

# ET1446 Software Development for Telecommunication System

## **THE DESIGN AND IMPLEMENTATION OF ONLINE SHOP**

**BY**

**Saaish Bhonagiri**

**([sabh17@student.bth.se](mailto:sabh17@student.bth.se))**

**19951114-2912**

# Introduction

## Online Shop:

Online shopping is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine, which displays the same product's availability and pricing at different e-retailers

The objective of this project is to develop a general purpose Online music shop where the User interface provides the details of each Music item and its properties (Artist, Album, Release date, Producer, Price, Quantity). The quantity property keeps track of the number of identical items available in the store.

Each User can self-register for the store and browse the products. The user can make a purchase only if the quantity is available. If the User exceed the quantity of the items, then an error message will be displayed. The users can also self-delete their accounts if they wish. The user can always make a successful purchase.

The effect of a purchase is to decrease the quantity of the items in the store and clear the cart. After the cart is emptied (either because the user cancelled its content or because a purchase was performed), the user interface changes to the initial view that lists available items

Apart from that, an administrator interface should be designed. The administrator also has an account. After successful login the admin has powers to edit the products of music store and can add a new product. The admin has also option to manage user accounts.

## Software Requirements

SNO	Requirement	Description	Test case
R1	Implementation of virtual machines with suitable connections	The online store will be implemented using 2 VirtualBox Ubuntu Linux VMs (1GB RAM per VM is sufficient). One VM is designated as front end (VM_WWW) and the other as the back end (VM_DB). VM_WWW and VM_DB will be able to communicate over a VirtualBox host-only network with the network prefix 192.168.100/24. The machine hosting this 2 VMs acts as a client and interacts with VM_WWW using NAT port forwarding	T1
R2	Database connection	The back end (VM_DB) will consist of an SQL database system. The front end should not keep information regarding the store. Instead the frontend should fetch, insert, update and delete from the database in the backend VM.	T2
R3	User registration	The user interface is provided with user registration page. Here the user can create an account with username and password. The error message will be displayed when the password does not match, and the fields are empty.	T3

<b>R4</b>	Login page	The login page is provided where the user can login with the created credentials. The error message is displayed when the user enters wrong password, when the user id does not exist.	T4
<b>R5</b>	Admin login page	The admin page is provided where the admin can login with the created credentials. The error message is displayed when the user enters wrong password.	T5
<b>R6</b>	Products page	The product page should fetch the data from the database and list the details of the available items in the music store with item properties (Artist, Album, Release date, Producer, Price, Quantity).	T6
<b>R7</b>	Adding item to the cart.	Add to cart option is displayed after every individual item. When the user performs it, the item should be added to the cart.	T7
<b>R8</b>	Limited Quantity error	An error message will be displayed when ever the user wants to add an item with less than zero quantity.	T8
<b>R9</b>	Cart page	The cart page should list the items added in the cart.	T9
<b>R10</b>	Remove item from the cart	The delete option is provided in the cart page to remove any item in the cart. The deleted item is added back to the products homepage.	T10

