Specifications

Mars AI Simulator

February 4, 2025

Danlin Luo, William Rodzen-Staff, Priyanshu Bhandari, Musab Umair, Basel Ahsan, Rafi Latif, Aryan Chaturvedi

Planetary Procrastinators

Contents

1. Background

1.1. Motivation

The objective of this Mars AI Simulator project is to evaluate the performance of various AIs with differing initialized parameters in discovering water and other resources. Our goal is to improve the AI's capability to assess the potential for life on other planets through rover simulations, thereby reducing the need for costly large-scale missions. This particular project focuses on simulating this concept using existing Mars terrain data. With variable spawn points and rover characteristics chosen by the user, the project will assist in identifying the most efficient type of rover for exploration. This initiative was primarily requested by NASA's Exoplanet Division, whose primary mission is to detect signs of life on planets beyond Earth, with the presence of water serving as a key indicator.

1.2. Stakeholders

Stakeholders	Name	Coordinates	Success Measure: Win Condition	Success Measure: Lose Condition
Director	Janet Petro	Phone number1, email1, office number1.	The AI simulator effectively demonstrates the ability to detect water on Mars. The project meets scientific and financial expectations, attracting further funding. Stakeholders such as scientists, engineers and funders, trust and support the project.	The simulator does not meet project objectives, failing to produce meaningful insights. Investors or funding bodies withdraw support due to lack of progress. The project receives negative publicity or criticism from the scientific community.
Team Manager	Bobette Rodriguez	Phone number2, email2, office number2.	The development process is efficient, and milestones are met on schedule. The simulation runs reliably, and data integrity is maintained. The team works cohesively, resolving issues proactively.	The simulation is delayed due to poor workflow, mismanagement, or unresolved technical challenges. Bugs, inaccuracies, or inefficiencies slow down research progress. Team morale declines, leading to delays or communication breakdowns.
Researcher	Indiana Jones	Phone number3, email3, office number3.	Clear configuration of AI biases and adjustable parameters, easy to start and use the simulation, access to personal & leaderboard results for comparative analysis, accurate detection of water. The simulation produces	Complex interface that slows productivity, inaccurate results and tracking of statistics. AI simulation is unreliable, unrealistic and does not translate to real-world rover performance.

			accurate and reproducible scientific results, and AI simulation integrates with real-world data to improve Mars mission planning.	
Casual Users	John Doe	Phone number4, email4.	Responsive simulation that reacts to user inputs smoothly. Leaderboards to compare performance with others. Simple but engaging interface that doesn't require technical expertise.	Simulation is too slow, unresponsive, or unintuitive, leading to frustration. Results feel arbitrary, making it unclear how performance is measured. Difficulty accessing, saving, or comparing results with other users.
Indirect User: Financial Stakeholder Providing Funding	SpacePharma	Phone number5, email5, office number5.	The simulator shows potential for commercial or scientific applications, leading to a positive return on investment, or the project demonstrates significant scientific or technological advancements, justifying continued funding.	The project goes over budget without delivering useful results, fails to attract interest from scientific communities or commercial partners.

2. Overview

2.1. Usage

The Director will use the AI Mars Rover Simulator primarily for high-level project oversight. Their main goal is to assess the success of the simulation in meeting strategic objectives, ensuring that financial stakeholders are satisfied with the investment. The Director will review reports and key performance metrics to determine whether the project is worth continued funding. They will interact with the program during major project milestones, funding reviews, and stakeholder presentations.

The Team Manager will use the simulator to monitor development progress and ensure efficient team coordination. They will track bugs, oversee feature implementation, and ensure the simulation meets research requirements. Additionally, they will facilitate communication between engineers, researchers, and stakeholders. The Team Manager will interact with the program on a daily or weekly basis to review progress, address issues, and prepare for sprint planning or project reviews.

Researchers will use the program as a scientific tool to simulate different Mars environments and rover configurations. Their goal is to test AI performance, analyze water detection accuracy, and refine the simulation's biases and detection methods. Researchers will adjust simulation parameters, run extensive tests, and extract data for scientific publications. They will interact with the program frequently (daily or weekly) as part of their research workflow, especially after AI updates or before publishing results.

They will also use the program to simulate potential Mars rover missions before deployment. They will test AI predictions and rover configurations to enhance real-world exploration strategies. If this simulation becomes integrated into actual mission planning, they will interact with it during pre-mission planning and after AI model improvements to ensure that simulations remain accurate and useful for real-world applications.

Casual users will treat this program as an interactive simulation game, attempting to find the optimal rover configuration for detecting water on Mars. They will engage with the software in their free time, experimenting with different strategies and competing on leaderboards. Their primary interests lie in the entertainment, educational, and competitive aspects of the simulation rather than deep scientific research. Casual users will interact with the program occasionally, when they want to explore the simulation, test their problem-solving skills, or participate competitively in their community.

Financial stakeholders will not interact directly with the simulator but will use investment reports, project updates, and financial summaries to assess its viability. Their goal is to determine whether the project is financially sustainable and has the potential for commercialization or further funding. They will interact with reports and results generated from the simulation quarterly, annually, or whenever major funding decisions are required.

2.2. Design

Development Start and Completion Dates

The expected development start is on February 8, 2025, and the expected project completion date is April 14, 2025.

The Storyboard

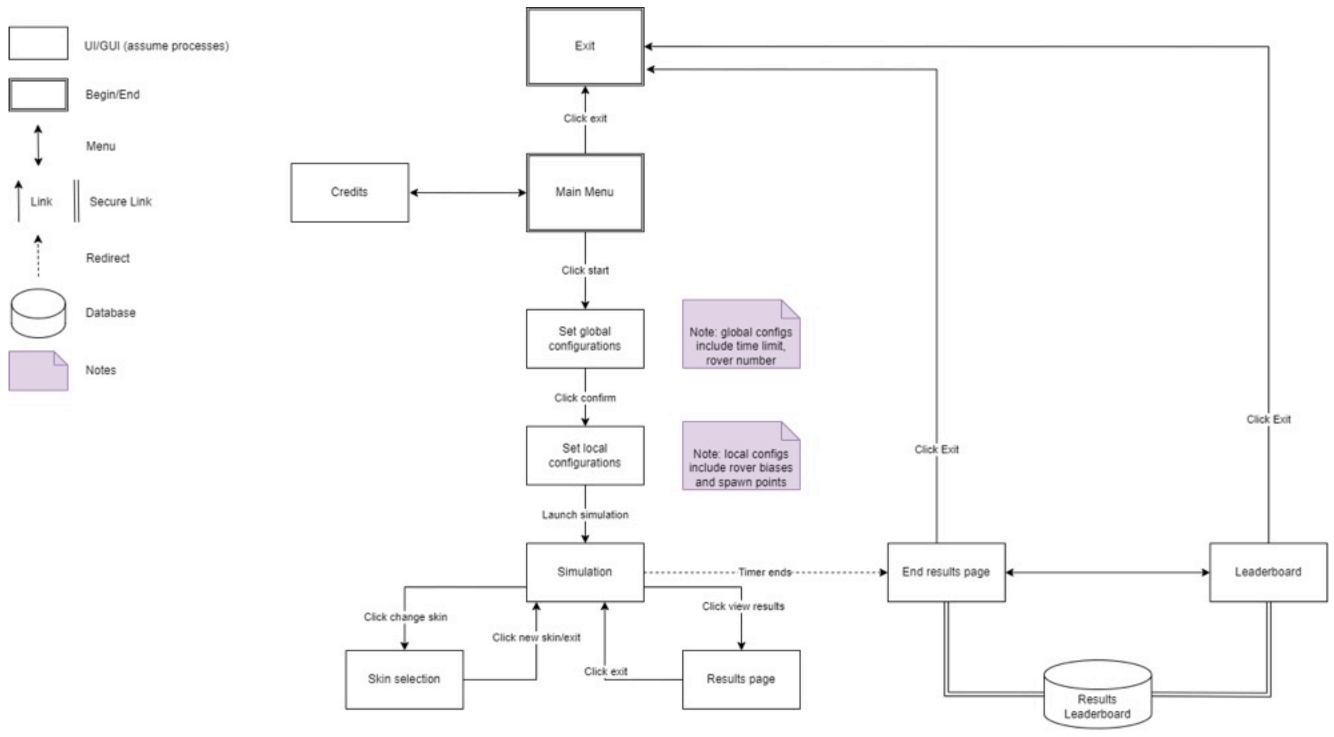


Figure 1: The storyboard for the Mars AI Simulation. The flow of the application is depicted, with the simulation portion to be described in greater detail in the following section.

Main Menu: This is the welcome screen that users view when the application is first launched. The screen contains the name of the program, the team name, start, credits, and exit.

Exit: The program closes.

Credits: This page contains information about the creators of this program, including the name and role of each contributor.

Set global configurations: This screen will give the user the option to select configurations that affect the game globally, such as the time limit and the rover number.

Set local configurations: This screen will give the user the option to select configurations that affect the game locally. The map of Mars will be displayed, and the user selects a point on the map as a starting spawn point as well as the biases that the rover is initialized with. The user is required to initialize rovers according to the number that was set in global configurations.

Simulation: After initializing the global and local configurations, the rover is spawned into the map. Depending on the rover's biases, it will learn to traverse the map differently from another rover and adapt to the terrains it has experienced itself. The rover's skin will also affect how it traverses the terrain, having different advantages and disadvantages depending on the chosen skin. The rover's main goal would be to find water over Mars' terrain. The rover that finishes with the highest number of underground water bodies found from the land it has traversed will be the winner.

Skin selection: When the user clicks on a rover, a side menu will appear displaying the list of available skins that the rover may change to. The user can choose a skin or exit the menu.

Results page: When the user clicks on a view results button, they will be taken to a page displaying the current performance of each rover ranked at the time of entering the results page. They are ranked by the number of water bodies found. In case of a tie, the rovers will be ranked by a secondary metric: terrain discovered. Clicking exit returns them to the simulation.

End results page: Once the time limit ends, the user will be redirected to the results page where each rover's performance is ranked by the number of water bodies found. In case of a tie, the rovers will be ranked by a secondary metric: terrain discovered.

Leaderboard: The leaderboard displays the all time results of the rovers, ranked by their ability to find water. In case of a tie, the rovers will be ranked by a secondary metric: terrain discovered.

Results leaderboard: This database stores all previous results of rovers, including the rover's performance, the date of the game, the rover biases, and skins the rover used.

The Deployment Diagram

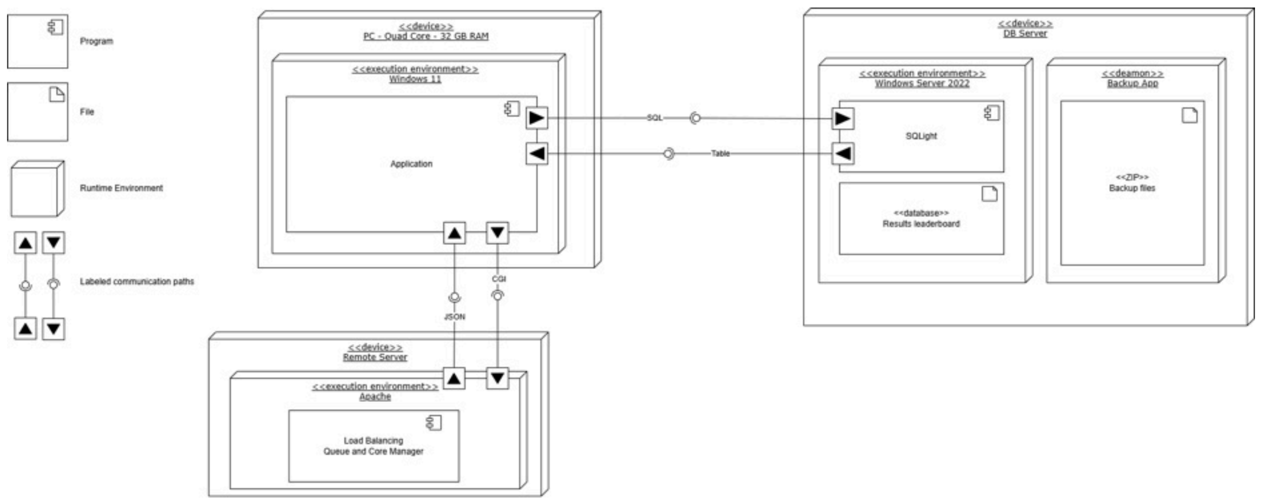


Figure 2: The deployment diagram for the Mars AI Simulation. The main components consist of the application, the database used to store the results, and the external data queried through an API.

