Documentation Percussion Islands

Salvador Salgado Normandia A01422874
Andrés Briseño Celada A01352283
Iwalani Amador Piaga A01732251
Iván Rodríguez Cuevas A01781284

User story:

User story: #1 Focus on instrument percussion	
As a association **I want** a fun and interactive video game **So that** we can teach students about the vast world of percussion instruments and their importance in music.	
Validation: • Survey to find out if the game was fun • Observe new knowledge acquired	**Priority:** 10 **Estimate: 25h**

User story: #2 Mechanics focused on pedagogy	
As a association **I want** establish pedagogically focused mechanics within the videogame **So that** we can teach in a fun and effective way.	
Validation: • Scoring that evaluates the player's performance in order to verify learnings	**Priority:** 10 **Estimate: 20h**

User story: #3 Student progress record	
As a teachers/collaborators of Percussive art society **I want** to evaluate the progress of those who use the game **So that** we can corroborate that the teaching method is effective.	
Validation: • Scoring that evaluates the player's performance.	**Priority:** 10 **Estimate: 5h**

User story: #4 playable on web browsers	
As a association **I want** a videogame accessible in web browsers **So that** users can play and share it with ease	
Validation: ● Performance tests on Google Chrome and Firefox	**Priority:** 10 **Estimate: 3h**

User story: #5 Single player	
As a association **I want** the video game to be a single player game. **So that** we can evaluate each student individually	
Validation: ● Score that evaluates the player's performance (individually)	**Priority:** 10 **Estimate: 10h**

User story: #6 SQL schema connection to Web	
As a association **I want** the data that is created on the web to be stored in schemas **So that** we can analyze and visualize the results of the students.	
Validation: ● Visible data storage, with dates and input values.	**Priority:** 10 **Estimate: 10h**

User story: #7 PK Auto_increment value	
As a association **I want** there to be a PK that correctly identifies the columns to generate values **So that** values can be generated when inserting a record in a table-schema	
Validation: ● PK helps prioritize the index to use and detects duplicate UNIQUE keys	**Priority:** 8 **Estimate: 4h**

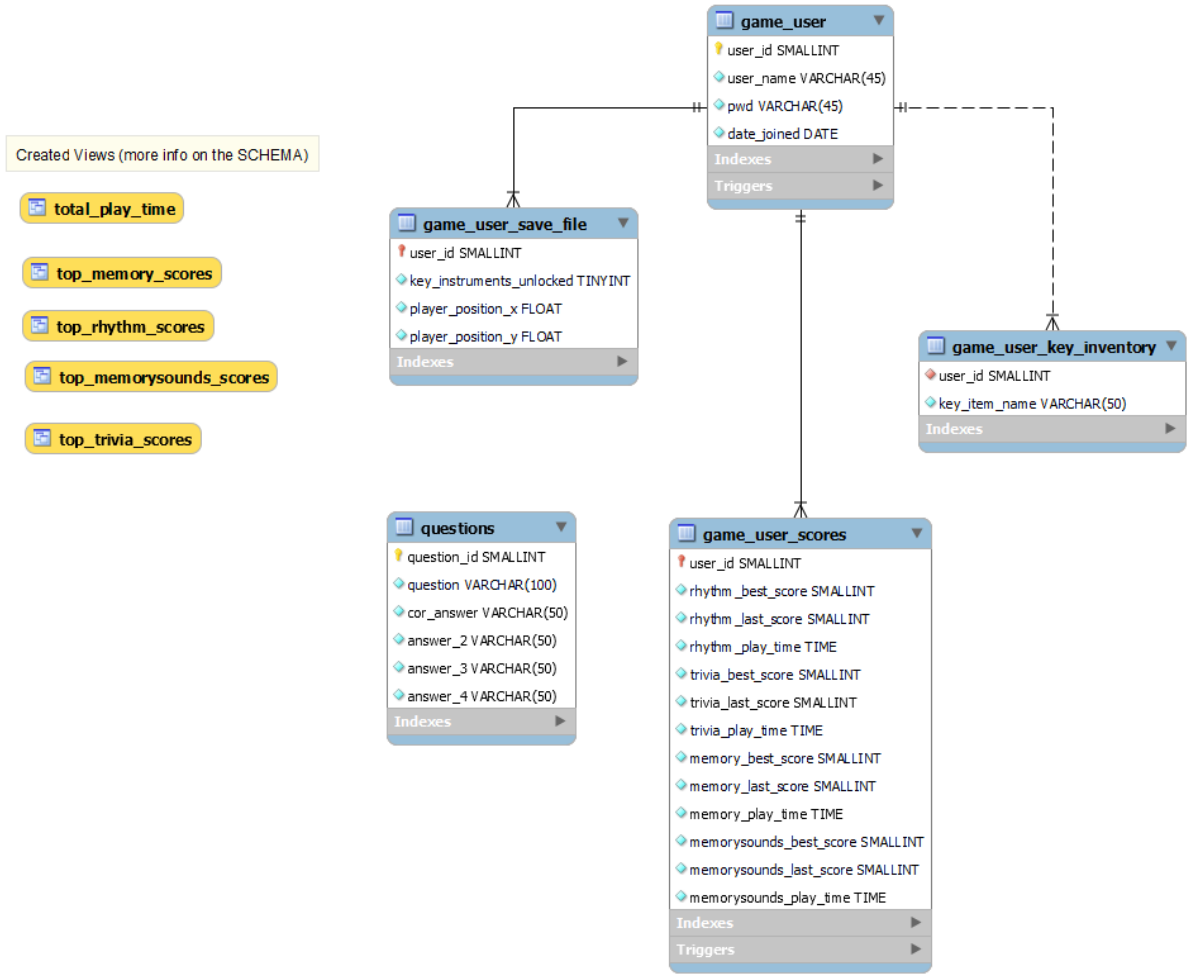
User story: #8 Score inquiry on the web page	
As a teachers/collaborators of Percussive art society (PAS) **I want** to easily see the scores of the players without accessing other apps **So that** we can evaluate the performance of our students	
Validation: ● Scores are displayed along with the name, without leaving the game page.	**Priority:** 8 **Estimate: 3h**

