**CPSC 304:** Milestone 3

*Hunt: Showdown*

Project Check-In

**Victor Shen** 73048563 e1w5d

**Ranyi (Jake) Tao** 27268507 s1w7b

**Yun (Brina) Li** 16840829 t9k4s

## 

**Responsibility Distribution:**

* **Front-End + End-to-End:** Brina
* **Back-End + End-to-End:** Victor, Jake

**Timeline:** Completion Date **-** November 25, 2022

| Task | Task Description | Person and Date (dd/mm/year) |
| --- | --- | --- |
| Landing Page | Must have a navigation menu built on the left hand side with database manipulation navigation options in the middle of the screen; will utilize buttons | Brina  9/11/2022 |
| Buttons | Add buttons for all intended functionalities   * Includes navigation buttons * Includes execution buttons | Brina  9/11/2022 |
| Implement Buttons | Implement page navigation functionality and execution functionality   * Execution functionalities will tie into query functionalities detailed further below | Jake  11/11/2022 |
| Linking Button Functionality with GUI | * Connect the front-end GUI with the back-end button functionality | Brina + Jake  13/11/2022 |
| GUI for Drop-Downs | Should include all options that users can choose   * Users should be able to click on the drop-down menu and make selections | Brina  9/11/2022 |
| Drop-Downs | Implement drop-down functionality   * Should accurately reflect available options/attributes given the drop-down * User selections should dynamically affect code behavior as intended | Victor  11/11/2022 |
| Linking Drop-Down Functionality with GUI | * Connect the front-end GUI with the back-end drop-down functionality | Brina + Victor  13/11/2022 |
| GUI for Insertion | Create GUI elements for the insertion functionality   * Users should be able to input text to set values for various table attributes * GUI should clearly indicate all attributes of the relevant table into which we insert | Brina  11/11/2022 |
| Implement Insertion Functionality | Implement insertion functionality for the database   * Insertion should ideally be possible for all tables, but focus will be on the ‘Trait’ table * Insertion input should adhere to the SQL table and schema, but using text boxes for user input - no SQL   + Error if input is invalid * Return some indicative success result if insertion is successful | Victor  14/11/2022 |
| Linking Insertion Functionality with GUI | Connect the front-end GUI with the back-end insertion functionality; must be dynamic | Brina + Victor  16/11/2022 |
| GUI for Deletion | Create GUI elements for the deletion functionality   * Users should be able to specify an input for the deletion operation using textboxes | Brina  11/11/2022 |
| Implement Deletion Functionality | Implement deletion functionality for the database   * Deletion should ideally be possible for all tables, but focus on allowing a Cascade-On-Delete situation involving the ‘Consumable’ table and a user-input name * Deletion input should adhere to the SQL table and schema, but using dropdowns and text boxes for user input - no SQL   + Error if input is invalid * Return some indicative success result if deletion is successful | Victor  14/11/2022 |
| Linking Deletion Functionality with GUI | Connect the front-end GUI with the back-end deletion functionality | Brina + Victor  16/11/2022 |
| GUI for Update | Create GUI elements for the update functionality   * Users should be able to specify which data they would like to update via drop-downs and text boxes | Brina  11/11/2022 |
| Implement Update Functionality | Implement update functionality for the database   * Ideally should be able to update for any table, but focus will be on hard-coding for ‘Tool’ table where the user selects a tool whose description is to be updated. * Adjustments should be made to inserted tables when relevant window is reloaded * Must allow for dynamic inputs regarding which attribute is updated and which tuples should be updated | Jake  14/11/2022 |
| Linking Update Functionality with GUI | Connect the front-end GUI with the back-end update functionality | Brina + Jake  16/11/2022 |
| GUI for Selection | Create GUI elements for the selection interface   * Allow users to select for tools that cost at most some amount of dollars that the they specify via the GUI * Display a table based on the user’s selection request if successful | Brina  16/11/2022 |
| Implement Selection Functionality | Implement selection functionality for the database   * Given a specific attribute and value, the function should be able to select what the user is looking for from the database   + Return a table based on the user’s selection request   + Return an empty table if no matching data is found   + Focus on allowing users to select based on user input tool cost * Should adhere to SQL tables/schema   + Error if input is invalid | Victor  18/11/2022 |
| Linking Selection Functionality with GUI | Connect the front-end GUI with the back-end selection functionality | Brina + Victor  21/11/2022 |
| GUI for Projection | Create GUI elements for the projection interface   * Will be hardcoded for users most likely * Display a table based on the user’s projection request if successful | Brina  16/11/2022 |
| Implement Projection Functionality | Implement projection functionality for the database   * Given a specific attribute, the function should be able to project what the user is looking for from the database   + Return all columns specified by the user in a table   + Focus will be on allowing users to find name, cost, melee damage and heavy melee damage of tools   + Focus will be on allowing users to find names and levels of hunters * Should adhere to the SQL table/schema   + Error if input is invalid | Victor  18/11/2022 |
| Linking Projection Functionality with GUI | Connect the front-end GUI with the back-end projection functionality | Brina + Victor  21/11/2022 |
| GUI for Join | Create GUI elements for the join functionality   * Focus on hard-coding for joining ‘Hunter’ and ‘Firearm’ functionality in GUI * Must provide selection interface that allows for two where clauses * Display resulting table if successful | Brina  16/11/2022 |
| Implement Join Functionality | Implement join functionality for the database   * Ideally, provided two or more table names as an input, must be able to join all individual tables into a single table, but focus will be on hard-coding for a join between ‘Hunter’ and ‘Firearm’ * Joins will default to being a natural join * Should be able to select on the joined table provided valid user input * Should be able to return a joined and potentially selected table | Jake  18/11/2022 |
| Linking Join Functionality with GUI | Connect the front-end GUI with the back-end join functionality | Brina + Jake  21/11/2022 |
| GUI for ‘Aggregation with GROUP BY’ | Create a front-end GUI for the ‘aggregation with GROUP BY’ functionality   * Will display a hard-coded query * Will provide an execute button * Will display a result table post execution | Brina  18/11/2022 |
| Implement ‘Aggregation with GROUP BY’ Functionality | Implement ‘Aggregation with GROUP BY’ functionality within the database   * Depending on the query, should allow for min, max, average, or count operations * Focus will be on hardcoding for the prompt: “count the number of unique traits available at every price point” * Result must correctly group values * Input must adhere to SQL tables/schema   + Error if input is invalid | Jake  21/11/2022 |
| Linking ‘Aggregation with GROUP BY’ functionality with GUI | Connect the front-end GUI with the back-end ‘Aggregation with GROUP BY’ functionality | Brina + Jake  23/11/2022 |
| GUI for ‘Aggregation with HAVING’ | Create GUI elements for the ‘Aggregation with HAVING’ interface   * Will display a hard-coded query * Will provide an execute button * Will display a result table post execution | Brina  18/11/2022 |
| Implement ‘Aggregation with HAVING’ Functionality | Implement ‘Aggregation with HAVING’ functionality within the database   * Depending on the query, should allow for users to find all tuples that match some criteria involving aggregations * Hardcoded for prompt: “find the names of all firearms that cost more than the average cost of a firearm” * Input must adhere to SQL tables/schema   + Error if input is invalid | Jake  21/11/2022 |
| Linking ‘Aggregation with HAVING’ functionality with GUI | Connect the front-end GUI with the back-end ‘Aggregation with HAVING’ functionality | Brina + Jake  23/11/2022 |
| GUI for ‘Nested Aggregation with GROUP BY’ | Create GUI elements for the ‘Nested Aggregation with GROUP BY’ interface   * Will display a hard-coded query * Will provide an execute button * Will display a result table post execution | Brina  18/11/2022 |
| Implement ‘Nested Aggregation with GROUP BY’ functionality | Implement ‘Nested Aggregation with GROUP BY’ functionality within the database   * Hardcoded for prompt: “find the average number of hunters that hunt each monster” * Input must adhere to SQL tables/schema   + Error if input is invalid | Victor + Jake  21/11/2022 |
| Linking ‘Nested Aggregation with GROUP BY’ functionality with GUI | Connect the front-end GUI with the back-end ‘Nested Aggregation with GROUP BY’ functionality | Brina  + Victor + Jake  23/11/2022 |
| GUI for Division | Create GUI elements for the division interface   * Will display a hard-coded query * Will provide an execute button * Will display a result table post execution | Brina  18/11/2022 |
| Implement Division Functionality | Implement division functionality for the database   * Hardcoded for prompt: “find the names of all hunters who carry both a knife and throwing knives” * Input must adhere to SQL tables/schema   + Error if input is invalid | Victor  21/11/2022 |
| Linking Division Functionality with GUI | Connect the front-end GUI with the back-end division functionality | Brina + Victor  23/11/2022 |

