



BURGER QUEEN

SWEN3165 – Software Testing

Abstract

This report describes the testing of a food ordering service application.

CodeMetric Technology

codemetricswen@gmail.com

Software Testing: Report

Lecturer:

Pro. Thomas Canhao Xu

Group Name:

CodeMetric Tech

Group Leader:

Scott Allen

Other Members:

Khadeja Clarke

Mischka Neill

Nigel Francis

Table of Tasks/Responsibilities

Student ID No.		Student Name	Tasks
UWI	GIST		
620099245	180923	Allen, Scott Nicholas	Project preparations
			Food Fragment
620098443	180908	Clarke, Khadeja Renee	User Interface Standardisation
			General Event Details Fragment
415001546	180925	Francis, Nigel Fitzgerald	Database Testing
			User Related Functionalities
620100135	180913	Neill, Mischka Zoila	Diagrams and Models Event Type and Guests Fragment

Table of Content

INTRODUCTION	4
PURPOSE OF THIS DOCUMENT.....	4
INTENDED AUDIENCE	4
SCOPE	4
BACKGROUND INFORMATION	4
PURPOSE OF THIS PROJECT	4
TESTING PROCESS	5
MANUAL TESTING	5
AUTOMATED TESTING	7
<i>Testing Environment.....</i>	<i>7</i>
<i>Favourites.....</i>	<i>8</i>
<i>Cart.....</i>	<i>9</i>
DATABASE TESTING	14
<i>Overview</i>	<i>14</i>
<i>Details</i>	<i>14</i>
<i>Data Integrity Results</i>	<i>15</i>

Introduction

Purpose of this Document

The main purpose of this document is to provide a detailed explanation of this project from start to finish.

Intended Audience

This document is intended for the lecturers of The University of the West Indies (UWI) and Global Institute of Software Technology (GIST).

Scope

This document should provide the intended audience with details about the testing process of the Burger Queen application, and all recommendations the team at CodeMetric Technologies has for the developers of the Burger Queen application.

Background Information

Purpose of this Project

To fulfil the requirements of the course SWEN3165 – Software Testing.

Testing Process

Manual Testing

The team at CodeMetric Technologies ran the Burger Queen application without having any knowledge of its interior workings. Listed below were our observations.

1. When the application started, there was a launch screen. When the time for that expired, the Login screen came up. Since we had no account, we had to create one. Unfortunately, the font used for the text “If you don’t have an account, Sign Up Here” was very hard on the eyes.

2. A total of five attempts were made to complete the registration form.

On the first attempt, a blank form was submitted. The field for Email Address rendered an error message that said “Invalid Username”.

On the second attempt, only the field for Email Address had a value that was not a valid email address. The same error message was rendered.

On the third attempt, only the field for Email Address had a value. This time, a valid email address was inputted. The form was successfully submitted. That means, records in the database have null values.

On the fourth attempt, all fields were filled out with appropriate values. The form also submitted successfully.

On the final attempt, the same values from the fourth attempt were inputted. Again, the form submitted successfully. This means that the database contains duplicate records.

3. Two attempts were made to login to the app.

On the first attempt, the blank credentials were submitted. We were given access to the app.

We were able to return to the login screen by simply hitting the back button, instead of pressing the logout option in the three vertical dots on the top app bar. It is to be noted that when this option is selected, however, nothing happens.

On the second attempt, the credentials used on the fourth and fifth attempts at registration were used. We regained access to the app.

4. There is a carousel on the home screen and the favourites screen for daily specials. There is a “Read More” button embedded in it. On click of the button, an item was added to the cart instead of being redirected to wherever the information on the special is.

5. Below the carousel is a tabbed layout with a recycler view in each tab. From there the user can select from an assortment of Burgers, Drinks, Sides, Desserts. Each view holder was designed with: (1) a title, (2) a description, (3) an associated image, (4) an associated cost, and (5) two action buttons (add to cart, add to favourites). The “Add to Cart” action

button was represented by a plus; “Add to Favourites” by an outlined heart. Each view holder was a clickable item that launched a dialog.

There were very few instances where the image matched the item. There were more cases where the same image was used for multiple menu items, or the image was just blank.

