Test Cases

Black-Box Test Cases

These test cases are defined based on the available documentation and execution of the program. The code was not inspected.

Test Class	Test Case ID	Precondition	Test Steps	Expectation	Observations
1	TC-01	User should have app with login	1. Open the app 2. Verify "Welcome to Learning Buddy" screen	User sees: - "Welcome to Learning Buddy" as header - "Read General Information" button - Some context - "Start Quiz" button - "Exit" button	pass
2	TC-02	User should have app with login	1. Open the app 2. Click on "Read General Information" button	User sees some context in text area in non-editable mode	pass
3	TC-03	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Select Category" screen	pass
4	TC-04	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Select Category" screen	pass
5	TC-05	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Select Category" screen	pass
6	TC-06	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees: - Selected subject name as header with information icon -	pass

				Subject context in text area - "Ok" and "Close" buttons	
7	TC-07	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Category Information" screen as closed and selects the quiz difficulty	pass
8	TC-08	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Category Information" screen as closed	pass
9	TC-09	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees: - "Select Quiz Difficulty" as header with question mark icon - "Choose your difficulty" field along with drop down - "Ok" button	pass
10	TC-10	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees drop down with levels and selected data in field	pass
11	TC-11	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Quiz Questions" screen	pass
12	TC-12	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees: - "Quiz Questions" as header - Questions with question mark icon - 4 options buttons - "Quit" button	pass
13	TC-13	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User selects the option button for given question and sees	pass

				feedback screen whether selected option for question is correct or incorrect	
14	TC-14	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User selects the correct option button for given question and sees correct screen dialog box	pass
15	TC-15	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User selects "X" icon and sees dialog box as closed and next question displays. If it is last question then sees "Quiz Over" screen	pass
16	TC-16	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Ok" button in correct screen dialog box and next question displays. If it is last question then sees "Quiz Over" screen	pass
17	TC-17	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees quiz as quit	pass
18	TC-18	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees: - "Quiz Over" as header with "X" icon - Some text with information icon - "Ok" as button	pass
19	TC-19	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Quiz Over" screen as closed	pass

20	TC-20	User should have app with login	1. Open the app 2. Click on "Start Quiz" button	User sees "Quiz Over" screen as closed and exits from application	pass
21	TC-21	User should have app with login	1. Open the app 2. Click on "Read General Information" button	User exits from application	pass
22	TC-22	User should have app with login	1. Open the app 2. Click on "Read General Information" button	User selects options for questions, and if lives are completed, exits the quiz	pass

White-Box Test Cases

These additional test cases were defined during inspection of the code.

#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation
1	Test getters and setters for Category class	Create a Category object with initial values	1. Verify initial values using getters. 2. Use setters to change values. 3. Verify new values using getters.	Getters return initial values. Setters update the values correctly. Getters return updated values.	Pass
2	Test empty string validation during initialization	None	Attempt to initialize a Category object with empty strings.	IllegalArgumentException is thrown during initialization.	Pass
3	Test empty string validation after initialization	Create a Category object with initial values	1. Attempt to set empty strings using setters. 2. Verify that IllegalArgumentException is thrown. 3. Verify that getters return initial values.	IllegalArgumentException is thrown for empty strings. Getters return initial values.	Pass

4	Test null value validation during initialization	None	Attempt to initialize a Category object with null values.	IllegalArgumentException is thrown during initialization.	Pass
5	Test null value validation after initialization	Create a Category object with initial values	1. Attempt to set null values using setters. 2. Verify that IllegalArgumentException is thrown. 3. Verify that getters return initial values.	IllegalArgumentException is thrown for null values. Getters return initial values.	Pass
6	Test setting name to empty string after setting valid value	Create a Category object with initial valid name	1. Set a valid name using the setter. 2. Attempt to set the name to an empty string. 3. Verify that IllegalArgumentException is thrown. 4. Verify that the name remains the valid value set previously.	IllegalArgumentException is thrown for empty string. The name remains as the valid value set previously.	Pass
7	Test setting study information to null after setting valid value	Create a Category object with initial valid values	1. Set valid study information using the setter. 2. Attempt to set study information to null. 3. Verify that IllegalArgumentException is thrown. 4. Verify that the study information remains the valid value set previously.	IllegalArgumentException is thrown for null value. The study information remains as the valid value set previously.	Pass
8	Test setting whitespace strings using setters	Create a Category object with initial values	1. Attempt to set whitespace strings using setters. 2. Verify that IllegalArgumentException is thrown. 3. Verify that getters return initial values.	IllegalArgumentException is thrown for whitespace strings. Getters return initial values.	Fail
9	Test setting study information to empty string after valid value	Create a Category object with initial valid values	1. Set valid study information using the setter. 2. Attempt to set study information to empty string. 3. Verify that IllegalArgumentException is thrown. 4. Verify that the study information remains the valid value set previously.	IllegalArgumentException is thrown for empty string. The study information remains as the valid value set previously.	Pass
10	Test setting name to null after setting valid value	Create a Category object with initial valid values	1. Set a valid name using the setter. 2. Attempt to set the name to null. 3. Verify that IllegalArgumentException is thrown. 4. Verify that the name remains the valid value set previously.	IllegalArgumentException is thrown for null value. The name remains as the valid value set previously.	Pass

