CS/SSW 555 Scrum Team 6

by

Hazem Abo-Donia, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis, Thomas Ung

Stevens.edu

May 1, 2025

CS/SSW 555 Scrum Team 6

Hazem Abo-Donia, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis, Thomas Ung
Stevens.edu

Table 1: Document Update History

Date	**Updates**
02/17/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Created the initial iteration of the project. • Completed planning and role assignments. • Established Jira board for task tracking. • Drafted initial functional and non-functional requirements.
02/24/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Updated weekly Reports. • Added use cases. • Added user stories.
03/13/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Updated Chapter: Weekly Reports (added meeting notes and Sprint 1 progress). • Updated Chapter: Implementation (added Sprint 1 implementation details).
04/03/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Updated Chapter: Weekly Reports (added Sprint 2 summary, pair programming reflection, CI/CD improvements, Sprint 2 review, and planning for Sprint 3). • Updated Chapter: Implementation (added Sprint 2 implementation details, mini-game integration, and CI/CD pipeline improvements). • Updated Chapter: Document Update History (added Sprint 2 update log).

Table 1: Document Update History

Date	Updates
04/17/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Updated Chapter: Weekly Reports (added Sprint 3 execution summary, demonstration details, sprint review, and Sprint 4 planning). • Added snapshots of the Kanban board, burndown chart, unit test results, and Sprint 4 planning Jira board. • Updated Chapter: Document Update History (added Sprint 3 update log).
05/01/2025	Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, Ryan Davis: • Updated Chapter: Weekly Reports (added Sprint 4 recap, team reflection, and project wrap-up). • Added Chapter: Project Reflection to summarize project completion and team experience. • Updated Chapter: Document Update History (added Sprint 4 update log).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Team Declaration

– Hazem Abo-Donia

The objective of this project is to develop an educational space exploration game that introduces players to real-world space mission challenges, astronaut teamwork dynamics, and resource management. The game will incorporate key scientific principles, making it engaging and informative for K-12 students. Our team aims to balance realism and interactivity, ensuring an immersive learning experience while meeting client expectations.

The game will focus on **teaching space mission logistics, astronaut cooperation, and survival decision-making in harsh environments.** Players will learn about topics such as **oxygen regulation, energy distribution, psychological challenges in space, and emergency procedures.** Through interactive decision-making and problem-solving tasks, students will **develop a deeper understanding of space exploration and teamwork.**

1.1 Project Links

For easy access to our project resources, use the links below:

- **GitHub Repository**: Overleaf Document
- **Jira Board**: Jira Project Board

1.2 Mission Statement

Mission Statement

Develop an interactive and educational space exploration game that simulates astronaut missions, focusing on teamwork, resource management, and survival in extreme environments.

Key Drivers

- Realistic space mission simulation incorporating scientific accuracy

- Engaging gameplay that fosters teamwork and decision-making
- Accessibility for K-12 students with an intuitive interface
- Alignment with agile development practices for iterative improvements

Key Constraints

- Limited development time within the semester
- Need for scientific accuracy without overcomplicating gameplay
- Ensuring smooth functionality on web-based platforms
- Balancing educational content with entertainment value

Milestones

- Submit initial project proposal
- Receive client feedback and refine proposal
- Complete basic game mechanics prototype
- Implement resource management and astronaut interactions
- Conduct internal playtesting and refine UI
- Finalize game mechanics and complete QA testing
- Submit final project for evaluation

1.3 About the Team

1.3.1 Hazem Abo-Donia (Scrum Master)

Hazem is responsible for ensuring smooth project execution, maintaining communication with clients, and organizing team sprints. Hazem keeps the team on track to meet deadlines effectively.

1.3.2 Thomas Ung (Game Mechanics & Backend Developer)

Thomas is responsible for designing core gameplay mechanics, astronaut tasks, and resource management. He will implement backend logic to support game functionality and ensure that the interactive features align with the mission objectives.

1.3.3 Mohamed Bengabsia (Frontend & UI/UX Developer)

Mohamed is in charge of designing and implementing the game's user interface and experience. He will ensure that the visuals are engaging, intuitive, and accessible for K-12 students, making learning an interactive experience.

1.3.4 Gianna Cerbone (Game Logic & Testing Lead)

Gianna will develop the game's logic flow, debugging, and testing frameworks. She will ensure smooth transitions between missions and interactions while implementing a structured testing approach for stability.

1.3.5 Ryan Davis (Tech & Infrastructure Lead)

Ryan is responsible for managing the game's repository, development tools, and ensuring smooth integration between different components. He will also oversee system performance and optimization strategies.

Chapter 2

Weekly Reports

2.1 Week Report 3 (02/17/2025)

What We Did This Past Week

This past week, we successfully set up our **Jira board** to track project tasks and progress. We also **reached out to the client (Iser Pena) for initial feedback** and to ensure alignment with project goals.

We executed Sprint 1 by completing key user stories and implementing fundamental gameplay mechanics. The team focused on developing core systems rather than auxiliary mechanics, ensuring a solid foundation for future sprints. Additionally, we committed all progress, including README updates, source code, and test files, to our GitHub repository.

As part of our Agile process, we updated the **burndown chart** and calculated sprint velocities using Jira reports. The snapshots of these reports are included in the Overleaf document.

2.2 Sprint 1 (03/13/2025)

Demonstrating Sprint 1

We shared our implementation progress with the customer via Slack, providing an overview of our initial feature set. A brief summary of implementation results is as follows:

- Established core mechanics for the game, focusing on task 1.3.
- Added environmental scenery to enhance player immersion and establish the game world.
- Created a prototype mini-game instance to demonstrate core gameplay concepts.
- Implemented health system with replenishment mechanics for the astronaut character.
- We did not complete user story 1.4 as initially intended. In our sprint planning, we over-assigned work and overlooked the dependency on core game mechanics.
- Successfully committed source code and documentation updates to GitHub.

- Implemented astronaut movement, resource tracking, and initial unit tests.
- Created comprehensive testing suite for core game components.
- CI/CD pipeline implementation is currently in progress, with initial framework established.

Reviewing Sprint 1

As a team, we reviewed the results of Sprint 1 and identified areas of success and areas for improvement.

Keep Doing

- **Clear and consistent communication** within the team and with the customer.
- **Regularly spaced meetings** that allow for proper workload management.
- **Notifying team members of updates** so developers can pull the latest changes.
- **Prioritizing manageable goals** to ensure consistent progress.
- **Collaborative work on tasks** rather than working in silos.

Needs Improvement

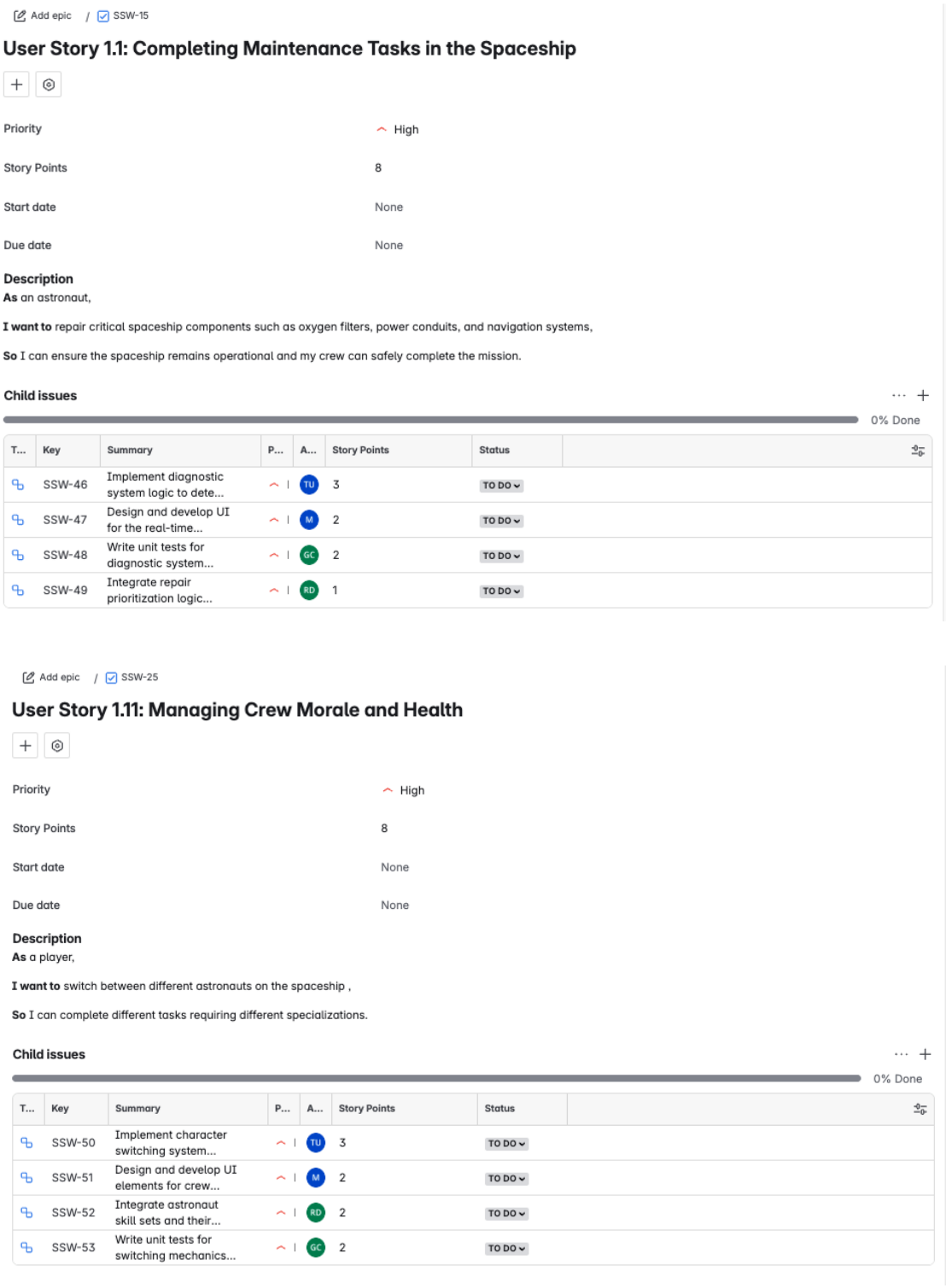
- **More proactive use of Jira** to ensure task creation and tracking is handled ahead of time.
- **Refining the burndown chart process** to provide a more accurate representation of progress.
- **Better resource allocation** to ensure all necessary features are developed in parallel.
- **Improved dependency analysis** when planning sprints to avoid over-assignment of work.
- **More realistic scope definition** for each sprint based on dependencies between user stories.

