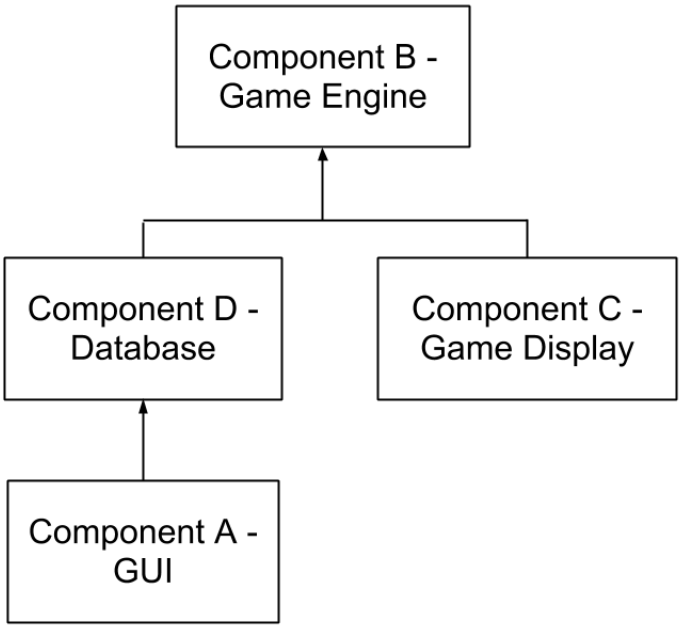
All components



Component A : GUI

Input: Mouse click

Output: scores, main menu, menu bar, registration page, login page

Dependent components: Database

Component B : Game Display

Input: Direction keystroke

Output: Map, character model, enemies, doors, treasure

Dependent components: Game Engine

Component C : Game Engine

Input: Direction keystroke

Output: Scores, effects of collision, movement of character model and enemies

Dependent components: Game Display

Component D : Database

Input: Username, password

Output: Status string, scores, account information
Dependent components: GUI, Game Engine

Bottom-up incremental testing

We follow bottom-up incremental testing due to several reasons.

Firstly, at the moment we carry out the incremental testing, we have fully developed all modules of our program. A requirement for this approach is completion of the program, and we have done so.

Second, the bottom-up approach prevents overlap of design and testing. As the approach can only be used when the program is fully developed, we will not make unwise decisions for omitting desirable improvements of upper levels.

Third, it is more convenient to produce driver modules than stub modules. By starting with terminal modules, we proceed from modules of lower levels to those of upper levels. Depending on total number of modules, this requires production of corresponding number of driver modules. Hence, a more convenient module production will be preferred.

Lastly, the bottom-up approach is favored in industry, so it is better for us to follow the most accepted convention.

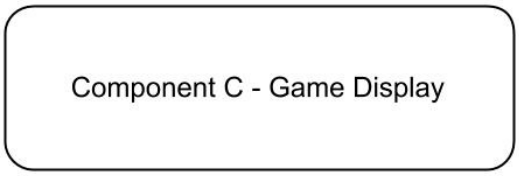
Incremental Test Cases

Step 1: **Test GUI**
Step 2: **Test game display**
Step 3: **Test database and GUI**
Step 4: **Test game engine, database and GUI**
Step 5: **Test game engine and game display**

Step 1: **Test GUI**

Defect Number	Description	Severity	How Corrected
1	Buttons on menu bar does not have correct responses.	1	Add action listeners for the menu bar buttons.
2	Level difficulty radio buttons are not mutually exclusive. Users can select more than one button.	2	Check the conditions of the enability of the radio button using if-else statement. For example, if "easy" level radio button is selected, the "medium" and "hard button" should be in setSelected(false).
3	Nothing happens when the switch user button is pressed. The user cannot log out from the game.	2	Correct listener for the switch user button to correctly bring up the logi screen.
4	The restart button does not work.	2	Add listeners for the restart button.
5	The switch user button does not work correctly.	2	Add listeners for the switch user button.
6	When you click restart another menu pops up with a choice of “open”	2	Removed the second menu as it served no functonality.

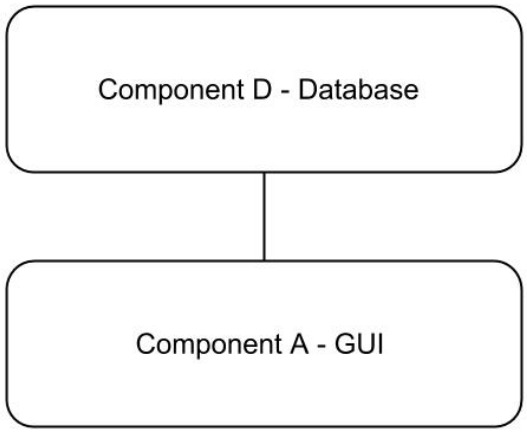
Step 2: **Test game display**



Defect Number	Description	Severity	How Corrected
1	The display refreshes in a slow	3	Set the timer to be shorter in

