```
1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
4 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
5 import il.ac.hit.CashFlowManagement.model.UserLogic;
6 import org.junit.After;
7 import org.junit.Before;
8 import org.junit.Test;
9
10 import java.sql.SQLException;
11
12 import static org.junit.Assert.assertEquals;
13 import static org.junit.Assert.assertTrue;
14
15 public class UserLogicTest {
       UserLogic userLogic = new UserLogic();
16
       String userName = "testUserName", password = "testPassword",
17
   country = "testCountry", gander = "testGander";
18
19
       @Before
20
       public void setUp() {
           userLogic.addUser(userName, password, country, gander);
21
22
       }
23
24
       @After
25
       public void tearDown() throws JDBCDataBaseException, SQLException
    {
           while(userLogic.checkIfExist("userName") || userLogic.
26
   checkIfExist(userName)) {
               JDBCDataBase.getInstance().getStatement().executeUpdate("
27
   DELETE FROM Registered Users WHERE UserName = 'testUserName'");
               JDBCDataBase.getInstance().getStatement().executeUpdate("
28
  DELETE FROM Registered_Users WHERE UserName = 'userName'");
29
           }
30
       }
31
       @Test
32
       public void checkIfUserExist() throws JDBCDataBaseException,
33
   SQLException {
34
           assertEquals(false, userLogic.checkIfExist("userName"));
35
           assertTrue(userLogic.checkIfExist(userName));
36
       }
37
38
       @Test
39
       public void addUser() {
40
```

```
assertEquals(false, userLogic.checkIfExist("userName"));
41
           userLogic.addUser("userName", password, country, gander);
42
           assertTrue(userLogic.checkIfExist("userName"));
43
44
       }
45
       @Test
46
       public void checkUserPassword() {
47
           assertEquals(true, userLogic.checkUserPassword(userName,
48
   password));
           assertEquals(false, userLogic.checkUserPassword(userName,
49
  password + "1"));
50
       }
51 }
52
```

```
1 package il.ac.hit.CashFlowManagement.test;
 2
 3 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
 4 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
 5 import org.apache.log4j.BasicConfigurator;
 6 import org.junit.Test;
 7
 8 import static junit.framework.TestCase.assertNotNull;
 9
10 public class JDBCDataBaseTest {
       public JDBCDataBaseTest(){
11
12
           BasicConfigurator.configure();
13
       }
14
15
       @Test
       public void getInstance() {
16
           try {
17
               assertNotNull(JDBCDataBase.getInstance());
18
19
           } catch (JDBCDataBaseException e) {
20
               e.printStackTrace();
21
           }
22
       }
23
24
       @Test
       public void getStatement() {
25
26
           try {
               assertNotNull(JDBCDataBase.getInstance().getStatement());
27
           } catch (JDBCDataBaseException e) {
28
               e.printStackTrace();
29
30
           }
       }
31
32
33
       @Test
       public void getConnection() {
34
35
           try {
               assertNotNull(JDBCDataBase.getInstance().getConnection
36
   ());
37
           } catch (JDBCDataBaseException e) {
               e.printStackTrace();
38
39
           }
       }
40
41
42 }
43
```

```
1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.model.Expense;
4 import il.ac.hit.CashFlowManagement.model.ExpensesLogic;
5 import il.ac.hit.CashFlowManagement.model.JDBCDataBaseLogic;
6 import il.ac.hit.CashFlowManagement.view.LoginForm;
7 import il.ac.hit.CashFlowManagement.viewmodel.IViewModel;
8 import il.ac.hit.CashFlowManagement.viewmodel.ManagementViewModel;
9 import org.apache.log4j.BasicConfigurator;
10 import org.junit.After;
11 import org.junit.Test;
12
13 import java.sql.ResultSet;
14
15 import static org.junit.Assert.assertEquals;
16
17 public class ExpensesLogicTest {
18
       private JDBCDataBaseLogic dataBaseLogic = JDBCDataBaseLogic.
   getInstance();
       private String expenseDate = "01/01/2020", expenseClassification
19
    = "testClassification", expenseDetails = "testDetails";
       private double expenseCost = 5.55;
20
21
       private IViewModel viewModel;
       private String nameOfTable;
22
       private ExpensesLogic expensesLogic;
23
       private Expense expense = new Expense(expenseDate,
24
   expenseClassification, expenseDetails, expenseCost);
25
       public ExpensesLogicTest(){
26
           BasicConfigurator.configure();
27
           LoginForm.username = "testUserName";
28
           viewModel = ManagementViewModel.getInstance();
29
           nameOfTable = LoginForm.username.toUpperCase() + " Expenses".