User story: #9 Player's experience	
As a association **I want** the game not to be difficult and tedious from the player's perspective. **So that** we can pass on the knowledge to those who do not know about percussion.	
Validation: ● The game gives results on learning and introduction to percussion. ● They complete the game effectively and with good results.	**Priority:** 8 **Estimate: 10h**

User story: #10 Game progress	
As a Player / PAS students **I want** the game to be intuitive **So that** I can navigate the islands and make use of the features added to the game.	
Validation: ● The player adapts to the game mechanics, understands the movement between islands, makes use of the inventory, plays all the mini-games, interacts with the NPCS ● Completes the game with good results and knowledge	**Priority:** 9 **Estimate: 40h**

Entity Relationship Diagram (ERD)

Once we understood the various user stories, as well as the functional and non-functional requirements required for this project, it was time to design the Entity-Relationship Diagram used for our database.



As observed in the image above, we created five tables in our Diagram, some containing foreign keys that link them with each other.

Game User (game_user)

This table contains the most relevant information about the game users. To correctly identify each user, we declared a unique primary key (user_id) that will allow us to better access the data for later modifications.

As we all know, it is also important to have users with unique names, by which we also declared the user_name value as unique.

It is necessary to mention that once a new user is created, we implemented a trigger that would generate a new record in the table with its foreign key associated with user_id. This

trigger allowed us to later update the information of the other tables without having the need of creating them at the moment.

Similarly, we established the option of deleting and updating the tables that contain the user_id. If a user is deleted or updated in the table game_user, it will reflect on the others.

Game User Save File (game_user_save_file)

This table contains the information required to load the user progress of the game. It is necessary to mention that its primary key consists of user_id, which means it is a foreign key from another table. This relation (1:1) allows the access of specific information from the users with only the user_id.

Game User Save File (game_user_save_file)

This table contains the different instruments that the user has obtained in their inventory. Similar to the one mentioned above, the table has a foreign key as its primary key (user_id).

Game User Scores (game_user_scores)

This table contains the best score, last score, and total playtime for each minigame played by the user.