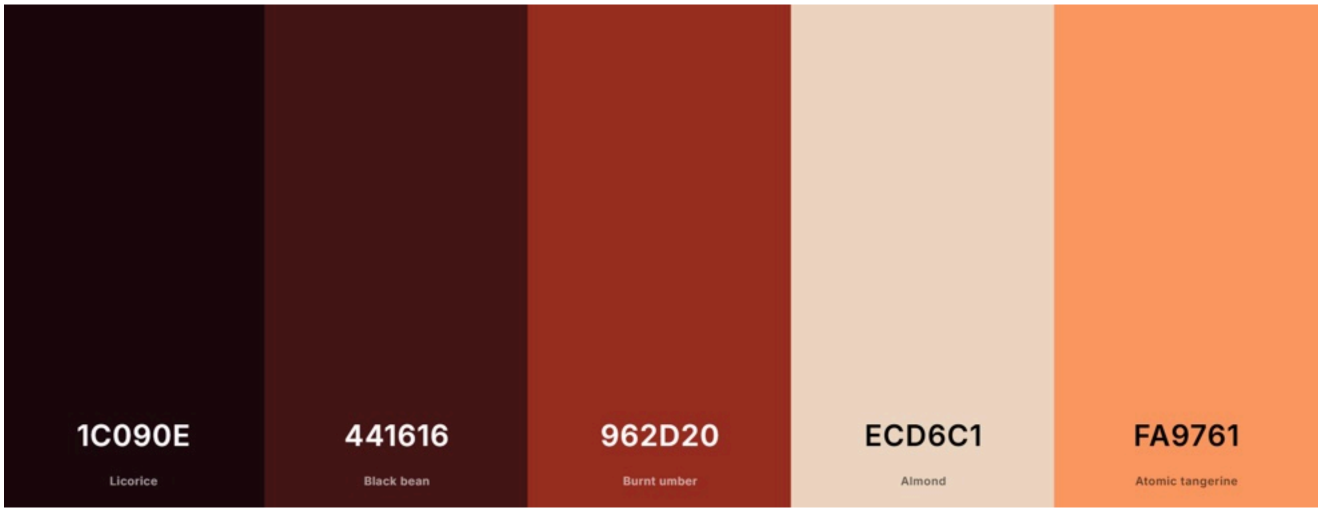
PC: The PC runs a Windows 11 operating system, providing a user-friendly environment for the Mars AI simulator Application. This application handles the running of the AI operations and the whole rest of the simulation. The computer does not store any local data; instead, it sends result inputs (e.g. result ID) to the server for processing via SQL queries. Responses from the server are received as a Table, processed by the program and is displayed to the user in a clean format. The execution environment ensures a lightweight and responsive interface tailored for at home use. Communication with users is facilitated through TCP/IP for wireless connections and Wi-Fi for wireless devices. TCP/IP will get the Mars terrain data via API which will be returned via a JSON file.

DB Server: The DB Server offers a robust and secure environment to store the system's leaderboard. It hosts a critical program: the SQL Database for storing results of the simulation. The server is responsible for executing logic, maintaining data consistency, and ensuring tasks, such as updating the leaderboard, are completed efficiently.

Remote Server: The Remote Server receives user inputs sent by the PC to the remote server for processing via CGI requests. Responses sent back to the PC are in JSON format and displayed to the user. The terrain is information that is found on the internet and will be retrieved through API calls.

3. Definitions

3.1. Branding



Planetary Procrastinators

The colour palette above was chosen to align with the visual identity and thematic elements of Mars while maintaining readability, usability, and aesthetic appeal. The logo for Planetary Procrastinators following the branding colors is displayed under the colour palette.

Mars has an iron-rich oxidized surface, which gives it its signature red colour. The Black Bean and Burnt Umber, as well as the earthy tones in the palette resemble Martian soil and rocky terrain, creating a visual experience that reflects the planet's real-world conditions.

The Almond also provides a neutral, readable contrast against the reds and browns, ensuring clarity in UI elements and texts.

The Licorice adds another layer of darker colours to represent the shadows and outer space of Mars, enhancing the natural and rocky aesthetic of the palette and allowing for the other colors to shine through. Finally, Atomic Tangerine can be used as an accent to guide the user's eyes while maintaining the earthy feel of the palette.

The two primary fonts used in this project are Orbitron and Press Start 2P. Orbitron has a modern, space-themed design, making it an ideal choice for our Mars AI Simulator. The font invokes a sense of technology and innovation, aligning with the cutting-edge nature of the simulation. Orbitron's futuristic and digital style connects the interface to the concept of space missions. This font was also selected as the font for Planetary Procrastinator's logo to communicate the same sense of technology and innovation of the developing team.

On the other hand, Press Start 2P is inspired by retro video games. It can evoke nostalgia, connecting the user experience to the early days of video games, and in some ways, the early days of space exploration when technology was more basic and systems were much simpler. It can engage a wider audience, making the application feel more approachable or gamified.

Together, Orbitron and Press Start 2P with the rusty color palette of Mars gives the Mars AI Simulator a unique look that blends modern technology with a nostalgic user interface.

3.2. Required Technologies

The requirements for the development of this project are as follows:

Home Hardware: At least one personal computer is required for development.

Software: The program will be developed using C# for frontend and backend, SQLite for the database, and Mono from Unity's game engine as the runtime environment.

3.3. Project and System Constraints

The project and system constraints for this project are as follows:

Budget limitations: A small budget of $50 for costs that may arise including subscriptions to other software or setting up a database for the application.

Equipment limitations: We will only have access to each team member's personal computer and one server for development and testing.

Tech limitations: The application must be able to run locally on personal computers and connect to the internet to query terrain data.

3.4. Client Expectations

The client has listed the following expectations:

- The Mars AI Simulator must function as a realistic simulation that integrates authentic Mars terrain data to provide an accurate testing environment.

- The system should allow users to instantiate robot avatars on the surface of Mars, ensuring that their movement adheres to physically realistic constraints.
- Each robot avatar must feature an interchangeable AI "brain", allowing for easy replacement or modification to test different decision-making algorithms.

The project will be considered successful if the rover's AI demonstrates adaptive learning, effectively adjusting to the terrain and improving its ability to locate water sources. However, it will be deemed unsuccessful if the AI fails to learn from the environment or if it incorrectly detects or reports the presence of water.

3.5. Impact, Ethics, and Performance Statements

Impact Statement

The Mars AI Simulator has the potential to drive significant advancements in space exploration and AI-driven autonomy, shaping the future of planetary missions. By serving as a testing ground for autonomous rover systems, this simulation enhances the ability of AI to adapt, learn, and navigate challenging extraterrestrial terrains without human intervention.

Improving AI-based water detection methods is a crucial step toward the long-term goal of human exploration on Mars. Identifying water sources is essential for sustaining future missions, enabling fuel production, and advancing planetary science. Additionally, this project fosters collaboration among AI researchers, engineers, and planetary scientists, accelerating innovations in robotic autonomy, resource utilization, and planetary exploration strategies.

Beyond space applications, the advancements in AI-driven autonomy developed through this simulation could have broader benefits on Earth. By refining AI adaptability in extreme conditions, this project contributes to the development of more resilient and intelligent autonomous systems that can address critical global challenges.

Ultimately, the Mars AI Simulator pushes the boundaries of exploration and technological innovation, laying the foundation for future self-sustaining robotic missions while delivering valuable advancements that extend beyond space exploration.

Ethics Statement

The Mars AI Simulator is designed to advance education, research, and public engagement in space exploration while upholding ethical principles of transparency, accuracy, and responsible AI representation. As a tool for learning, it offers valuable insights into AI-driven decision-making and resource management in a safe and controlled environment, free from direct human risk. By simulating real-world challenges, the simulator inspires curiosity in STEM fields and fosters a deeper understanding of planetary exploration.

However, there are ethical concerns regarding misuse or misrepresentation of the simulator. If the AI models or simulations present oversimplified, biased, or misleading outcomes, users may develop misconceptions about AI's actual capabilities in space exploration. To mitigate this, the simulator must remain grounded in scientific integrity, ensuring that all data and predictions align with realistic constraints and known planetary science.

A key ethical consideration is transparency in communicating the simulator's limitations. It is essential that users recognize that the simulation does not represent real-world AI performance but rather a controlled environment for testing and learning. Additionally, scientific data must be represented accurately to avoid exaggerating AI's ability to detect water or navigate Martian terrain.

Beyond mitigating risks, the simulator also presents opportunities for ethical sharing and accessibility. Offering free or discounted access to schools, universities, and research institutions ensures that the technology serves an educational purpose and remains widely accessible. Partnering with reputable scientific organizations and educational institutions further reinforces its credibility and ethical integrity, ensuring that the project remains aligned with responsible research and public outreach.

Through a commitment to scientific accuracy, responsible AI representation, and equitable access to education, the Mars AI Simulator seeks to inspire, educate, and innovate in a manner that benefits both scientific progress and society.

Performance Statement

The Mars AI Simulator is designed to provide a realistic and computationally efficient simulation of rover autonomy and water detection on Mars. The system must balance simulation complexity, responsiveness, and hardware limitations to ensure smooth operation.

Runtime: The run time of the simulation depends on factors such as terrain complexity, AI model computation, and the number of active rover avatars. Given that each rover operates with its own AI "brain" and must process terrain navigation and water detection in real time, the simulation must be optimized for parallel processing to ensure smooth execution. The estimated run time per session will vary based on user-defined simulation length but should allow for continuous operation without excessive delays.

Network Load: The network load will depend on whether the simulator operates in a local or cloud-based environment. If running locally, the network impact is minimal. However, if a cloud-based architecture is implemented for data storage, leaderboard comparisons, or AI updates, network load may increase. To optimize efficiency, only essential data (such as rover performance statistics and environmental conditions) should be transmitted, reducing unnecessary bandwidth usage.

Data Warehousing Needs: The simulator requires storage for Mars terrain data, AI model states, and simulation results. Given the complexity of terrain mapping and AI adaptation, a structured data warehousing system is necessary to manage simulation logs and performance metrics. The system should support efficient retrieval and storage of historical simulation data, allowing for comparative analysis between AI models and user experiments.

Turnaround Time: Turnaround time is defined as the time required to process an AI-driven simulation request and return a result. Based on expected computational complexity, a maximum turnaround time of one minute is justified due to the following:

- Loading Mars terrain data and initializing rover avatars requires data retrieval and preprocessing.
- AI computations, including terrain analysis and decision-making, must be processed efficiently.

- Final output generation, such as water detection reports and AI performance metrics, must be computed and presented to the user.

Given these factors, a maximum turnaround time of one minute ensures that users receive feedback without significant delays, while allowing for complex AI computations to execute effectively.

Restart Time: Restart time differs from turnaround time as it includes hardware initialization and software boot-up. Since the system relies on multiple hardware components (such as a server, local processing units, and possible cloud services), restart time must account for:

- System boot-up time for each piece of hardware (typically 30-60 seconds for a workstation).
- Loading of Mars terrain data and AI models (similar to turnaround time).
- Initialization of network services or cloud-based features.

Given these factors, the expected restart time is approximately 2-3 minutes, ensuring that all system components are fully initialized before a new simulation can begin.

