
Phillip Shields

Milestone 3

DAT602 FINAL SUBMISSION

MILESTONE ONE	4
Description of Game	4
Basic functional requirements	4
Game concept	4
STORYBOARDS	5
Login	5
Login Alerts	6
Register	8
Register Alert	9
Home Menu	10
Main Game Display	11
Main Game Display - Element Selection	12
Main Game Display - Element Mined	13
Admin Menu	14
Menu Design Explanation	18
Entity Relationship Diagram	19
Entity Relationship Diagram Explanation	20
tblPilot	20
tblExplore	20
tblExplorePlay	20
tblExploreScore	20
tblElement	20
tblExplorePlayStock	20
tblExploreGalaxy	20
tblExploreGalaxyVector	20
tblExploreVectorElement	20
CRUD Table	20
CRUD Table Explanation	22
Pilot Registration	22
Pilot Login	22
Lock Pilot	22
Delete Pilot	22
Pilot Log Out	22
Join Galaxy	22
New Galaxy	22

Pilot Moves	22
Admin Menu	23
Destroy Galaxy	23
Milestone One Summary	23
MILESTONE TWO	24
ACID	24
Atomicity	24
Consistency	25
Isolation	25
Durability	25
MILESTONE THREE	26
EXPLANATION	26
GALAXIAS 2.0 INTRODUCTION	27
FINISHED ASPECTS	28
Login and Registration	28
Game instance	29
Game logic	31
Game display	32
UNFINISHED ASPECTS	33
Registration and Login	33
Game instance selection	34
CURRENT SINGLE GAME INSTANCE DB STRUCTURE	34
MULTIPLE GAME INSTANCE DB STRUCTURE	35
Admin controls and menu	35
Data persistence	37
FINAL DEFENSE OF GALAXIAS 2.0	38

MILESTONE ONE

Description of Game

This game will be a multi-user 2-D adventure tile game that will be playable via a point and click game system connected with a simple database. This will be a rudimentary prototype that can be used for the future development of a much more complex version.

Basic functional requirements

1. Registration of new players require:
 - a. User name
 - b. Password
2. Users are given three attempts at entering their passwords
3. Online players are visible to one another
4. Admin accounts will have the following abilities:
 - a. Adding new players
 - b. Updating existing players
 - c. Deleting existing players
 - d. Ending players game states
 - e. Deleting current games
 - f. Locking and unlocking player accounts
5. Players can start a new game or continue their previous games
6. Players start on a predefined home tile
7. Multiple different games can be played at the same time
8. Players move around the board by moving only onto adjacent tiles to their current position
9. A players location is stored in the database after every move
10. Tiles contain Elements that will increase or decrease the players' score
11. A players game, location, and score will be saved upon logout
12. A players game, location, and score will be restored when logging back in

Game concept

The game will be based on the idea of intergalactic travel. A game instance will occur within a galaxy where players will move around in search of elements to mine. Essentially, a galaxy is represented by a tiled gameboard. Each tile represents a location within the galaxy and

potentially contains some type of resource or element to be collected. A single tile could possess a single element, multiple elements, no elements, or even destructive elements.

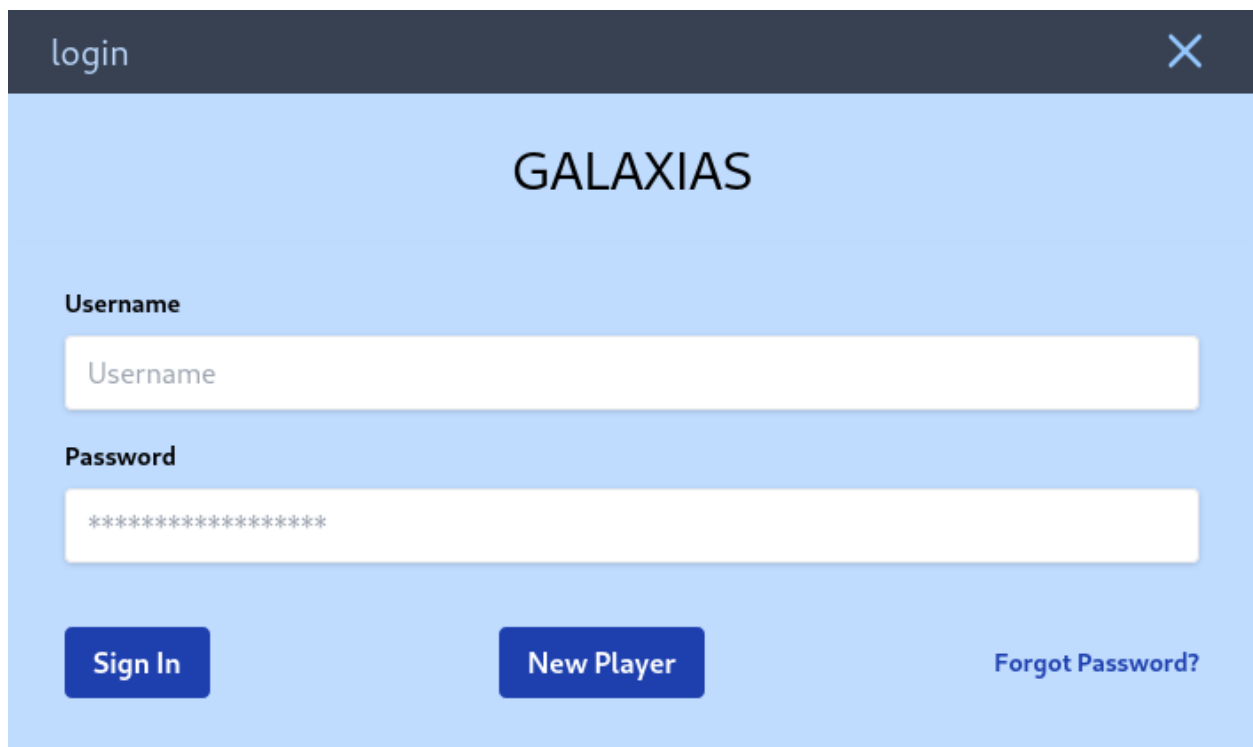
Elements will have different value points that increase the player's points and destructive elements will decrease the player's points. There may even be apocalyptic elements that kill the player and remove all points.

A galaxy will be considered explored and depleted if all elements, except destructive elements, are mined and collected by players.

Multiple players may explore the same galaxy, and each time a player starts a new game a new and random galaxy will be generated.

STORYBOARDS

Login

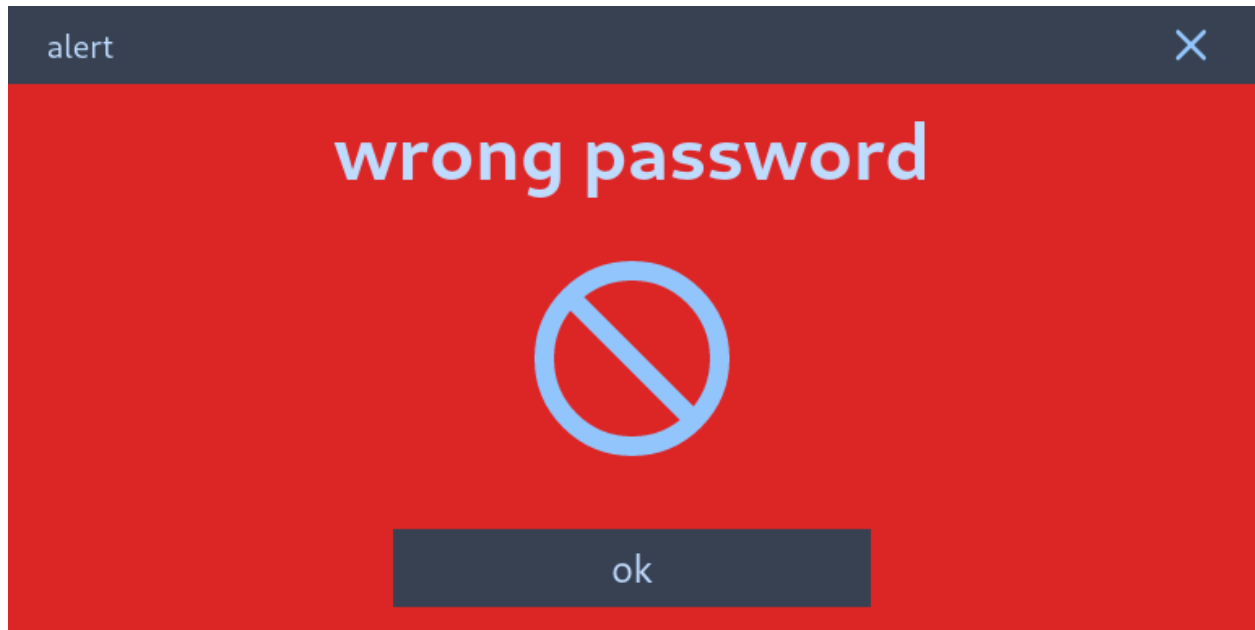


The login screen for GALAXIAS features a dark blue header with the word "login" on the left and a close button (X) on the right. The main background is light blue. The title "GALAXIAS" is centered in a large, bold, black font. Below the title, there are two input fields: "Username" and "Password". The "Username" field is a white rectangle with a light blue border and the placeholder text "Username". The "Password" field is a white rectangle with a light blue border and the placeholder text "*****". Below the input fields, there are three buttons: "Sign In" (a dark blue button with white text), "New Player" (a dark blue button with white text), and "Forgot Password?" (a link in dark blue text).