30
   toUpperCase();
31
           expensesLogic = new ExpensesLogic();
32
       }
33
34
       @After
35
       public void tearDown() throws Exception {
           dataBaseLogic.removeTable(nameOfTable);
36
37
       }
38
39
       @Test
       public void addExpense() throws Exception {
40
           expensesLogic.addExpense(expense);
41
           ResultSet rs = expensesLogic.getAllUserExpenses();
42
```

```
43
           rs.next();
           String date = rs.getString("Date");
44
           String cost = rs.getString("Cost");
45
           String classification = rs.getString("Classification");
46
           String details = rs.getString("Details");
47
           double costDouble = Double.parseDouble(cost);
48
49
50
           assertEquals(expenseDate, date);
           assertEquals((long)expenseCost, (long)costDouble);
51
           assertEquals(expenseClassification, classification);
52
           assertEquals(expenseDetails, details);
53
54
       }
55
       @Test
56
57
       public void getAllUserExpenses() throws Exception {
           int expensesCounter = 0;
58
           expensesLogic.addExpense(expense);
59
           ResultSet rs = expensesLogic.getAllUserExpenses();
60
61
           while (rs.next()){
62
               expensesCounter++;
63
           }
           assertEquals(1, expensesCounter);
64
       }
65
66 }
67
```

```
1 package il.ac.hit.CashFlowManagement.test;
2
3
4 import il.ac.hit.CashFlowManagement.exception.
   JDBCDataBaseLogicException;
5 import il.ac.hit.CashFlowManagement.model.JDBCDataBaseLogic;
6 import org.apache.log4j.BasicConfigurator;
8 import static org.hamcrest.CoreMatchers.is;
9 import static org.junit.Assert.*;
10
11
12 public class JDBCDataBaseLogicTest {
       private JDBCDataBaseLogic dataBaseLogic = JDBCDataBaseLogic.
13
   getInstance();
       private final String iNameOfTable = "Test";
14
15
16
       public JDBCDataBaseLogicTest(){
17
           BasicConfigurator.configure();
18
       }
19
       @org.junit.Before
20
       public void setUp() throws Exception {
21
           dataBaseLogic.createTableIfNotExist(iNameOfTable,"Password",
22
   "varchar(300)");
       }
23
24
25
       @org.junit.After
26
       public void tearDown() throws Exception {
27
           dataBaseLogic.removeTable(iNameOfTable);
28
29
       }
30
31
       @org.junit.Test
32
       public void createTableIfNotExist() throws Exception {
33
           String tableName = "CreateTableTest";
34
35
           assertEquals(true, dataBaseLogic.createTableIfNotExist(
  tableName, "Password", "varchar(300)"));
           assertEquals(false,dataBaseLogic.createTableIfNotExist(
36
  tableName, "Password", "varchar(300)"));
           dataBaseLogic.removeTable(tableName);
37
38
           try{
               dataBaseLogic.createTableIfNotExist(iNameOfTable,"
39
   Password");
               fail("Expected exception has not been thrown");
40
```

```
41
           } catch (Exception e) {
42
               assertThat(e.getMessage(), is("Please enter the
   parameters as follows: Param1Name ,Param1Type, Param2Name ,Param2Type
    .... , ParamName ,ParamType"));
43
44
       }
45
       @org.junit.Test
46
       public void insertValue() throws Exception {
47
           String params = "'" + "134" + "'";
48
49
50
           dataBaseLogic.insertValue(iNameOfTable,params);
51
           try {
               dataBaseLogic.insertValue(iNameOfTable, null);
52
               // If the exception is not thrown, the test will fail
53
               fail("Expected exception has not been thrown");
54
           } catch (Exception e) {
55
           assertThat(e.getMessage(), is("Argument for @NotNull
56
   parameter 'iParameters' of il/ac/hit/CashFlowManagement/model/
   JDBCDataBaseLogic.insertValue must not be null"));
57
           }
58
       }
59
       @org.junit.Test
60
       public void query() throws JDBCDataBaseLogicException {
61
           assertNotNull(dataBaseLogic.query("select * from " +
62
   iNameOfTable));
63
64
           try {
               assertNotNull(dataBaseLogic.query("select * from " +
65
   iNameOfTable + "1"));
               fail("Expected exception has not been thrown");
66
           } catch (Exception e) {
67
               assertThat(e.getMessage(), is("query execution failed
68
   Table/View 'TEST1' does not exist."));
69
           }
70
71
           try {