4. Screen Layouts

4.1. Table of Contents of Mockups

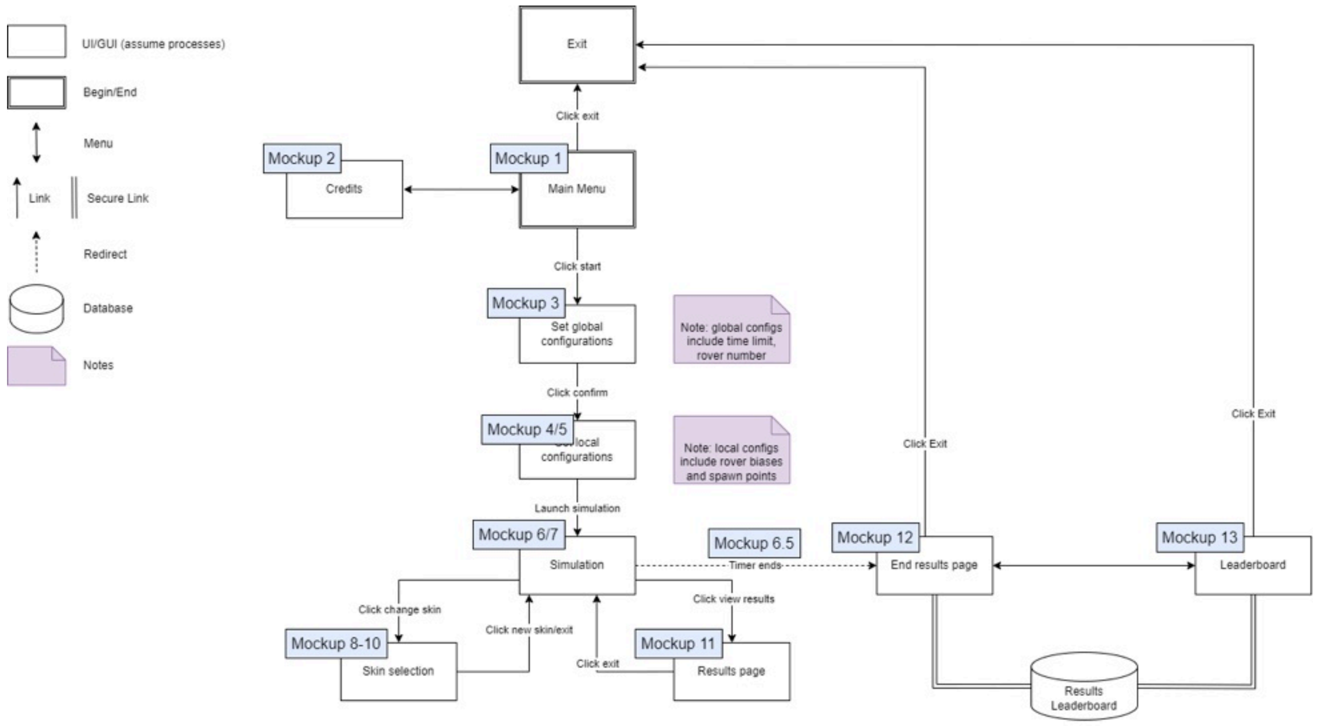
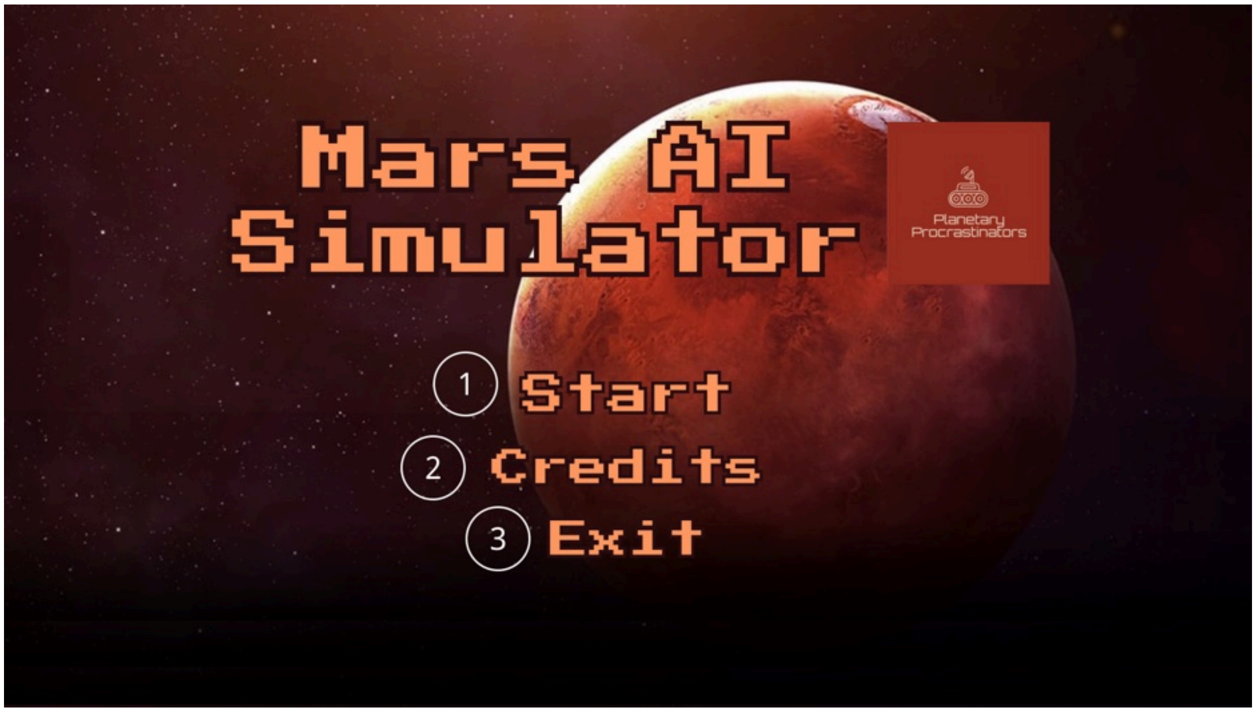


Figure 3: The figure above labels the storyboard with the relevant mockups. Certain sections have more than one mockup because there are responsive elements that need to be modeled visually.

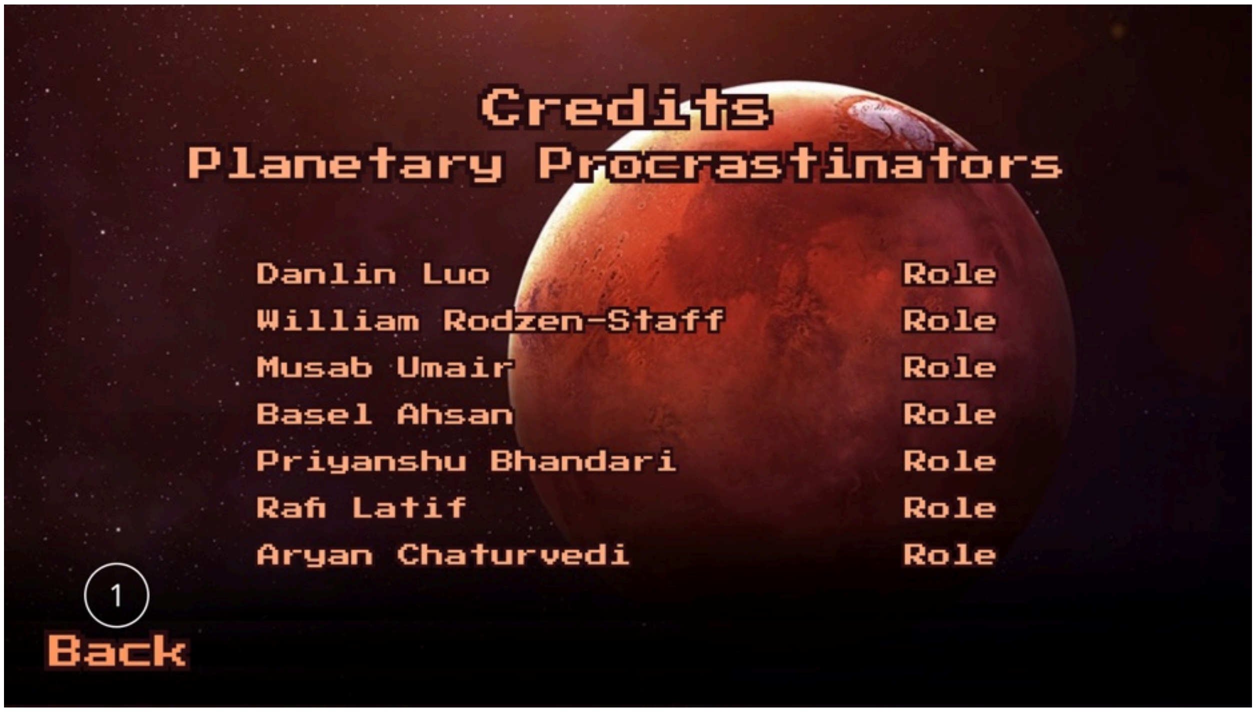
4.2. Mockup Screens

Mockup 1 - Main Menu



1. Pressing the start button launches a new game to the user and takes them to the start menu, displayed in Mockup 3.
2. Pushing the credits button takes the user to the credits menu, displayed in Mockup 2.
3. Pushing this button exits the game and closes the application.

Mockup 2 - Credits

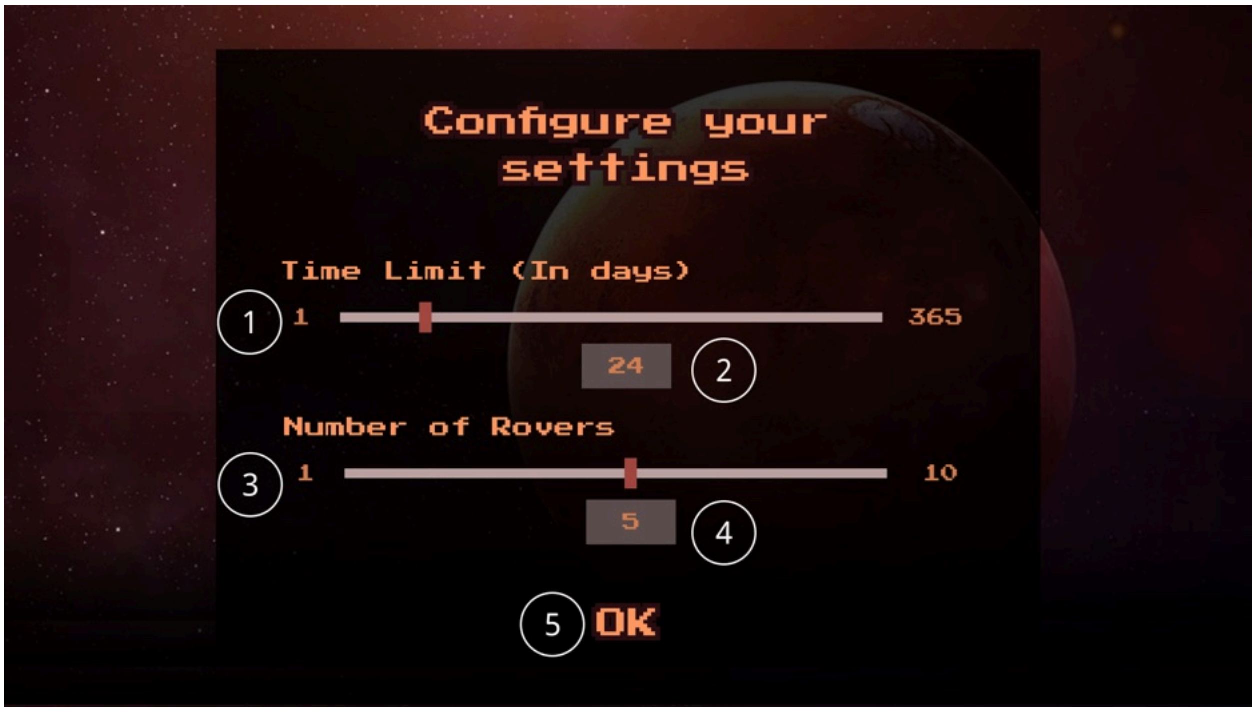


Credits
Planetary Procrastinators

Danlin Luo	Role
William Rodzen-Staff	Role
Musab Umair	Role
Basel Ahsan	Role
Priyanshu Bhandari	Role
Rafi Latif	Role
Aryan Chaturvedi	Role

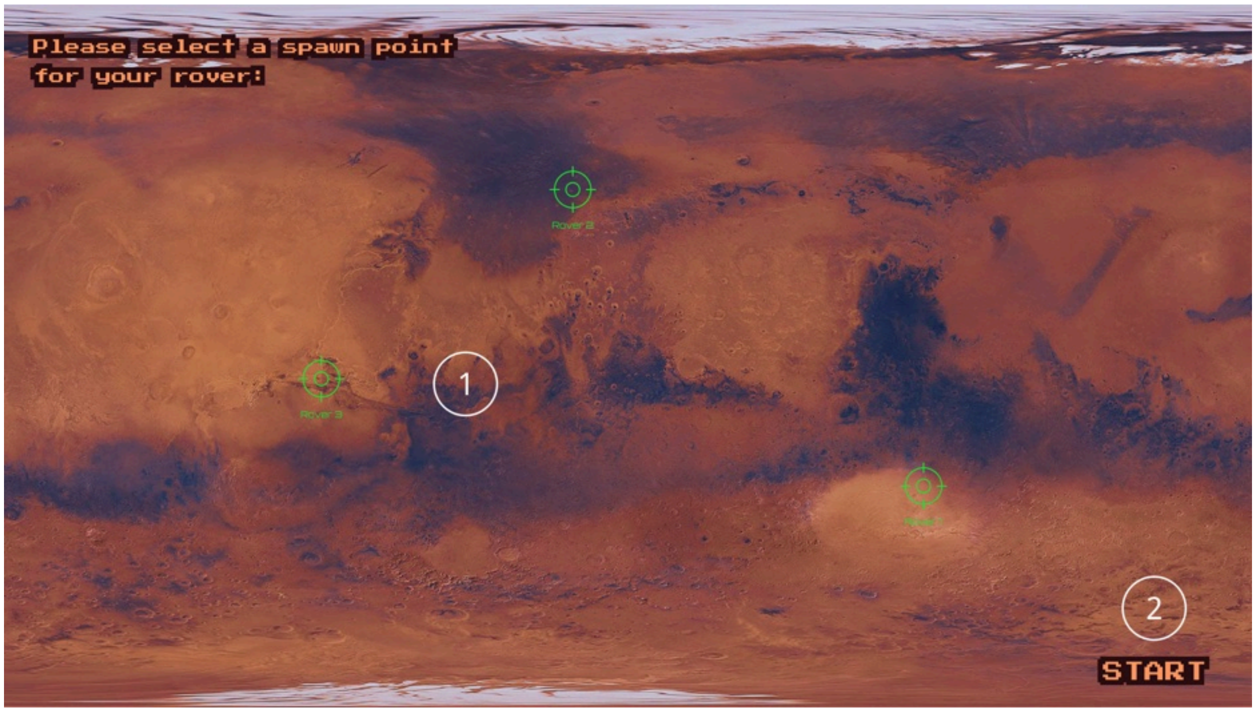
1. Pressing this button takes the user back to Mockup 1 - Main Menu.