Recycler View

On click of the heart, the item was added to the Favourite’s list. The interface did not reflect that the item was added to favourites, outside of a toast being displayed. We anticipated the heart being filled. We clicked the heart for the same view holder repeatedly and the message displayed by the toast was always “Adding to favourites” and never anything to indicate that the item had already been favoured.

Dialog

On click of the plus, the message in the toast was “Testing”. On click of the heart, the message in the toast was “TESTING BUTTON 1”.

6. When we had finished adding our items to favourites and the cart, we were unsure about how to access either. It was sheer happenstance that caused us to stumble upon both. The bottom app bar shared the same colour as the background of the page so it blended right in. The colour became the same as the top app bar when it was clicked.
7. In both Favourites and the Cart is the list of all items that were added to each respectful list. The view holders in each of these pages differs from the ones on the home page in that the only action button is “Remove”. On click of this button, a toast appears with the message “I am deleting this” but the item is not removed from the recycler view.
8. On the Favourites page, the item that was selected multiple times also appeared multiple times.
9. Coupons are available for purchases. The setting can be toggled on or off. But regardless of state (on/off), they are accessible. The same with Favourites.
10. The top app bar used on the Home page is not the same as the one used for Favourites and Cart. The one on the Home page offers *Search* capabilities and access to Application Settings. The one on the other two pages do not have these two things, which is an inconvenience.

Automated Testing

Testing Environment

Our testing environment is setup using the following frameworks:

1. Espresso – UI Testing
2. Cucumber & Gherkin – Test automation framework
3. AndroidJUnitRunner – Test runner

Feature Files

Feature files are based in the assets/features directory. These feature files are written in the gherkin language that will map each test feature to the relevant step definition. Each scenario in a feature file makes up the acceptance criteria for that feature.

Step Definitions

The Step definitions are based in the tests/steps directory. The step definitions use regular expressions to map each gherkin scenario to its matching step definition. The espresso functions that interact with the UI are wrapped inside the cucumber step definition function to allow automation of UI tests based on the gherkin scenarios.

Test Case

The cucumber options feature of the cucumber framework was used to customize test cases with the use of tags. To run specific test cases, we can simply specify a set of tags, and only tests with those tags will be executed.

Favourites

Cucumber and Espresso were used to test this functionality of the application. We tested Adding to Favourites and Removing from Favourites.

Adding to Favourites

In this Cucumber feature file were three scenarios:

1. Favourites is enabled in settings and the item has not already been added to the list of favourites.
2. Favourites is enabled in settings and the item has already been added to the list of favourites.
3. Favourites is disabled in settings.

SCENARIO	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
1	Item should be added to favourites.	Item is added.	✓
2	Item should not be added again to favourites.	Item is added.	✗
3	Favourites button should be disabled, therefore disallowing an item to be added to favourites.	Favourites button is not disabled, therefore allowing an item to be added.	✗

Removing from Favourites

In this Cucumber feature file was one scenario.

SCENARIO	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
1	Item should be removed from favourites.	Item is not removed.	✗

Cart

Cucumber and Espresso were used to test this functionality of the application. We tested Adding to cart and Removing from cart.

Manage cart

In this Cucumber feature file were two scenarios:

1. A user adds an order to their cart
2. A user deletes an order from their cart

SCENARIO	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
1	The specific order should be added to their cart.	Order is added.	✓
2	A user deletes a specific order from their cart.	Order is not deleted.	✗

Sign Up

Espresso and JUnit were used to test this functionality of the application. We tested creating a new user account.

SIGN UP	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
HAS WIDGETS	All widgets should be loaded into view.	Widgets loaded into view.	✓
VALID CREDENTIALS	User should be successfully created.	User successfully makes an account.	✓
DUPLICATE CREDENTIALS	User should not be able to make another account.	User did not see an error message, therefore a duplicate account was made.	✗
EMPTY FIELDS	User should not be able to create an account with null values.	User did not see an error message, therefore an account exists with null values.	✗
INVALID EMAIL 1 & 2	User should not be able to make an account with invalid email addresses.	User did not see an error message, therefore an account exists with invalid email addresses.	✗
INVALID PASSWORD CONFIRMATION	The user should not be able to make an account if the password confirmation does not match what was written in the password field.	User sees an error message, therefore the account was not created.	✓

Login

Espresso and Junit were used to test this functionality of the application. We tested logging in and logging out of the application.

