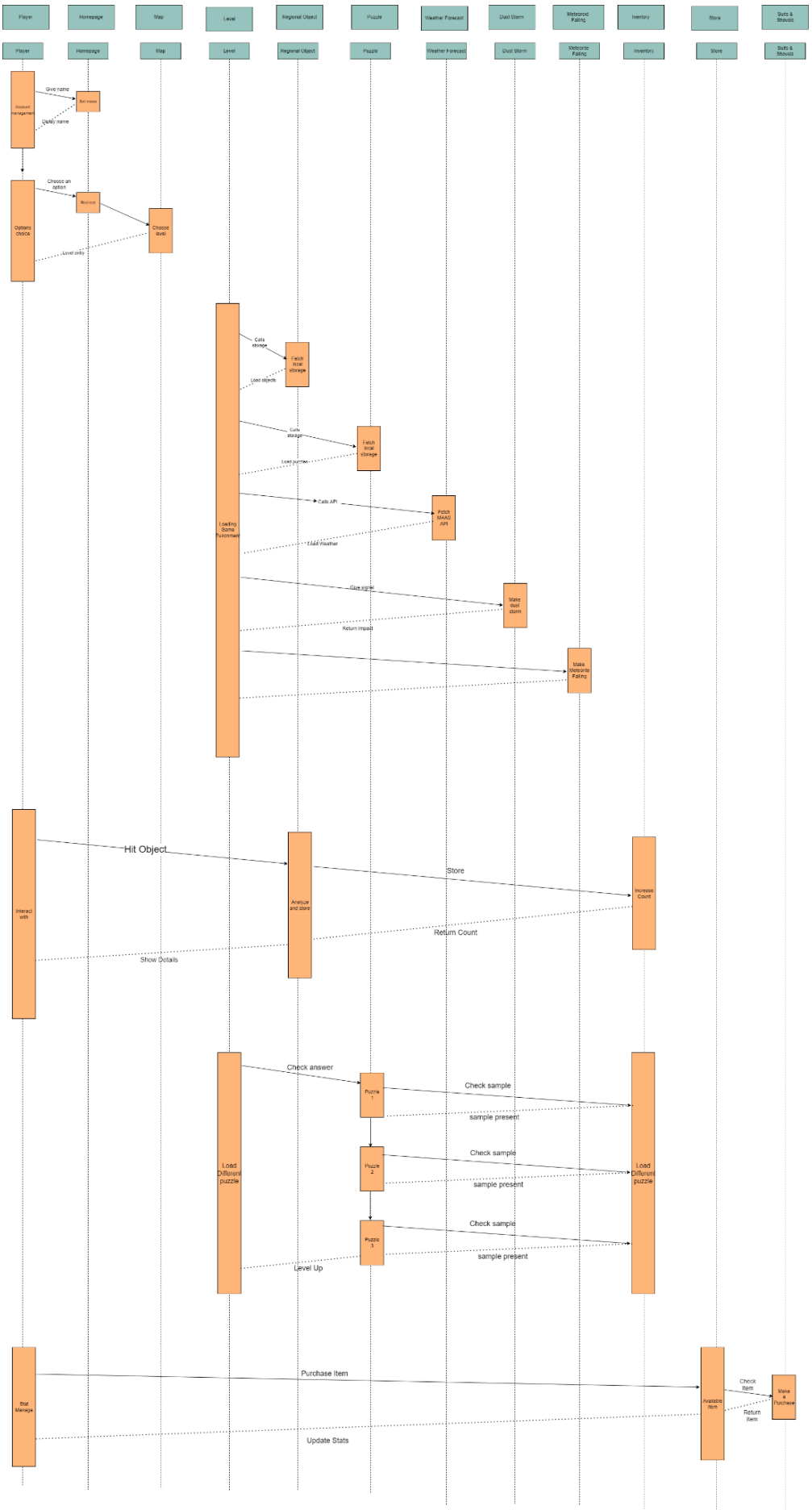
Name: WeatherForecast

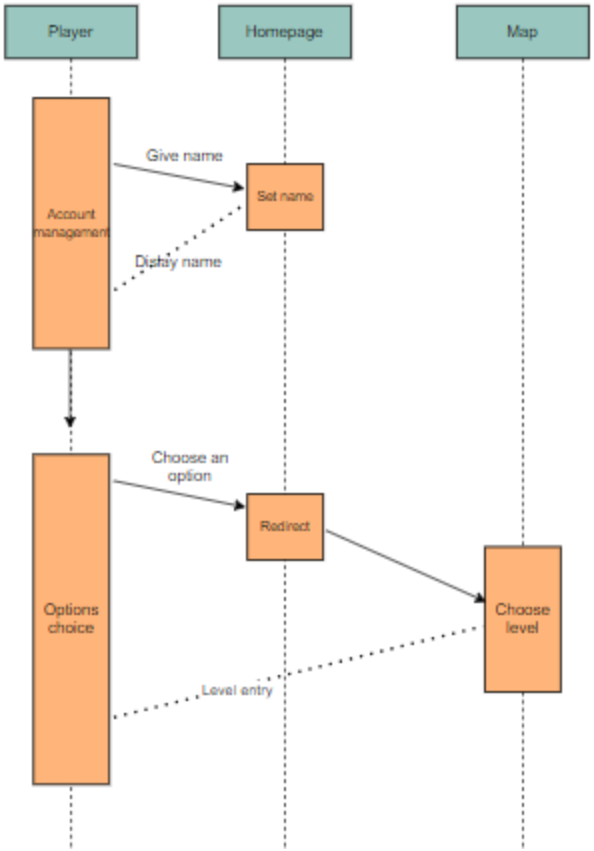


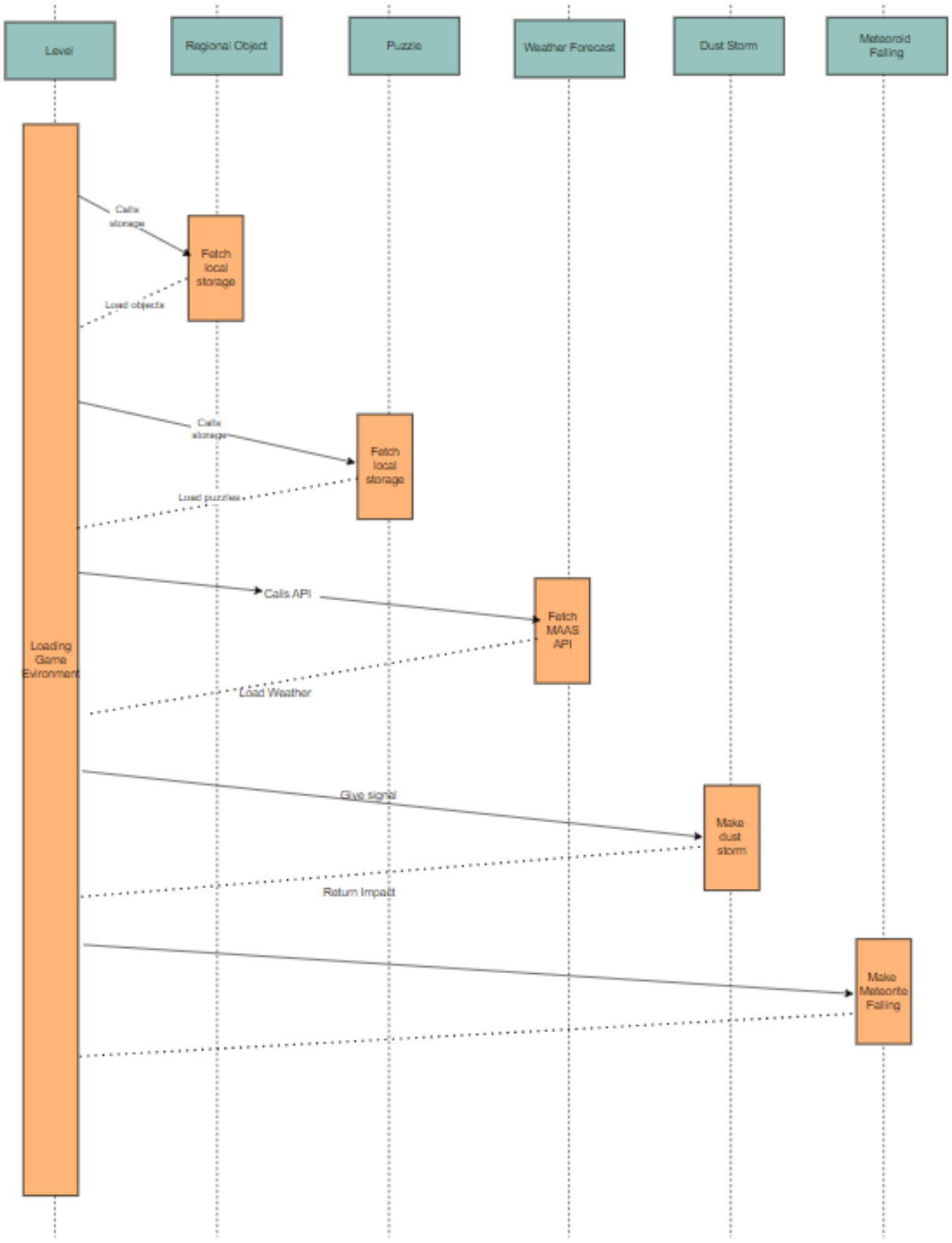
6.13 : WeatherForecast state transition

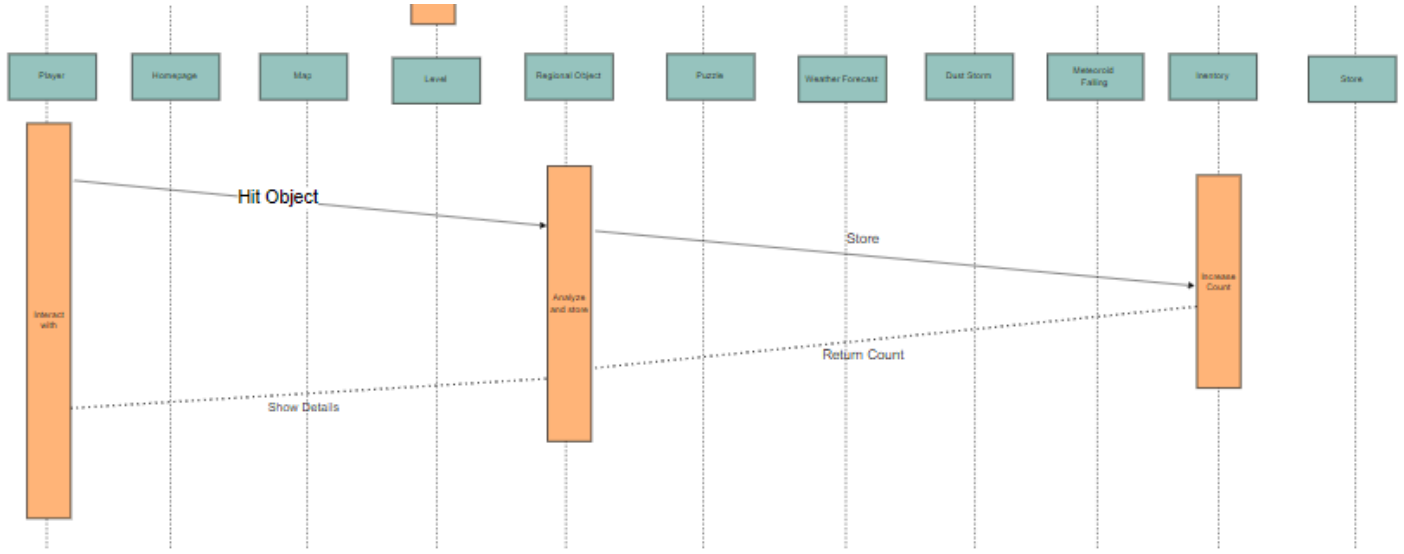