Mockup 3 - Global Configurations



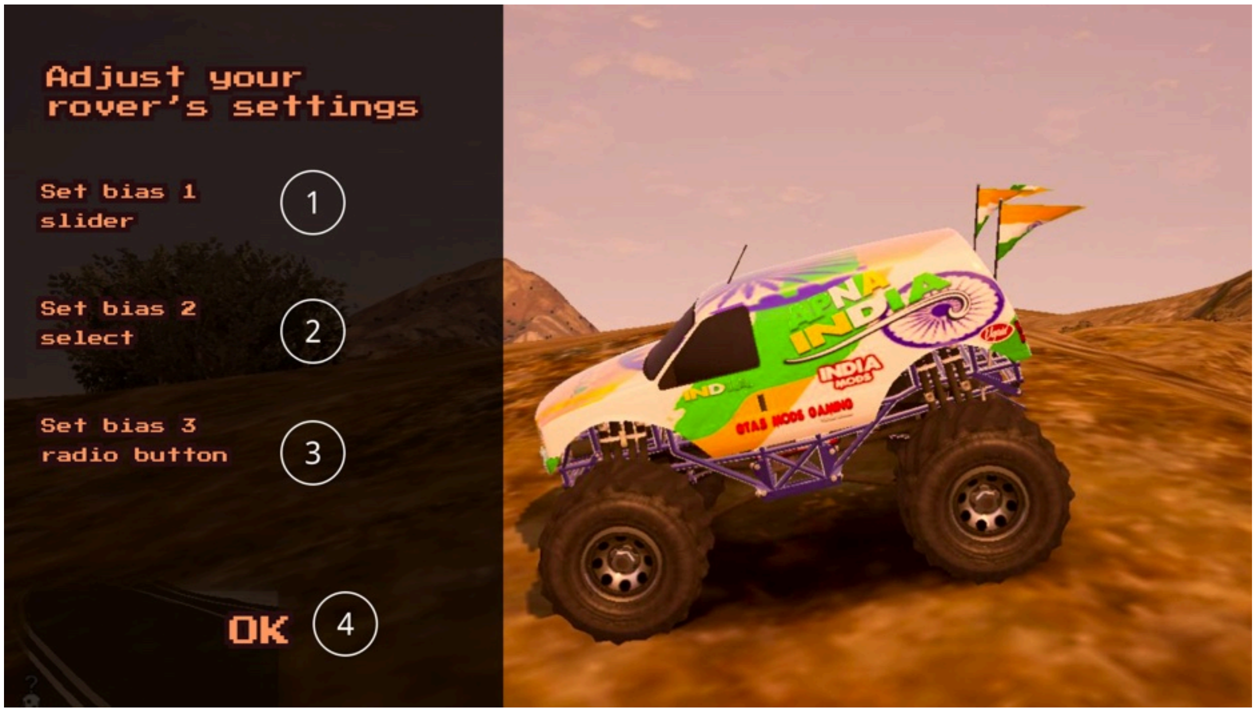
1. Adjusting this slider allows the user to choose their time limit in days.
2. The input text field allows the user to type in their desired time limit in days.
3. Adjusting this slider allows the user to choose their number of rovers in the simulation.
4. The input text field allows the user to type in their desired number of rovers.
5. Pressing OK takes the user to Mockup 4 - Local Configurations (Spawn Point).

Mockup 4 - Local Configurations (Spawn Point)



1. Clicking anywhere on the map instantiates a rover on a location of the map and takes the user to Mockup 5 - Local Configurations (Rover Biases) while the number of rovers on the map is less than the configured amount. Instantiated rovers are indicated by a green target on the map.
2. Pressing START is only available when all rovers are instantiated and takes the user to Mockup 6 - Simulation (Main).

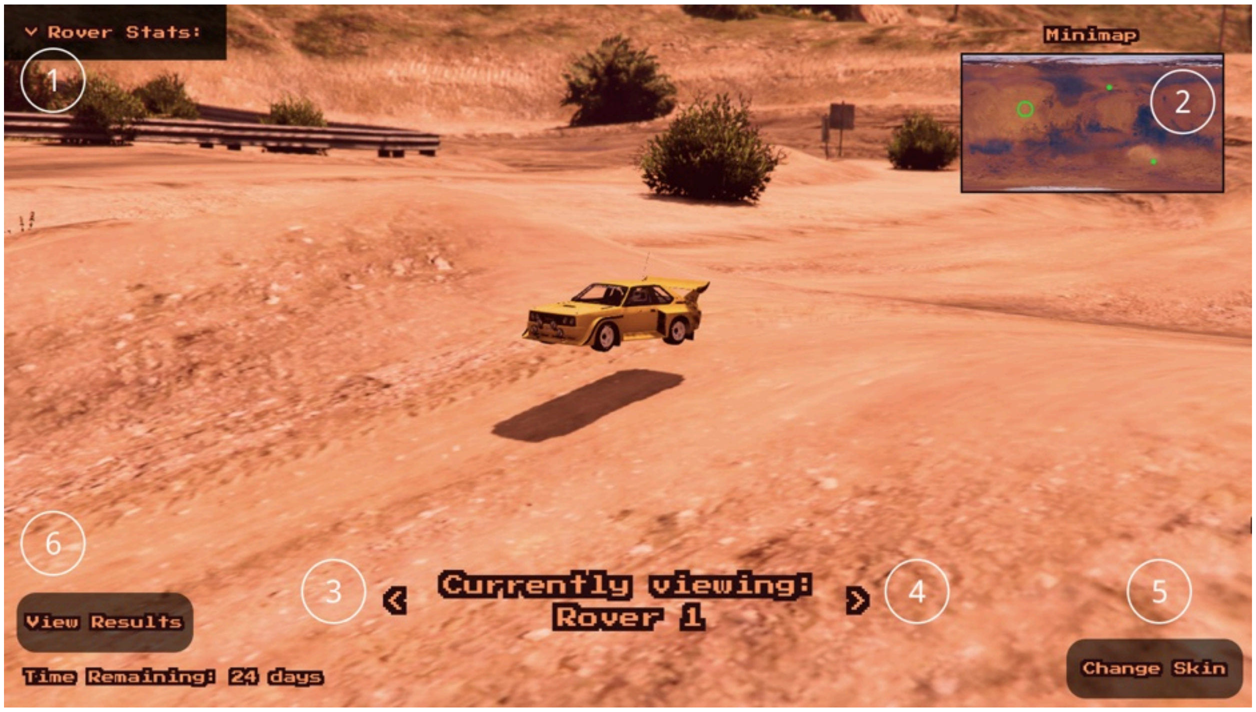
Mockup 5 - Local Configurations (Rover Biases)



1. Adjusting this example slider modifies the rover's property bias 1.

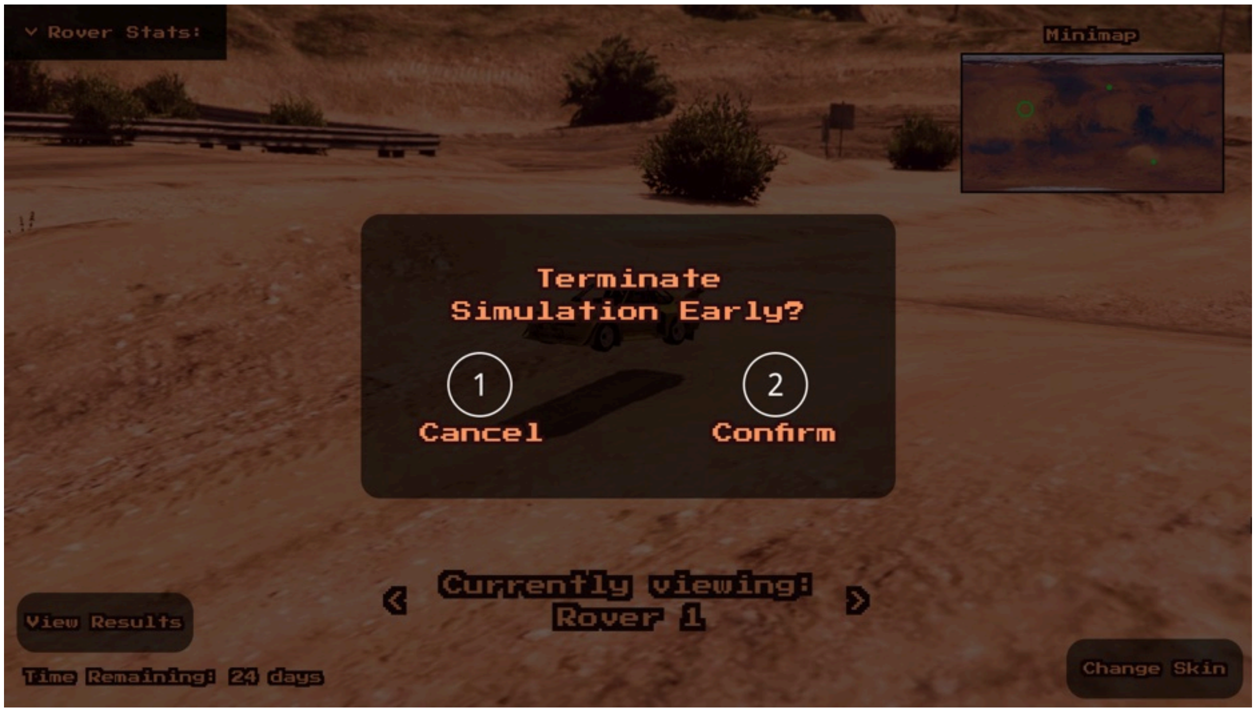
2. Choosing from this example select menu modifies the rover's property bias 2.
3. Selecting from these example radio button options modifies the rover's property bias 3.
4. Pressing OK confirms the rover's biases and takes the user back to Mockup 4 - Local Configurations (Spawn Point).

Mockup 6 - Simulation (Main)



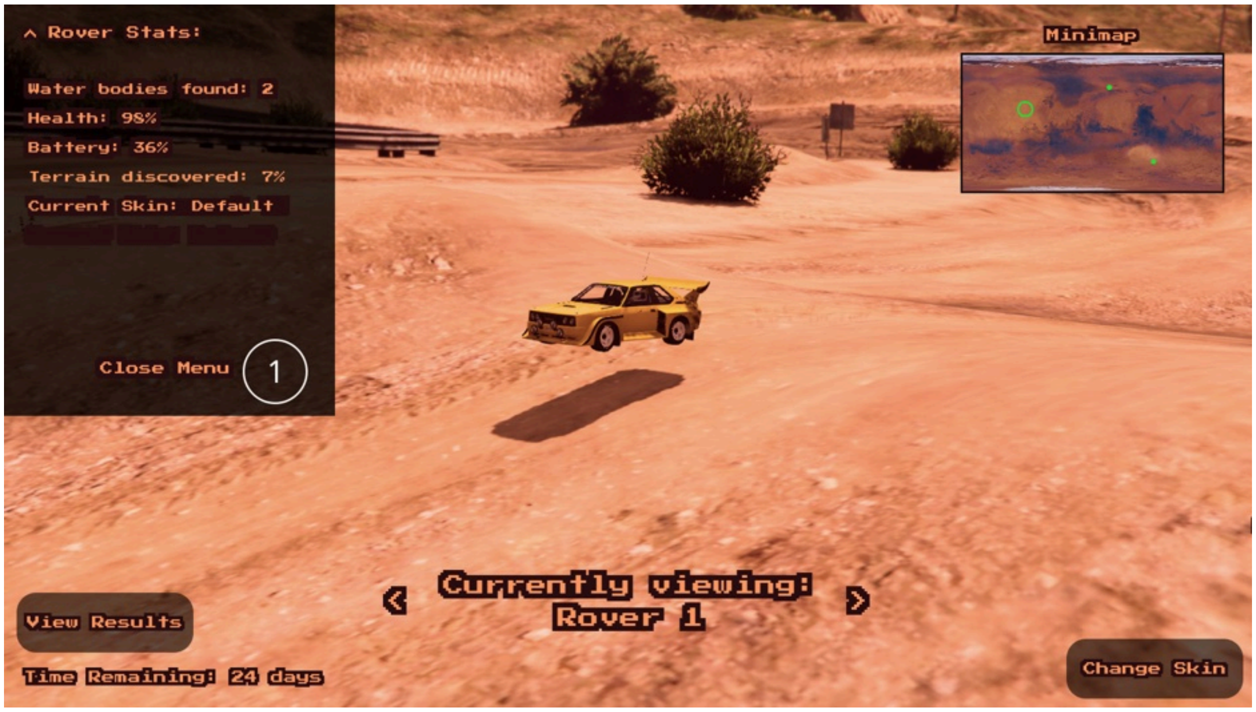
1. Pressing on the down arrow opens the current rover's stats. The menu layout is as shown in Mockup 7 - Simulation (Rover Stats).
2. The minimap displays the location of all rovers and the current rover with a larger circle and the others as dots.
3. Pressing the left arrow moves to view the previous rover.
4. Pressing the right arrow moves to view the next rover.
5. Pressing Change Skin opens the skin change menu as displayed in Mockup 8 - Simulation (Pre Skin Change)
6. Pressing View Results opens the temporary leaderboard as displayed in Mockup 11 - Results.
7. **For Mockup 6-11, the simulations, when the time limit is reached, the user will be automatically redirected to Mockup 12 - End Results. Additionally the user can choose to end the simulation early by pressing escape. Doing so will display a confirmation popup as seen in Mockup 6.5 - Simulation (Early Termination).