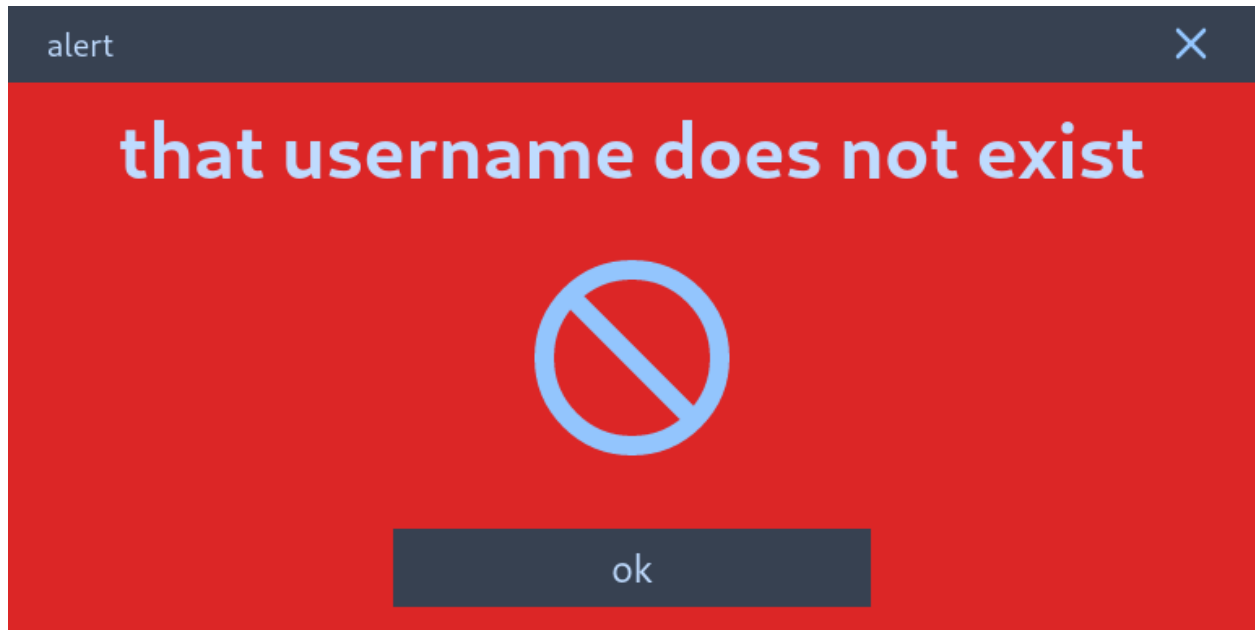
The login screen consists of a username input, a password input, a “sign in” button that submits the entered username and password to a database for verification, a “new player” button that opens the registration screen, and a forgot password link that would open some sort of menu to help the user recover or create a new password. For the scope of this project, password recovery won't actually be implemented.

Login Alerts

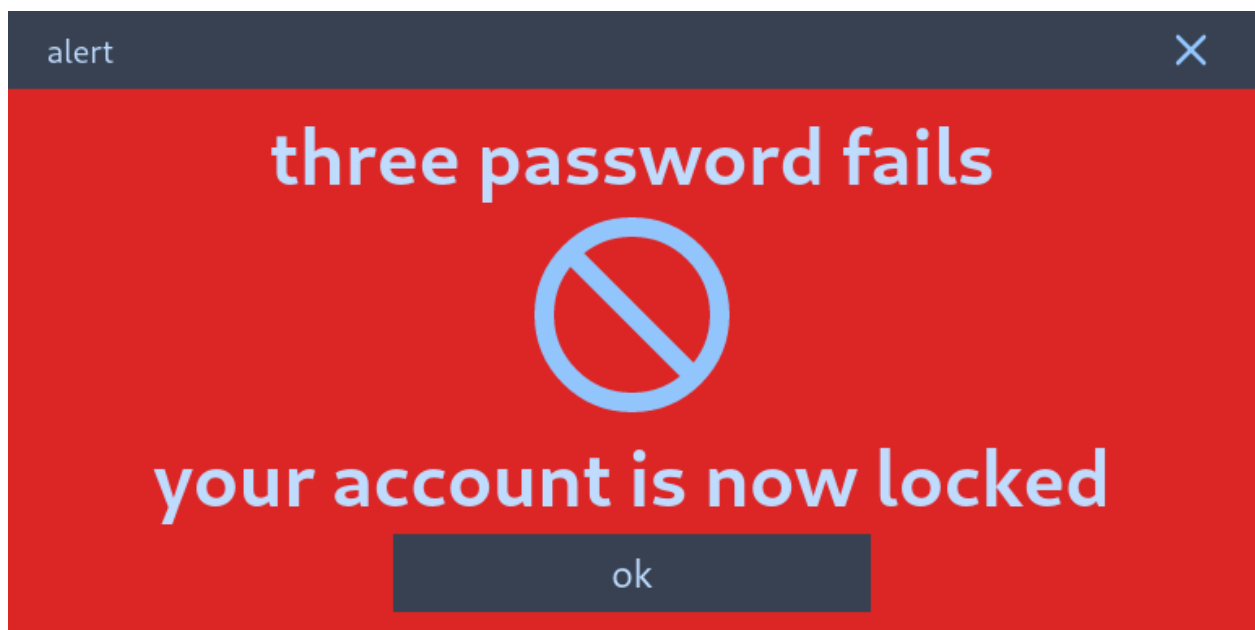
The following alerts are specific to invalid actions that occur via the Login menu.



This alert will trigger when a user tries to log in with a registered username and an incorrect password. The button returns to the login screen.



This alert will trigger when a user tries to log in with a username that is not yet registered.



This alert will trigger when a user enters the wrong password for a registered username three times.

Register

register

X

GALAXIAS REGISTER

Email

email@email.com

Username

Username

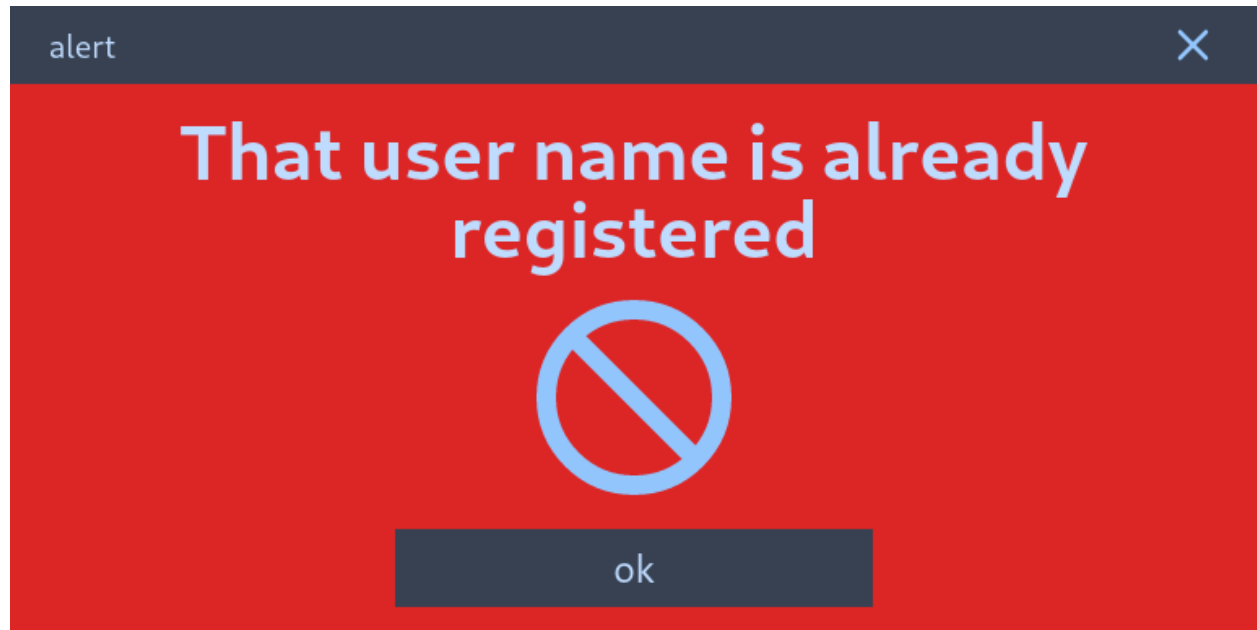
Password

Confirm Password

Sign Up

The register menu consists of an email input, a username input, a password input, a password confirmation input, and a “sign up” button that submits the info to the database for user creation.

Register Alert



This alert will trigger when a user tries to create a new account with a username that already exists. Clicking the “ok” button will return to the registration screen.