Planning Sprint 2

For Sprint 2, we have reassessed our user stories and decided to adjust our focus. Our next steps include:

- **Putting user story 1.4 on hold.**
- **Prioritizing user stories 1.1 and 1.11** since they have a strong foundation.
- **Updating Jira** to reflect Sprint 2 tasks, including ownership, estimated story points, and priorities.
- **Capturing snapshots of Jira updates** and including them in the Overleaf document.

- **Finalizing CI/CD pipeline implementation** to streamline future development.
- **Expanding unit tests** to cover additional game mechanics.
- **Enhancing the mini-game experience** with additional features and polish.
- **Refining the health and replenishment systems** based on initial testing feedback.



Add epic / SSW-15

User Story 1.1: Completing Maintenance Tasks in the Spaceship

Priority	High
Story Points	8
Start date	None
Due date	None

Description

As an astronaut,

I want to repair critical spaceship components such as oxygen filters, power conduits, and navigation systems,

So I can ensure the spaceship remains operational and my crew can safely complete the mission.

Child issues

0% Done

T...	Key	Summary	P...	A...	Story Points	Status
	SSW-46	Implement diagnostic system logic to dete...		TU	3	TO DO
	SSW-47	Design and develop UI for the real-time...		M	2	TO DO
	SSW-48	Write unit tests for diagnostic system...		GC	2	TO DO
	SSW-49	Integrate repair prioritization logic...		RD	1	TO DO

Add epic / SSW-25

User Story 1.11: Managing Crew Morale and Health

Priority	High
Story Points	8
Start date	None
Due date	None

Description

As a player,

I want to switch between different astronauts on the spaceship ,

So I can complete different tasks requiring different specializations.

Child issues

0% Done

T...	Key	Summary	P...	A...	Story Points	Status
	SSW-50	Implement character switching system...		TU	3	TO DO
	SSW-51	Design and develop UI elements for crew...		M	2	TO DO
	SSW-52	Integrate astronaut skill sets and their...		RD	2	TO DO
	SSW-53	Write unit tests for switching mechanics...		GC	2	TO DO

Figure 2.1: These screenshots show the planned user stories and their respective child tasks for sprint 2. We have determined their owners as well as their estimated story points.

Lessons Learned

Moving forward, we are confident that we won't face the same issues with user story implementation. The solid foundation of core game mechanics we've established will provide a better platform for implementing other user stories. Key takeaways include:

- **Foundation-first approach:** We've learned the importance of prioritizing core mechanics before building additional features.
- **Dependency mapping:** We will now map dependencies between user stories more carefully during sprint planning.
- **Realistic scheduling:** Our estimations will account for the technical dependencies between components.
- **Incremental implementation:** We'll build features incrementally on top of established systems rather than attempting parallel development of interdependent components.

Project Artifacts

Required Screenshots

- Screenshot of Jira Board (task statuses)
- Screenshot of Sprint Burndown Chart

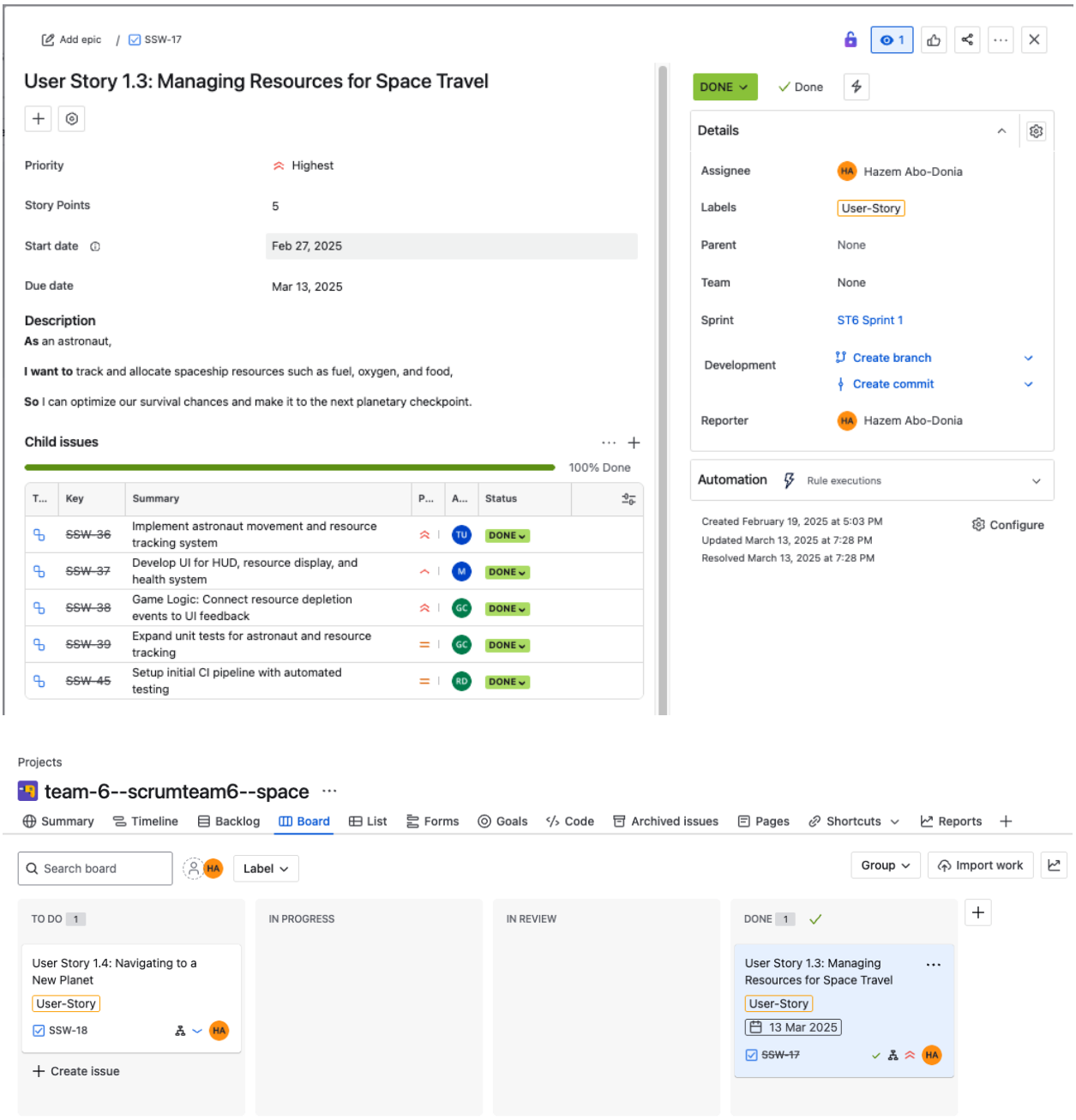


Figure 2.2: These screenshots display the completed tasks and their respective owners for User Story 1.3, which was the main focus of Sprint 1. It highlights the progress made on core mechanics, including astronaut movement and resource tracking. Additionally, User Story 1.4 is shown as deferred due to dependencies on unfinished systems, ensuring accurate backlog management and sprint tracking.