Mockup 6.5 - Simulation (Early Termination)



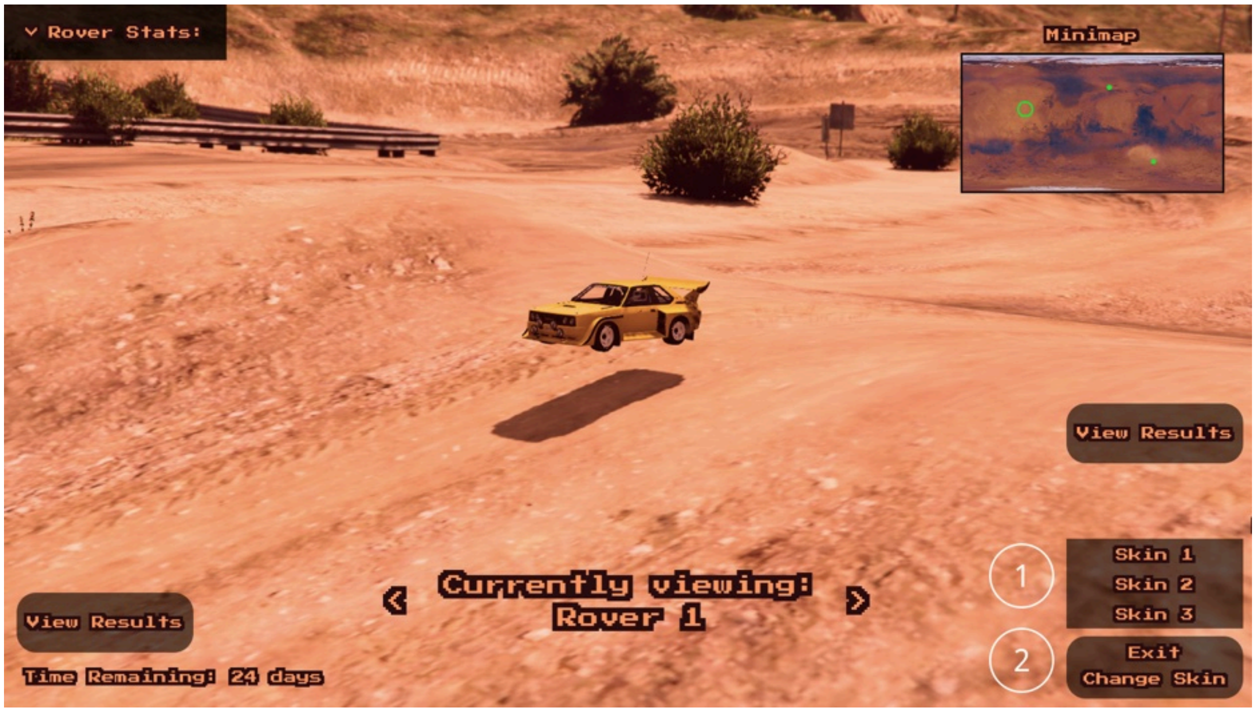
1. Pressing Cancel brings the user back to Mockup 6 - Simulation (Main).
2. Pressing Confirm takes the user to Mockup 12 - End Results.

Mockup 7 - Simulation (Rover Stats)



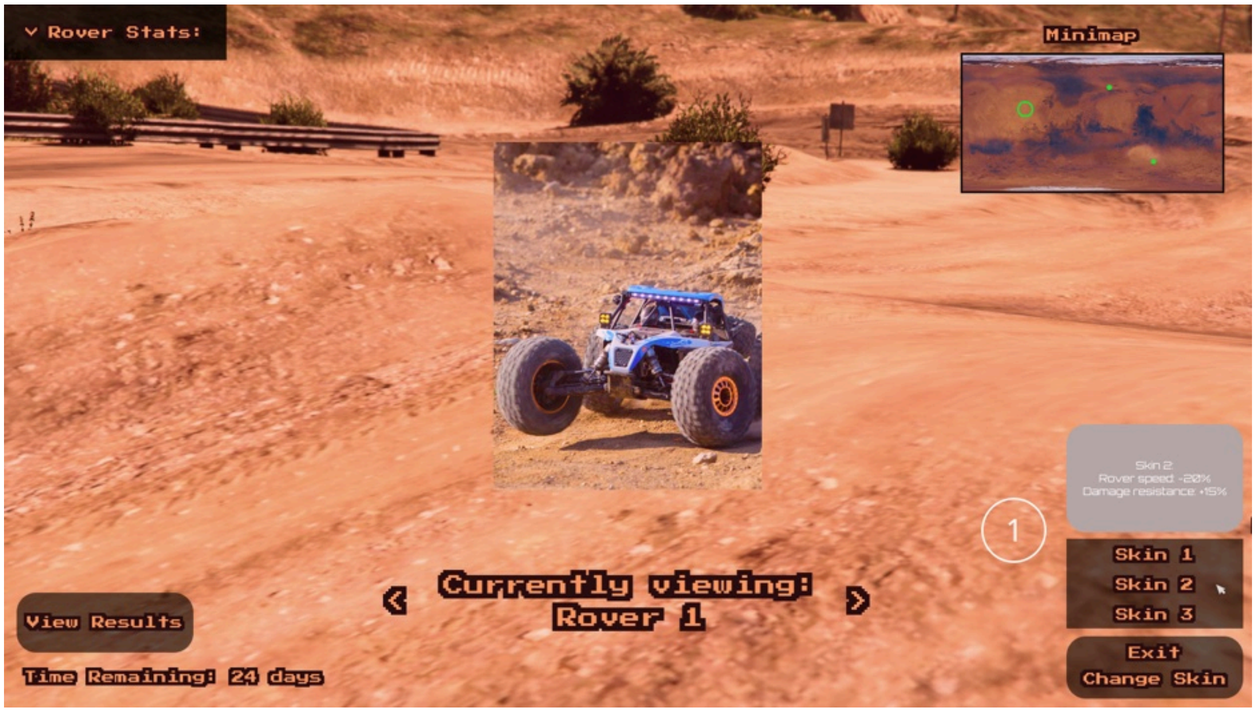
1. All other functionality is the same as Mockup 6 - Simulation (Main). However, pressing Close Menu allows the user to minimize the Rover Status Menu. The layout will look like Mockup 6 once again.

Mockup 8 - Simulation (Pre Skin Change)



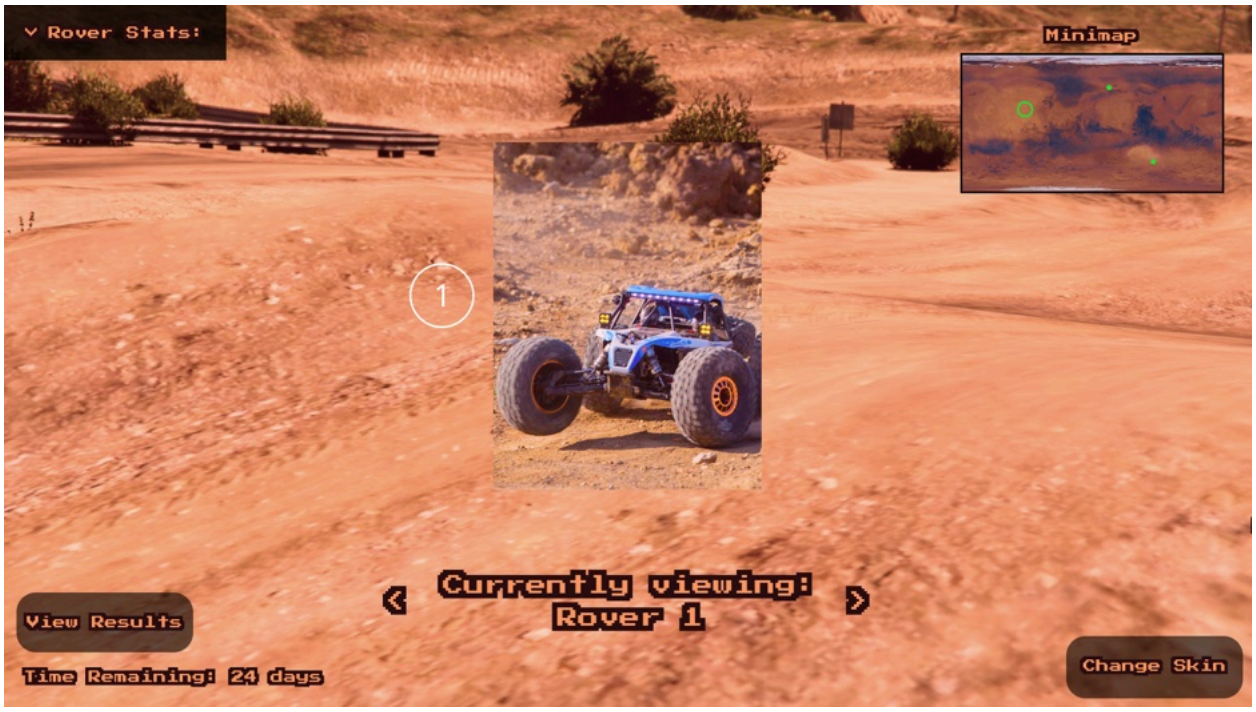
1. Interacting with the list of available skins will cause a UI change as displayed in Mockup 9 - Simulation (On Hover Skin Change).
2. Pressing Exit Change Skin minimizes the menu, taking the user back to Mockup 6 - Simulation (Main).

Mockup 9 - Simulation (On Hover Skin Change)



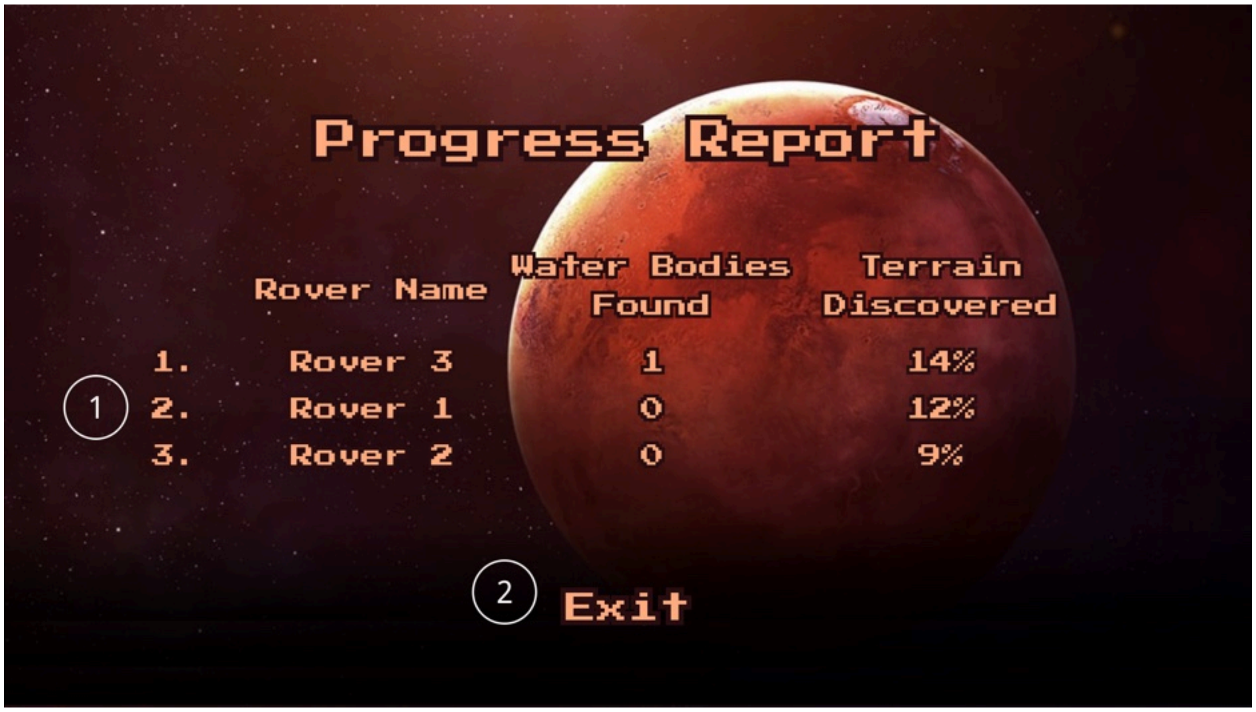
1. Hovering over a skin will display the gray menu which previews how the rover stats will be modified as well as the look of the sin. Pressing on a skin will select it for the rover, which takes the user to Mockup 10 - Simulation (Post Skin Change).