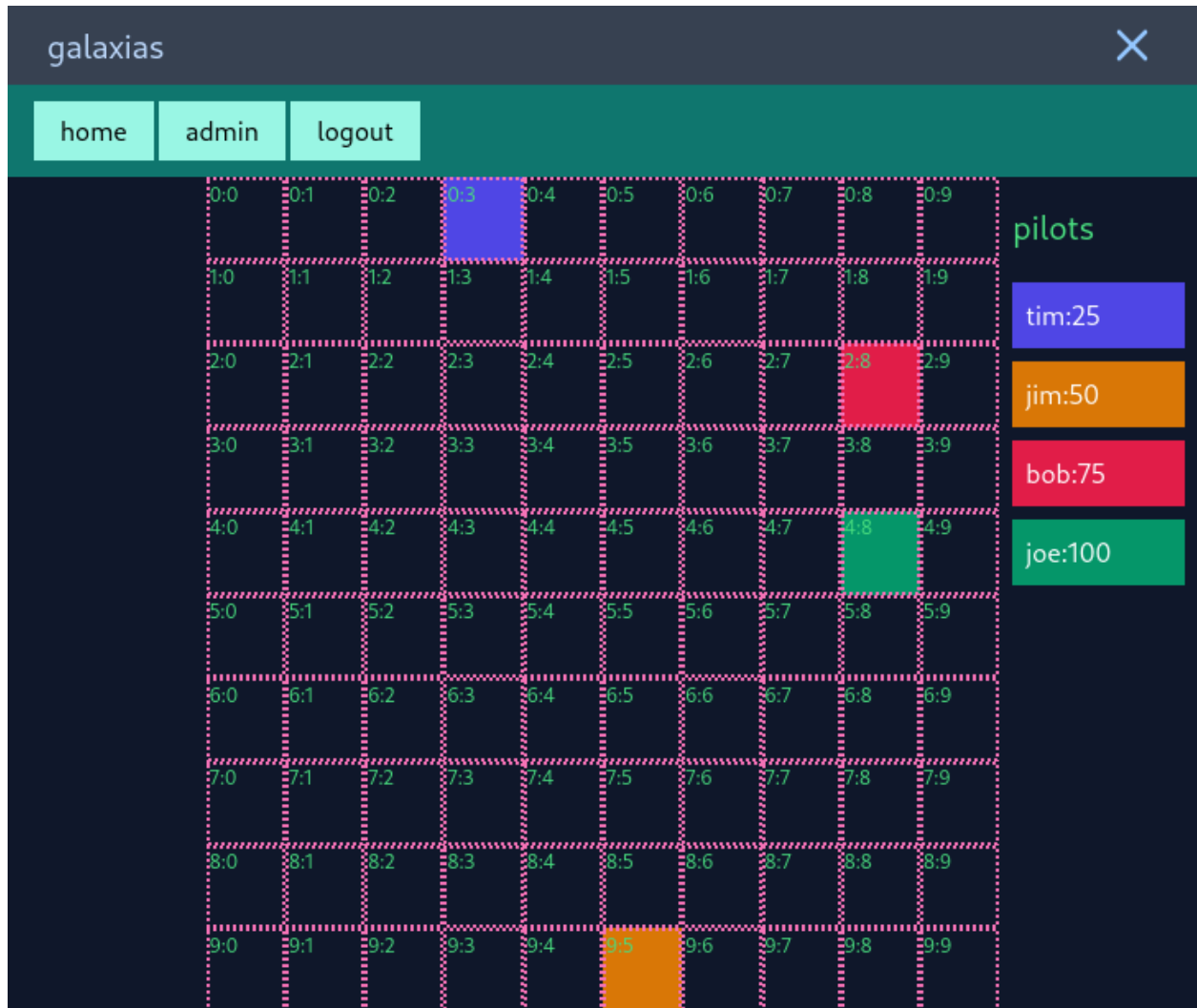
Home Menu



The home menu consists of a top row of navigation buttons: a home button that returns to this screen, an admin button that is only clickable by administrators and opens an admin menu, and a logout button that logs the user out of the game and closes the menu.

Below the navigation buttons, are two displays that show current ongoing games via the “Open Galaxies” display box, and a list of other players currently online via the “Free Pilots” display box. Below the Open Galaxies display box is an “enter galaxy” button that allows the player to join a current game, while the “new galaxy” button creates a new game.

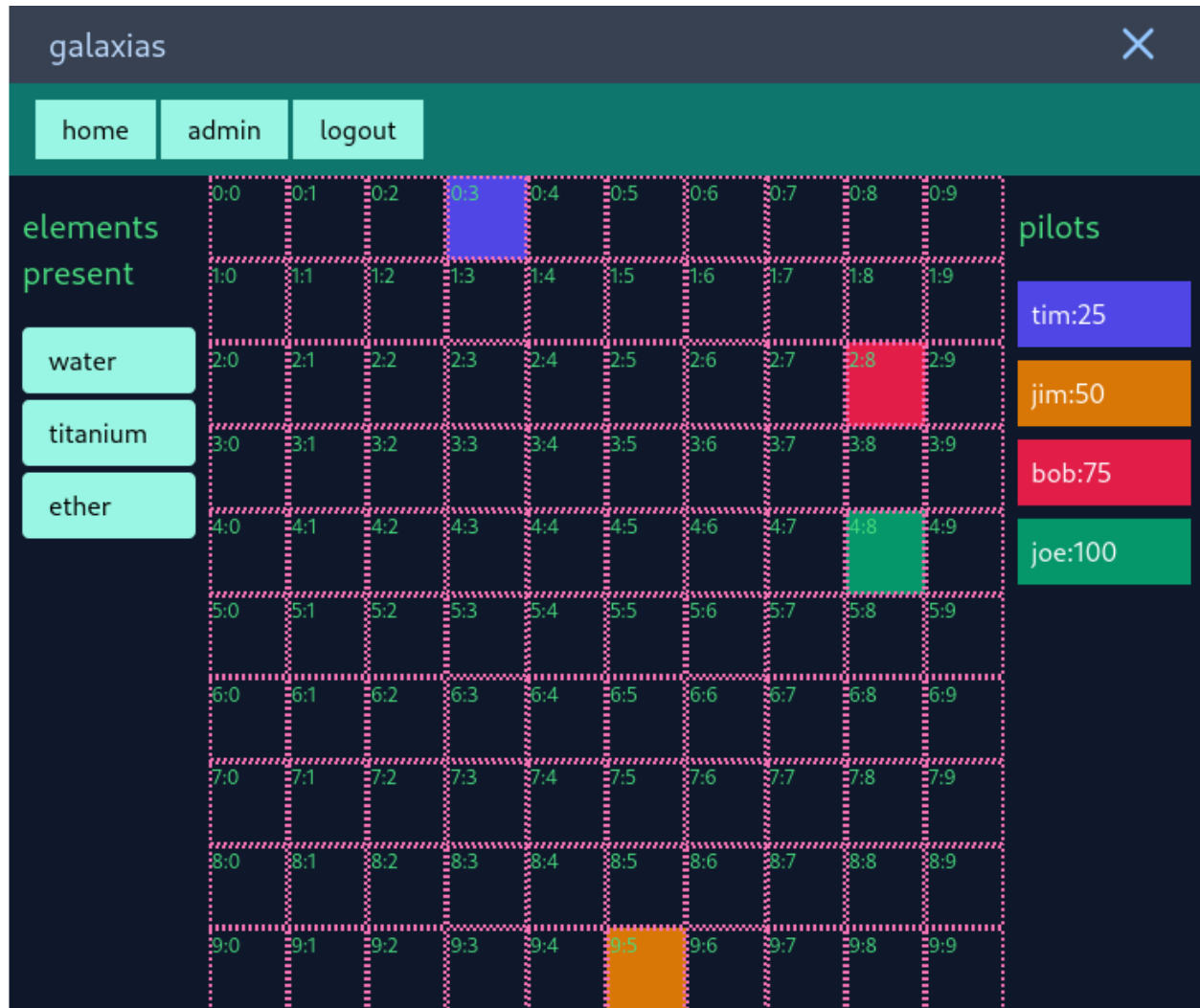
Main Game Display



The game display contains the same navigation menu as the login screen. The lower section of the game menu screen contains three main sections: on the left-hand side is an element selection menu bar that is blank in the above screenshot. In the middle is a grid that displays the galaxy or game board. And on the right is a list of players currently in this galaxy and their current scores.