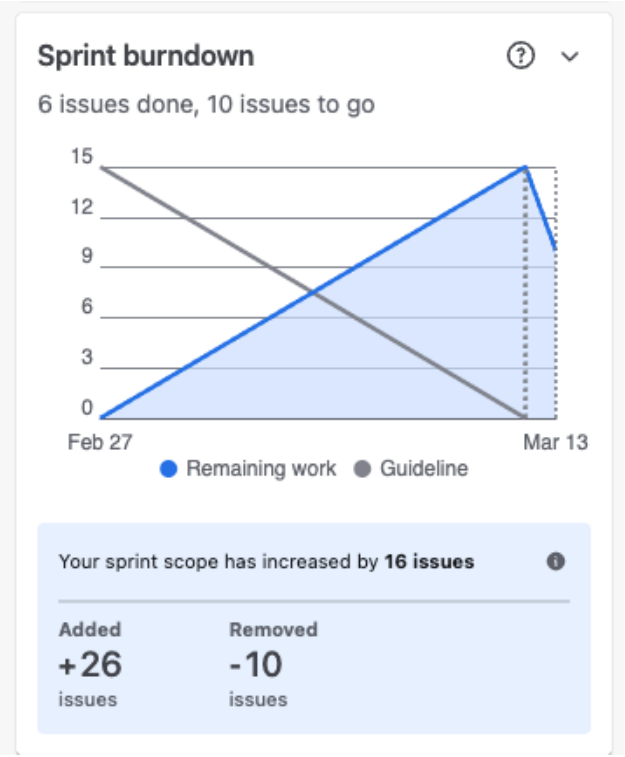


Figure 2.3: The burndown chart shows remaining work at the end of the sprint because we opted to keep incomplete tasks rather than deleting or moving them, ensuring accurate velocity calculations. User Story 1.4 was moved back to **TO DO** due to a reassessment of dependencies. This was not due to incomplete work, but rather a strategic decision since this user story relied on unfinished core mechanics. We determined it would be better handled in a future sprint after foundational systems were finalized, ensuring smoother integration. This approach maintains an accurate historical record and helps with better sprint planning moving forward.

Velocity Calculation Analysis

During Sprint 1, our team worked on two primary user stories:

- **User Story 1.3** - Consisted of 5 child tasks, all of which were completed.
- **User Story 1.4** - Consisted of 9 child tasks, but was deferred due to dependencies.

Since we completed **User Story 1.3** in its entirety, our team successfully finished 5 child tasks. However, Jira's velocity calculation did not account for these subtasks correctly.

Jira's Velocity Report Discrepancy

Jira tracked our user stories as single issues rather than recognizing their respective subtasks. As a result, the velocity report recorded only **one completed issue** instead of acknowledging the 5 individual child tasks completed under User Story 1.3. This led to an artificially low reported velocity.

Manual Velocity Calculation

To obtain a more accurate velocity, we manually calculated it based on the actual number of completed tasks:

$$\text{Velocity} = \frac{\text{Total completed tasks}}{\text{Number of sprints}}$$

Given that we completed all 5 tasks from User Story 1.3, our manual velocity is:

$$\text{Velocity} = \frac{5}{1} = 5$$

This represents a more accurate measure of our sprint output, as it properly accounts for completed work at the task level rather than at the user story level. Moving forward, we will ensure that Jira is configured to properly track subtasks or adjust our reporting to reflect the actual velocity.

Looking Ahead: Action Items

- Finalize Jira updates for Sprint 2.
- Ensure all development progress is documented and committed.
- Continue iterating on core game mechanics and refining implementation.
- Complete CI process implementation to automate testing and deployment.
- Maintain proactive communication with team members and stakeholders.

Meeting Notes

February 27, 2025 - Sprint 1 Development

Instead of direct task assignments, we allowed members to claim tasks based on interest and expertise. We organized our work into five broad Jira tasks:

- **Core Game Mechanics Implementation:** Basic astronaut movement, resource tracking, and event triggers.
- **UI/UX Development & Integration:** Creating menus, HUD, and interaction screens.
- **Testing & Quality Assurance:** Setting up unit tests and integration tests.
- **AI Assistant & Decision System:** Prototyping AI for player guidance.
- **Documentation & Client Communication:** Refining the mission statement, learning objectives, and weekly client updates.

March 10, 2025 - Sprint 1 Check-In

We reviewed progress and identified blockers:

- **Completed:** Astronaut movement, resource tracker, unit tests for astronaut file, initial scenery implementation, health system prototype.
- **In Progress:** Oxygen resource tracking, UI implementation, CI/CD setup, mini-game development.
- **Blocked:** CI/CD errors and integration of the health replenishment system.
- **Next Steps:** Complete resource tracking logic, UI updates, expand unit tests, and finalize CI implementation.
- **Identified Issue:** User story 1.4 blocked due to dependency on core mechanics not initially accounted for in planning.

March 12, 2025 - Sprint 1 Final Adjustments

Sprint 1 nearing completion:

- **Refined user stories for Sprint 2.**
- **Identified gaps in current mechanics and testing.**
- **Updated Jira to reflect sprint completion and priorities for the next phase.**
- **Successfully integrated health system with replenishment mechanics.**
- **Created working prototype of mini-game for client review.**
- **Continued work on implementing CI process with automated testing.**
- **Addressed planning issues:** Discussed lessons learned regarding user story 1.4 and agreed on improved dependency analysis for future sprints.

Frequent check-ins helped maintain alignment and ensure timely sprint completion.

2.3 Sprint 2 (04/03/2025)

What We Did This Past Sprint

During Sprint 2, we focused on completing the integration of the mini-game into the main branch, improving the crew switching logic, and enhancing our CI/CD pipeline. The team aimed to ensure a functional and stable version of the game to be showcased during the demo presentation. Additionally, we practiced pair programming to address challenges collaboratively and efficiently.

Pair Programming

To streamline problem-solving and increase code quality, we practiced pair programming during this sprint by forming two groups:

- Group 1: Hazem and Ryan
- Group 2: Gianna, Mohamed, and Thomas

These sessions were conducted online via Zoom with screen sharing, which allowed real-time collaboration and code review. We found this practice especially useful during the integration of the mini-game and troubleshooting CI/CD pipeline issues. Pair programming facilitated brainstorming sessions that significantly reduced the time spent debugging and increased overall productivity.

CI/CD Integration

Our CI/CD pipeline underwent substantial improvements since Sprint 1. We utilized **GitHub Actions** to automate the build and test process whenever changes were pushed to the main branch. The pipeline was configured to run on an **Ubuntu** environment to maintain compatibility with **Godot 4.3**.

For more details on the pipeline and its current status, visit our GitHub Actions page.

Improvements from Sprint 1:

- Resolved CI pipeline failures caused by incorrect configuration of the Godot headless version.
- Implemented automated testing for all scene-based unit tests to reduce manual testing workload.
- Enhanced the pipeline to automatically fetch the latest Godot version and execute tests for each pull request.

CI/CD Pipeline Snippet

```
name: Godot CI Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
  workflow_dispatch:

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
```

```
16      - name: Download Godot Headless
17        run: |
18          wget https://github.com/godotengine/godot/releases/download/4.3-stable/Godot_v4.3-stable_linux.x86_64.zip
19          unzip Godot_v4.3-stable_linux.x86_64.zip
20          chmod +x Godot_v4.3-stable_linux.x86_64
21      - name: Run All Scene-Based Unit Tests
22        run: |
23          for scene in tests/*.tscn; do
24            ./Godot_v4.3-stable_linux.x86_64 --headless --path . --scene "$scene"
25          done
```

Listing 2.1: CI/CD Pipeline for Godot 4.3

Sprint Reflection

During our Sprint 2 meeting, we discussed our progress, challenges, and areas for improvement. The key takeaways were:

Successes:

- Pair programming sessions led to more efficient code development and bug resolution.
- CI/CD pipeline improvements significantly reduced build failures and automated the testing process.
- Successful integration of the mini-game into the main branch, with stable performance during internal tests.

Challenges:

- Integrating new features into the existing game logic caused temporary instability.
- Managing story scope remained difficult, as some user stories took longer than estimated.
- Balancing game functionality with educational content required clearer prioritization.

Sprint Artifacts

In Sprint 2, we generated the following key artifacts to document our progress and results:

Required Screenshots

- Screenshot of CI Passes (GitHub Actions)
- Screenshot of Updated Jira Burndown Chart
- Screenshot of Completed User Stories (Jira Board)
- Screenshot of Mini-Game Integration

Run All Scene-Based Unit Tests 0s

```
▶ Run for scene in tests/*.tscn; do
Running scene: tests/RunAppleTests.tscn
Godot Engine v4.3.stable.official.77dcf97d8 - https://godotengine.org

Running Apple Unit Tests...

body entered
Current Health: 70
+20 Health!
body entered
Current Health: 90
+20 Health!
body entered
✅ Health increase test passed!
✅ Apple removal test passed!
✅ Apple pickup restriction test passed!
All Apple tests completed.

Running scene: tests/RunTests.tscn
Godot Engine v4.3.stable.official.77dcf97d8 - https://godotengine.org

Running Astronaut Unit Tests...

Current Health: 70
✅ Initial health test passed!
✅ Movement test passed for direction: left
✅ Movement test passed for direction: right
✅ Movement test passed for direction: up
✅ Movement test passed for direction: down
✅ Health drain test passed!
✅ Health recovery test passed!
All Astronaut tests completed.