Mockup 10 - Simulation (Post Skin Change)



1. The UI is the same as Mockup 6 - Simulation (Main), with the difference in the visual appearance of the rover as well as its behavior following the updated effects of the new skin.

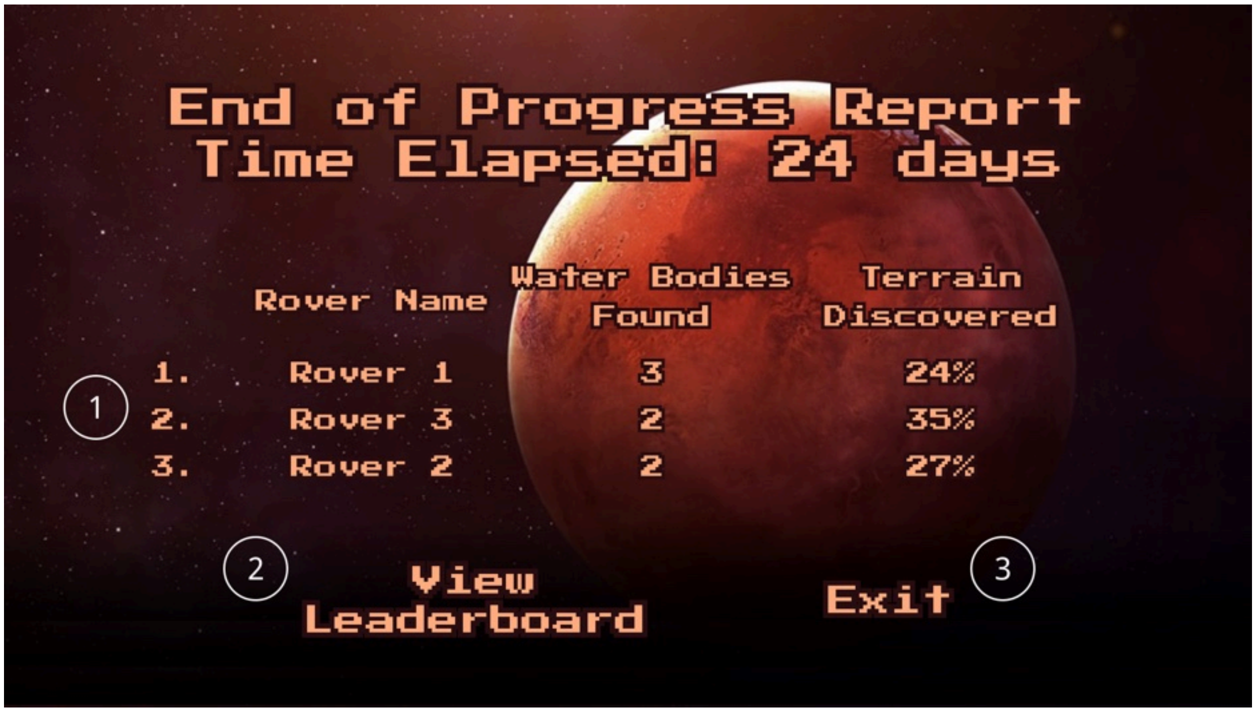
Mockup 11 - Results



	Rover Name	Water Bodies Found	Terrain Discovered
1.	Rover 3	1	14%
2.	Rover 1	0	12%
3.	Rover 2	0	9%

1. The leaderboard ranks the performance of all rovers in this simulation, first by the main goal which is the number of water bodies found. In case of ties, secondary metrics are used such as the amount of terrain discovered by the rover.
2. Pressing Exit takes the user back to Mockup 6 - Simulation (Main).

Mockup 12 - End Results

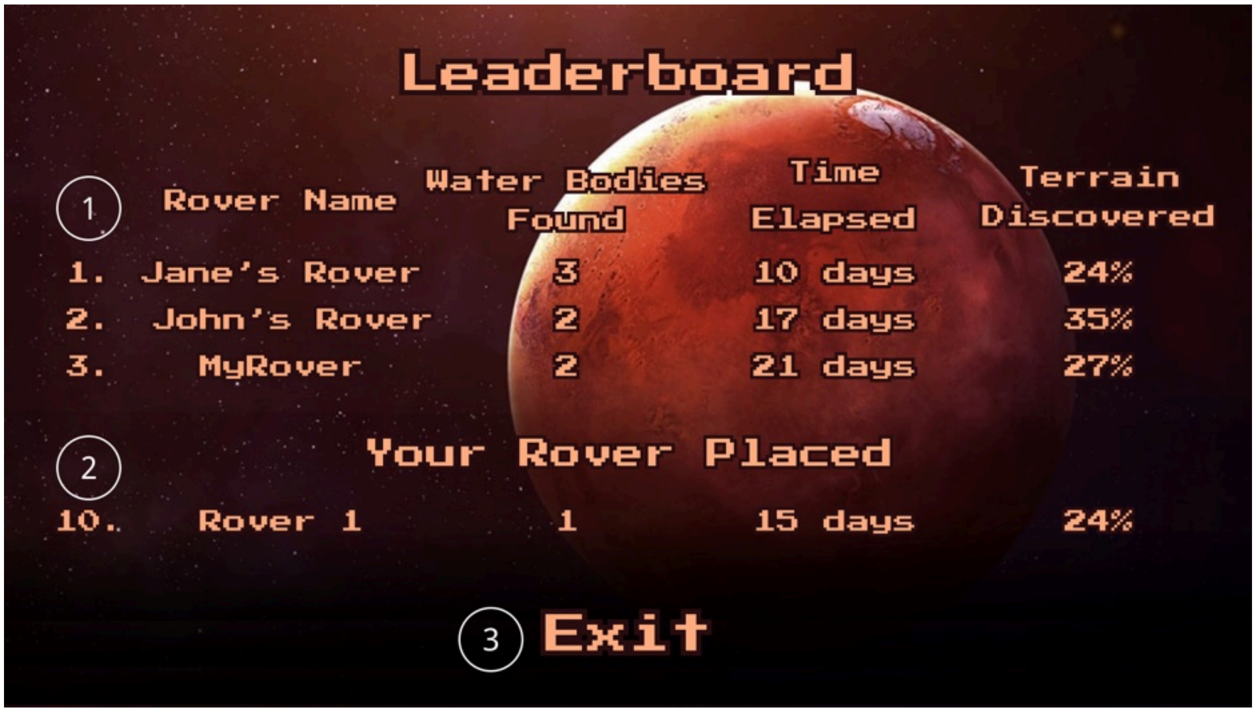


End of Progress Report
Time Elapsed: 24 days

	Rover Name	Water Bodies Found	Terrain Discovered
1.	Rover 1	3	24%
2.	Rover 3	2	35%
3.	Rover 2	2	27%

1. The leaderboard ranks the performance of all rovers in this simulation, first by the main goal which is the number of water bodies found. In case of ties, secondary metrics are used such as the amount of terrain discovered by the rover.
2. Pressing View Leaderboard takes the user to Mockup 12 - Leaderboard.
3. Pressing Exit quits the game and closes the application.

Mockup 13 - Leaderboard



	Rover Name	Water Bodies Found	Time Elapsed	Terrain Discovered
1.	Jane's Rover	3	10 days	24%
2.	John's Rover	2	17 days	35%
3.	MyRover	2	21 days	27%
10.	Rover 1	1	15 days	24%

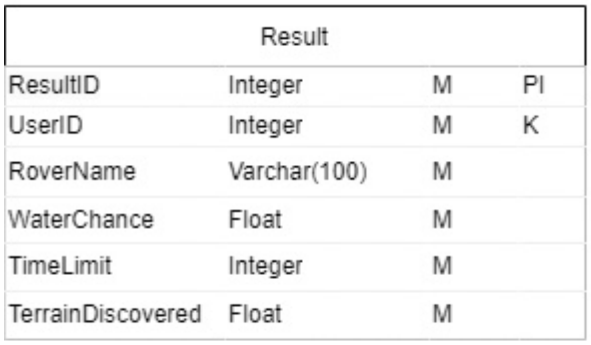
1. The upper section of the leaderboard shows the top n performing rovers of all time. Only the top performing rover of each session qualifies for this leaderboard. In this section, time limit is used as the first tiebreaker to rank rover performances.
2. The lower section of the leaderboard displays the rank of the user's top performing rover compared to all previous existing rovers.
3. Pressing Exit quits the game and closes the application.

5. Schemas & Queries

5.1. Defining File Structures

5.1.1. Database Diagram

The figure below displays the database diagram for the results leaderboard, which is only required to store the results, as these are the objects that need to be queried to display to the users at the end of the simulation.



Result			
ResultID	Integer	M	PI
UserID	Integer	M	K
RoverName	Varchar(100)	M	
WaterChance	Float	M	
TimeLimit	Integer	M	
TerrainDiscovered	Float	M	

Figure 4: The database diagram for the results leaderboard. Every result object corresponds to one rover within a particular simulation

5.1.2. Database Table Structure

Below is the table structure representing the database diagram of the results leaderboard from section 5.1.1.

Filed Name	Type	Length	Key, Primary, Sort	Constraints	Legal Values
ResultID	Integer		Primary Key	unique	Random number generated based on order of date created
UserID	Integer		Foreign Key	References Users	Valid UserID from Users table
RoverName	String	100 characters		Only alphanumeric values	Max 100 chars
WaterBodies	Integer		Sort by Decreasing	Valid range of 0 to 100	
TimeElapsed	Integer			Valid range of 1 to 365	Number of days
TerrainDiscovered	Float	2 decimals		Valid range of 0.00 to 100.00	Percentage rounded to 2 decimal places

5.1.3. Formatted Text Files

ResultID INTEGER PRIMARY KEY AUTOINCREMENT,
UserID INTEGER NOT NULL,
RoverName TEXT NOT NULL,
WaterChance REAL NOT NULL,
TimeLimit INTEGER NOT NULL,
TerrainDiscovered TEXT NOT NULL

5.1.4. Defined Operations

```
CREATE TABLE results (
        ResultID INTEGER PRIMARY KEY AUTOINCREMENT,
        UserID INTEGER NOT NULL,
        RoverName TEXT NOT NULL,
        WaterChance REAL NOT NULL,
        TimeLimit INTEGER NOT NULL,
        TerrainDiscovered TEXT NOT NULL
);
```

```
INSERT INTO results (UserID, RoverName, WaterChance, TimeLimit, TerrainDiscovered)
VALUES (1, 'Rover1', 75.5, 120, 'rocky');
```

5.2. Remote API Definitions

Results

Signature

POST results/add [result]
GET results/list

Description

POST: the API takes as input a result object in JSON format, where the fields of the object correspond to the result object in the database table structure. Expected response time: 1-2 seconds.
GET: the API retrieves all of the results of the leaderboard. Expected response time: 5 seconds.