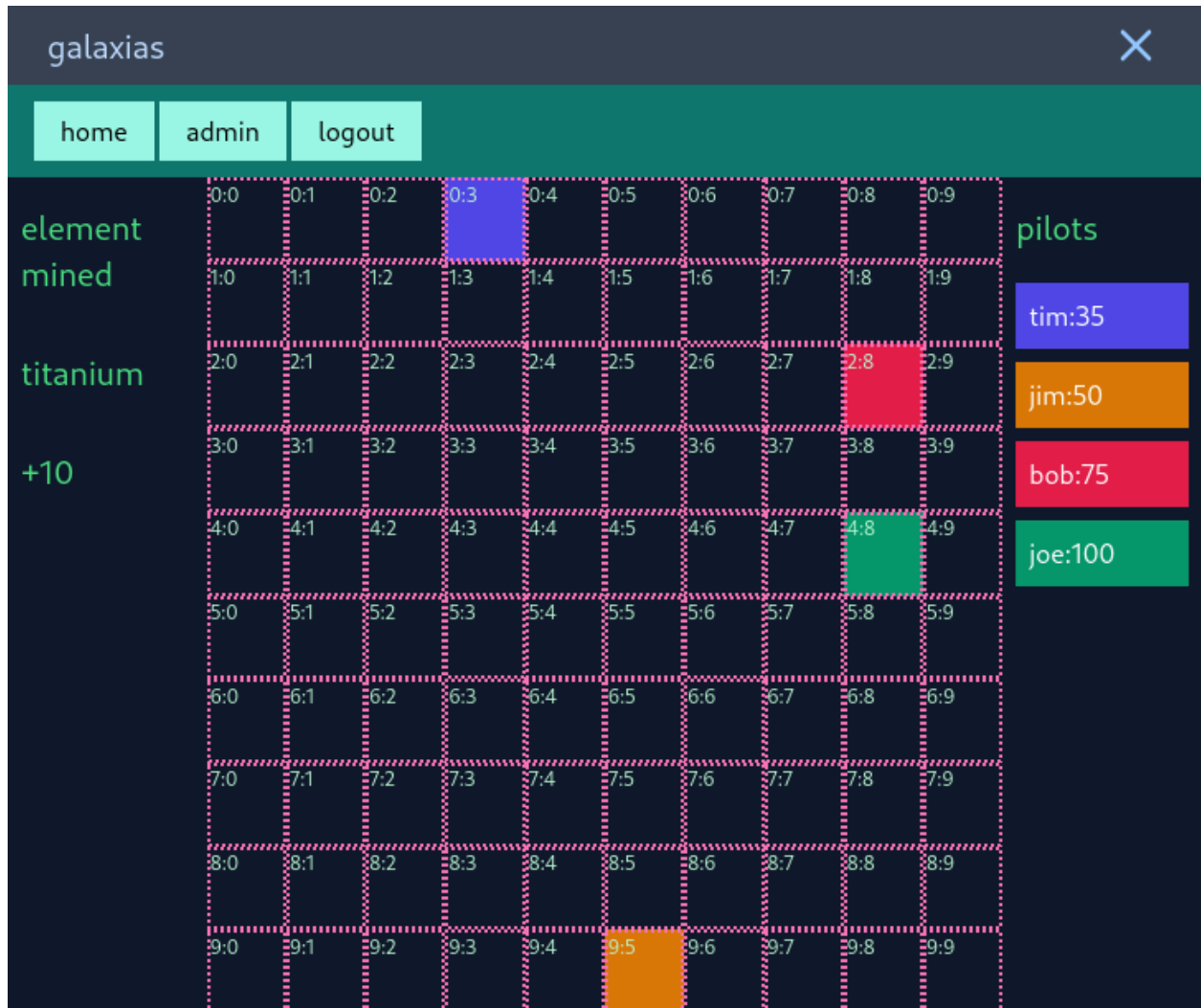
The galaxy grid has each vector's coordinates displayed in the upper left corner of the square. Colored vectors represent each pilot's location.

Main Game Display - Element Selection



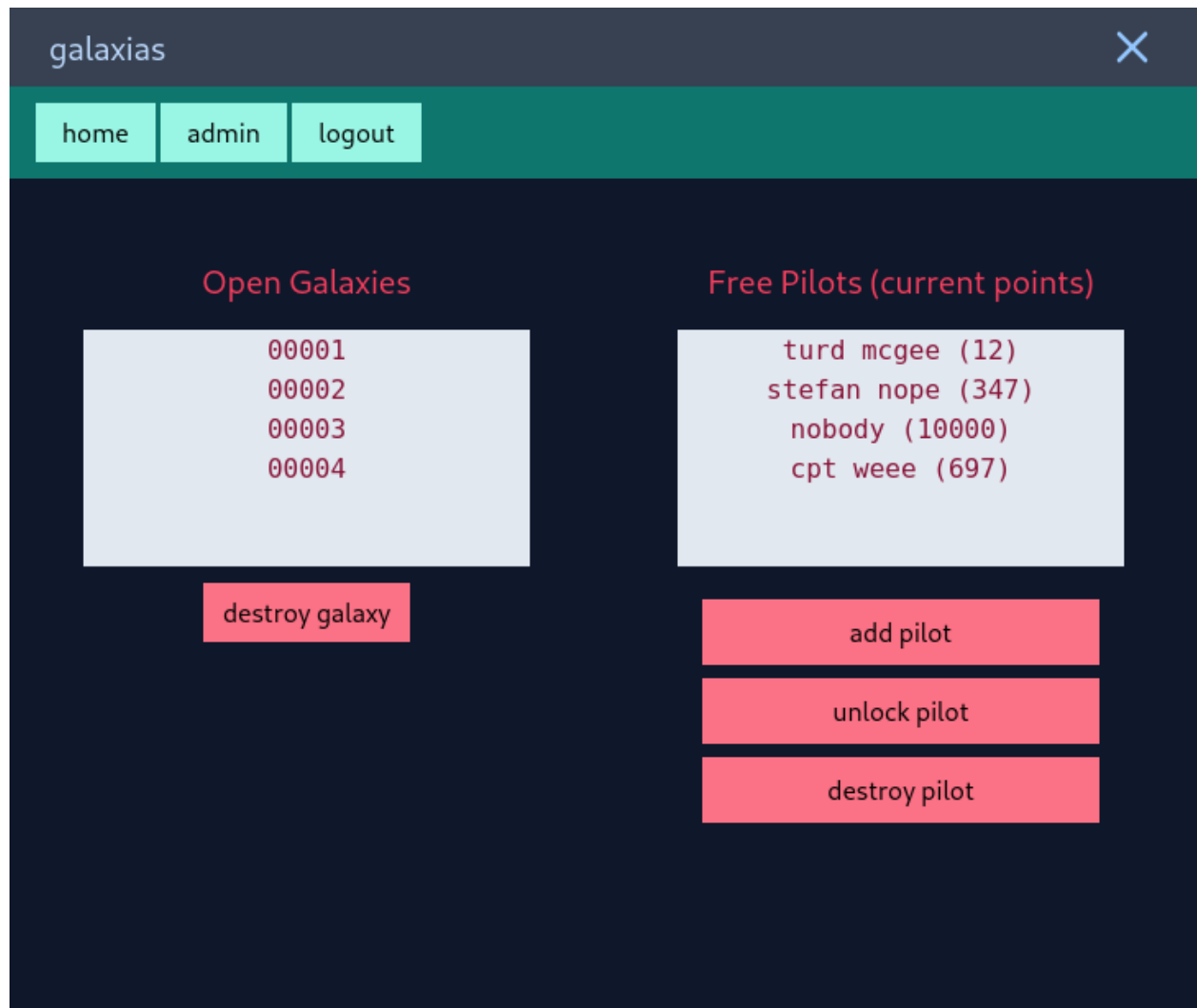
The game menu screen above shows the element selection menu when a player lands enters a vector with multiple elements present. A player can only select one element at a time, but may leave and then return to a vector to mine multiple different elements.

Main Game Display - Element Mined

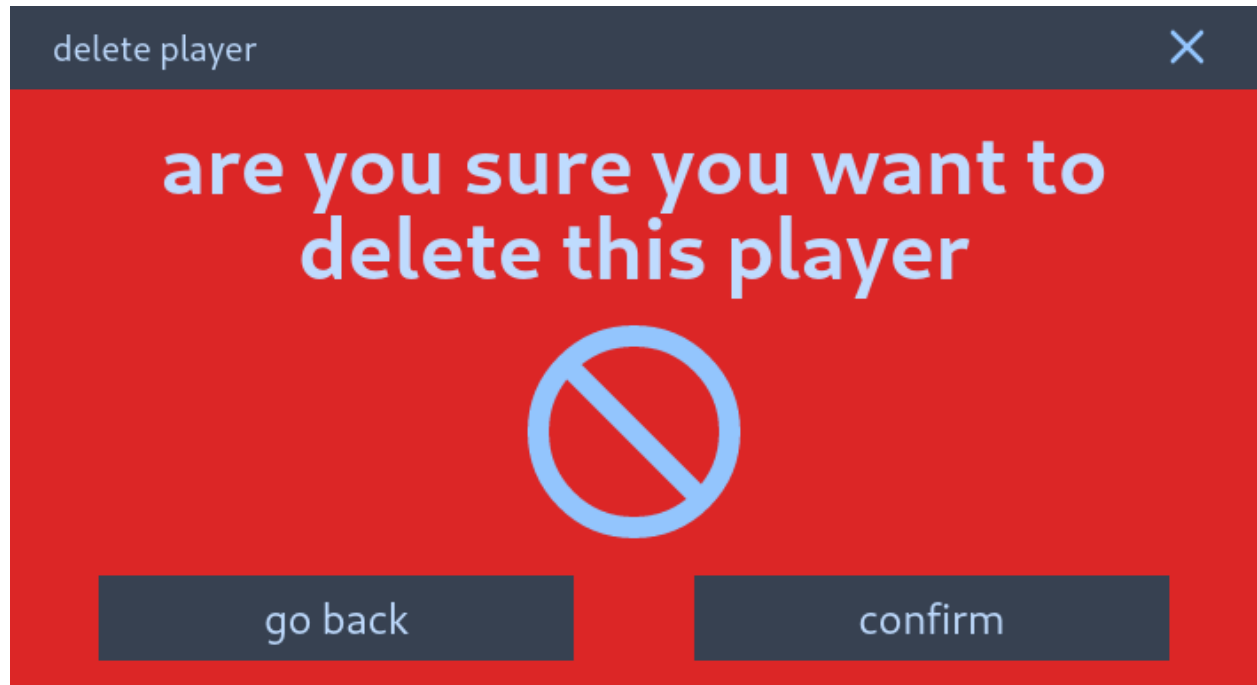


The game display above shows the element has been mined, the points added, and an updated score for the player tim to 35 points. The element menu on the left will not revert to an element selection menu until the player has moved into a different vector.