At first hand, it might seem that it is too big. In the beginning, we had planned to create different tables for each minigame but found out that obtaining the information would be far more tedious.

Grouping all the minigames in one table allowed us to modify the values more efficiently. Because all these scores are associated with each user, we declared the primary key as the foreign key of game_user.

It is necessary to mention that each minigame is independent of the others. Hence we needed to implement triggers that updated only the values corresponding to the current minigame played. (More details of the triggers in the schema)

One of the main objectives of this project was to show the data obtained from the players in a better way. With this in mind, we decided to create views in our database that would select specific information about the current tables. Some of the views created were the sum of every minigame playtime and the top ten scores in every minigame.

Questions (questions)

This table contains the Questions and the respective answers used in the Trivia minigame. Each of the Questions has its unique id (question_id) and columns that contain the question, the correct answers, and the false answers. Once the Trivia Minigame begins, the game makes an API call which obtains all the records of the table in order to show them in the game. Because this table isn't related to our user we don't need to add a foreign key to establish a relation.

ERD Normalization

While designing Entity Relationship Diagrams it is important to take into consideration the normalization, which allows for a better organization of data that will optimize the search and organization.

First Normal Form (1NF)

- Each of the Tables contains a unique primary key (in some of them we consider the foreign key as its own primary key)
- The primary keys don't allow for null attributes
- All the columns have the same number of elements. We prevented this by declaring default values.
- Every attribute has been atomized in the smallest way possible
- The values of each attribute can no longer be divisible
- There is no repetition of data in the tables

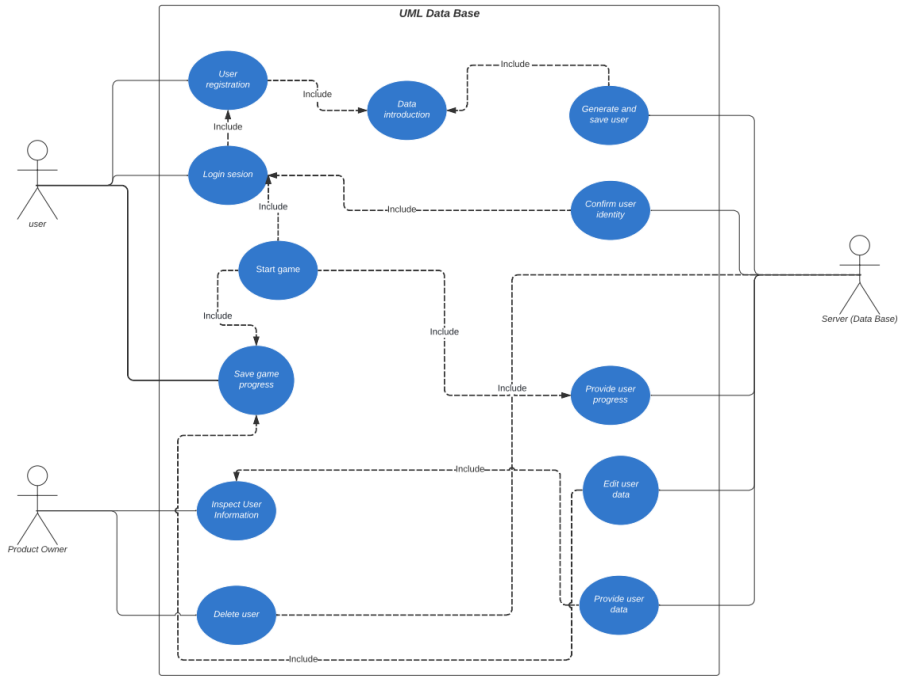
Second Normal Form (2NF)

- It doesn't exist functional dependencies in our tables, meaning that all the columns in the tables depend solely on their primary key.
- If there were functional dependencies it was necessary to create subtables (game_user_scores, game_user_save_file, game_user_inventory)

