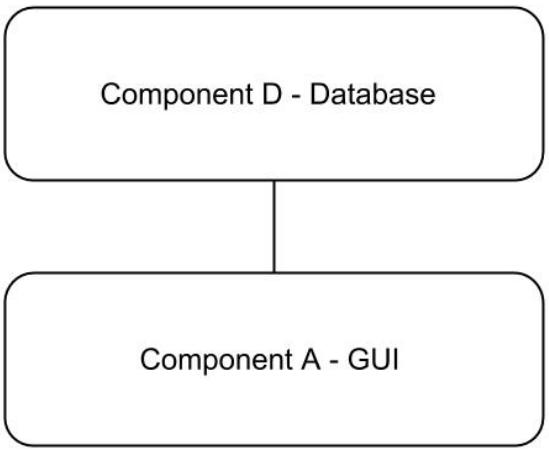
Regression Test Cases

Step 3: **Test database and GUI**



The main component is the database (Component D) which has test cases for login and registration for a new user, and access and store scores for existing users. The test cases for Component D verify that the database could store user information and scores correctly. The test cases also verify that the code to access the database performs correctly both when the user registers an account and logins to the game.

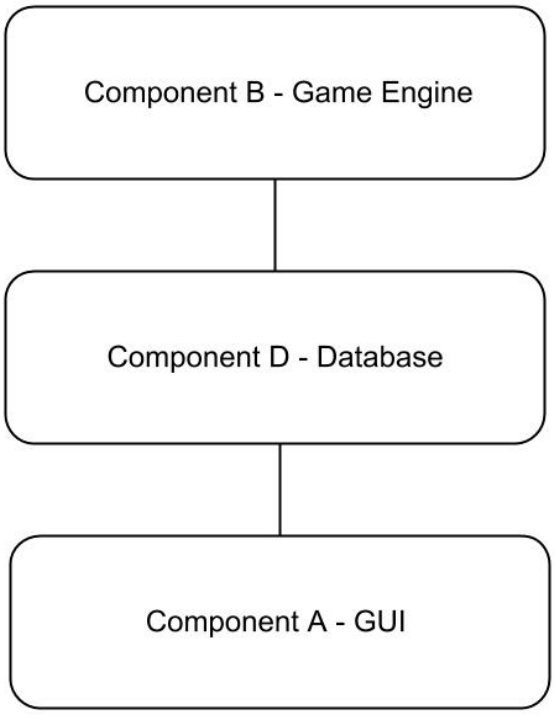
Component A is a GUI component which depends on Component D. The test cases for Component A verify that operations like signing up, signing in and storing scores are performed correctly and test boundary cases. In addition, the test cases make sure that no users can perform malicious attacks to the database through SQL injection. In conclusion, in regression testing, once defects in component A are found, we rerun the test cases that revealed the defects in debugging mode. If the defect is on GUI side, we rerun GUI test cases; if the defect is on database side, we rerun all the database and GUI test cases because they are logically affected by changes. We also also rerun a random subset of all test cases to make sure no unexpected defects are generated by the fixes.

Defect / Reference Number	Description	Severity	How Corrected
1/1	Originally sqlite-jdbc-3.7.2.jar was not added to the	1	Add sqlite-jdbc-3.7.2.jar to a folder in the project folder, and add the relative path to

	classpath. Members cannot run the database code correctly after pulling it from github. After fixing the defect by adding the classpath for sqlite-jdbc-3.7.2.jar, the code still cannot run correctly on other members' computer. This was because the directory added was a full path on a particular person's computer.		the jar file to the project build path.
2/2	Scores cannot be saved correctly and program crashes every time it attempts to save a score. From the error tracing information, a defect in database code was found: there was a typo about table name in the SQL statement.	1	Change the table name in the sql statement string.
3/1	There was an error -- "SQLite cannot be accessed" every time a user attempts to signs up. The reason was that SQLite only allows a single database connection at a time.	1	Rewrite the program so that connections are closed whenever operations are done.
4/6	The password can be accessed if a user try to access the database file from command line by typing "sqlite3 dungeon.db". Error resides in the algorithm to save password.	1	Save the hashed password.
5/4	A user can register with	2	Check for the first character

	name or password that starts with numbers, has spaces and special characters. This is because the program did not check for validity of username in the GUI part.		of the name as well as other characters in the name. Check for the first character of the password as well as other characters in the name.
6/1	A user can select multiple difficulty levels. The error is in GUI part.	2	Add ButtonGroup so that only one radio button can be selected at a time.
7/1	After the user clicks on “switch user” button, a “signup” page pops up.	3	Change the default tab so that signin page will be displayed.
8/1	The length of buttons on the menu page changes whenever length of username changes. This is because of the label “Welcome back ‘username’!” aligns with buttons.	3	Fix the length of label to 40. Pad spaces to the label if length is not reached.
9/1	The play button is clickable even when user is not signed in.	3	Disable the button when user is not signed in. Enable it when user logs in or signs up.