Admin Menu



The admin menu shows the same information as the home menu regarding open galaxies and pilots currently in exploration. Below the display boxes are the admin buttons for deleting a current game instance, adding a new user, unlocking a user, and deleting a user.



This alert menu will trigger when a player is selected and the “destroy pilot” button is clicked. Pressing the “go back” button will return to the admin menu without any modifications to the player or database. Pressing the confirm button will delete the player from the database and return to the admin menu.

admin

home

admin

logout

admin : add player

Email

email@email.com

Username

Username

Password

password

current score

highscore

☐ player is locked

☐ player is an admin

add player

The admin add player menu consists of an email input, a username input, a password input, a current score input, a checkbox to lock or unlock the player, a checkbox to declare if the player is an administrator with admin privileges, and an “add player” button that will submit the new player to the database.

admin

home

admin

logout

admin : update player

Email

phill@phill.com

Username

cpt phill

Password

●●●●●●●●●●●●●●●●

current score

444

☐ player is locked

☒ player is an admin

update player

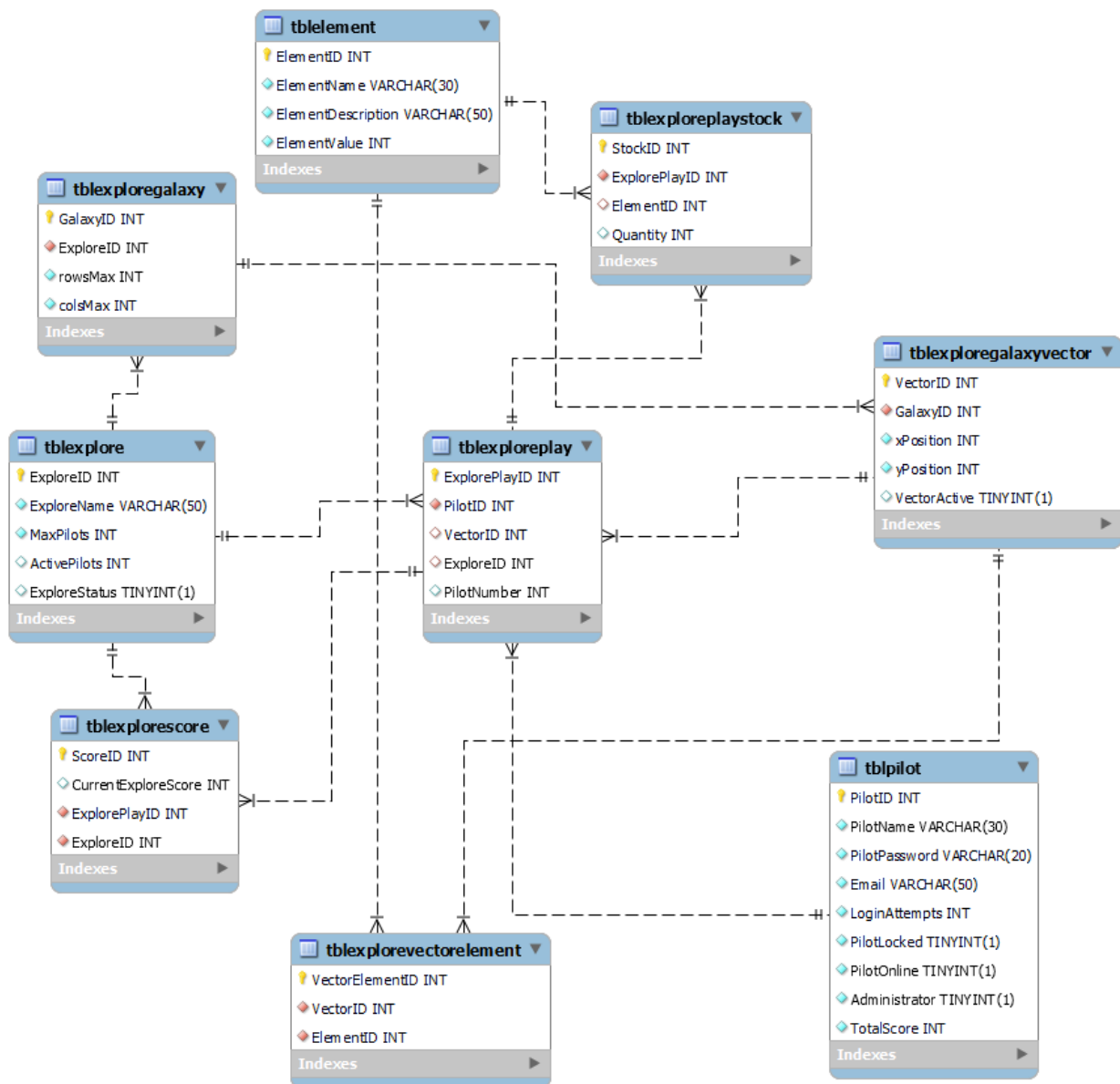
The update player menu has all the same elements as the add player menu, but the elements will be filled with the data of the player that is being modified. Clicking on the “update player” button will update the modified information in the database.

Menu Design Explanation

The game design storyboards provide a rough draft layout and general theme for the base of the game prototype. Some changes are expected, especially after user testing and feedback. An emphasis will be made to maintain simplicity and reusability of different menu components, this

will make the UI easy to use and the use of reusable components will shorten the time required for game development.

Entity Relationship Diagram



Entity Relationship Diagram Explanation

tblPilot

The Pilot table stores individual pilots' data.

tblExplore

The Explore table stores records of each exploration being played

tblExplorePlay

The ExplorePlay table bridges a pilot to an exploration.

tblExploreScore

The ExploreScore table updates and stores a score for each pilot in exploration.

tblElement

The Element table stores all the potential elements that can be discovered and collected during an exploration

tblExplorePlayStock

The ExplorePlayStock table holds all elements collected by a pilot in each exploration

tblExploreGalaxy

The ExploreGalaxy table creates an instance of a galaxy for each exploration.

tblExploreGalaxyVector

The ExploreGalaxyVector table holds all the vectors for each galaxy in each exploration.

tblExploreVectorElement

The ExploreVectorElement holds references to the various elements and their vectors within each galaxy in each exploration

CRUD Table

		pilot registration	pilot log in	lock pilot	delete pilot	pilot log out	join galaxy	new galaxy	pilot moves	admin menu	destroy galaxy
Pilot	C	R		D	RU	R	R		R		
PilotID	C			D		R	R		R		
PilotName	C	R		D		R			R		
PilotPassword	C	R		D							
Email	C			D							
LoginAttempts	C	RU		D							
PilotLocked	C	R	U	D							
PilotOnline	C			D	U	RU					
Administrator	C			D		R					
TotalScore	C			D	RU	R				RU	
Explore					U	R	C	U	R	U	
ExploreID					U	R	C		R		
ExploreName						R	C		R		
MaxPilots							C				
ActivePilots					U		C	U			
ExploreStatus					U				R	U	
ExploreGalaxy							C	R			
GalaxyID							C				
ExploreID							C				
rowsMax							C				
colsMax							C				
ExploreGalaxyVector							C	R			
VectorID							C	RU			
GalaxyID							C	R			
xPosition							C	R			
yPosition							C	R			
VectorActive							C	RU			
ExplorePlay				D	U	R	C		R	D	
ExplorePlayID				D	U	R	C		R	D	
PilotID				D		R	C		R	D	
VectorID				D	U	R	C	U		D	
ExploreID				D	U					D	
PilotNumber				D	U		C			D	
ExplorePlayStock				D			C				
StockID				D			C				
ExplorePlayID				D			C				
ElementID				D			C				
Quantity				D			C				
ExploreScore				D		R	C			R	
ScoreID				D		R	C				
CurrentExplorerScore				D	R	R	C	U		R	
ExplorePlayID				D		R	C				
ExploreID				D			C				
ExploreVectorElement							C	RU			
VectorElementID							C	R			
VectorID							C	R			
ElementID							C	R			
Element							R	R			
ElementID							R	R			
ElementName							R	R			
ElementDescription							R	R			
ElementValue							R	R			

CRUD Table Explanation

Pilot Registration

Trigger an insert command with related pilot info to populate and add a pilot to the Pilot table.

Pilot Login

Retrieve login-related info, reset login attempts to zero, and update PilotOnline to true..

Lock Pilot

After three unsuccessful login attempts the PilotLocked field will be set to true and the pilot will not be allowed to log in until reverted by an admin.

Delete Pilot

Delete all records from the pilot table and cascade to any foreign key references in other tables.

Pilot Log Out

Update PilotOnline to false, update the pilot's score and current exploration-related fields.

Join Galaxy

Retrieve the user information and all current exploration information from other current galaxies being played. This is found via the ExploreStatus field being true for an exploration. Then retrieve all pilot, galaxy, and exploration-related tables and data.

New Galaxy

Starting a new galaxy will create a new entry into the explore table and create related table entries need to start a new game.

Pilot Moves

Update the pilot location and score according to the last move.

Admin Menu

A pilot must have their Administrator field set to true to access this menu. This menu will retrieve all current exploration, galaxy, and pilot information.

Destroy Galaxy

Revert ExploreStatus to false and update all current pilot score data.