Return format

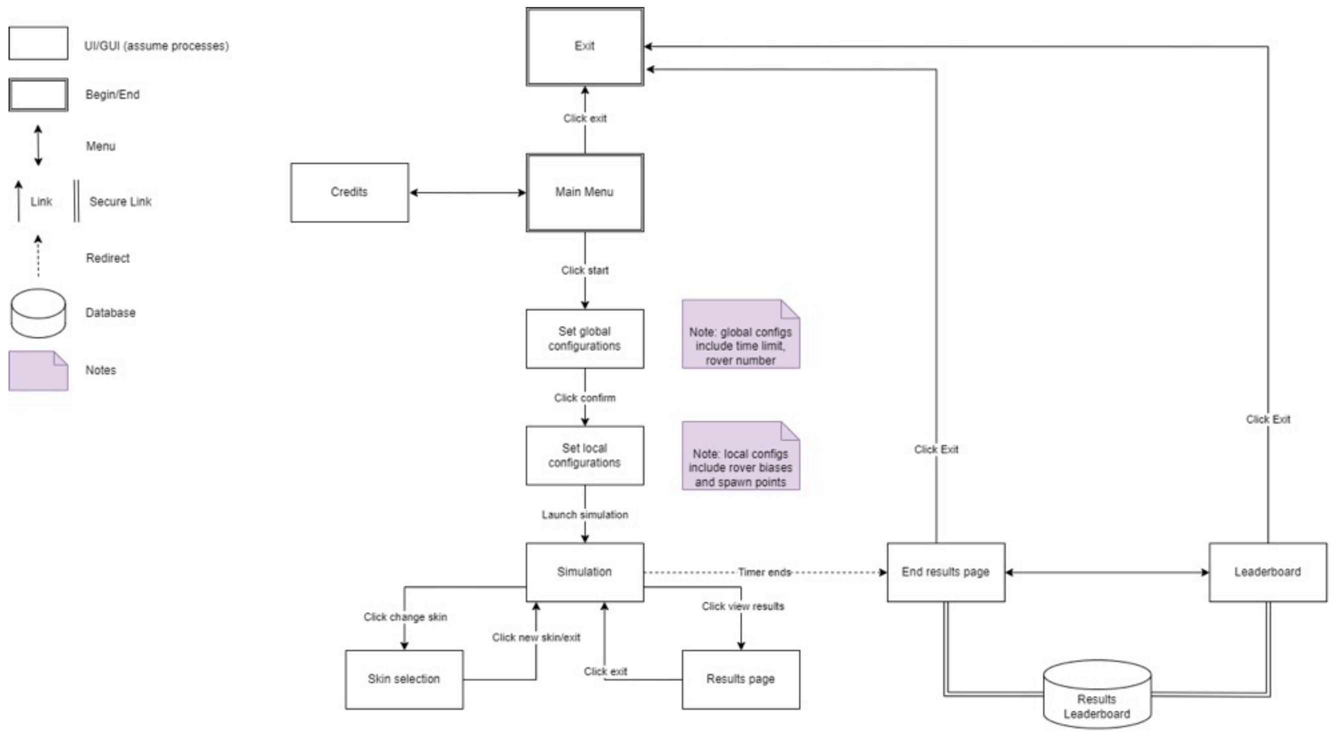
POST: Returns a message indicating success and failure
{“message”: “SUCCESS”}

GET: Returns a list of all the results

```
{ “result” :
    [
        {
            “ResultID”: 586152,
            “UserID”: 783120,
            “RoverName”:”MyRover”,
            “WaterBodies”: 2 ,
            “TimeElapsed”: 21,
            “TerrainDiscovered”: 32.16
        },
        {
            “ResultID”: 523415,
            “UserID”: 791849,
            “RoverName”:”BetterRover”,
            “WaterBodies”: 1,
            “TimeElapsed”: 29,
            “TerrainDiscovered”: 35.16
        }
    ]
}
```

6. Critical Sections

6.1. Table of Contents



The critical section is located from the “Set global configurations” section of the storyboard to the “End results page” and “Leaderboard” sections.

6.2. Critical Sections

The title of this critical section is the Simulation. It is the main interactive component of the application, from the configuration of the settings to the user changing the skins and camera during the simulation itself.

The purpose of this section is to ensure that the visual display of the application is responsive to the user and results in proper communication to the backend. It is the most critical section of the application because without this section, the simulation cannot even start without setting up configurations. Without the simulation portion, no data on the rovers will be able to be stored and used by the stakeholders, as the rovers require the terrain and user input to make decisions and search for water. Finally, even if the visuals are properly stored, without the data being sent to the backend, the rover AIs cannot properly learn and function to return correct data. It is highly important to ensure the correctness of this process.

The proposed solution is a clear flow of the simulation runs, and how it must stay connected and running from the start, when the user presses Start from the main menu, to the end, when the user terminates the simulation early or the time limit has been reached. Below is the state diagram depicting the states that the application travels in between depending on user inputs.

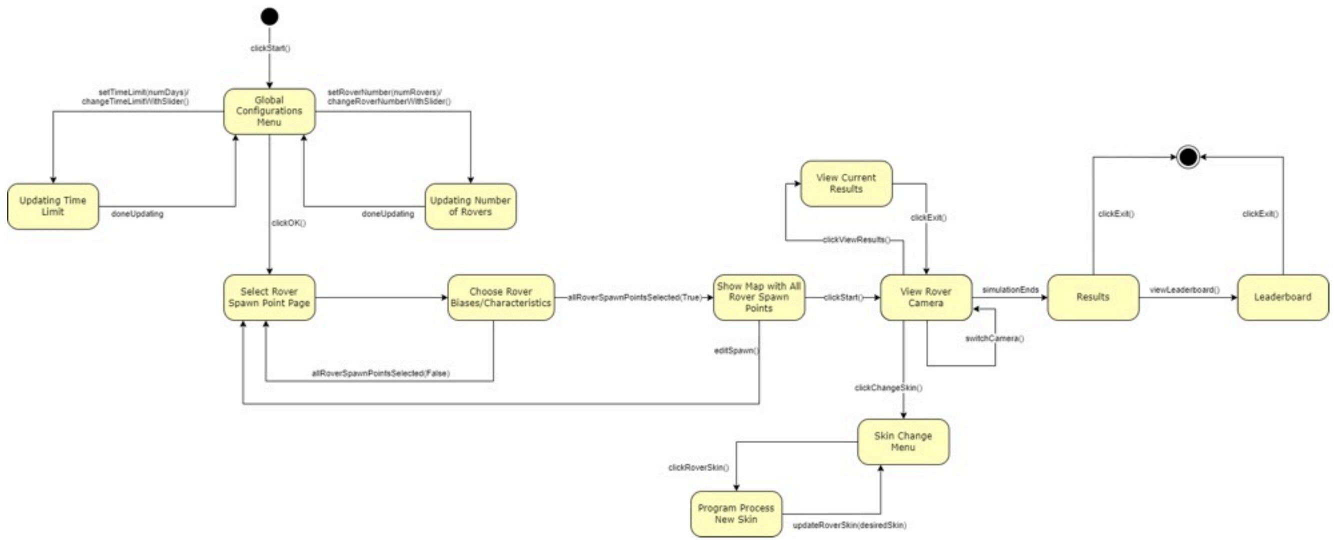

clickStart()
Global Configurations Menu
setTimeLimit(numDays)/ changeTimeLimitWithSlider()
setRoverNumber(numRovers)/ changeRoverNumberWithSlider()
Updating Time Limit
doneUpdating
Updating Number of Rovers
doneUpdating
clickOK()
Select Rover Spawn Point Page
Choose Rover Biases/Characteristics
allRoverSpawnPointsSelected(True)
allRoverSpawnPointsSelected(False)
Show Map with All Rover Spawn Points
editSpawn()
clickStart()
View Current Results
clickViewResults()
clickExit()
View Rover Camera
switchCamera()
simulationEnds
Results
viewLeaderboard()
Leaderboard
clickExit()
clickExit()
clickChangeSkin()
Skin Change Menu
clickRoverSkin()
Program Process New Skin
updateRoverSkin(desiredSkin)

Figure 5: The state diagram displays the state that the program goes through as the user sets up then runs the simulation.

The above diagram represents the flow of the system, from the configuration phase to the final results. The process begins when the user clicks the Start button from the main menu, which leads to the Global Configurations Menu. Users can continuously adjust settings, including the simulation time limit and the number of rovers. For both settings, the program enters a state of updating these settings in the backend before allowing the users to modify these variables to ensure correctness.

Once configurations are finalized, clicking OK moves the user to the Select Rover Spawn Point Page, where they determine where the rovers will begin their mission. The user is required to initialize the locations and characteristics of rovers. Only once all rovers are initialized can the system proceed to the past this stage. Otherwise, the user remains in the selection phase until all points are assigned.

The next step, Show Map with All Rover Spawn Points, presents a visual overview of where the rovers will start. At this stage, users can either edit spawn points if needed or proceed by clicking Start Simulation.

Once the simulation begins, the system transitions to the View Rover Camera state, where users can follow the rover's perspective.At this state, they have several options: they can switch between different rover cameras, view the current results, or exit either through early termination or end of time limit. Additionally, they have the option to change the appearance of the rover by clicking Change Skin, which takes them to the Skin Change Menu. If they proceed with changing the skin, the Program Process New Skin state is activated, updating the rover’s appearance before returning to the View Rover Camera.

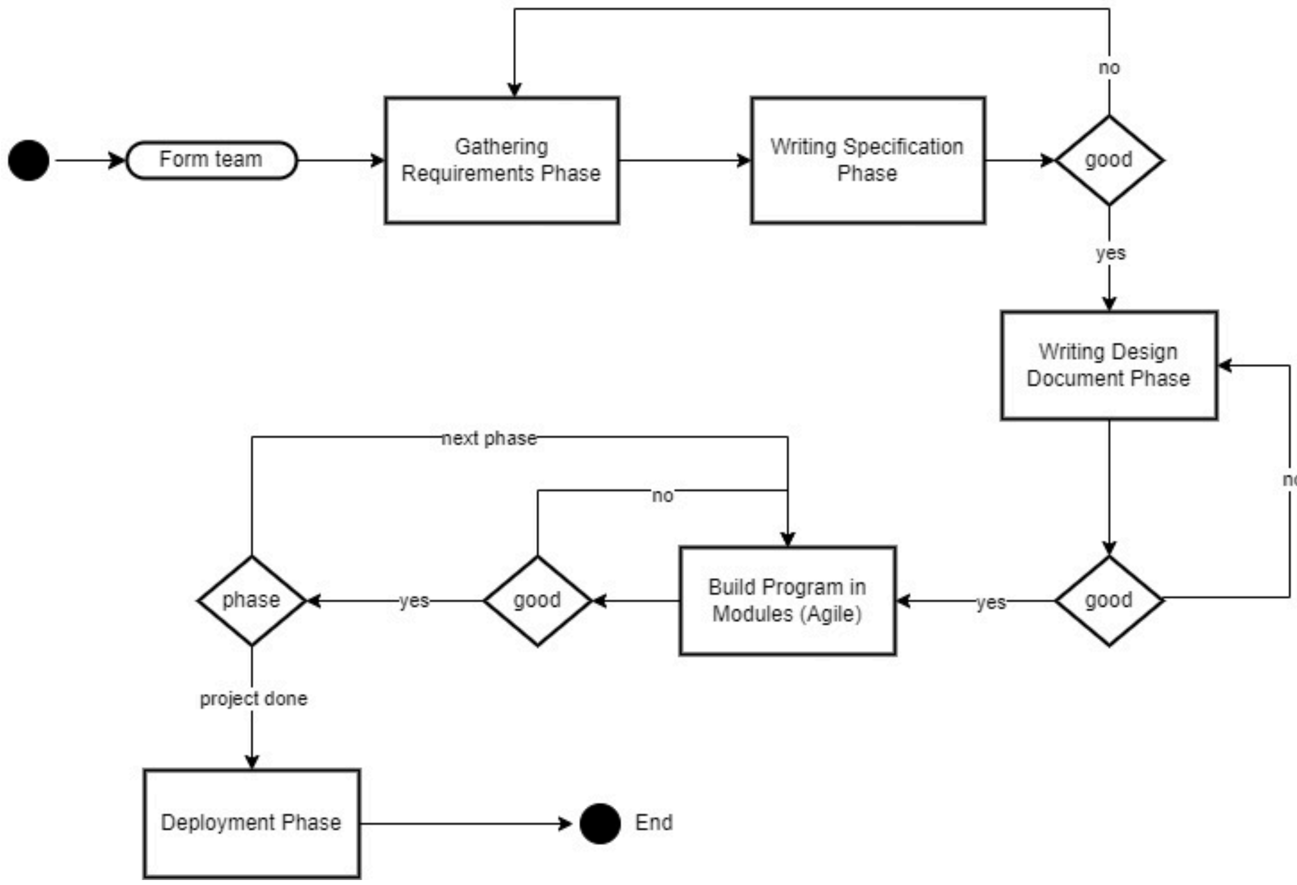
During the simulation, users may check the View Current Results screen to see a current leaderboard of the performance of all current rovers. When the simulation ends, it moves to the Results state, where users can analyze final outcomes. From this stage, they can choose to view the Leaderboard or exit the simulation. The Leaderboard as previously described displays past results that the users may compare their rovers to.