4.4.2. Sequence Diagram

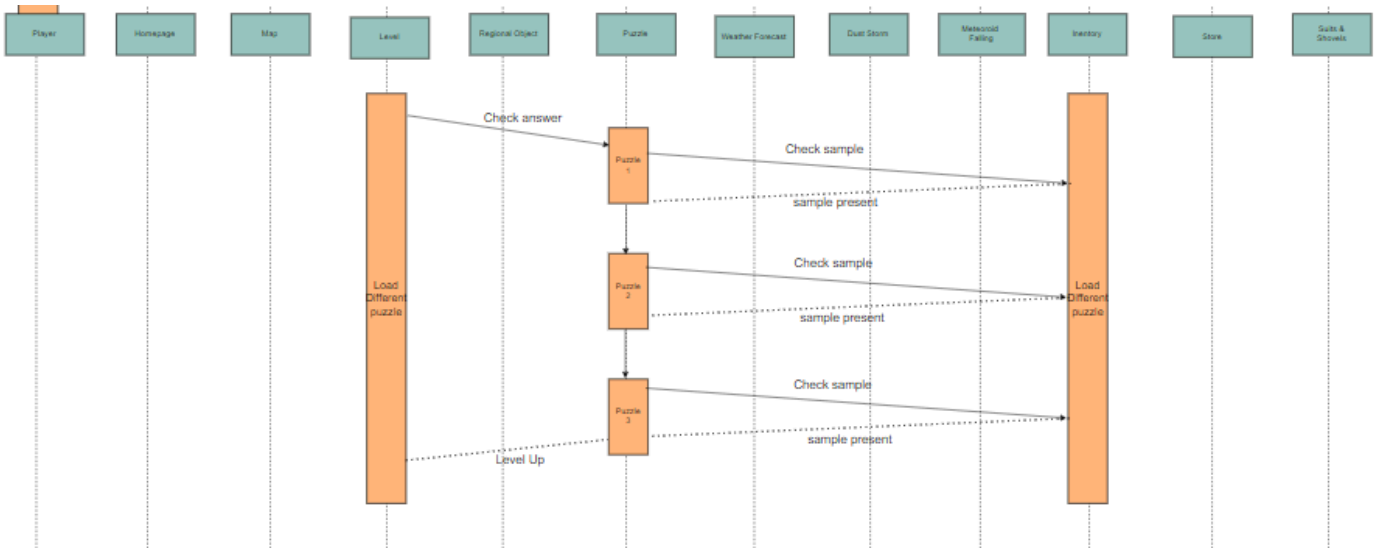
The second type of behavioral representation, called a sequence diagram in UML, is a representation of how events cause flow from one object to another as a function of time. In essence, the sequence diagram is a shorthand version of the use case. It represents key classes and the events that cause behavior to flow from class to class.