Milestone One Summary

For this milestone, a great many elements of the game were planned and developed. Great care and effort were made to consider all the broad strokes needed to develop this game. Various technologies were used, including:

- Mysql workbench
- Visual studio code editor
- SvelteJs (storyboard designs)
- GitHub
- Microsoft Excel

The first part developed was the game concept and functional requirements. A general idea of what the game is and what sort of basic functionality were needed before anything further could be developed. I have always enjoyed learning about space and playing video games involving space exploration, so a similarly themed game was an easy first choice. I used the assessment outline to align my game's basic functionality with the course requirements.

The second part was creating the table UML needed for the game. I sketched a rough draft and then used the MySQLworkbench GUI to create the tables, during this process, I fine-tuned the tables and necessary key constraints required for the game. Then I used MySQLworkbench to generate both the DDL and ERD for me. Using the GUI first to visually piece together the tables made a lot more sense to me and saved me a lot of time and frustration. Using the built-in features to generate the DDL and ERD also allowed me to quickly recognize issues and design flaws, then make the necessary fixes to the database infrastructure.

After the tables, DDL, and ERD were complete I wrote the required test queries for each table.

The third part was spent creating the storyboard designs. For this part, I used SvelteJs with TailwindCSS, which for me is way more comfortable and enjoyable than using other wireframe

software. I can simply load a live server, then use HTML with utility classes to quickly prototype my screen designs. I am aware that my screen designs will not be consistent with the final game designs considering C# and windows forms will most likely be used to complete the project.

To complete the final part of this milestone, I used Microsoft Excel to create the CRUD table. The CRUD table outlines the interactions with the database that are required to complete various actions during the game's life cycle.

The time required to complete this milestone was immense and unfairly took large chunks of time needed for other classes, again. There was not much learned during this milestone because most of this planning process has been covered ad nauseam in prior classes. There are little to no relevant resources for creating a game in this manner online, and finding helpful and focused resources on the moodle course section added to frustrations and time lost.

MILESTONE TWO

ACID

The acronym ACID refers to the four main attributes of a solid database system. The four attributes, atomicity, consistency, isolation, and durability, are all used to analyze each transaction that occurs within the system.

Atomicity

Atomicity is an all or none approach to each transaction. This means that either that transaction completes all the procedures within the transaction, or if there is a problem, it completes none of the procedures within the transaction.

For Galaxias, each procedure was created as a single function to ensure atomicity. When a procedure does not finish properly, the database is not modified or updated. A good example of this being important is that only one pilot can move within a galaxy at a time. This procedure of limiting game activity to a single pilot movement procedure removes the possible database conflicts that would arise from multiple improper calls that would make changes to a galaxy.

Consistency

Consistency refers to each procedure or transaction producing the same complete outcome. A procedure should never leave a database half modified, or incompletely updated. If a procedure has multiple changes, all changes must be reverted if the procedure fails for any reason.

For Galaxias, all procedures are written so that the database is only updated when the entire procedure is completed successfully. All galaxy data is removed and player data is updated accordingly when a galaxy is successfully conquered by a player.

Isolation

Isolation is the separation of procedures and transactions by purpose. It may at times seem efficient to group unrelated procedures that are called at the same time, but doing so can lead to conflicts and unforeseen consequences. Isolation helps guarantee separation of purpose and that focus is given to a single priority for each procedure.

For Galaxias, each procedure was written to accomplish a single purpose driven transaction. This is especially important with a multiplayer turn-based game. A strong focus and emphasis was placed on completing each single player transaction with the most recent data.

Durability

Durability refers to the database's ability to recover from any type of shutdown, normal or abnormal. Each procedure should update relevant data so that if the system is improperly shutdown, the data persists when the system is turned back on.

Galaxias achieves durability by updating and storing relevant data to the database after each successful transaction, so that if the next transaction were interrupted or the system shutdown, the prior data would still persist in the database.

MILESTONE THREE

Galaxias 2.0 GitHub repository : <https://github.com/Phillip-D-Shields/firebase-js-galaxias>

EXPLANATION

For this milestone I have submitted a different project than was worked on during the first two milestones. There are a number of reasons for this that will be presented and explained. Parts of this section will also be used in my process review.

The primary reason was a lack of understanding of the programming language C#. I have struggled with this language in multiple classes and have had no breakthrough. I spent an excessive amount of time on the first windows login form and accomplished no functionality. The frustration and desperation became overwhelming when I realized that I had a few days left until this milestone was due and had not been able to accomplish anything.

A secondary reason was the lack of time. This semester has been full on, my final project, a level seven paper, and this class have made time management a minefield. This class was third in priority of importance. And there was definitely not enough time for me to try and relearn the fundamentals of C#.

I decided at the beginning of this week to accomplish this milestone with a tech stack that I was more knowledgeable in and passionate about. This is the last assessment I will ever submit at NMIT and I would like to end things on a strong note. My first two milestone submissions demonstrate an adequate understanding and command of MySQL. This milestone demonstrates that I can also implement similar functionality via another technology stack.

GALAXIAS 2.0 INTRODUCTION

Galaxias is an online multiplayer game that can be played anonymously by any user with a web browser. The objective of the game is to collect as many stars as possible and earn the highest score. A user can change their pilot call sign (name), change their ship color, and if they can achieve the highest score, their name and score will be displayed at the top of the screen. A user can move around the screen using the up, down, left, and right arrow keys to collect stars. Each star is assigned a random color and value, emulating the chaos and unpredictability of the cosmos. Some stars even have a negative value.

All the game data is updated in real time and all player data is displayed correctly on other players screens.

The tech stack used to complete this game prototype is as follows:

1. Front end
 - a. HTML
 - b. CSS
 - c. JavaScript
2. Back end
 - a. Firebase real time database

I decided to use a fundamental front end stack of HTML, CSS, and JavaScript because it is simple and efficient. I can quickly prototype and I am very comfortable working with this stack. Most of the heavy lifting is done by JavaScript, which is used for game logic and connecting with the backend via firebase modules.

The back end uses Firebase, which is simple to work with and allows you to get a basic NoSQL database up and running with basic rules defined in under five minutes. There are a wealth of features that come along with using the Firebase realtime database that will be discussed in the following sections.

Since this was completed in about five days time, the primary focus was to complete a functional multiplayer game that allowed unique login and stored essential data to the database. That game instance would then be used to create an application that allowed for multiple game instances. Other features that were not completed, but are essential are discussed in the **UNFINISHED ASPECTS** section and an explanation of how they would have been completed is discussed.

FINISHED ASPECTS

Login and Registration

Galaxias uses an anonymous login feature provided by Firebase. This lets me implement user-specific security and storage features without requiring actual credentials from a user. A unique ID is created for each user specific to their browser connection instance. Once a user connects to the game this process is handled automatically for them, and a player object is created in the real time database that contains their ship color, unique id, chosen name, star points, and grid location. This player object updates every time a player moves or makes changes to their name or ship color. A screenshot example is provided below.



Game instance

There is only one instance of galaxias currently available. A paid tier would have been required to implement a database containing multiple game instances. An explanation of how to accomplish multiple game instances that players could join and leave is discussed in the **UNFINISHED ASPECTS** section. Any new users that connect will join the current single game instance. The database for this game updates in real time and contains three key pieces of information:

1. High-score:
 - a. Player name
 - b. Player score
2. Players:
 - a. Example id
 - i. Color
 - ii. Direction facing
 - iii. ID
 - iv. Name
 - v. Stars collected
 - vi. Current X coordinate
 - vii. Current Y coordinate
3. Stars:
 - a. Example star id (X Y coordinate)
 - i. Star value
 - ii. Star X coordinate
 - iii. Star Y coordinate

A screenshot example of the game instance database is shown below.

galaxias-js ▾

Realtime Database

[Data](#) [Rules](#) [Backups](#) [Usage](#)



Protect your Realtime Database resources from abuse, such as billing fraud or phishing

[Configure](#)

<https://galaxias-js-default-rtdb.asia-southeast1.firebaseio.com>

`https://galaxias-js-default-rtdb.asia-southeast1.firebaseio.com/`

▾ — high-score

name: "DEAR GOAT"
score: 5

▾ — players

▾ — d2SJhJ5RnOgFjaF7tLXpdG4fKX13

color: "pink"
direction: "right"
id: "d2SJhJ5RnOgFjaF7tLXpdG4fKX13"
name: "DEAR GOAT"
stars: -6
x: 18
y: 5