**Getting Started:**

* **Front-End:** Setting up all the pages, buttons, textboxes, etc. Initial focus will be on ensuring all graphical elements are present; functionality will be linked in at later dates.
* **Back-End:** Initializing the database and providing initial data. Initial focus will be on implementing basic hard-coded capabilities for insertion, deletion, and updates. Expansion beyond hard-coded capabilities will be extended as necessary according to the timeline, and further optional expansions will be performed if time allows.

**Challenges:**

* Learning how to properly manage our database!
  + Need to research via provided class files and tutorial materials on the topic.
* Accepting dynamic queries; we will need to allow for user-input to modify back-end functionality within our code.
  + Will hardcode many elements as necessary to reduce complexity.
  + Will write generalizable code that can be reused for multiple dynamic elements.
  + Need to further research end-to-end integration from user to database.
* Ensuring all GUI elements play nicely with our back-end implementation and allow the user to do what we envision them being able to do (but no more).
  + Will limit the number of dynamic GUI elements available to users.
  + Will hard code difficult queries into simple buttons.
* Implementing drop-down functionality; this is our most dynamic front-end GUI element. Care will need to be taken to ensure it plays nicely with our back-end functionality.
  + Need to research how to do this efficiently and properly.
  + May need to build generic drop-down interfaces that handle flags.

**TODO:**

* Everything!
* We will focus on getting things done in an order that adheres to our timeline.

**UI Design:**

* Images are provided as samples of how we envision our GUI to look for various pages.
* Not all possible pages are included as some are similar, and inclusion of multiple such pages would be redundant. However, at least one of each type of page we wish to include has been explicitly drawn out.
* In the name of readability, not all dynamic elements, such as pop-up result tables, are included in the sample images.
* GUI is potentially subject to change depending on the difficulties we run into; but ideally will remain fairly faithful to the attached sample drawings.
* We attach all our sample images starting on the next page; please continue downwards.