Step 4: **Test game engine, database and GUI**



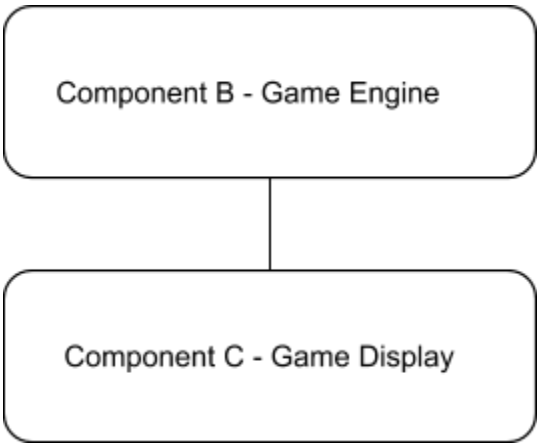
The main component is the game engine (Component B), which has test cases for scores. The database (Component D) depends on the game engine for determining the scores, while the GUI (Component A) depends on the database for recording the scores. The test cases for the game engine should verify that the scores are calculated correctly before storing them into the database. The test cases for the database should verify that the newest scores are stored at any given point of time. The test cases for the GUI should verify that the scores are displayed correctly, especially the personal score list and overall score list.

The regression test case should verify if the recording of the scores is carried out successfully when the user restarts a particular level, exits the game in middle of a level, and a blackout or sudden termination of the game program occurs. At any given situation, the database has to maintain the newest scores.

Defect / Reference Number	Description	Severity	How Corrected
1/6	Initially current score are saved to database	1	Save the score only at the end of the game or in catch

	whenever a user completes a level. This makes sure that even if the program crashes, the scores are appropriately saved. This introduces the new error: too many intermediate but not final scores are saved.		statements.
2/1	When the character model died in a particular level, the total score is the same as if it did not die.	1	Do not update the score at the end of a level. Call the save score function on user with the current score immediately after death.
3/2	When the user completes the last level, nothing happens.	1	Checks if current level is the last level at the end of each game, and save score to database if it is.
4/3	After the character model opens doors and treasures, the score remain unchanged.	1	Check for position of model with treasure and doors, and update scores accordingly.
5/4	The final score is the same for all difficulty levels with same steps.	2	Multiply score with a different number for different difficulty levels at the end of the game.
6/5	The top scores were originally ordered from lowest to highest in scoreboard. The error must reside in the select statement.	2	Added “ORDER BY SCORE DESC;” at the end of the select statement.

Step 5: **Test game engine and game display**



The main component is the game engine (Component B) which has test cases for collision detections and character movement. Game display component (Component C) depends on Component B. The test cases for Component B should verify that collision detection and character movement is working like intended. The regression test cases should verify that if you see a character model moving through obstacles that it is intended to move through, then in Component B the game engine (Component B) is functioning properly. Component B should also verify when the character model (Game Display) interacts with a treasure, it is reacting properly. Finally Component B is also responsible for verifying that when a user selects a difficulty the enemy and player speed is adjusted accordingly.

Similarly if in game display component (Component C) a character is ignoring collision detection or not moving properly. Then something is not functioning properly in Component B.

Defect / Reference Number	Description	Severity	How Corrected
1/1	After fixing the defect that the character model moving out of the game board, the character model is not able to go back to previous map levels.	1	Checked boundary coordinates of the game board in which the model will step on. If it is (0,0), which are coordinates of the top left corner, enable the collision. This will allow

			the model to walk past the boundary of the game board, and trigger the function directing to the previous map levels.
2/2	After fixing the character model is not able to pick up the treasure, another defect is found in which the model has been granted with the capability to “pick up” enemy units.	1	Assigned the enemy units with a different variable for checking collision status. If the model collides with the enemy units, the combat system will be triggered rather than letting the enemy units to be picked up by the model.
3/5	After fixing the defect in which the character model walks past the walls, another defect is found in which the model is not able to walk past the doors.	1	Assigned the doors with a different variable for checking collision status. If the model collides with the doors, it should “pick up” the doors rather than being stopped as if the doors are the walls.
4/7	After fixing the defect in which the character is unable to walk through a path even though visually it looks like you can, another defect is found in which the model walks past the walls.	1	Changed the fixing method of the previous defect. Removed the “invisible” wall rather than changing the variable for collision checking of the walls. (The previous defect is due to not adding some wall objects into the panel during level design. Rather than removing the wall objects, they were set in a rush such that the model can walk past them. However, the wall objects share a global variable for collision checking, making all the walls permeable to the model.)