               dataBaseLogic.query("create table TEST (Password varchar(
72
   300))");
               fail("Expected exception has not been thrown");
73
           } catch (Exception e) {
74
               assertThat(e.getMessage(), is("query execution failed
75
   Table/View 'TEST' already exists in Schema 'APP'."));
76
77
       }
```

```
78
79
       @org.junit.Test
       public void removeTable() throws JDBCDataBaseLogicException {
80
           String tableName = "RemoveTestTable";
81
82
           dataBaseLogic.createTableIfNotExist(tableName, "Password", "
83
   varchar(300)");
           dataBaseLogic.removeTable(tableName);
84
85
86
           try {
               dataBaseLogic.removeTable(tableName+"1");
87
               fail("Expected exception has not been thrown");
88
           } catch (Exception e) {
89
               assertThat(e.getMessage(), is("failed to remove
90
   RemoveTestTable1 from DB"));
91
           }
92
       }
93
94
       @org.junit.Test
       public void getInstance() throws JDBCDataBaseLogicException {
95
           assertNotNull(JDBCDataBaseLogic.getInstance());
96
97
       }
98
99 }
```

```
1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.exception.FormCastingException;
4 import il.ac.hit.CashFlowManagement.exception.
  GetAllUserExpensesException;
5 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
6 import il.ac.hit.CashFlowManagement.model.Expense;
7 import il.ac.hit.CashFlowManagement.model.ExpensesLogic;
8 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
9 import il.ac.hit.CashFlowManagement.model.UserLogic;
10 import il.ac.hit.CashFlowManagement.view.LoginForm;
11 import il.ac.hit.CashFlowManagement.viewmodel.ManagementViewModel;
12 import org.junit.After;
13 import org.junit.Before;
14 import org.junit.Test;
15
16 import java.sql.ResultSet;
17 import java.sql.SQLException;
18
19 import static org.junit.Assert.assertEquals;
20 import static org.junit.Assert.assertNotNull;
21
22 public class ManagementViewModelTest {
       private ManagementViewModel viewModel;
23
       private UserLogic userLogic = new UserLogic();
24
       private String userName = "testUserName", password = "
25
  testPassword", country = "testCountry", gander = "testGander",
  nameOfTable;
       private String expenseDate = "01/01/2020", expenseClassification
26
    = "testClassification", expenseDetails = "testDetails";
       private double expenseCost = 5.55;
27
       private Expense expense = new Expense(expenseDate,
28
   expenseClassification, expenseDetails, expenseCost);
29
30
31
       public ManagementViewModelTest(){
           LoginForm.username = "testUserName";
32
33
           viewModel = ManagementViewModel.getInstance();
           viewModel.setExpensesLogic(new ExpensesLogic());
34
           nameOfTable = LoginForm.username.toUpperCase() + "_Expenses".
35
   toUpperCase();
       }
36
37
       @Before
38
       public void setUp() {
39
           userLogic.addUser(userName, password, country, gander);
40
```

```
41
42
43
       @After
       public void tearDown() throws JDBCDataBaseException, SQLException
44
    {
           while(userLogic.checkIfExist("userName") || userLogic.
45
   checkIfExist(userName)) {
               JDBCDataBase.getInstance().getStatement().executeUpdate("
46
   DELETE FROM Registered_Users WHERE UserName = 'testUserName'");
               JDBCDataBase.getInstance().getStatement().executeUpdate("
47
   DELETE FROM Registered Users WHERE UserName = 'userName'");
48
           }
       }
49
50
51
       @Test
       public void getInstance() {
52
           assertNotNull(ManagementViewModel.getInstance());
53
54
       }
55
56
       @Test
57
       public void verifyUser() {
           assertEquals(true, viewModel.verifyUser(userName, password));
58
59
       }
60
       @Test
61
       public void register() {
62
           assertEquals(true, viewModel.register("userName", password,
63
   country, gander));
           assertEquals(false, viewModel.register(userName, password,
64
   country, gander));
65
       }
66
       @Test
67
       public void getAllExpenses() throws GetAllUserExpensesException,
68
   SQLException, FormCastingException {
           ResultSet rs = viewModel.getAllExpenses();
69
           int counter = 0;
70
71
           while (rs.next()){
72
73
               counter++;
74
           }
           assertEquals(0, counter);
75
       }
76
77 }
78
```