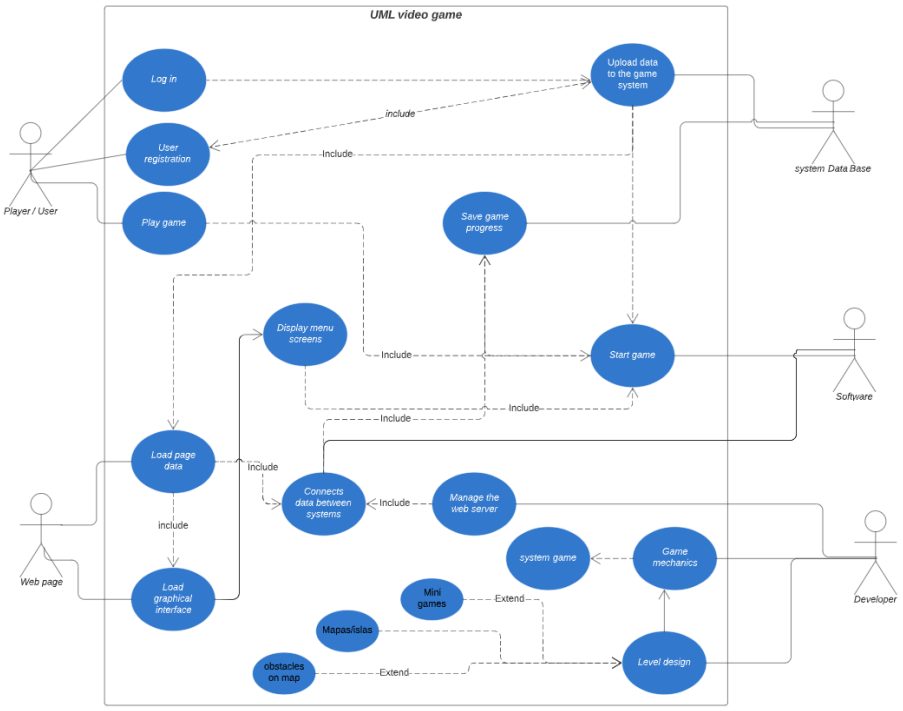
Third Normal Form (3NF)

- The tables in our database don't have transitive dependencies, meaning that every column depends solely on its primary key.
- It is important to mention that some columns are defined by update triggers(best_scores), but these are related to each primary key which allows modifications to that specific column.

Use Case Diagrams:



New Diagram UML - Data Base (LucidChart):
https://lucid.app/lucidchart/5b76b205-eb0e-428b-a025-952f22450ea4/edit?invitationId=inv_f57906ab-4373-491b-a5a4-24dcbdf40d6d



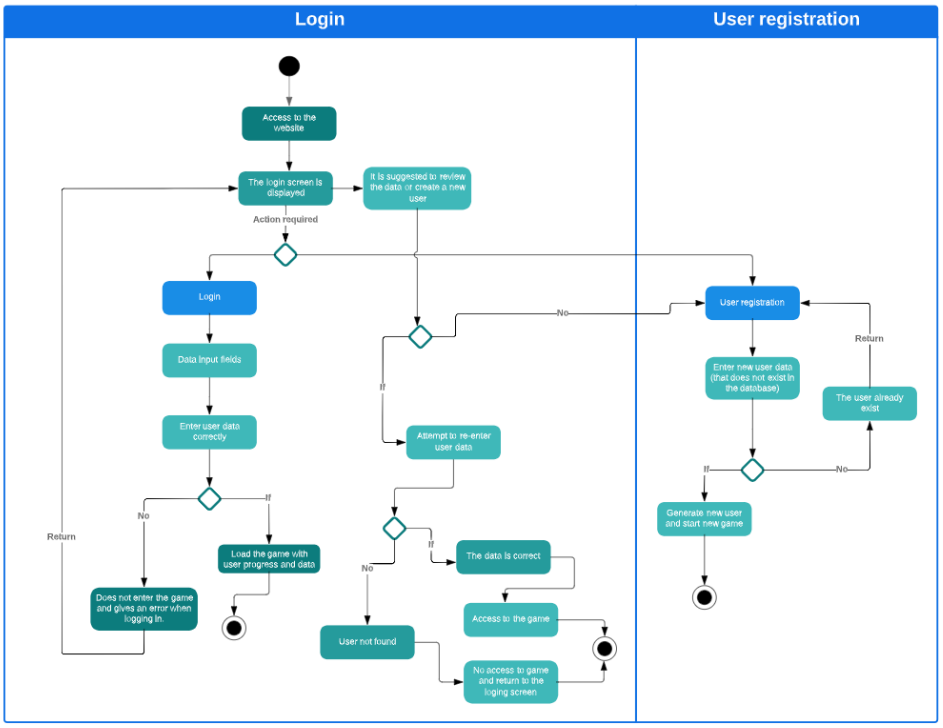
New Link Diagrama UML Videojuegos (LucidChart):
https://lucid.app/lucidchart/b06e335c-adb3-40ff-8275-7b3e08ad6a83/edit?invitationId=inv_e6ae3cc2-9562-4097-a77b-26cef37712ce