▾ — stars

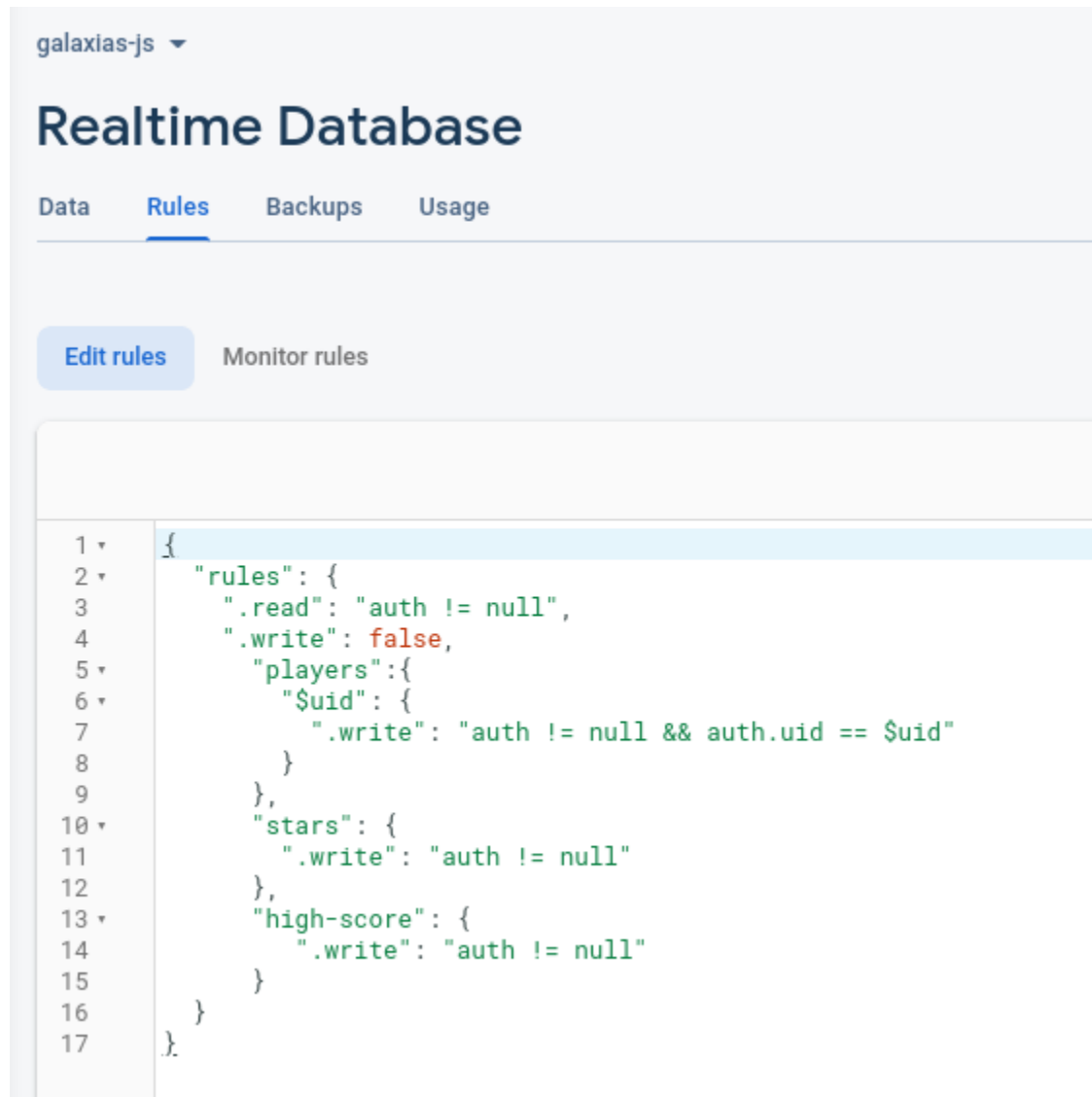
▾ — 0x0

value: 5
x: 0
y: 0

▶ 0x1

▶ 0x13

The realtime database is configured via a rules screen. A number of rules can be added and modified, then published and implemented in real time. For this game a very simple set of rules are established that require a user to be authorized before their player can access and write new data to the database. A screen shot example is provided below.

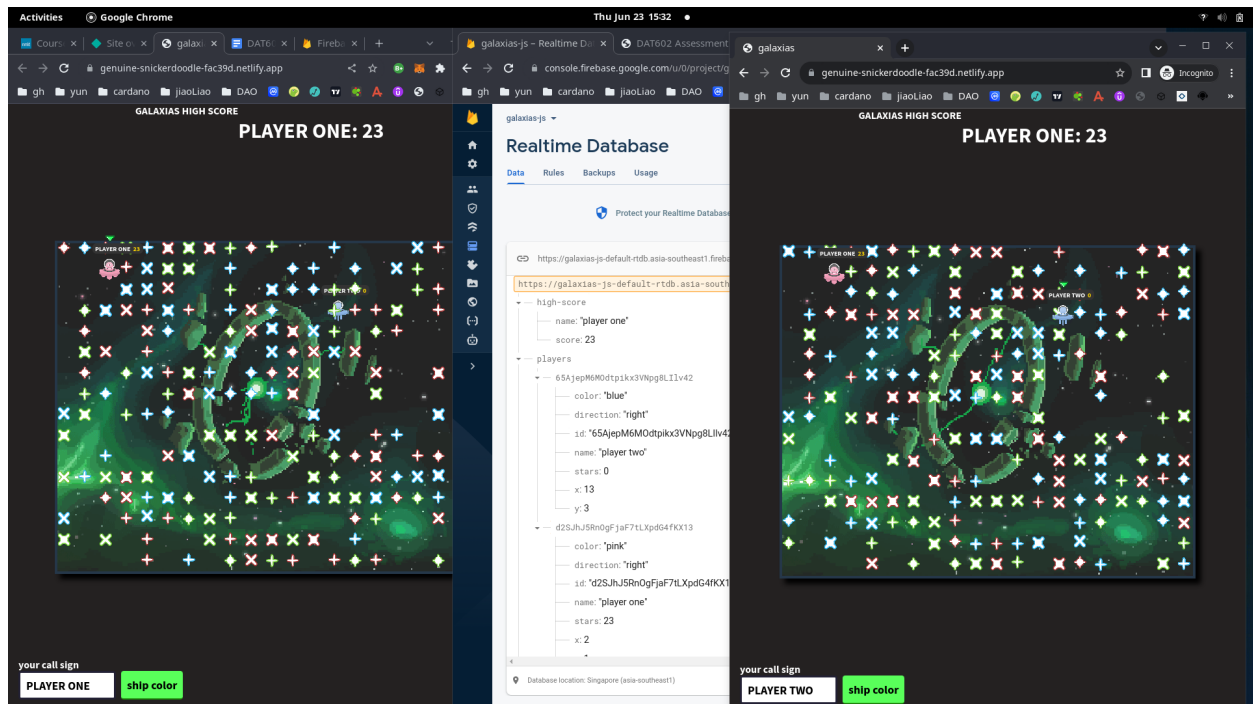


Game logic

The logic used for player movement, point allocation, and database actions are contained within the two JavaScript files, app.js and KeyPressListener.js submitted with this milestone. Comments have been added to the code to explain what each code section accomplishes.

Game display

A game play display has been successfully developed and finished. The display shows the board, stars, players, player names, player scores, current high score, and inputs that allow a player to change their name and ship color. The game display is updated with the database in real time and is consistent across all players screens. A screen shot has been included below.



A working demo of the game has been deployed via Netlify and can be accessed at this url : <https://genuine-snickerdoodle-fac39d.netlify.app/>

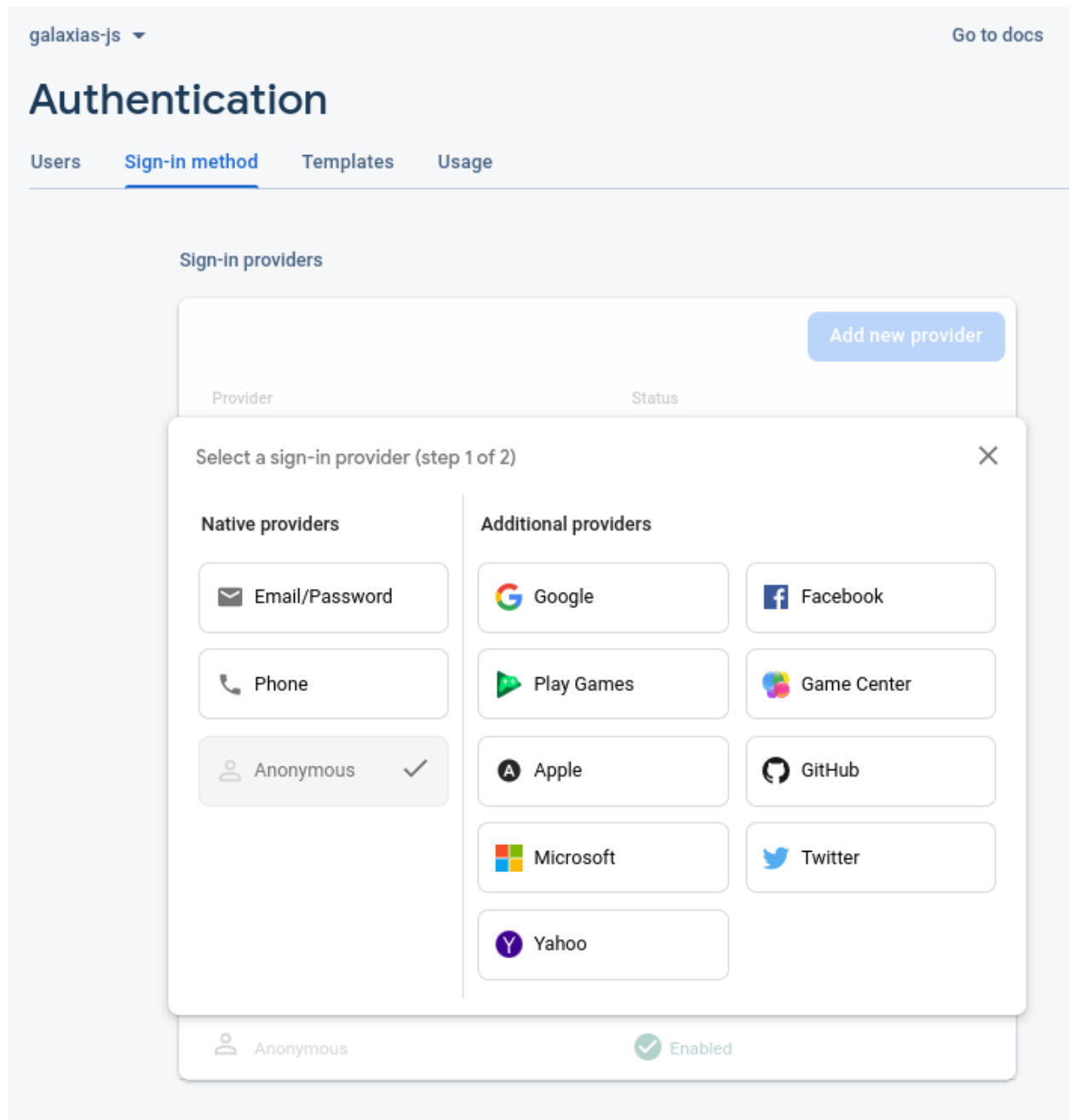
In order to simulate multiplayer you can open the game in your browser, then open the same url in a private or incognito browser instance. This simulates two different users on different machines accessing the game.