LOGIN	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
HAS WIDGETS	All widgets should be loaded into view.	Widgets loaded into view.	✓
VALID CREDENTIALS	User should be successfully logged in.	User successfully logs in.	✓
EMPTY FIELDS	User should not be able to login with null values.	User did not see an error message, therefore they were able to login.	✗
INCORRECT USERNAME	User should not be able to login with an incorrect username.	User saw an error message, therefore they were unable to login.	✓
INCORRECT PASSWORD	User should not be able to login with an incorrect password.	User saw an error message, therefore they were unable to login.	✓
BOTH CREDENTIALS INCORRECT	User should not be able to login with incorrect credentials.	User saw an error message, therefore they were unable to login.	✓
LOGOUT	The user should not be able to logout and re-enter the application without re-entering credentials.	The user was not stopped from logging in without re-entering their credentials	✗

Updating Account Details

Espresso and JUnit were used to test this functionality of the application. We tested updating the user's credentials which will be used to login in future instances.

UPDATE CREDENTIALS	BEHAVIOUR		PASS
	EXPECTED	ACTUAL	
HAS WIDGETS	All widgets should be loaded into view.	Widgets loaded into view.	✓
VALID EMAIL CHANGE	User should be able to successfully update their email.	User successfully updates their email.	✓
VALID USERNAME CHANGE	User should be able to successfully update their username.	User successfully updates their username.	✓
VALID PASSWORD CHANGE	User should be able to successfully update their password.	User successfully updates their password.	✓
VALID LOCATION CHANGE	User should be able to successfully update their location.	User successfully updates their location.	✓
EMPTY EMAIL FIELD	User should not be able to update their email with a null value.	User did not see an error message, therefore they were able to update their email.	✗
EMPTY USERNAME FIELD	User should not be able to update their username with a null value.	User did not see an error message, therefore they were able to update their username.	✗
EMPTY PASSWORD FIELD	User should not be able to update their password with a null value.	User did not see an error message, therefore they were able to update their password.	✗

INVALID EMAIL	User should not be able to update their password with an invalid email address.	User did not see an error message, therefore they were able to update their email.	x
------------------	---------------------------------------------------------------------------------	------------------------------------------------------------------------------------	---

Difficulties:

- Due to the speed at which the flow of automated testing occurs on my computer, it was failing to read certain widgets or values on the view fast enough. To fix this I made the thread sleep for a short period before doing some tests.
- After writing my Junit tests I then attempted to use feature files and cucumber. For some reason it would not run my feature files, but due to espresso and Junit being suitable for a local application I used the original tests as they thoroughly tested at necessary.

Database Testing

Overview

Grey box testing will be used for the Database testing as it provides combined benefits of both black box and white box testing. Additionally, it also reduces the long overhead of testing functional and non-functional types. Since the database used in this application is SQLite then database load testing will not be performed as it varies from device to device, instead data integrity and database security only will be tested.

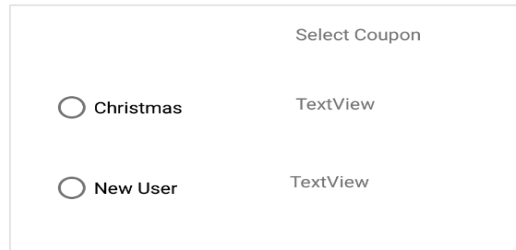
Details

- Data Integrity: Data will be verified using DB Browser for SQLite
- Database Security: Various methods will be employed to disrupt or corrupt the database.

Data Integrity Results

The database has seven tables as such data will be saved to the database and the data will be checked using DB Browser for SQLite.

Coupons Table:



Select Coupon

☐ Christmas TextView

☐ New User TextView

Figure 1 Coupons Table: Application Screenshot

	_id	couponname	couponid	discount	userid	couponused
		Filter	Filter	Filter	Filter	Filter
1	1	New User	WELCOME15	15	4	false
2	2	New User	WELCOME15	15	3	false

Figure 2 Coupons Table: Database Screenshot

The above results clearly show a disparity between what is displayed to the user on the application and what is in the database. The Christmas coupon is not saved in the database although it was selected. Instead only the New User coupon is saved.