New Link Diagrama UML Web (LudiChart):
https://lucid.app/lucidchart/56302772-c1ef-45ff-a1a4-f8c70b9718c8/edit?invitationId=inv_55ffc0eb-3c06-431c-9e79-2e19bccccc08

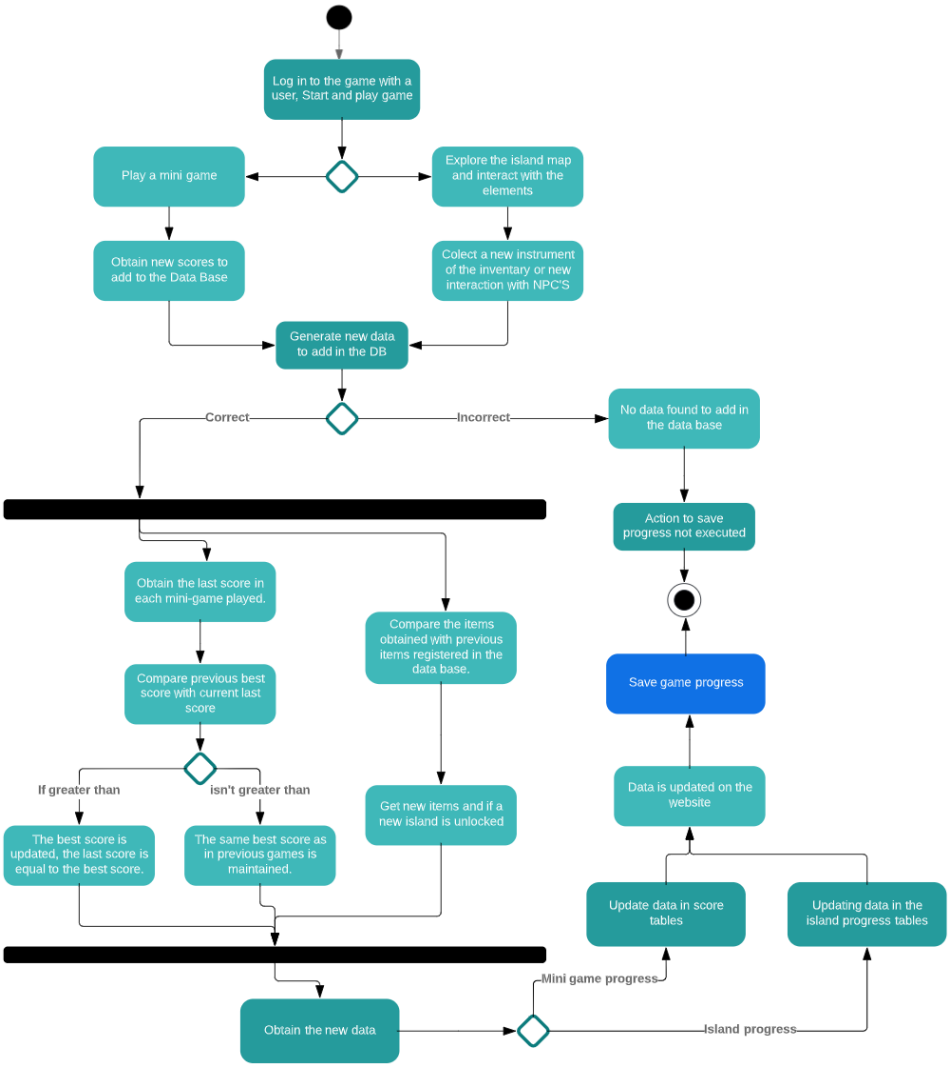
- Nota: modificar el diagrama de Web para incluir la interacción con el API
- Mostrar estadísticas en relación con el API-html