```

Figure 2.4: CI Pipeline Successful Runs

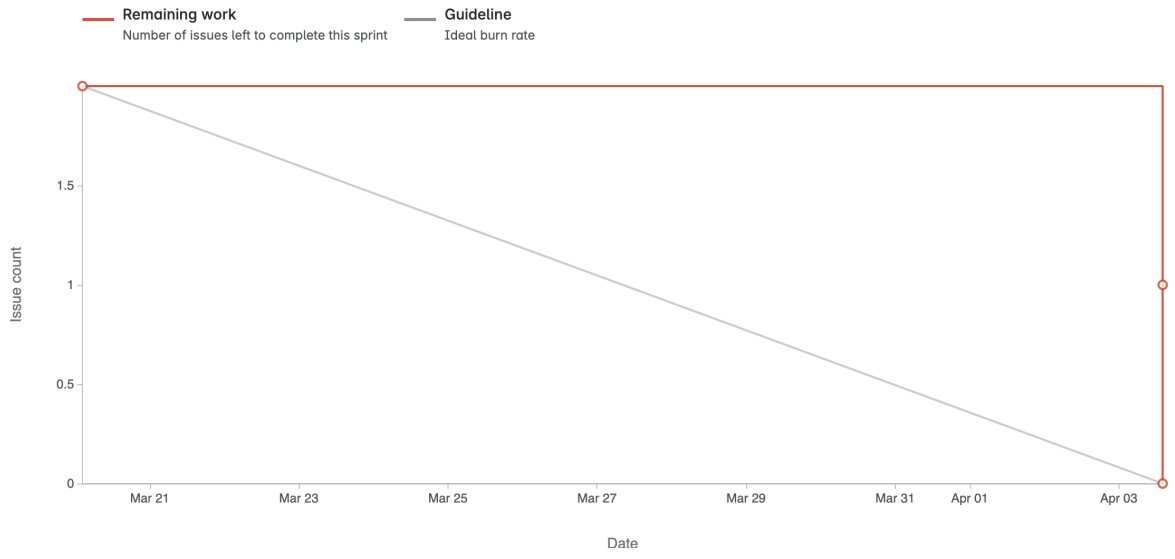
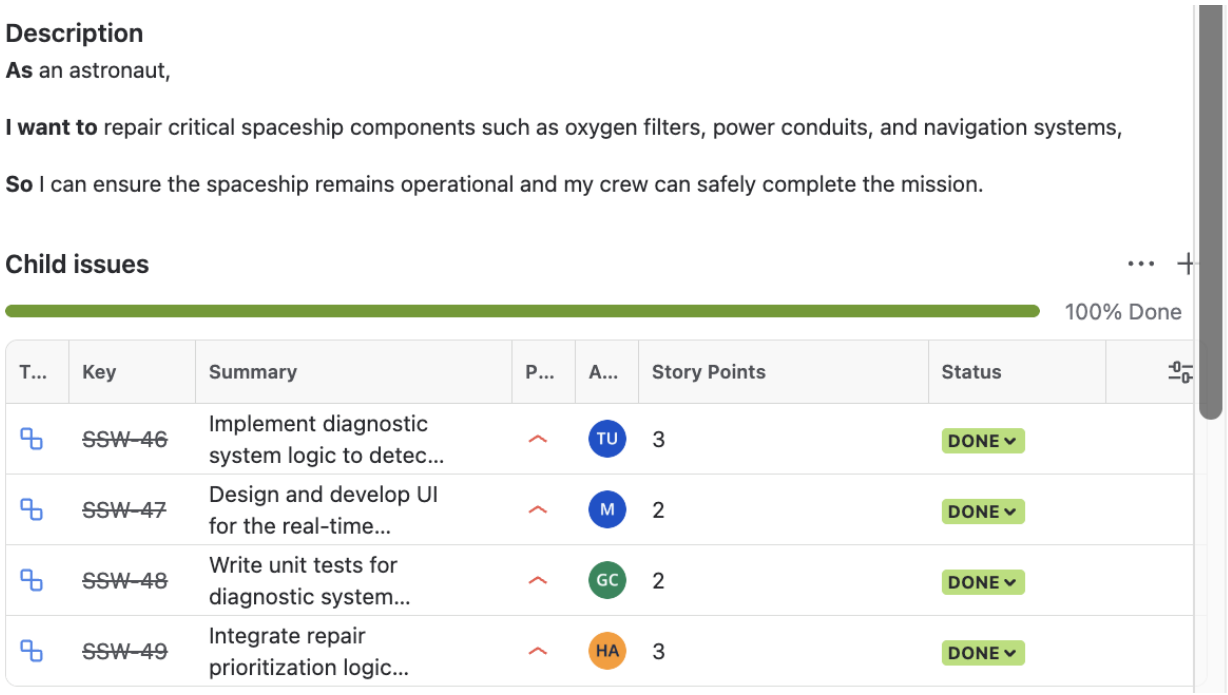


Figure 2.5: Updated Jira Burndown Chart for Sprint 2



Description

As an astronaut,

I want to repair critical spaceship components such as oxygen filters, power conduits, and navigation systems,

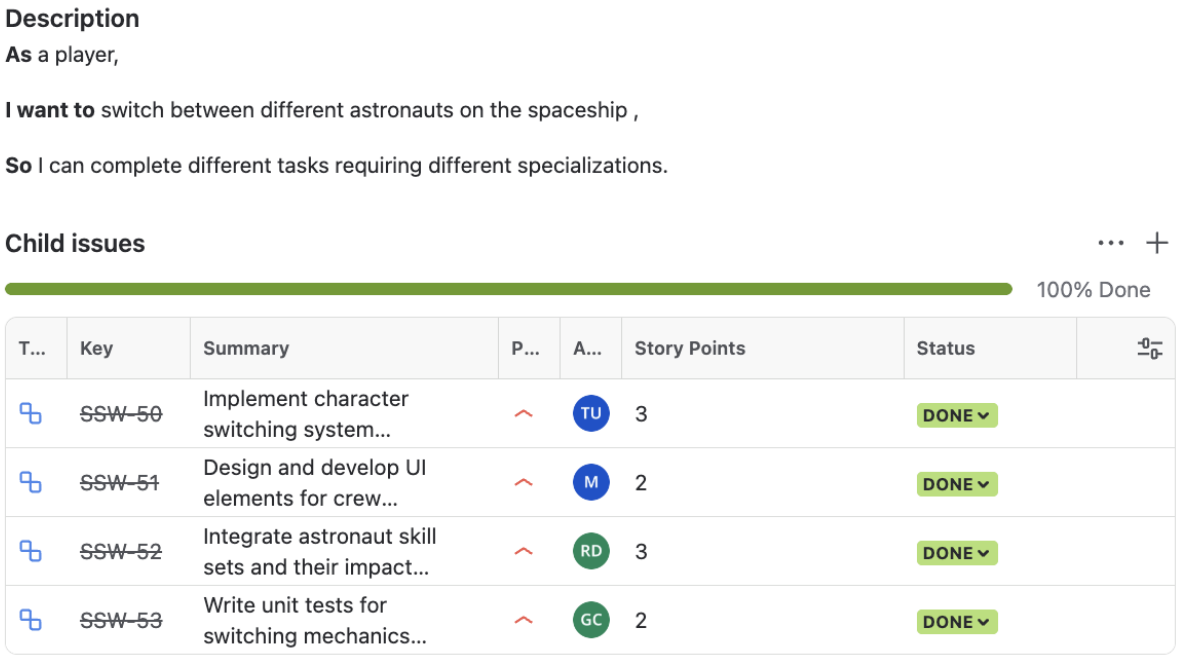
So I can ensure the spaceship remains operational and my crew can safely complete the mission.

Child issues

100% Done

T...	Key	Summary	P...	A...	Story Points	Status
	~~SSW-46~~	Implement diagnostic system logic to detec...		TU	3	DONE
	~~SSW-47~~	Design and develop UI for the real-time...		M	2	DONE
	~~SSW-48~~	Write unit tests for diagnostic system...		GC	2	DONE
	~~SSW-49~~	Integrate repair prioritization logic...		HA	3	DONE

Figure 2.6: Completed User Story 1.1 from Sprint 2



Description

As a player,

I want to switch between different astronauts on the spaceship ,

So I can complete different tasks requiring different specializations.

Child issues

100% Done

T...	Key	Summary	P...	A...	Story Points	Status
	~~SSW-50~~	Implement character switching system...		TU	3	DONE
	~~SSW-51~~	Design and develop UI elements for crew...		M	2	DONE
	~~SSW-52~~	Integrate astronaut skill sets and their impact...		RD	3	DONE
	~~SSW-53~~	Write unit tests for switching mechanics...		GC	2	DONE

Figure 2.7: Completed User Story 1.11 from Sprint 2

Planning Sprint 3

Based on lessons learned from Sprint 2, our primary focus for Sprint 3 will include:

- Completing user stories 1.4 (Navigating to Planets), 1.7 (Communicating with Mission Control), and 1.9 (Running System Diagnostics).
- Prioritizing educational content to align with project objectives.
- Finalizing the spacewalk feature, focusing on resource management and crew coordination.