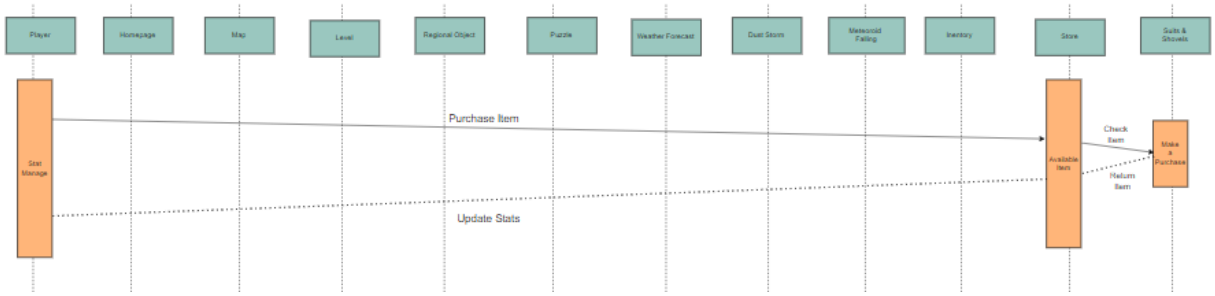

Player
Homepage
Map
Level
Regional Object
Puzzle
Weather Forecast
Dust Storm
Meteorite Falling
Inventory
Store
Stats & Shovels
Give name
Set name
Modify name
Choose an option
Next level
Choose level
Level only
Calls storage
Fetch local storage
Load objects
Load puzzles
Calls API
Fetch MAAS API
Load Weather
Loading Game Environment
Give signal
Make dust storm
Return Impact
Make Meteorite Falling
Interact with
Hit Object
Analyze and store
Store
Increase Count
Return Count
Show Details
Check answer
Puzzle 1
Puzzle 2
Puzzle 3
Check sample
sample present
Load Different puzzle
Level Up
Stat Manage
Purchase Item
Available Item
Check Item
Make a Purchase
Return Item
Update Stats


Player
Homepage
Map
Account management
Give name
Set name
Display name
Choose an option
Redirect
Options choice
Choose level
Level entry


Level
Regional Object
Puzzle
Weather Forecast
Dust Storm
Meteoroid Falling
Calls storage
Fetch local storage
Load objects
Calls storage
Fetch local storage
Load puzzles
Calls API
Fetch MAAS API
Loading Game Evironment
Load Weather
Give signal
Make dust storm
Return Impakt
Make Meteorite Falling


Player
Homepage
Map
Level
Regional Object
Puzzle
Weather Forecast
Dust Storm
Meteoroid Falling
Inventory
Store
Interact with
Hit Object
Analyze and store
Store
Increase Count
Return Count
Show Details


Player
Homepage
Map
Level
Regional Object
Puzzle
Weather Forecast
Dust Storm
Meteoroid Falling
Inventory
Store
Suits & Shovels
Load Different puzzle
Check answer
Puzzle 1
Check sample
sample present
Puzzle 2
Check sample
sample present
Puzzle 3
Check sample
sample present
Level Up
Load Different puzzle



5. Conclusion

MEG is an endeavor to incorporate gaming and education regarding the Martian environment. MEG aims to inspire curiosity, spark interest in space exploration, and provide an enriching experience for players worldwide. Throughout the design, we have tried to follow the best practices to ensure security and scalability.