A video demonstration of multiplayer gameplay and real time database changes can viewed on YouTube via this url : https://youtu.be/_O52A_luim0

UNFINISHED ASPECTS

Registration and Login

Firebase has a number of options available for registration and login, as shown in the screenshot below.



The anonymous login was the most direct solution to implement given the time constraints. But all the options are well documented and general code snippets and instructions can be located for each.

Game instance selection

Since only a single instance of this game was created, a menu to select from multiple instances was not yet created. To create multiple instances, the database rules would need to be changed to reflect that instead of three different variables, high-score, players, and stars, a single root variable containing all game instances would be created. Within this single game variable would be multiple game instances that each store four variables, game-id, high-score, players, and stars. This would have allowed for a simple menu to be added that allowed the player to select between current game instances available. I had already designed and planned this unfinished aspect but ran out of time before I could begin to implement it. An example of the different database structures is shown below.

CURRENT SINGLE GAME INSTANCE DB STRUCTURE

- High-score
 - Name
 - Score
- Players
 - Player id 00
 - Color
 - Direction
 - Id
 - Name
 - Stars
 - X
 - Y
- Stars
 - Star id
 - Value
 - X coordinate
 - Y coordinate

MULTIPLE GAME INSTANCE DB STRUCTURE

- Games
 - Game id
 - High-score
 - Players
 - Stars

The actual menu selection user interface would have been simple to implement, and the storyboards would have been referenced as a guideline.

Admin controls and menu

With the current game configuration and trajectory of being a web based browser experience an admin control screen was not a priority. For the time being all admin functionality is more easily accessed via the Firebase console. However, the logic and functionality could be implemented via JavaScript and an admin dashboard that would allow an admin user to:

1. Create new games
2. Modify current games
3. Delete current games
4. Create new players
5. Modify current players
6. Delete current players
7. Modify visual display presented to all players

This was not considered a priority for completing this prototype, but given further time could be completed. User and admin permissions can be controlled via Firebase configurations, so establishing a user as an admin with specific permissions can be done via the project settings menu as shown below.



Project settings



< General Cloud Messaging Integration Service accounts Data privacy Users and permission >

🔍 Search members

Add member

Member ↑

Roles

Email address(es)

❗ Enter at least one email address

Role(s)

- ☒ Owner ⓘ
- ☐ Editor ⓘ
- ☐ Viewer ⓘ
- ☐ Assign Firebase role(s)



Phillip
chemical.salamander.40@gmail.com

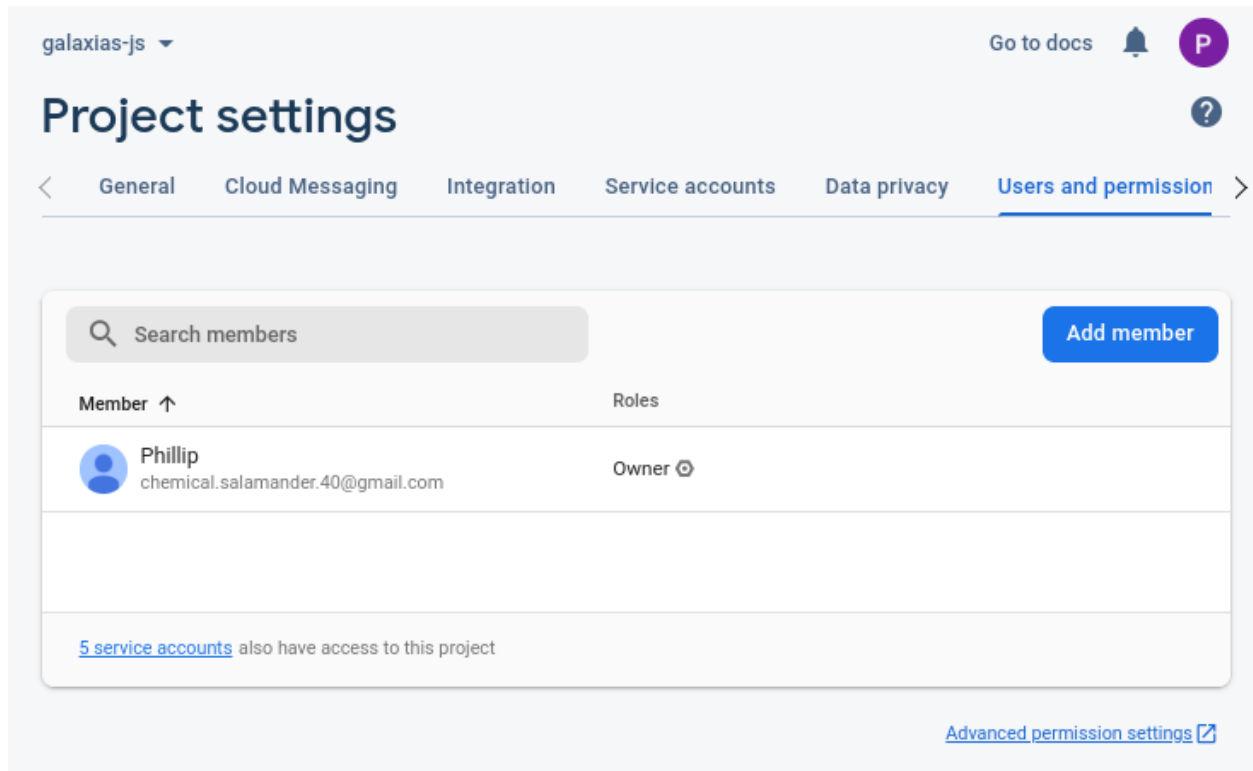
Owner ⓘ

Cancel

Done

[5 service accounts](#) also have access to this project

[Advanced permission settings](#) 🔗



Galaxias currently only has one main admin account, that is established with the owner role, the owner role gives permission and access to all options. An editor role is available that gives the user limited permissions and admin controls as defined by the owner account. Only the owner account would have direct access to the firebase console, while editor roles would be given permissions and allowed to make changes available via the proposed admin dashboard. This has been planned and designed but was not completed before the submission deadline.

Data persistence

Firebase handles this within its project configurations by default, and if the game were made into a stand alone application there are a few Firebase project configuration settings that would allow data to be stored locally when offline, then the database updated once the application was reconnected to the internet. A stand alone application for this game could be achieved via electron or similar technology, but was beyond the scope of this preliminary development.

FINAL DEFENSE OF GALAXIAS 2.0

My experiences and exposure to C# here at NMIT have left a miserable impression of the language. The language itself is powerful, popular, and probably much more useful than I currently understand. Multiple NMIT classes had required me to complete a high level application with C# before completing SDV601, which is essentially our introduction to C# and object oriented programming. When I did take SDV601, I struggled deeply for a number of reasons, and personally made no progress with the language and little progress with OOP. Had I been allowed to use JavaScript, Python, or Typescript for the entirety of those classes, I could have focused on OOP principles and completed projects and assessments that I am proud of, and that demonstrate my skills as a developer. Instead I have a series of incomplete projects, hacked together in an attempt to survive a marking rubric.

MySQL is another story. I've known and understood MySQL since before I started here. I am by no means an expert in the language, but know which documentation to reference when I need to accomplish complex query commands. My struggle in my previous attempt at this class was not related to using MySQL. The difficulty arose from implementing some game logic in MySQL and then using C# for the UI. Both those concepts seemed extremely foreign to me. I tried to express that I needed to better understand how to develop a high level C# application with such little understanding of the language. Perhaps I did not adequately express this when seeking additional assistance.

This time around, my first two milestones demonstrate my ability in MySQL. And to an extent, a basic understanding of C# through my console app submission. There was no way I would have been able to complete an entire collection of functional windows forms, displays, and menus to complete this milestone, especially considering my class load this semester.

Galaxias 2.0 is my final act of defiance here at NMIT, it is my attempt to end these three years on my own terms, with an assessment submission that I am proud of and that could be further developed for use within my digital portfolio. I was able to develop and deploy a Galaxias 2.0 prototype that achieved parallel functionality with some of the basic requirements in about five days time. There is no doubt that given more time, I could have achieved a more complete and robust version of Galaxias that met or exceeded the requirements given.