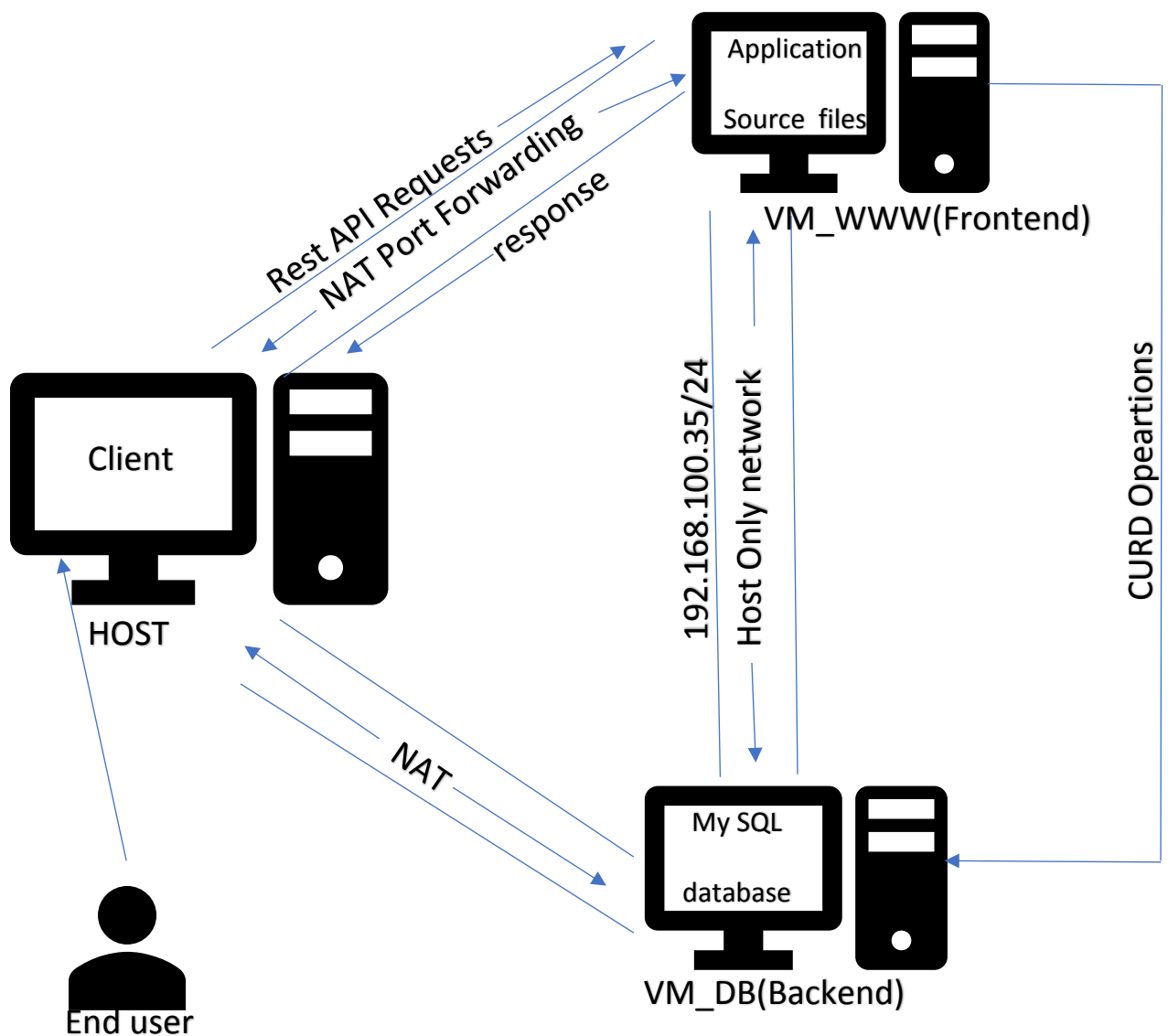
<b>R11</b>	Total amount	The total price of all the items in the cart is displayed in its page.	T11
<b>R12</b>	Purchase	The purchase option is provided to make the final purchase after adding required items to the cart. The successful purchase decreases the quantity of the products purchased.	T12
<b>R13</b>	Delete my account	A delete my account is provided in the products page to delete the individual account of the user.	T13
<b>R14</b>	Admin page for editing users	The admin page is provided with “edit users” option. The admin can manage user accounts in the edit users page.	T14
<b>R15</b>	Admin page for editing products	The admin page is provided with “edit products” option. The admin can add, edit, delete products in the edit users page.	T15
<b>R16</b>	Products information backend	The backend VM, music store database consists of information regarding products in products table.	T16
<b>R17</b>	User account information backend	The backend VM, music store database consists of information regarding user accounts in users table.	T17
<b>R18</b>	Cart information backend	The backend VM, music store database consists of information regarding items added to the cart in cart table.	T18

<b>R19</b>	Rest API communication between client and frontend VM	Complete Rest API should be implemented between client and front end following the Rest paradigm. The http methods GET, PUT, POST and Delete are implemented to retrieve, update, post and delete information respectively	<b>T19</b>
------------	---	--	------------