Activity Diagrams:

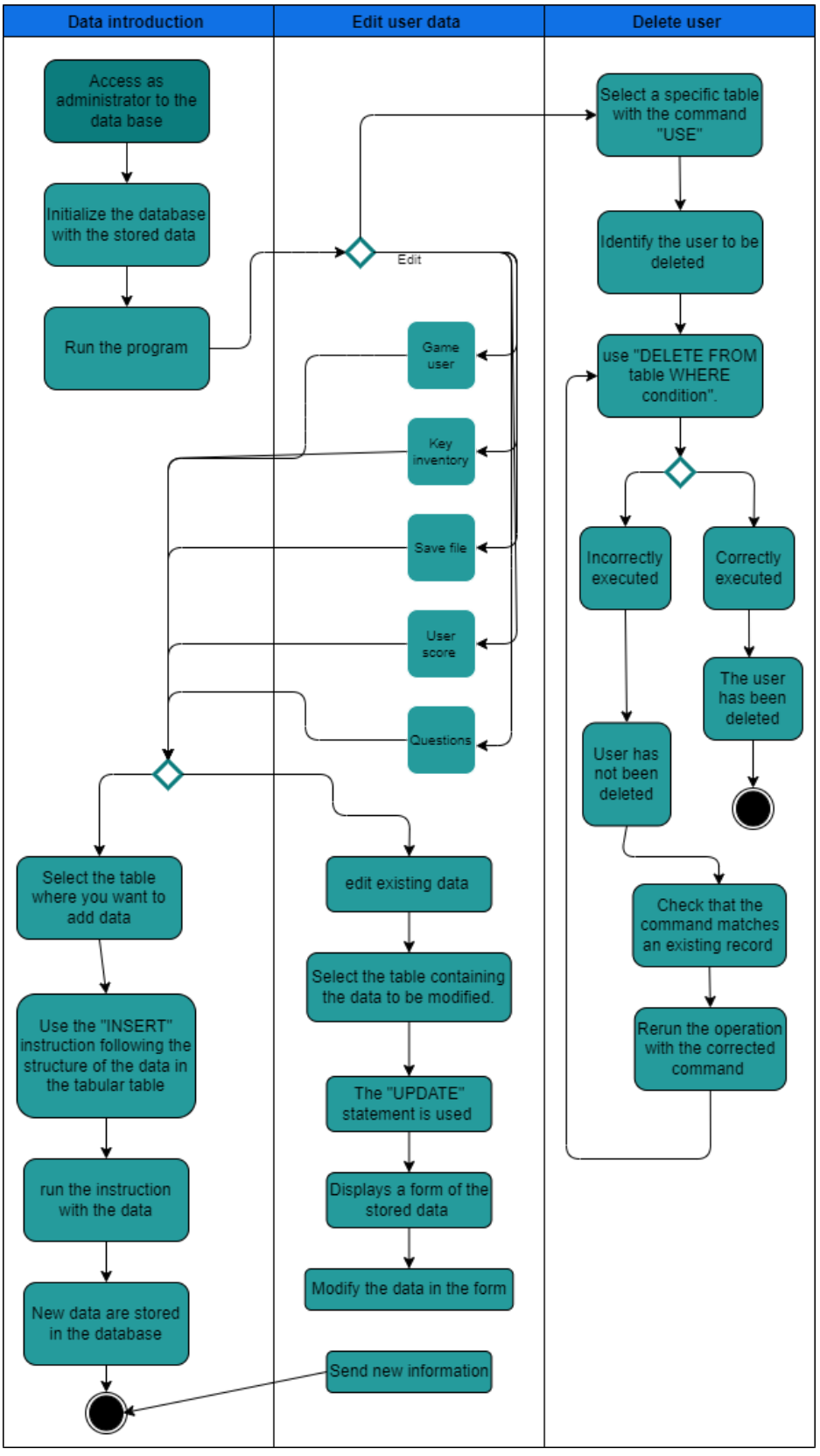
- Login and User registration (Diagram link)