Figure 2.8: Planned User Stories and Task Assignments for Sprint 3

2.4 Sprint 3 (04/17/2025)

Executing Sprint 3

For Sprint 3, we modified the way our Kanban board was formatted. In previous sprints, tasks contained subtasks which created problems when trying to update their statuses because they could not easily be dragged between swim lanes. To address this, we organized Sprint 3 around three main epics, each with individual tasks directly assigned to the Kanban board.

Additionally, we added a new lane called **Elaboration**, which served as an intermediary step for tasks that needed further discussion or refinement before moving to **In Progress**. We also increased the **In Progress** limit from 5 to 10 tasks to better accommodate parallel work, since many tasks could be completed independently by different team members.

Throughout the sprint, we continued running our CI/CD pipeline tests automatically on GitHub Actions. New unit tests were also written for the features implemented this sprint. As a team, we also practiced refactoring techniques, inspired by the individual assignment focused on code smells, and we identified several areas for improvement in our codebase moving forward.

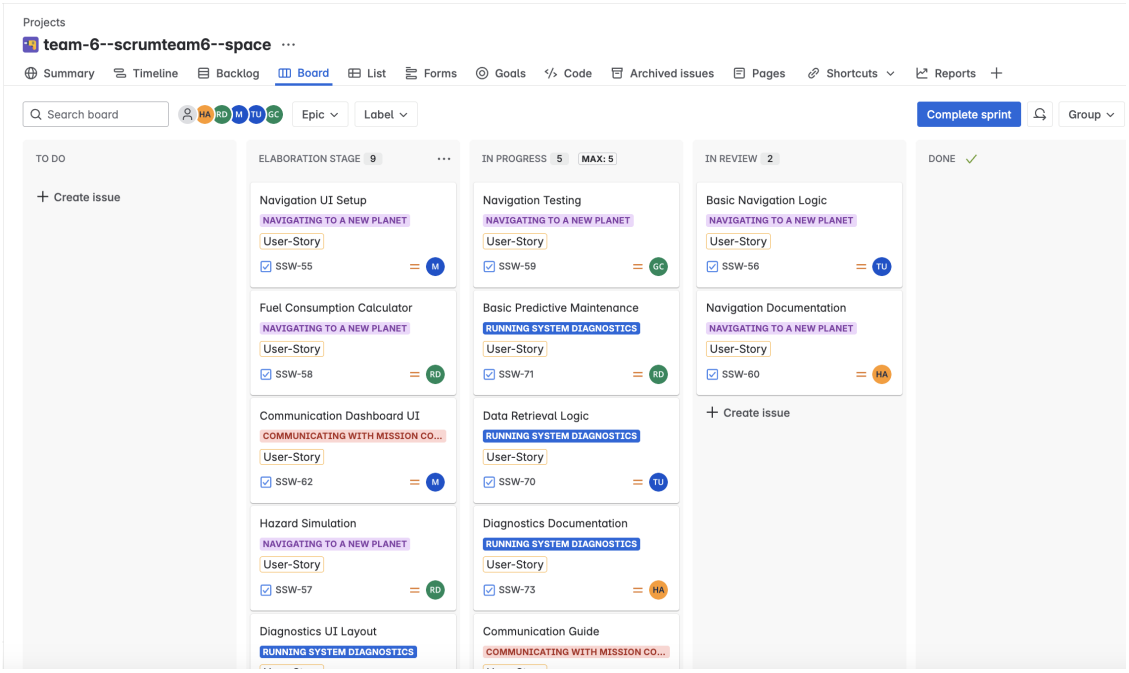


Figure 2.9: Screenshot of our Kanban board a few days before the end of Sprint 3

Figure 2.10: Sprint 3 Burndown Chart showing progress over time. (Note: we had to postpone tasks belonging to the mission control epic, so some tasks were moved to the backlog)

Demonstrating Sprint 3

During this sprint, we implemented another navigation mini-game and successfully linked the System Diagnostics dashboard to the mini-games so that it updates dynamically when a mini-game is completed. We also worked on writing unit tests for these new features and created a communication method between the user and Mission Control.

A demonstration of our current progress was shared with the client through Slack. The shared demo file shows the new functionality and improved interactivity between game systems.

Testing Results

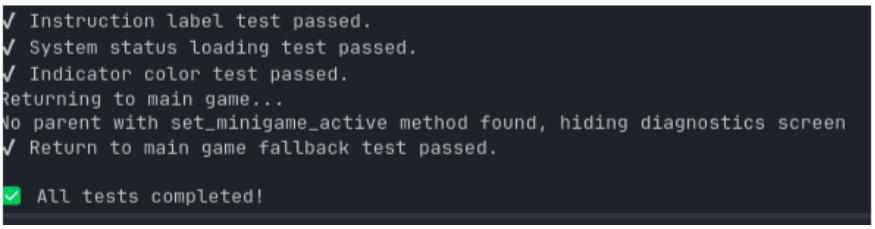


Figure 2.11: Systems Diagnostics Tests

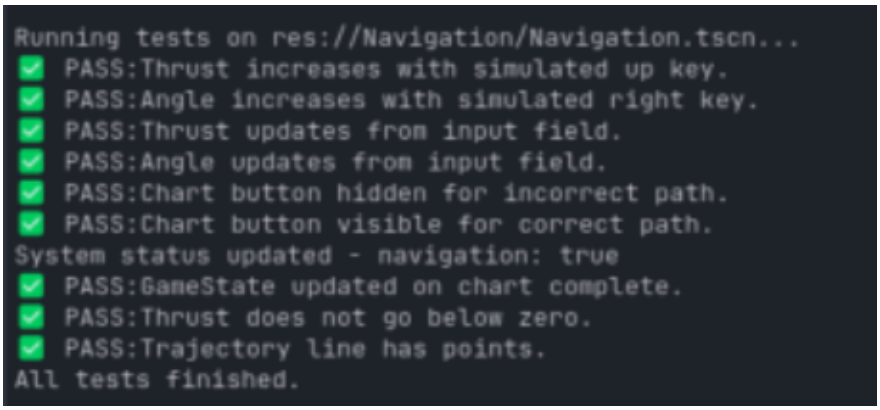


Figure 2.12: Navigation Tests

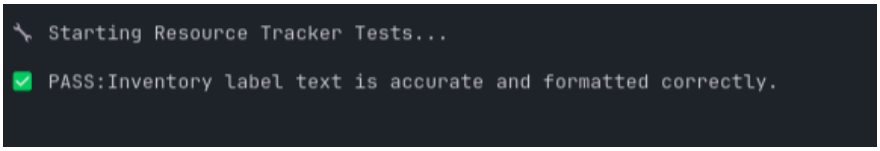


Figure 2.13: Resource Tracker Tests

Reviewing Sprint 3

Overall, Sprint 3 went smoothly. We were able to work effectively in parallel on different features, which improved our overall development velocity. Having different team members focus on different mini-games and system components worked well and is something we want to continue.

One challenge we identified was that many tasks still had dependencies on the completion of others, which occasionally caused delays. In future sprints, we plan to reframe tasks to minimize dependencies and ensure that different members can continue making progress even when certain pieces are still under development.

Additionally, communication within the team was strong throughout the sprint. Our meetings were effective, and we maintained good momentum by proactively addressing blockers when they came up.

Keep Doing

- Working in parallel on different mini-games and features.
- Clear and proactive team communication.
- Effective sprint meetings to align progress and priorities.
- Continued unit testing and CI/CD usage.

Needs Improvement

- Better management of task dependencies to avoid workflow bottlenecks.
- Further refinement of task breakdowns to allow for more independent work.

Planning Sprint 4

For Sprint 4, our primary focus is on incorporating a cohesive storyline into the game to tie all the systems and mini-games together.

Our planned tasks for Sprint 4 include:

- Redesigning the UI for each task so that not all interfaces use monitors.
- Creating a new mini-game focused on **ISS Human Waste Management**, as suggested by the client. This topic is a lesser-known but important aspect of space exploration and provides an engaging learning opportunity for students.
- Implementing the character switching mechanic into gameplay.
- Adjusting navigation sensitivity settings so that difficulty can be customized for different users.

We have updated our Jira board with these new tasks, including ownership assignments, story points, and priority settings.

Throughout Sprint 4, we will continue practicing refactoring to improve code quality, and the main objective will be to build a structured, guided narrative that enhances the educational value and overall player experience.