Favourites Table:

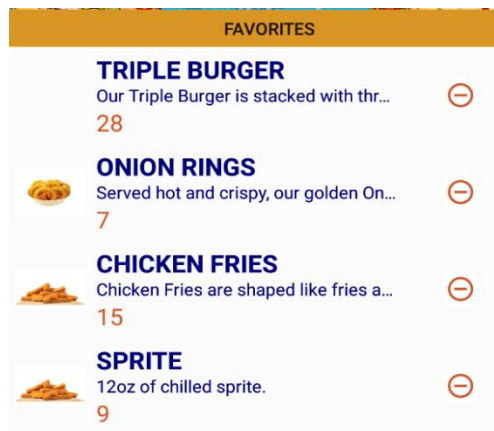


Figure 3 Favourites Table: Application Screenshot

	_id	itemid
	Filter	Filter
1	1	12
2	2	6
3	3	8
4	4	17

Figure 4 Favourites Table: Database Screenshot

The data displayed in the application matches the data saved in the database. The item id's in the database were verified against the menu items and the id's match what is displayed in the application.

Menu Table:

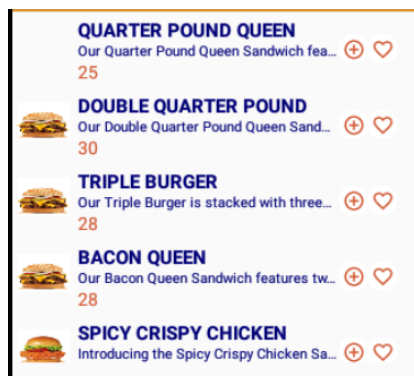


Figure 6 Burgers Menu: Application Screenshot

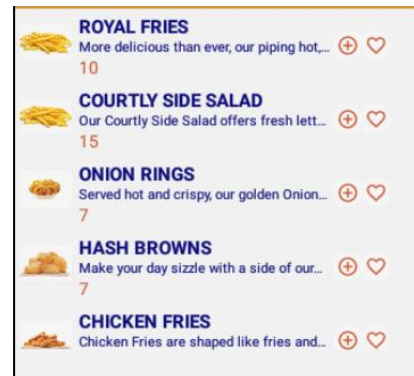


Figure 5 Sides Menu: Application Screenshot

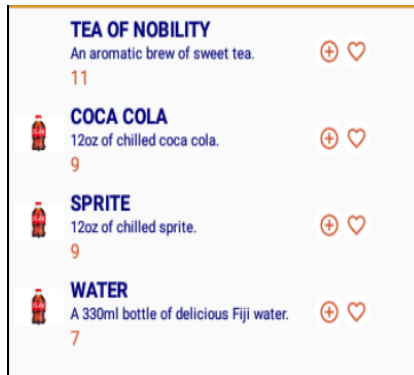


Figure 7 Drinks Menu: Application Screenshot



Figure 8 Desserts Menu: Application Screenshot

	_id	itemid	itemname	itemimage	itemcategory	itemdescription	itemprice
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	Vanilla Shake	onions_rings	Desserts	Cool down with ...	20
2	2	2	Royal Court Pie	onions_rings	Desserts	Say hello to our ...	30
3	3	3	Chocolate Chip ...	chocolate_chip	Desserts	A rich cookie sp...	5
4	4	4	Royal Fries	royal_fries	Sides	More delicious t...	10
5	5	5	Courtly Side Salad	onions_rings	Sides	Our Courtly Sid...	15
6	6	6	Onion Rings	onion_rings	Sides	Served hot and ...	7
7	7	7	Hash Browns	hashbrowns	Sides	Make your day s...	7
8	8	8	Chicken Fries	chicken_fries	Sides	Chicken Fries ar...	15
9	9	9	Crispy Chicken ...	chicken_wrap	Sides	Get our Crispy ...	12
10	10	10	Quarter Pound ...	onions_rings	Burgers	Our Quarter Po...	25
11	11	11	Double Quarter ...	dblqtrpndqueen	Burgers	Our Double Qua...	30
12	12	12	Triple Burger	onions_rings	Burgers	Our Triple Burg...	28
13	13	13	Bacon Queen	onions_rings	Burgers	Our Bacon Quee...	28
14	14	14	Spicy Crispy Chi...	spicy_chicken	Burgers	Introducing the ...	21
15	15	15	Tea of Nobility	onions_rings	Beverages	An aromatic bre...	11
16	16	16	Coca Cola	coca_cola	Beverages	12oz of chilled ...	9
17	17	17	Sprite	onions_rings	Beverages	12oz of chilled ...	9
18	18	18	Water	onions_rings	Beverages	A 330ml bottle ...	7
19	19	19	Special1	smnelse	Special	Burger Combo.	4
20	20	20	Special2	special2	Special	Raining tomatoes.	3

Figure 9 Menus: Database Screenshot

The data in the database matches perfectly the data that is displayed in the application.