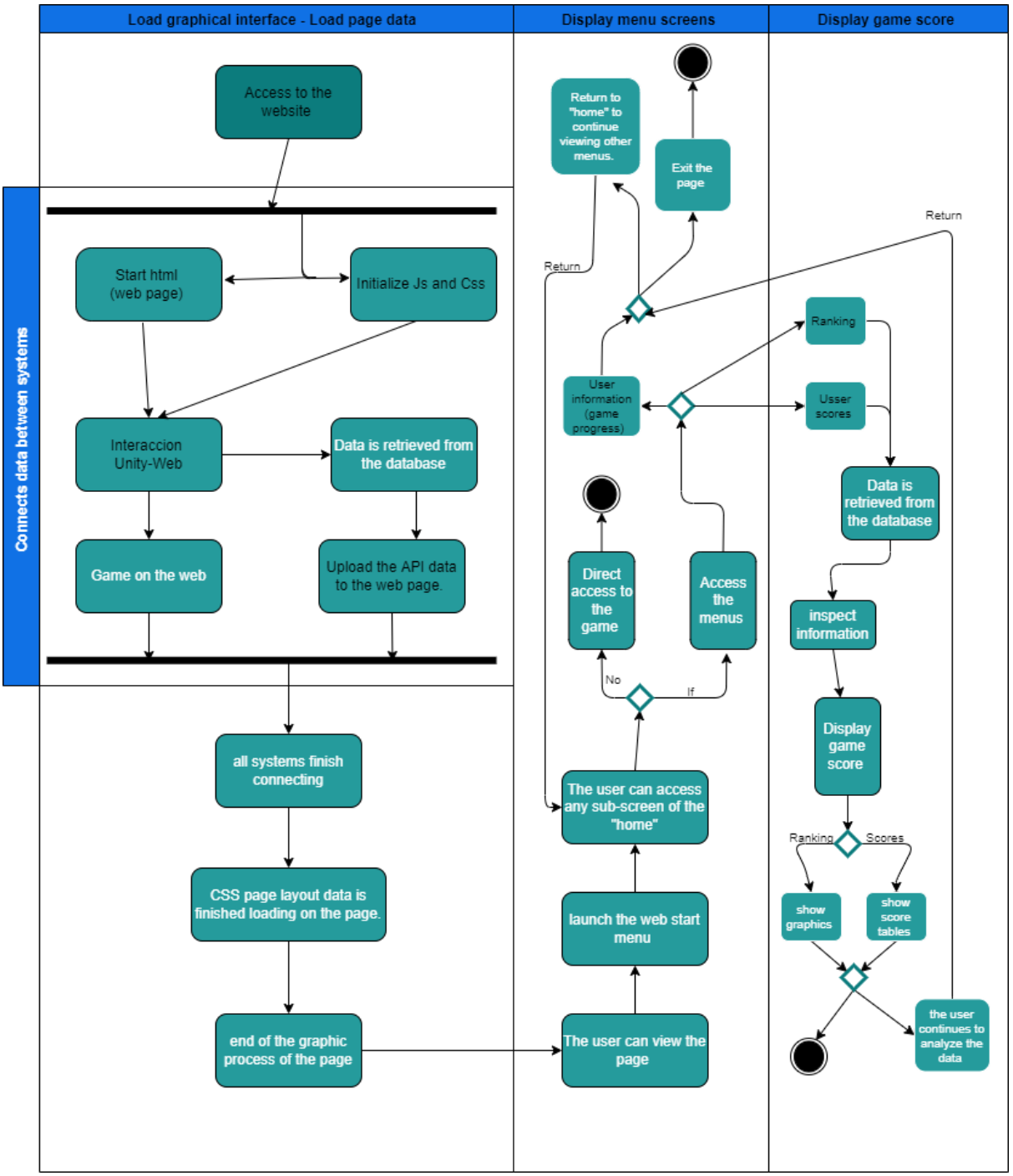
- [Save game progress (Diagram link)]



- Data introduction/Edit user data/delete user diagram



- Data introduction/Edit user data/delete user diagram



- Inspect user information