	manner.		duration. A shorter timer increases the display speed.
2	When the user completes the las level, nothing happens.	1	Checked whether the current level completed is the last level. so, link it to ending screen.
3	Nothing happens when the character model reaches exit of a level.	1	Linked the exit with entrance of the next level by implementing a listener that triggers when position of the model is the same as the exit's.
4	The doors do not disappear after the character model walks past them.	1	Tracked position of the character model during the game. If the position of the model is the same as the doors', set visibility of the doors to be invisible.
5	Players move faster down than up	2	Corrected values for move distance on down arrow
6	Character model seems to get stuck on corners when it shouldn	2	Corrected the size of the character picture to match the size allocated for the character model
7	When the user resized a window, the game does not resize properly, so gray is showed.	1	Made it so a user can't resize it and instead the size is locked.

Step 3: **Test database and GUI**

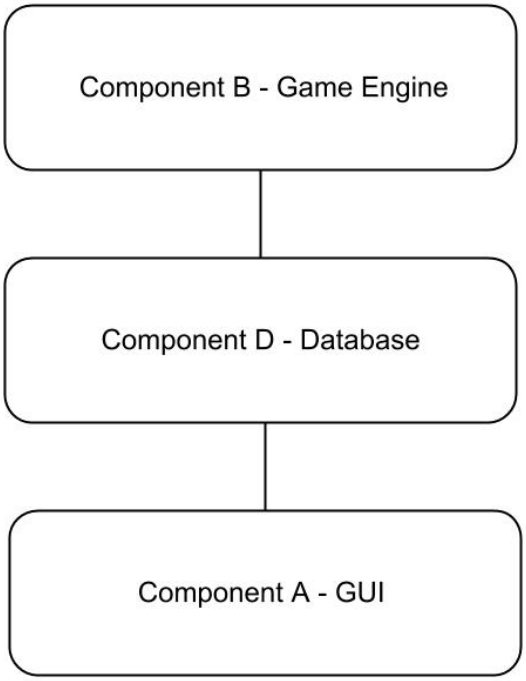


Defect	Description	Severity	How Corrected

Number			
1	The database does not record account information when an account is created.	1	Checked the sql insert command and corrected it.
2	The database does not record scores correctly.	1	Checked the sql insert command and corrected it.
3	The database allows the user to register with an existing account.	1	Checked whether the registered username and password are already exist in the database. If there is an entry in the database matches the credentials entered, a pop-up message will appear to notify the user.
4	Users can sign up for an account using password with invalid length.	1	Checked to make sure that length of the password is between 4 and 15 before proceeding. If the length of the password is out of the range, display a warning message and prompt the user to re enter his or her username and password.
5	The user can login by merely entering username.	1	Added a check to ensure password correct.
6	User is allowed to login if password is incorrect.	1	Added password check. Based on the database, compare the account username with the inputted password.
7	The user is able to sign up for a username even when the username is already in the database.	1	Before adding the username and password to the database, checked if the username already exists in the database first. If the username exists, the sign up process fail. The user is showed with a message indicating the issue, and prompted to enter a different username.

8	Did not check for special characters in username.	2	Checked for special characters in username. Ensure the username does not begin with a special character.

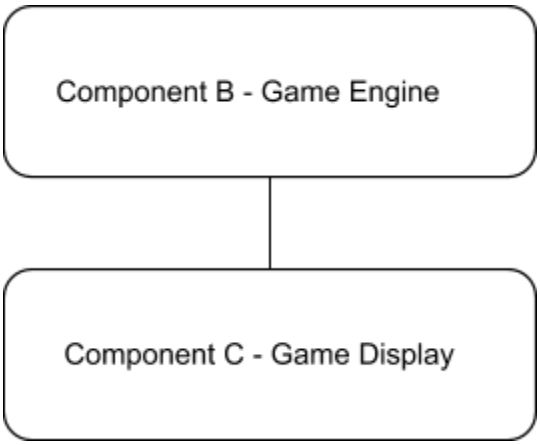
Step 4: **Test game engine, database and GUI**



Defect Number	Description	Severity	How Corrected
1	The total score changes when the character model dies in a particular level.	1	Saved net score change of a particular level into a temporary variable. If the user completes the level, the net score change is added into the total score. Otherwise, it is reset to zero without adding it to the total score.

2	The score for not opening any door is the same as that for opening doors.	1	Checked whether position of the character model is the same as the doors'. If the positions are the same, trigger a function that reduces a particular amount of score immediately from the current total score.
3	The score does not increase when picking up the treasure.	1	Checked whether position of the character model is the same as the treasure's. If the positions are the same, trigger a function that increases a particular amount of score from the current total score.
4	The score is the same for all difficulty levels.	1	Assigned each difficulty level with a different score multiplier.
5	The overall score list is not updated correctly.	1	Sorted a particular score with the current top five overall score, and assigned the top five scores after sorting into the list.
6	The newest score is not recorded after a blackout or sudden termination of the game program.	1	Save the newest score immediately after completion of each level so that the database has a copy of the newest score.