Orders Table:

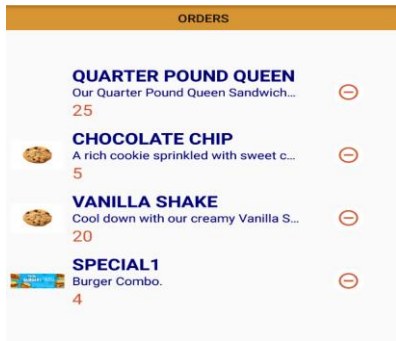


Figure 10 Orders Table: Application Screenshot

	_id	itemid
	Filter	Filter
1	1	10
2	4	3
3	6	1
4	7	19

Figure 11 Orders Table: Database Screenshot

The data displayed in the application matches the data saved in the database. The item id's in the database was verified against the menu items and the id's match what is displayed in the application.

Specials Table:

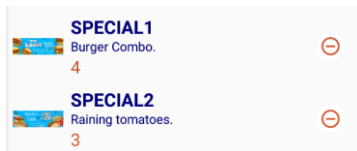


Figure 12 Specials Table: Application Screenshot

_id	itemid
Filter	Filter
1	19
2	20

Figure 13 Orders Table: Database Screenshot

The data displayed in the application matches the data saved in the database. The item id's in the database was verified against the menu items and the id's match what is displayed in the application.

Users Table:

	_id	username	password	email	home_restaurant	date_registered	first_login
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	John Doe	123	john@123.com	Suzhou	2018-11-12	NULL
2	2	nfrancis	nigel	nigel.f.francis@...	#8, Jinjihu Lu, Si...	2019-05-23 01...	NULL
3	3	nfrancis		nigel.f.francis@...	Choose nearest ...	2019-05-23 01...	2019-05-23 01...
4	4			nigel.f.francis@...	Choose nearest ...	2019-05-23 01...	2019-05-23 01...
5	5						NULL

Figure 14 Users: Database Screenshot

The Database has null values, duplicate emails and usernames also the order list and favorites are not linked to any user name in the database. The registration page and the update user info allow data to be written to the database and as there are no checks then anything or nothing can be written to the database.

Data Integrity Results

It was noticed that a dummy user is hard coded into the application as John Doe with a password of 123. This is a backdoor where the user has access to full functionality of the application. The user passwords are stored in clear text and not encrypted. Anyone with access to the database will have access to all the passwords.

It was also noticed that thought the code that there are "Log.i" which allows users access to internal information in the code. E.g. `Log.i("UserName: ", user_session_obj.getUsername());`.

There is a huge security risk where SQL injection can be done in register and update user forms. We were able to crash the application by injecting the following code `-- ';"DROP TABLE user";/*`. We were also able to register a user with no time stamp by injecting the following code `-- ","",",");` `"DROP TABLE user";/*`.