This stage diagram outlines the strict flow of events that must take place in order to ensure the correctness of the application. It ensures that users have full control over configuration and monitoring while maintaining a structured approach to simulation management.

7. Effort Estimation

7.1. Optimal Workflow Graph

The decided workflow for this project will be waterfall iterative by module, where each module will be worked using Agile methodologies. Below is a flowchart depicting the proposed workflow.



Form team
Gathering Requirements Phase
Writing Specification Phase
good
no
yes
Writing Design Document Phase
no
good
yes
Build Program in Modules (Agile)
good
no
yes
phase
next phase
project done
Deployment Phase
End

Figure 6: This flowchart depicts the workflow of the project from the formation of the team to the finished product. Each module will be built using Agile methodologies.

The proposed workflow offers several benefits, enhancing flexibility, efficiency, and product quality. By breaking down the project into independent modules, developers can work in parallel, reducing dependencies and accelerating development. The ability to work in parallel is especially critical for a team of our size. Each module undergoes incremental improvements through iterative sprints, allowing for early issue detection and continuous refinement.

Following Agile principles ensures that feedback is incorporated early, reducing costly rework later in development. Additionally, Agile's continuous integration and testing practices enhance software stability, as each module is validated before integration. The project first being divided into modules allows for a more structured schedule with the benefits of continuous integration rather than developing the entire application through Agile methodologies.

The workflow also improves adaptability to changing requirements. Because each module is developed independently, teams can pivot quickly without disrupting the entire system. Furthermore, it allows teams to help each other out depending on their relative speed of development. This makes it ideal for projects where priorities evolve frequently. Moreover, the modular approach enhances risk management, as issues in one module do not necessarily impact others, allowing teams to isolate and resolve problems efficiently.

Ultimately, the dynamic aspects of this workflow while maintaining certain aspects of structure makes it a great fit for the development of the Mars AI Simulator.

7.2. Effort Estimation Table

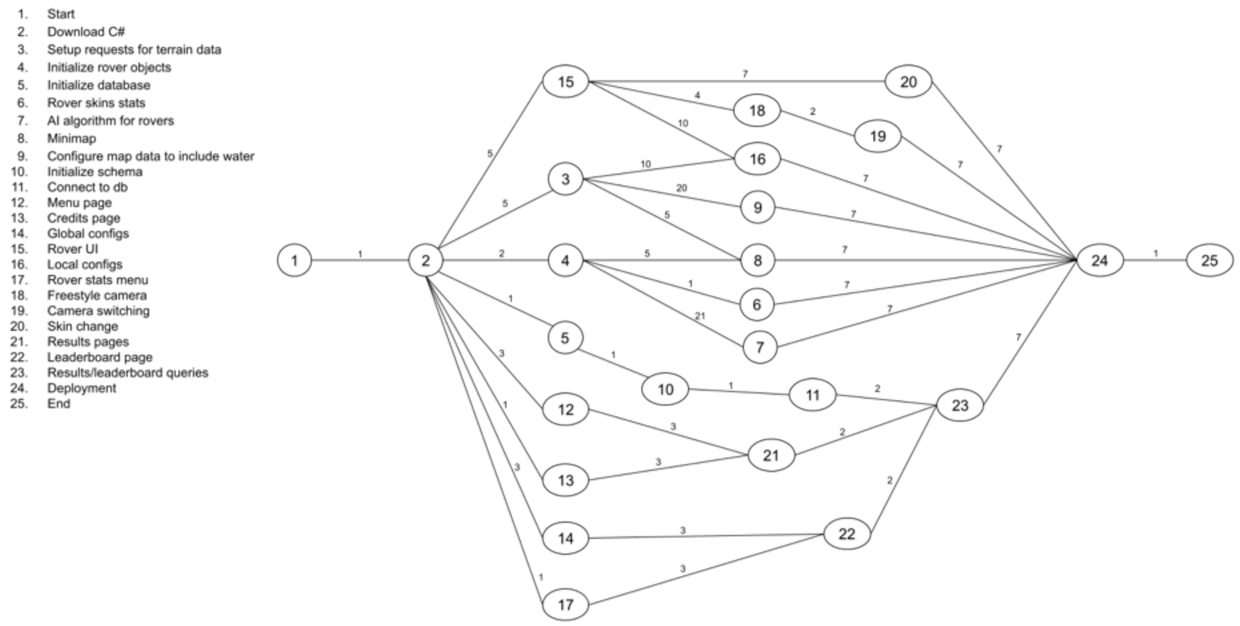


Figure 6: The graph displays the estimated effort required to complete the application, where each node represents a task as indicated on the left. Each edge is the estimated amount of time required for completion in days.

The graph above depicts the effort estimation for the project in addition to the requirements for each node. The same estimations have also been translated into a table format as seen below. We are assuming that the design document for this project will be completed in parallel as the team follows the development path.

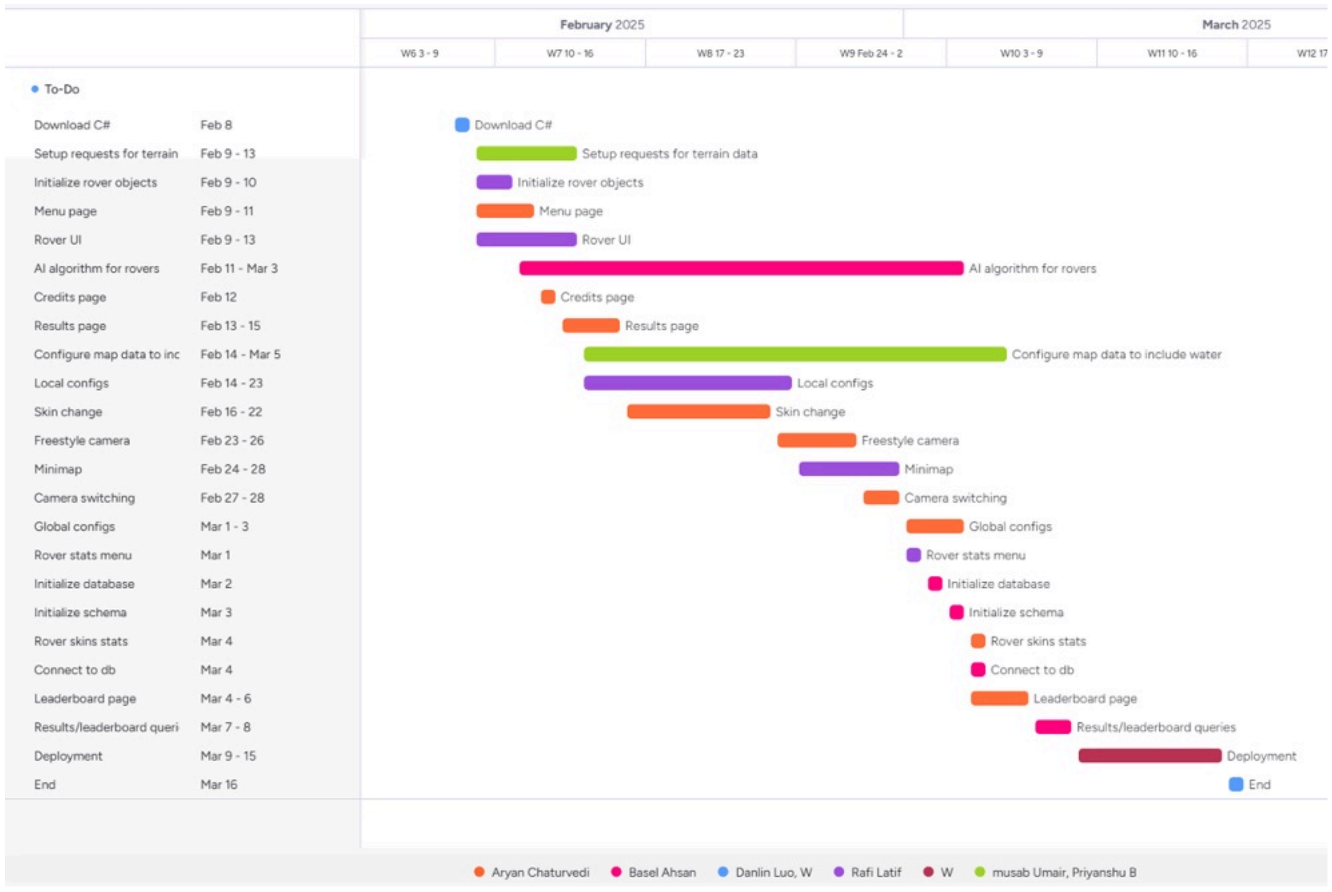
Node Number	Task Name	Duration (Days)	Pre-Conditions
1	Start	0	
2	Download C#	1	Start
3	Setup requests for terrain data	5	Download C#

4	Initialize rover objects	2	Download C#
5	Initialize database	1	Download C#
6	Rover skins stats	1	Initialize rover objects
7	AI algorithm for rovers	21	Initialize rover objects
8	Minimap	5	Setup requests for terrain data, Initialize rover objects
9	Configure map data to include water	20	Setup requests for terrain data
10	Initialize schema	1	Initialize database
11	Connect to db	1	Initialize schema
12	Menu page	3	Download C#
13	Credits	1	Download C#
14	Global configs	3	Download C#
15	Rover UI	5	Download C#
16	Local configs	10	Setup requests for terrain data, Rover UI
17	Rover stats menu	1	Download C#
18	Freestyle camera	4	Rover UI
19	Camera switching	2	Freestyle camera
20	Skin change	7	Rover UI
21	Results pages	3	Menu page, Credits
22	Leaderboard page	3	Global configs, Rover stats menu
23	Results/leaderboard queries	2	Connect to db, Results pages, Leaderboard page
24	Deployment	7	Rover skin stats, AI algorithm for rovers, Minimap, Configure map data to include water, Local configs, Camera switching, Skin change, Results/leaderboard queries
25	End	1	Deployment

The total estimated time is 110 days, the optimal estimated time is 34 days, and the practical estimated time is 37 days.

7.3. Gantt Chart

The Gantt Chart was created following the division of tasks and estimated time required for each task. The assignee for each task is subject to vary depending on each individual's progress. The Gantt Chart was created assuming that all developers will be able to work 7 days a week.



Download C#	Feb 8
Setup requests for terrain	Feb 9 - 13
Initialize rover objects	Feb 9 - 10
Menu page	Feb 9 - 11
Rover UI	Feb 9 - 13
AI algorithm for rovers	Feb 11 - Mar 3
Credits page	Feb 12
Results page	Feb 13 - 15
Configure map data to inc	Feb 14 - Mar 5
Local configs	Feb 14 - 23
Skin change	Feb 16 - 22
Freestyle camera	Feb 23 - 26
Minimap	Feb 24 - 28
Camera switching	Feb 27 - 28
Global configs	Mar 1 - 3
Rover stats menu	Mar 1
Initialize database	Mar 2
Initialize schema	Mar 3
Rover skins stats	Mar 4
Connect to db	Mar 4
Leaderboard page	Mar 4 - 6
Results/leaderboard queri	Mar 7 - 8
Deployment	Mar 9 - 15
End	Mar 16

Figure 7: The Gantt Chart outlining the expected timeline of the project completion with all members assigned various tasks. Each task is color coded according to the assigned team, where the task's assignee is subject to change going forward.

This projected Gantt Chart assumes that developers will work according to the estimated time requirements and be able to pick up a new task as soon as the previous one is completed. With these assumptions, the estimated date of completion is March 16, 2025, which gives for additional time before the deadline in case there are unexpected issues that require more time to resolve, or changes in requirements to the project must be revised. Additional features may also be considered such as UI enhancements if there is remaining time, which will require a future reassessment of the project.

7.4. Estimated Project Cost

The main estimated cost of this project will be to host a database in order to virtually store the results. The most cost effective manner of hosting a database would be to rent a cloud instance and host the database through the instance, such as Amazon Lightsail. It would also be doable due to the simplicity of the database. The costs are estimated for the duration of the development of the project, which is estimated to be 3 months. The total as shown below comes out to $33.00.

Element	Unit Cost	Number of Units	Additional	Subtotal

			Costs	
Cloud instance	$11/month	3 months	$0.00	$33.00
			TOTAL:	$33.00

8. Resources

Appendix A: References

Project Inspiration

Much of the format of the project as well as inspiration for the rendering of the terrain was sourced through an online video named "I Tried Creating a Game Using Real-World Geographic Data" by Sebastian Lague.

Estimated Costs

The estimated costs of the project were referenced from creating an Amazon Lightsail instance in Canada. Assuming Linux/Unix as the platform, using Ubuntu as the OS, and using a dual-stack network, we determined that a package with 1 GB of memory and 40 GB of storage would be enough for the project. This package costs $7 USD per month, which after converting and rounding up, resulted in an estimated monthly cost of $11 CAD.