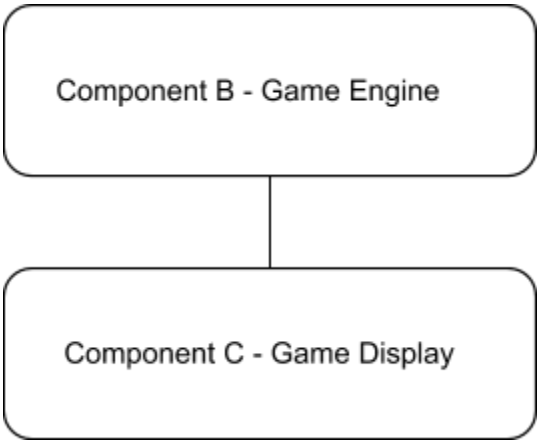
Step 5: **Test game engine and game display**



Defect Number	Description	Severity	How Corrected
1	The character model moves out of the game board.	1	Added boundary check so the character model cannot move through it.
2	The character model is not able to pick up the treasure. The model walks through the treasure.	1	Added code to replace the treasure model with a floor mode after the character model walks through it.
3	The character model walks past the enemy units without being teleported back to entrance.	1	Added tests for enemy collision to the engine.
4	Movement speed of the character model and the enemy units is the same regardless of difficulty.	1	Increased or decreased the speed of model character while keeping differences between different difficulty levels.
5	The character model walks past the walls.	1	Added checks for walls whenever location of the character model changes. If the positions of the walls and the enemy model are the same, set the wall to be impermeable for the enemy models.
6	Character speeds up when walkin up	1	Calculation was off when doing speed going up. Fixed incorrect values
7	Character is unable to walk through a path even though	1	Lowered the height and width of the character model to properly

	visually it looks like you can		reflect it.
8	Character would start at wrong location	1	Changed the starting coordinates to properly reflect visuals.

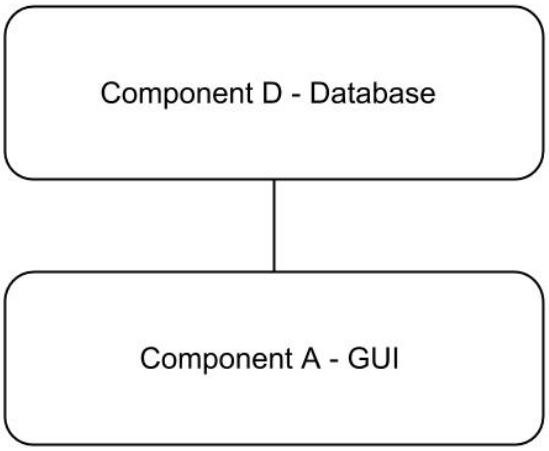
Regression Test Cases



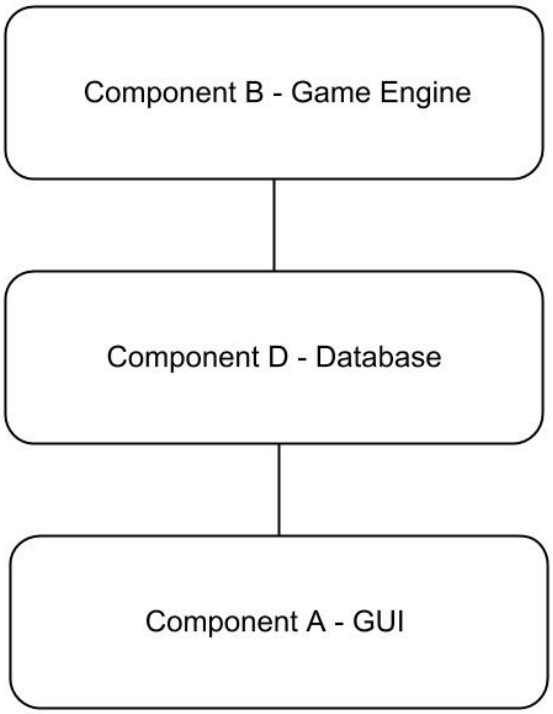
The main component is the game engine (Component B) which has test cases for collision detections and character movement. Game display component (Component C) depends on Component B. The test cases for Component B should verify that collision detection and character movement is working like intended. 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.



The main component is the database (Component D) which has test cases for login information and registration for the new user. The GUI (Component A) depends on the GUI. The test cases for Component A should verify that the database has stored user information successfully when the user creates a new account. The test cases should also verify whether the database performs correctly both when the user registers and logins. For both the operations, they should verify whether the new account has been created or not. The regression test case should verify if login without creating the new account, and register with an existing account work correctly in both the components. It should also verify that the scores are stored correctly and do not change.



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.