11	Test setting name to whitespace string	Create a Category object with initial values	1. Attempt to set the name to a whitespace string. 2. Verify that IllegalArgumentException is thrown. 3. Verify that the name remains the initial value.	IllegalArgumentException is thrown for whitespace string. The name remains as the initial value.	Pass
1	Test reading general information	Ensure GeneralInformation.txt exists with content	1. Click "Read General Information". 2. Wait for the file content to be loaded. 3. Check the content of the TextArea.	The content of TextArea should not be empty.	Pass
2	Test start quiz button functionality	Mock categories in DataStoreSql	1. Mock categories in DataStoreSql. 2. Click "Start Quiz". 3. Wait for the dialog to appear.	The category selection dialog should be displayed.	Pass
3	Test showing category information	Create a Category object	1. Create a Category object. 2. Call showCategoryInformation(category). 3. Wait for the dialog to appear.	The category information dialog should be displayed.	Pass
4	Test asking a question and answering correctly	Create a Question object	1. Create a Question object. 2. Call askQuestion(question). 3. Wait for the dialog to appear. 4. Click the correct answer.	The user should continue the quiz after answering correctly.	Pass
5	Test asking a question and answering incorrectly	Create a Question object	1. Create a Question object. 2. Call askQuestion(question). 3. Wait for the dialog to appear. 4. Click an incorrect answer.	The user should continue the quiz after answering incorrectly.	Pass
6	Test asking a question and choosing to quit	Create a Question object	1. Create a Question object. 2. Call askQuestion(question). 3. Wait for the dialog to appear. 4. Click "Quit".	The user should quit the quiz.	Pass
7	Set up the database	Create the database schema with tables Categories and Questions	1. Connect to the database. 2. Drop existing Categories and Questions tables if they exist. 3. Create Categories and Questions tables.	The database schema should be set up correctly.	Pass
1	Test database setup and table creation	Ensure the H2 database driver is available and database is clean	1. Clean the database. 2. Create Categories and Questions tables.	The database should be set up with Categories and Questions tables.	Pass

2	Test populating the database with initial data	Ensure the database is set up with Categories and Questions tables	1. Call DataBasePopulator.populateDatabase(). 2. Verify the number of categories. 3. Verify the number of questions for each category.	There should be 5 categories, each with 6 questions.	Pass
3	Test adding a category from a file	Ensure the database is set up with Categories table and a file with category information exists	1. Create a file with category information. 2. Call DataBasePopulator.addCategoryFromFile(categoryName, filePath). 3. Verify the category is added. 4. Delete the file.	The category should be added from the file, and the file should be deleted after verification.	Pass
1	Test adding a category	Ensure the H2 database driver is available and database is set up with Categories table	1. Call DataStoreSql.addCategory(). 2. Query the database to verify the category was added.	The category should be added to the Categories table with the correct name and study information.	Pass
2	Test adding a question	Ensure the database is set up with Categories and Questions tables and a category is added	1. Call DataStoreSql.addCategory(). 2. Retrieve the category ID. 3. Call DataStoreSql.addQuestion(). 4. Query the database to verify the question was added.	The question should be added to the Questions table with the correct category ID, difficulty, question text, answers, and right answer.	Pass
3	Test reading categories	Ensure the database is set up with Categories table and multiple categories are added	1. Call DataStoreSql.addCategory() twice. 2. Call DataStoreSql.readCategories(). 3. Verify the number of categories and their details.	There should be at least two categories, and their names and study information should match the added values.	Pass
4	Test reading questions by category	Ensure the database is set up with Categories and Questions tables and a question is added for a specific category	1. Call DataStoreSql.addCategory(). 2. Retrieve the category ID. 3. Call DataStoreSql.addQuestion(). 4. Call DataStoreSql.readQuestionsbyCategory(). 5. Verify the question details.	There should be at least one question, and its details should match the added values.	Fail
1	Test Question constructor and getters	None	1. Create a Question object with specific values. 2. Verify each value using the getters.	The values retrieved by the getters should match the values provided in the constructor.	Pass

2	Test Question constructor with different difficulty	None	1. Create a Question object with different difficulty. 2. Verify each value using the getters.	The values retrieved by the getters should match the values provided in the constructor.	Pass
3	Test Question constructor with null values	None	1. Create a Question object with null values. 2. Verify that the getters return null.	The getters should return null for each field.	Pass
#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation (“pass” or failure description)