test case for City table creation: Verify that the City table is created with the expected columns and data types.

-- Define expected columns and data types

SET @expected = 'CityID INT, CityName TEXT NOT NULL, Population INT NOT NULL, Region TEXT, Latitude FLOAT, Longitude FLOAT';

-- Verify the actual table structure matches the expected structure

SELECT GROUP\_CONCAT(column\_name, ' ', data\_type) AS actual

FROM information\_schema.columns

WHERE table\_name = 'City'

INTO @actual;

SELECT IF(@expected = @actual, 'PASS', 'FAIL') AS result;

Test case for Page table creation: Verify that the Page table is created with the expected columns and data types, and that the CityID column has a foreign key constraint referencing the City table's CityID column.

-- Define expected columns and data types

SET @expected = 'PageID INT, PageTitle TEXT NOT NULL, PageContent TEXT NOT NULL, PageType TEXT NOT NULL, CityID INT';

-- Verify the actual table structure matches the expected structure

SELECT GROUP\_CONCAT(column\_name, ' ', data\_type) AS actual

FROM information\_schema.columns

WHERE table\_name = 'Page'

INTO @actual;

-- Verify that the CityID column has a foreign key constraint

SELECT COUNT(\*)

FROM information\_schema.table\_constraints

WHERE constraint\_type = 'FOREIGN KEY' AND table\_name = 'Page' AND column\_name = 'CityID'

INTO @has\_fk;

SELECT IF(@expected = @actual AND @has\_fk = 1, 'PASS', 'FAIL') AS result;

down

Test case for AboutUsPage table creation: Verify that the AboutUsPage table is created with the expected columns and data types.

-- Define expected columns and data types

SET @expected = 'AboutUsID INT, AboutUsContent TEXT NOT NULL';

-- Verify the actual table structure matches the expected structure

SELECT GROUP\_CONCAT(column\_name, ' ', data\_type) AS actual

FROM information\_schema.columns

WHERE table\_name = 'AboutUsPage'

INTO @actual;

SELECT IF(@expected = @actual, 'PASS', 'FAIL') AS result;#

test case for HomePage table creation: Verify that the HomePage table is created with the expected columns and data types.

-- Define expected columns and data types

SET @expected = 'AboutUsID INT, AboutUsContent TEXT NOT NULL';

-- Verify the actual table structure matches the expected structure

SELECT GROUP\_CONCAT(column\_name, ' ', data\_type) AS actual

FROM information\_schema.columns

WHERE table\_name = 'AboutUsPage'

INTO @actual;

SELECT IF(@expected = @actual, 'PASS', 'FAIL') AS result;

Test case for InfoPage table creation: Verify that the InfoPage table is created with the expected columns and data types.

-- Define expected columns and data types

SET @expected = 'InfoPageID INT, InfoPageTitle TEXT NOT NULL, InfoPageContent TEXT NOT NULL';

-- Verify the actual table structure matches the expected structure

SELECT GROUP\_CONCAT(column\_name, ' ', data\_type) AS actual

FROM information\_schema.columns

WHERE table\_name = 'InfoPage'

INTO @actual;

SELECT IF(@expected = @actual, 'PASS', 'FAIL') AS result;

Graphical user interface, application

Description automatically generated

a set of unit tests designed to verify the correctness of a database schema and ensure that it meets the expected requirements. The tests are performed by executing SQL queries against the database and comparing the expected results with the actual results obtained from the queries.

The first set of tests are for verifying the table creation for each of the tables (City, Page, AboutUsPage, HomePage, InfoPage). The expected columns and data types are defined for each table, and SQL queries are executed to retrieve the actual columns and data types for each table from the information schema. The results of the queries are then compared to the expected values to determine if the table creation is correct. If the actual results match the expected results, the test passes, otherwise it fails.

The next set of tests are for verifying the integrity constraints for the database schema. In the case of the example provided, the test verifies that the CityID column in the Page table has a foreign key constraint referencing the City table's CityID column. To perform this test, an SQL query is executed to retrieve the number of foreign key constraints on the CityID column in the Page table. If the number of foreign key constraints is equal to one, the test passes, otherwise it fails.

Finally, there are tests for inserting data into the database tables. The purpose of these tests is to verify that the data can be inserted correctly into the tables and that the integrity constraints defined for the schema are enforced. For example, in the test for inserting a new City record, an SQL query is executed to insert a new record into the City table. Then, another SQL query is executed to retrieve the inserted record from the City table. The retrieved record is then compared with the expected record to determine if the insert operation was successful. If the expected and actual records match, the test passes, otherwise it fails.

Overall, these unit tests provide a way to verify the correctness of the database schema and ensure that it meets the expected requirements. They can be automated and executed regularly to catch any potential issues early in the development process.