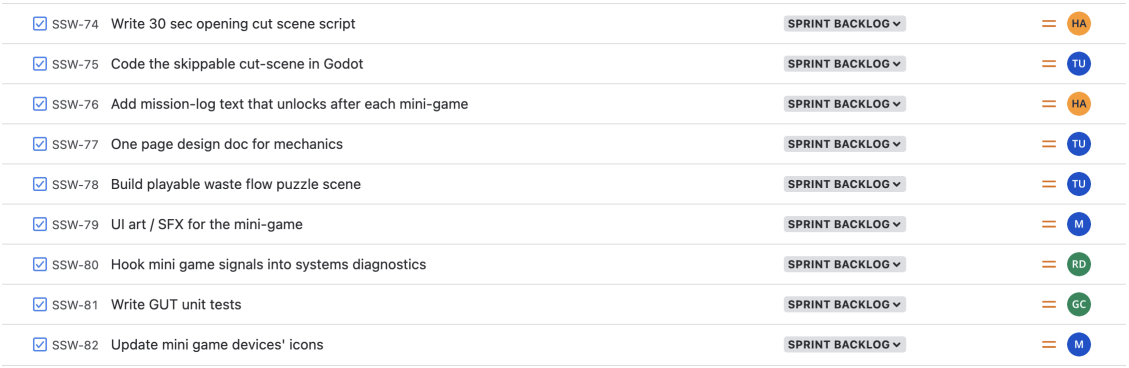


Figure 2.14: Planned user stories and tasks for Sprint 4

2.5 Sprint 4 (04/17/2025 – 05/01/2025)

Recap of Sprint 4

Sprint 4 was focused on bringing everything together and polishing the final version of our game before the presentation and client delivery. We worked on tying together the storyline, finalizing gameplay systems, and adding all necessary polish to ensure a smooth and engaging player experience.

Some of the key areas we tackled this sprint included:

- Final integration of storyline elements and connecting mini-games to the main flow.
- Polish passes on UI and adding missing sound effects and visual feedback.
- Adding tutorial and instructional elements to make the game easier to learn and play.
- Creating the intro cutscene and mission narrative to establish context for players.
- Ensuring system mechanics like health, task access restrictions, and game loss conditions were fully implemented.
- Finalizing the mini-games and connecting them to the diagnostics system and mission flow.

Throughout the sprint, we also paid close attention to previous client feedback from after Sprint 2 and made sure to address many of their concerns around onboarding, clarity of objectives, and storytelling.

Team Reflection

Overall, this sprint went very smoothly. The team worked really well together to finish everything on time. Everyone wore multiple hats and stepped up where needed to help each other out. We met about 4 times throughout this sprint and communicated daily to stay aligned.

The workload was consolidated well across the team, and our strong communication allowed us to push through any blockers quickly. Focusing on the client's feedback really helped us prioritize the most important aspects of the project to deliver a polished product.

Sprint 4 Burndown Chart

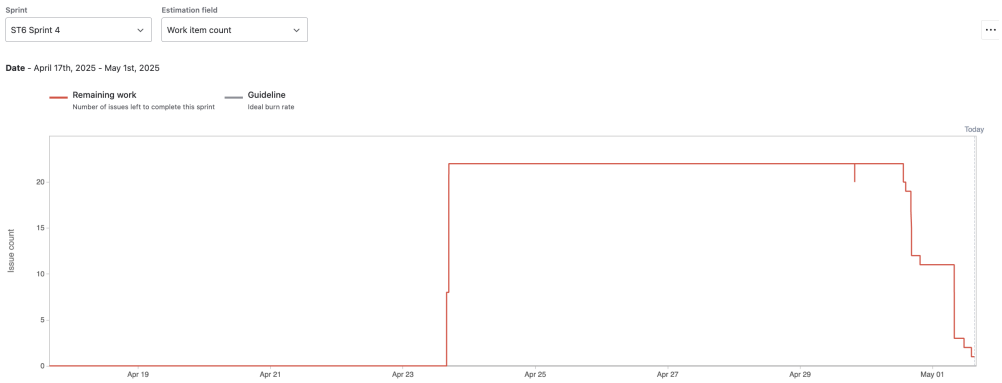


Figure 2.15: Sprint 4 Burndown Chart

Project Reflection

With the completion of Sprint 4, our team has officially wrapped up development for this project. Looking back across all the sprints, our team has grown significantly in both our workflow and collaboration. Throughout each phase, we adapted based on client feedback, iterated on our design, and continually refined our game to improve the user experience and educational value.

One of the key strengths of our team was our ability to communicate and support each other. Everyone contributed in multiple roles when needed, and we maintained steady progress even during challenging moments. Frequent check-ins, daily online communication, and shared responsibility allowed us to stay organized and aligned from start to finish.

In addition, we made a strong effort to incorporate feedback from earlier client reviews, particularly after Sprint 2. This helped us prioritize clarity, player guidance, and overall polish to make the final product engaging and intuitive. Our focus on iterative development and responsiveness to feedback shaped the final version of the game into something we are proud to present.

Overall, this project was a valuable learning experience in working as a team to deliver a complete software product from concept to delivery. We look forward to sharing the results with our client and demonstrating the hard work and dedication that went into every part of this game.

Chapter 3

Development Plan

– Ryan Davis and Thomas Ung

3.1 Introduction

It is the goal of this project to develop an elementary level game, introducing players to concepts of space travel and aerospace systems principle. The game will be developed using agile methods, and the team will communicate frequently on issues of development.

3.2 Roles and Responsibilities

The roles for this project include:

1. Development Lead- Hazem Abo-Donia
2. Architect- Ryan Davis
3. Developers Thomas Ung, Mohamed Bengabsia
4. Test Lead- Gianna Cerbone
5. Testers- Gianna Cerbone, Thomas Ung, Mohamed Bengabsia, Ryan Davis
6. Documentation- All
7. Documentation Editor- Hazem Abo-Donia
8. Designer- Thomas Ung
9. User advocate- Hazem Abo-Donia
10. Risk Management- Gianna Cerbone
11. System Administrator- Ryan Davis
12. Requirements Resource- Thomas Ung
13. Customer Representative- Hazem Abo-Donia

3.3 Method

These are unique to software development, although there may be some overlap.

3.3.1 Software

We will be using the Godot 4.3 game engine. We want to use this since it is open source, lightweight, and has both front and back end capabilities. We will program in Godot's native programming language GDScript which has syntax similar to python. Godot has its own code editor so no ide will be required.

Godot can export as a web file which can run locally on your computer. To upload our game to be played by online users we will use itch.io. Itch.io is an indie game distributor which takes no ownership of games and takes revenue cuts less than industry norms and will not charge us to upload our game.

Both itch.io and Godot are free, widely used in the industry, and user-friendly, allowing us to complete our project efficiently.

Additionally, we will use Jira for project management and task tracking, and GitHub for version control and team collaboration.

1. Language(s) with version number including the compiler if appropriate

   We will code in GDScript which is native to Godot 4.3

2. Operating System(s) with release number

   Since our project is web based, all major computer operating systems–such as MacOs, Windows, and Linux–which can handle modern web browsers.

3. Software packages/libraries used with release/version number

   Godot Engine 4.3 will be the only software which we will need for development.

4. Code conventions – this should preferably be a pointer to a document agreed to and followed by everyone

   We will follow Godot's GDScript style guide, which emphasizes frequent commenting, proper indentation, and consistent spacing and punctuation for clean and readable code.

3.3.2 Hardware (optional)

Not applicable to this project.

1. Development Hardware
2. Test Hardware
3. Target/Deployment Hardware

3.3.3 Backup plan (individual and project)(optional)

In the event our current tech stack fails we will switch over to Unity.

3.3.4 Review Process

1. Will you do architecture, usability, design, security, privacy or code reviews?
2. What approach will you use for the reviews (formal, informal, corporate standard)?
3. Who is responsible for the reviews and resolving any issues uncovered by the reviews?
4. Code readings?

3.3.5 Build Plan

1. Revision control system and repository used
2. Continuous integration
3. Regularity of the builds – daily
4. Deadlines for the builds – deadline for source updates
5. Multiplicity of builds
6. Regression test process – see test plan

3.4 COMMUNICATION PLAN

3.4.1 Heartbeat Meetings

The team, including Hazem Abo-Donia, Thomas Ung, Mohamed Bengabsia, Gianna Cerbone, and Ryan Davis, will hold weekly meetings on Wednesday at 4. Each meeting covers the progress of the project's sprints. Weekly meetings will have a set agenda and proper notes will be taken by the scrum master.

3.4.2 Status Meetings

Status Meetings will be held monthly over zoom in order to track the progress of each members tasks. The scrum master will address each member of the team based on their assigned roles and their individual tasks. Progress reports will be provided over such meetings.

3.4.3 Issues Meetings

When a problem arises, a meeting will be scheduled with the scrum master, and if necessary, other members of the team or stakeholders. Over such meeting, a risk managing strategy will be composed in order to resolve any issue.

3.5 Timeline and Milestones

Milestones will be tracked based on the sprints and status of the project. There will be about 4-10 milestones, and changes and features will be tracked.

3.6 Testing Policy/Plan

Throughout development, we plant to adopt a test-driven development (TDD) approach, where tests are written before development to ensure the correct output of the code. Unit testing will be conducted on individual components to ensure the correctness of functions. Integration testing will verify the proper interaction between the system's modules, and their ability to contribute to each other after each component is developed. System testing will evaluate the complete system's functionality against the requirements to confirm that all components work together as intended. Acceptance testing will be performed with stakeholders ensure the program meets expectations and requirements. Regression testing will be ran to ensure functionality after refactoring. The team's testing will ensure that the program works properly.

At least indicate if certain type of testing will be practiced, if so, when to start, and what is the stopping criteria. For example, unit testing is recognized as important, and recommended. But sometimes start-up companies intentionally skip unit testing for speed to market.

