Test plan - savings app

Requirements:

\* view a balance for a user

\* view the balances for all users

\* add funds for a user

\* remove funds for a user (funds must not go below 0)

\* add a new user with a balance of 0

\* remove a user

\* exit the app

Complete the table to plan how you will test each of the requirements. Each will need one or more tests to ensure that all possible cases have been tested.

You should test with valid users, non-valid users, valid and non-valid data types, etc.

TRY TO FILL THIS TABLE IN FIRST, then run the app, create 10 users with various balances and modify the data column to match the data you have in the app.

REMEMBER - the objective is to get the app to fail (it should do this quite easily if you have tested it fully).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test no.** | **What is being tested?** | **Data (user\_id, amount) used?** | **Expected outcome** | **Actual outcome** | **Notes** |
| 1 | Views all records in the savings table  get\_data(conn) | get\_data(db\_conn) | user\_id balance  --------- ---------  1 44  2 5  3 42  4 32  5 25  6 5  7 26  8 39  9 45  10 0 | user\_id balance  --------- ---------  1 44  2 5  3 42  4 32  5 25  6 5  7 26  8 39  9 45  10 0 | This test works |
| 2 | View the balance for a valid user  get\_user\_data(user\_id, conn) | get\_user\_data(1, db\_conn) | 44 | 44.0 | This test works |
| 3 | View the balance for a non-valid user  get\_user\_data(user\_id, conn) | get\_user\_data(15, db\_conn) | User does not exist | 44.0 | This test works |
| 4 | View the balance with an incorrect data type – one-digit string  get\_user\_data(user\_id, conn) | get\_user\_data(“1”, db\_conn) | Invalid user\_id data type |  | This test doesn’t work as the string data type is accepted. Need to add data type validation |
| 5 | Add new user with id that already exists and correct data type - integer add\_new\_user(user\_id, conn) | add\_new\_user(1, db\_conn) | User already exists |  | Need to add validation of user\_id to check if already exists |
| 6 | Add new user with id that does not exist and correct data type - integer  add\_new\_user(user\_id, conn) | add\_new\_user(11, db\_conn) | 1) “User 11 record added”  2) There should be a new entry in the sqlite3 database  3) The balance should be 0 | “User 11 record added”  True | This test works |
| 7 | Add new user with id that already exists and incorrect data type – one-digit string  add\_new\_user(user\_id, conn) | add\_new\_user(”1”, db\_conn) | 1) “Invalid user\_id data type”  2) no change in the database – a record was not added |  | This test doesn’t work as the invalid data type – string is accepted, even though unique constraint failed. Need to add data type validation |
| 8 | Add new user with id that does not exist and incorrect data type – two-digit string  add\_new\_user(user\_id, conn) | add\_new\_user(“12”, db\_conn) | 1) “Invalid user\_id data type”  2) no change in the database – a record was not added |  | This test doesn’t work as the invalid data type – string is accepted and a record was added to the database. Need to add data type validation |
| 9 | Add new user with id that already exists and incorrect data type – float  add\_new\_user(user\_id, conn) | add\_new\_user(1.0, db\_conn) | 1) “Invalid user\_id data type”  2) no change in the database – a record was not added |  | This test doesn’t work as the invalid data type – float is accepted, even though unique constraint failed. Need to add data type validation |
| 10 | Add new user with id that does not exist and incorrect data type – float  add\_new\_user(user\_id, conn) | add\_new\_user(13.0, db\_conn) | 1) “Invalid user\_id data type”  2) no change in the database – a record was not added |  | This test doesn’t work as the invalid data type – float is accepted and a record was added to the database. Need to add data type validation |
| 11 | Add new user with id that does not exist and incorrect data type – non-digit string  add\_new\_user(user\_id, conn) | add\_new\_user(“p”, db\_conn) | 1) “Invalid user\_id data type”  2) no change in the database – a record was not added |  | This test doesn’t work as the invalid data type – non-digit string is accepted and an error raised |
| 12 | Add funds for a user with id that exists, valid data type – integer and funds as valid data type – float  add\_funds(user\_id, amount, conn) | add\_funds(1, 10.0, db\_conn) | 1) “Record for user: 1 updated - 10.0 added”  2) the balance for user 1 changes to 50.0 | “Record for user: 1 updated - 10.0 added” | The user balance did not change and remained 44.0. |
| 13 | Add funds for a user with id that does not exist, valid data type – integer and funds as valid data type – float  add\_funds(user\_id, amount, conn) | add\_funds(14, 10.0, db\_conn) | 1) “User does not exist”  2) no changes in the database | 1) “User does not exist”  2) no change in the database | This test works |
| 14 | Remove funds for a user with id that already exists and correct data type – integer, funds as correct data type – float, amount not exceeding the balance  remove\_funds(user\_id, amount, conn) | remove\_funds(2, 3.0, db\_conn) | 1) “Record for user: 2 updated - 3.0 removed”  2) the balance for user\_id 2 should change from 5.0 to 2.0 | 1) “Record for user: 2 updated - 3.0 removed”  2) | The balance for did not update |
| 15 | Remove funds for a user with id that already exists and correct data type – integer, funds as correct data type – integer, amount not exceeding the balance  remove\_funds(user\_id, amount, conn) | remove\_funds(2, 3, db\_conn) | 1) “Record for user: 2 updated - 3.0 removed”  2) the balance for user\_id 2 should change from 5.0 to 2.0 | 1) “Record for user: 2 updated - 3 removed”  2) | The balance for did not update |
| 16 | Remove funds for a user with id that already exists and correct data type – integer, funds as correct data type – integer, amount exceeding the balance  remove\_funds(user\_id, amount, conn) | remove\_funds(3, 45, db\_conn) | 1) “Insufficient funds”  2) no change in the database |  | Even though the balance did not change, the message that the funds were removed was printed. The function needs corrected |
| 17 | Remove a user for user with id that already exists and correct data type – integer  remove\_user(user\_id, conn) | remove\_user(1, db\_conn) | 1) “User 1 deleted”  2) user 1 should be removed from the database |  | Even though the message about user being removed was prnted, there was no change in the database |
| 18 | Remove a user with id that does not exist and correct data type – integer  remove\_user(user\_id, conn) | remove\_user(15, db\_conn) | 1) “User does not exist”  2) no change in the database | “User does not exist” | This test works |
| 19 | Exit the app while the app is still open, test performed in correct order  exit\_app(conn) | exit\_app(db\_conn) | 1) "Exiting app..."  2) the next test calling the function exit\_app(db\_conn)should fail | 1) "Exiting app..."  2) the next test did not fail | The next test did not fail which means the app either was not closed in this test or it mistakenly prints incorrect message |
| 20 | Exit the app while the app is closed, test performed in correct order  exit\_app(conn) | exit\_app(db\_conn) | “App is already closed” |  | There app was closed during the previous step but still proceeded to print “Exiting the app...” message |