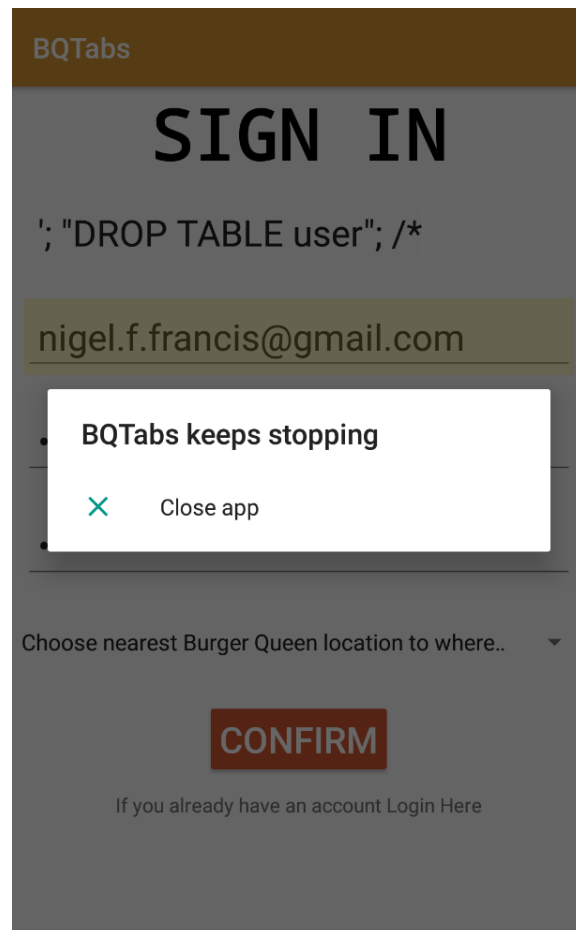


Figure 15 Application after SQL Injection

Recommendations

Automated Testing

Favourites

ADDING TO FAVOURITES	
SCENARIO	RECOMMENDATION
2	Disallow multiple instances of an item to be added to the user's Favourites.
3	Disable the Favourites button, or set its visibility to 'gone'
REMOVING FROM FAVOURITES	
SCENARIO	RECOMMENDATION
1	Remove the item from the Recycler View and notify that the data set has been changed.

Cart

Managing cart	
SCENARIO	RECOMMENDATION
2	Create a feature to allow the user to quickly glance their cart without navigating to the cart page.
2	A success message is displayed; however, the list of orders in the cart is not actually updated. The item should be removed from the list and the UI updated when there is any change to the list.

Sign-Up:

- testEmptyFields failed as there was no error alert found to stop the user from registering with null values. Due to the fact that a database should never have null values, the suggested fix for this issue is to implement such an error which would stop the registration process from continuing.
- testEnterValidDuplicateCredentials failed as there was no error alert found to stop the user from registering with values identical to those already in the database. To adhere to uniqueness rules of the database, the suggested fix is to implement an error alert which would stop the registration process from continuing.
- test1InvalidEmail and test2InvalidEmail failed as there was no error alert found to stop the user from registering with invalid email addresses. Due to wanting to have real email address stored and not non-existent values, code should be implemented to test the email address entered by the user, and an alert to stop the user from creating an account with an invalid email address.

Login:

- test2EmptyLogin failed because there was no error alert found stopping the user from logging in with null values. The suggested fix is to implement an error which will stop the login process from continuing if no values are entered into the username and password fields.
- Logout failed as there was nothing to stop the user from re-entering the application without re-entering their credentials. As such the session should end when the user exits the application and they should be forced to re-enter their credentials to gain access.

Update Account Credentials:

- test3EmptyEmailChange failed as no error alert was found to stop the user from updating their username with a null value. The suggested fix is to implement an error which will stop the user from updating their account if no values are entered into the email field.
- test3EmptyPwChange failed as no error alert was found to stop the user from updating their password with a null value. The suggested fix is to implement an error which will stop the user from updating their account if no values are entered into the password field.
- test3EmptyUsernameChange failed as no error alert was found to stop the user from updating their username with a null value. The suggested fix is to implement an error which will stop the user from updating their account if no values are entered into the username field.

- test3InvalidEmailChange1 and test3InvalidEmailChange1 both failed as no error alerts were found to stop the user from updating their email with invalid email addresses. The suggested fix is to implement code to test the email address entered by the user, and an alert to stop the user from updating their account with an invalid email address.

Database Testing

Database

1. The Christmas coupon should be saved into the database.
2. The favorites table and order list table should be linked to the users table.
3. Encrypt all passwords in the database.
4. Check the database against the username and email before insertion.
5. Check the edit text for empty string and null value before inserting into database
recheck again and after inserting into the database check it again in the database. Also,
when retrieving perform the check again.

Security

1. Remove the line of code adding the dummy user.
2. Remove all Logcat from the application.
3. To prevent SQL injection, create prepared statements using a place holder. By using a placeholder SQLite automatically treats the data as input data and it does not interfere with parsing the actual SQL statement.
4. Use a regular expression for user name and password. That way you can limit the type of special characters accepted.