3.7 Risks

- Time management- The project will be arranged into sprints in order to schedule different tasks throughout the project.
- Learning new skills- The team will familiarize themselves with the different components of the system.
- Managing other course loads- The sprints, arranged in a deadline system will allow for each team member to work diligently.
- Managing Agile development process- The scrum master will manage each team member and make sure they are making progress with each sprint. Heartbeat meetings will qualitatively track the progress of the project.

3.8 Assumptions

It can be assumed that the project is at an elementary level. We assume that the user will be able to navigate simple game functionalities. We are also assuming that the user will be using a computer in order to use the application.

3.9 DISTRIBUTION LIST

Members of the team, the stakeholder of the project, and the instructors of the class.

Chapter 4

Requirements

– Gianna Cerbone & Mohamed Bengabsia

4.1 Functional Requirements

Core Gameplay Mechanics

- The player shall complete tasks inside the spaceship while traveling between planets.
- The player shall manage their available resources, including fuel, oxygen, food, and repair materials.
- The players shall be able to share resources and interact with AI characters.
- The player shall be able to switch between different characters in order to complete tasks.
- The missions shall be timed, requiring players to complete tasks within a set duration.

Educational and Survival Elements

- The game shall provide educational content about different planetary environments.
- The game shall include an astronaut survival mechanic, such as health deterioration and oxygen depletion.

Progression and Safety Features

- The game shall include checkpoints at each planet to save progress.
- The game shall ask for the player's age for safety purposes.
- The game shall allow players to progress from Earth through different planets until reaching the final goal.

4.2 Non-functional (Quality) Requirements

Performance, Scalability, and Compatibility

- The system shall run efficiently in Godot, ensuring smooth performance on the target platform.
- The system performance shall support smooth gameplay without significant lag or frame drops, with a target of minimum 60fps at all times.
- The system shall be compatible with Godot's engine capabilities, including physics, animations, and UI elements.

Security and Error Handling

- The system shall support proper error handling for security purposes (utilizing try-catch mechanisms to prevent crashes and unexpected behavior).

User Experience

- The user interface shall be intuitive and easy to navigate.

Data Integrity and Progress Tracking

- The system shall store and track player progress accurately.

Testing and Quality Assurance

- The game shall be tested to ensure it runs without critical bugs before submission.

Project Management and Deployment

- All functional requirements shall be numbered and tracked in Jira for development tracking.
- The build process shall be integrated with GitHub for version control and proper collaboration.
- The final version shall be packaged properly for submission, ensuring it runs without setup issues.

Chapter 5

Use Cases

– Gianna Cerbone & Ryan Davis

This chapter presents the use case diagrams that satisfy the requirements. Detailed descriptions of each use case are given in separate chapters for each use case.

5.1 Table of Use Cases

To prevent gold-plating all use cases must be mapped to at least one requirement. For right now, the team has decided to map the use case to a requirement section since we are still working through refining the requirements. Next, a table to keep track of this mapping is shown. Along with diagrams with requirements, actors, and more information on the use cases. A use case model still needs to be made for each.

Table 5.1: Use Cases Table

Use Case	Requirements	Name and Description
UC_1	$]CoreGameplay$	ucDistributeResources Describes the abilities of astronauts with gathered resources. Astronauts can distribute their belongings to meet the needs of their crew and the ship.
UC_2	$]UserExperience$	ucNavigateToPlanet Describes the process of navigating to a new planet. The astronaut uses the onboard navigation system to chart a course, adjusting for space anomalies, fuel limitations, and potential hazards, ensuring safe arrival at the destination.

5.2 Use Case Diagrams

Use Case 1 (ucDistributeResources) *Distribute Resources: Astronaut allocates energy, oxygen, and food supplies to the crew.*
Requirements: *CoreGameplay*??
Brief description: The system enables an astronaut to allocate energy, oxygen, and food supplies based on crew needs and mission duration, while factoring in emergency reserves and stress-induced consumption.
Primary actors: Astronaut
Secondary actors: Crew Members.
Preconditions: 1. Resources (energy, oxygen, food supplies) have been acquired. 2. Crew needs and mission parameters are defined.
Main flow: 1. The astronaut accesses the resource management interface. 2. The astronaut selects a resource type (energy, oxygen, or food). 3. The astronaut allocates the selected resource to its designated location. 4. If an emergency or stress condition is detected then 4.1. The system automatically allocates emergency reserves.
Postconditions: The resources are distributed to the designated locations according to their type.
Alternative flows: User stores resources in personal inventory.

Use Case 2 (ucNavigateToPlanet) *Navigating to a New Planet: Astronaut charts a course to a new planet while adjusting for space anomalies, fuel limitations, and potential hazards.*
Requirements: $UserExperience_{??}$
Brief description: The astronaut uses the onboard navigation system to chart a course to a new planet, factoring in space anomalies, fuel limitations, and potential hazards to ensure a safe journey.
Primary actors: Astronaut
Secondary actors: Mission Control (optional)
Preconditions: 1. The astronaut is aboard the spacecraft. 2. The onboard navigation system is operational.
Main flow: 1. The astronaut accesses the onboard navigation system. 2. The astronaut selects the destination (new planet). 3. The system displays relevant data (space anomalies, hazards, fuel levels). 4. The astronaut adjusts the course to account for space anomalies, fuel limitations, and hazards. 5. The astronaut confirms the final course. 6. The system continuously updates the course and alerts the astronaut to any changes.
Postconditions: The spacecraft is on the planned course to the new planet.
Alternative flows: 1. If an unexpected anomaly is detected, the system suggests an alternate route. 2. If fuel levels are critically low, the system prompts for a refueling route or recalculates course adjustments.

Chapter 6

User Stories

– Gianna Cerbone, Ryan Davis

This chapter presents a series of user stories that outline the key functional requirements and experiences envisioned for our space mission simulation. Each story is written from the perspective of astronauts, engineers, mission commanders, or players, detailing scenarios from routine system maintenance and diagnostics to emergency responses and crew management. These narratives serve as a blueprint for designing a responsive, robust system that addresses the unique challenges of space travel.

User Story 1.1: Completing Maintenance Tasks in the Spaceship As an astronaut,
I want to use a real-time diagnostic system to detect spaceship malfunctions, prioritize repairs based on urgency, and deploy the correct tools under time constraints,
So I can ensure mission survival and avoid catastrophic failures.
Story Points: 8

User Story 1.2: Conducting Life Support System Checks As an astronaut,
I want to monitor oxygen regulation, CO_2 levels, and cabin temperature through a dynamic life support interface, making real-time adjustments under stress,
So I can keep the environment stable and prevent crew incapacitation.
Story Points: 5

User Story 1.3: Managing Resources for Space Travel As an astronaut,
I want to allocate energy, oxygen, and food supplies based on crew needs and mission duration, factoring in emergency reserves and stress-induced consumption,
So I can increase survival chances and sustain the mission.
Story Points: 5
Related Use Case: Distribute Resources

User Story 1.4: Navigating to a New Planet As an astronaut,
I want to use the onboard navigation system to chart a course while adjusting for space anomalies, fuel limitations, and potential hazards,
So I can safely reach my destination without unnecessary risk.
Story Points: 3
Related Use Case: Navigate to Planet

User Story 1.5: Performing a Spacewalk to Repair External Damage As an astronaut,
I want to conduct an EVA (extravehicular activity) while managing oxygen levels, physical

exertion, and radiation exposure,
So I can complete external repairs before life support runs out.
Story Points: 8

User Story 1.6: Learning About Space Exploration for Mission Success As an astronaut,
I want to interact with an onboard AI assistant capable of answering questions, giving mission briefings, and dynamically responding to decisions,
So I can make informed choices that improve survival chances.
Story Points: 5 for now (could become 13+)
maybe integrate a companion with an LLM

User Story 1.7: Communicating with Mission Control As an astronaut,
I want to maintain constant communication with mission control through a dashboard, navigating realistic signal delays and using AI to relay emergency protocols,
So I can receive mission updates, report critical issues, and get expert guidance.
Story Points: 3

User Story 1.8: Testing Emergency Protocols As an astronaut,
I want to react to real-time emergencies such as fire outbreaks, rapid decompression, or navigation failures under stress,
So I can ensure survival and crew safety.
Story Points: 8

User Story 1.9: Running System Diagnostics As a spaceship engineer,
I want to run diagnostic scans that provide real-time data on system performance and predict potential failures,
So I can fix issues before they become life-threatening.
Story Points: 3

User Story 1.10: Managing Crew Morale and Health As a mission commander,
I want to track psychological stress, fatigue, and social interactions among crew members,
So I can intervene before morale drops too low and mission efficiency suffers.
Story Points: 5
(make a bar that measures overall morale, timely completion of missions can increase it)

User Story 1.11: Switching Between Crew Members As a player,
I want to switch between astronauts with different skill sets, psychological traits, and expertise,
So I can complete specialized tasks that require teamwork and problem-solving.
Story Points: 8

Chapter 7

Implementation

– Team 6

7.1 Sprint 1 Implementation

7.1.1 Introduction

The Preliminary Implementation Demo showcases the early development stages of our educational space exploration game. The demo highlights the project's progress by presenting a working prototype of the core gameplay mechanics, including astronaut movement, collision detection, health management, and an interactive trigger to launch a mini game that will introduce the educational aspects of the system. This stage serves as a foundation for further refinements and feature expansions in preparation for the final submission.

7.1.2 Demo Objectives

The Preliminary Implementation Demo focuses on achieving key milestones in the early development of the system. The specific goals of the prototype demo include:

1. **Validating Key Functionalities**
   (a) Astronaut movement within the spaceship
   (b) Collision detection using bounding walls
   (c) Health management via a dynamic health bar
2. **Establishing Feasibility**
   (a) Interactive screen detection and response
   (b) Triggering a mini game (via the `E` key) to lay the groundwork for educational content
3. **Gathering Initial Feedback**
   (a) Evaluation of player control responsiveness and collision accuracy
   (b) Assessment of the health system and interactive elements

7.1.3 Prototype Description and Demo Results

In Sprint 1, the implementation focused on integrating several core gameplay mechanics:

- **Astronaut Movement & Collision Mechanics:** Players can control an astronaut who navigates the spaceship. Collision detection is enforced by bounding walls, ensuring that the astronaut remains within the playable area.
- **Dynamic Health Bar:** A health bar positioned above the astronaut gradually deteriorates over time. Health can be replenished by consuming an apple, simulating food intake.
- **Interactive Screen and Mini Game Trigger:** Within the spaceship, various screens are present. One screen becomes highlighted when the astronaut approaches. Pressing the `E` key near the highlighted screen initiates a mini game, which will later be expanded to incorporate educational content.

7.1.4 Future Work and Next Steps

As the project progresses, the team will focus on further refining the gameplay mechanics and expanding the mini game initiated by the interactive screen. Planned improvements include:

- Enhancing astronaut control responsiveness and collision precision.
- Expanding the educational content integrated into the mini game.
- Implementing additional game mechanics and refining user interface elements.

7.1.5 Conclusion

The Sprint 1 demo successfully demonstrates the foundational gameplay mechanics of our educational space exploration game. The integration of astronaut movement, collision detection, dynamic health management, and interactive elements provides a strong basis for future development. Feedback from this prototype will inform the enhancements planned for subsequent sprints.

7.2 Sprint 2 Implementation

7.2.1 Implementation Details

During Sprint 2, the team focused on integrating the mini-game into the main branch and refining the CI/CD pipeline to ensure stability and efficient testing. Additionally, crew switching logic was improved to support seamless transitions between characters.

7.2.2 CI/CD Pipeline Integration

We utilized **GitHub Actions** for automating build and test processes. The pipeline runs on an **Ubuntu** environment with the Godot 4.3 engine. The configuration includes downloading the Godot headless version and running all scene-based unit tests to verify game functionality.

Pipeline Configuration

```yaml
name: Godot CI Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Download Godot Headless
        run: |
          wget https://github.com/godotengine/godot/releases/download/4.3-stable/Godot_v4.3-stable_linux.x86_64.zip
          unzip Godot_v4.3-stable_linux.x86_64.zip
          chmod +x Godot_v4.3-stable_linux.x86_64
      - name: Run All Scene-Based Unit Tests
        run: |
          for scene in tests/*.tscn; do
            ./Godot_v4.3-stable_linux.x86_64 --headless --path . --scene "$scene"
          done
```

Listing 7.1: GitHub Actions Pipeline for Godot 4.3

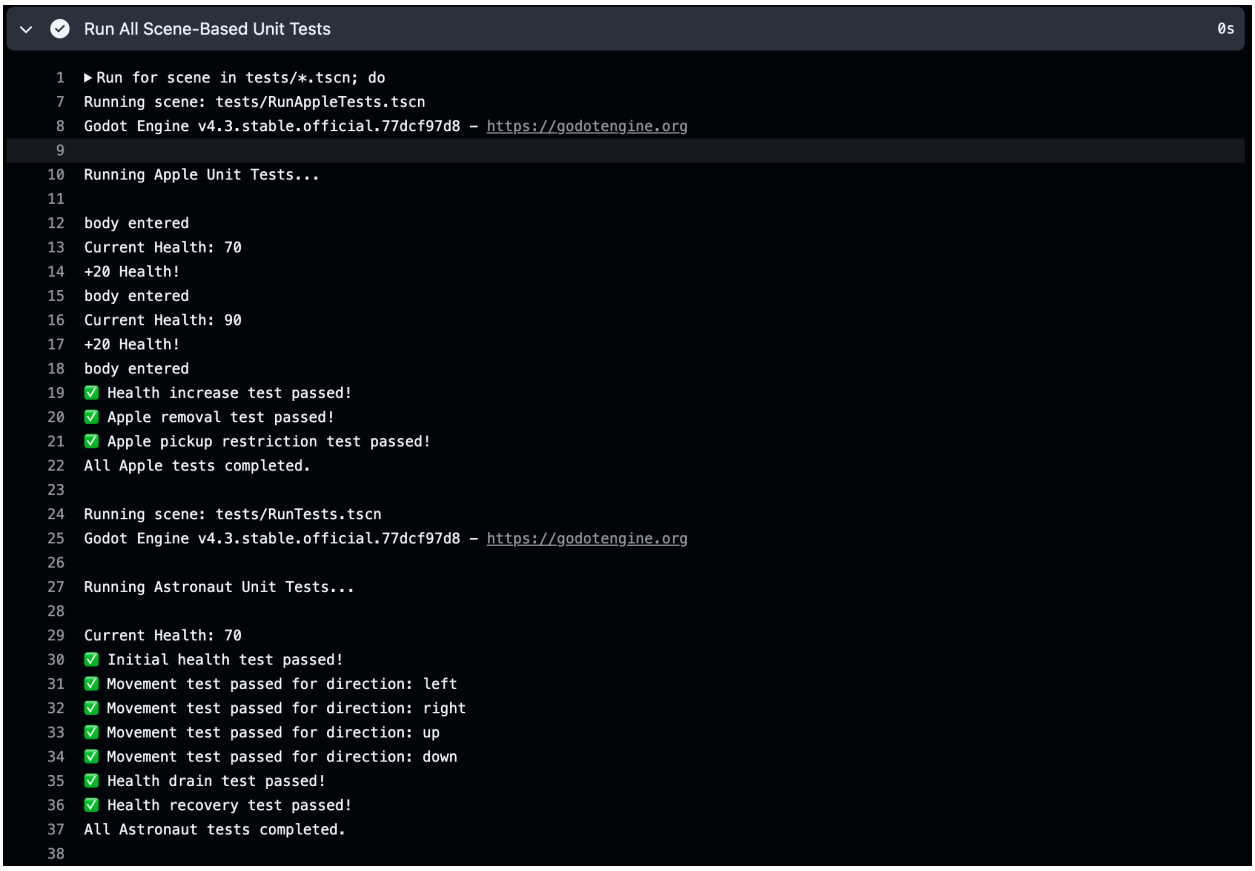


Figure 7.1: Snapshot of CI Passes

7.2.3 Mini-Game Integration

The mini-game was successfully integrated into the main game branch. Thomas and Mohamed collaborated on merging the code, and Gianna conducted unit testing to ensure stability. The primary challenge was maintaining the smooth transition between the main game and mini-game environments.

7.2.4 Next Steps

- Enhance the responsiveness of astronaut controls during the mini-game.
- Integrate more realistic astronaut tasks to align with educational objectives.
- Implement the spacewalk feature, focusing on oxygen management and external repairs.

7.2.5 Conclusion

Sprint 2 marked significant progress in stabilizing the game and refining the CI/CD pipeline. The successful integration of the mini-game demonstrates our ability to incorporate complex features while maintaining core gameplay functionality. Moving forward, we will concentrate on adding educational elements and preparing for the upcoming demonstration.

7.3 Final Implementation and Deployment

7.3.1 Overview

As development came to a close during Sprint 4, the team focused on finalizing and polishing the game to prepare it for delivery. This phase included integrating all remaining gameplay systems, completing the storyline, refining UI and visual feedback, and addressing final client feedback to ensure a smooth and engaging experience.

7.3.2 Final Features and Polish

Key tasks completed during the final sprint included:

- Integrating narrative elements and cutscenes to guide the player through the game.
- Connecting all mini-games seamlessly with the main gameplay flow.
- Enhancing UI clarity with titles, roles, and task indicators.
- Implementing final audio and visual polish, including sound effects and UI animations.
- Adding win/loss conditions and improving onboarding through tutorials and hints.

7.3.3 Deployment

Once the final version was tested and approved, the team deployed the completed game for public access. The game is now available on **Itch.io**, providing an easy and accessible platform for users to play the game.

- **Final Game Release:** https://scrumteam6.itch.io/outerflight

7.3.4 Conclusion

The final release represents the culmination of all sprint efforts, team collaboration, and client feedback throughout the semester. We are proud of the final product and look forward to presenting and sharing OuterFlight with our client and the broader community.

Bibliography

Index