# DESIGN

## Model:

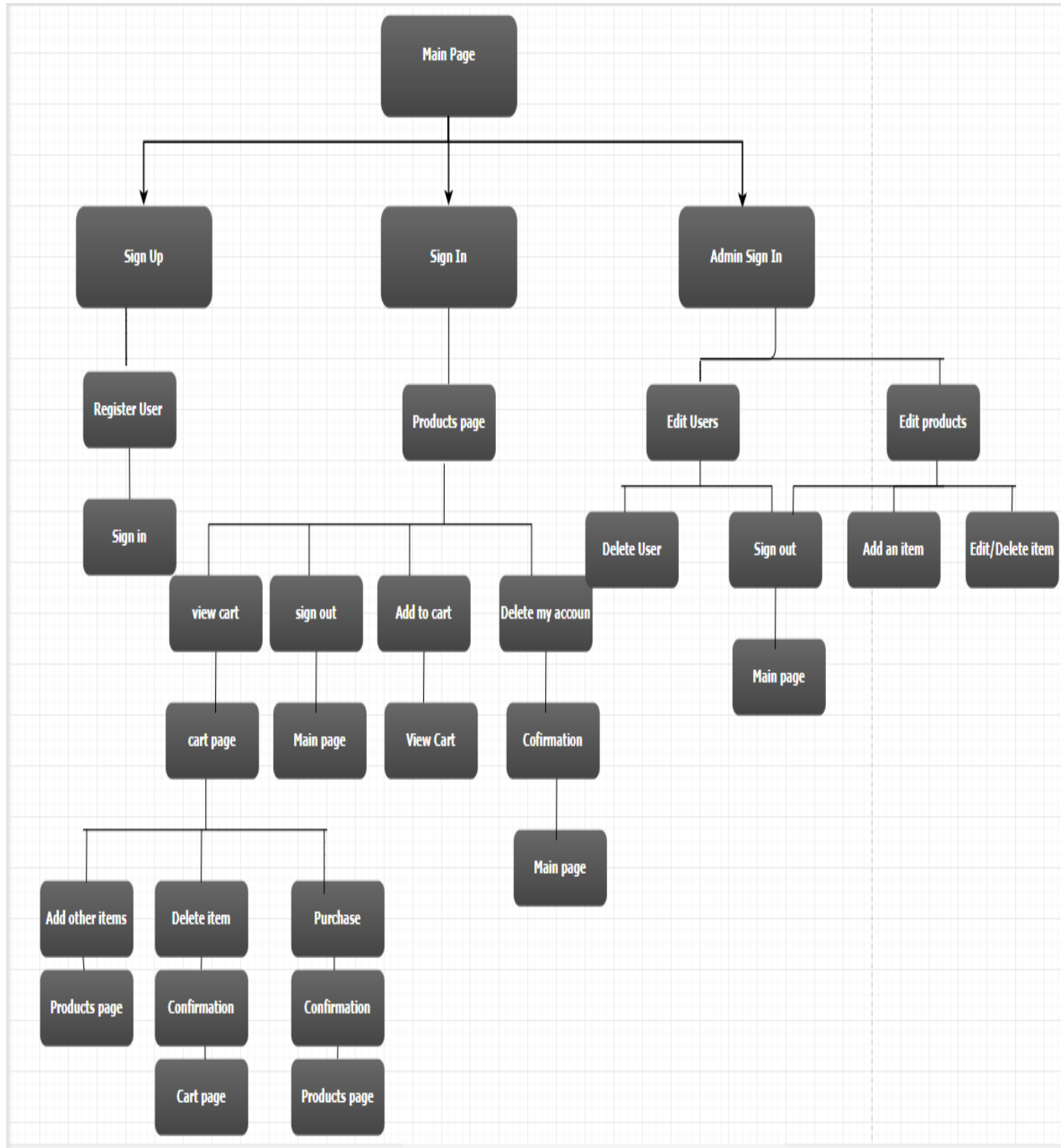
The complete online store is modelled in two virtual machines VM\_WWW, VM\_DB for front end and backend respectively. The machine hosting the two VMs is the client and it makes use of REST API to send requests to the frontend VM which is hosting web server application. The frontend will then make use of backend VM database to process the request. The entire model is represented in the following figure.





## Functional Decomposition Diagram:

The online store web application is represented in the Functional Decomposition diagram as shown in the figure.



## Database Schema:

In this section, the basic structure of the tables composing the database for the project are shown along with information.

### **Products table**

Field	Type	Nul l	Key	Default	Extra
Id	INT (11)	No	PRI	NULL	Auto_increment
Artist	VARCHAR (30)	Yes		NULL	
Album	VARCHAR (30)	Yes		NULL	
Date	DATE	Yes		NULL	
Producer	VARCHAR (30)	Yes		NULL	
Price	INT (30)	Yes		NULL	
Quantity	INT (30)	Yes		NULL	

### **Cart Table**

Field	Type	Nul l	Key	Default	Extra
Id	INT (11)	No	PRI	NULL	Auto_increment

Artist	VARCHAR (30)	Yes		NULL	
Album	VARCHAR (30)	Yes		NULL	
Date	DATE	Yes		NULL	
Producer	VARCHAR (30)	Yes		NULL	
Price	INT (30)	Yes		NULL	

### **Users Table**

Field	Type	Nul l	Key	Default	Extra
Id	INT (11)	No	PRI	NULL	Auto_increment
Username	VARCHAR (20)	Yes		NULL	
Password	VARCHAR (30)	Yes		NULL	
Confirm password	VARCHAR (30)	Yes		NULL	

# Implementation

## Communication settings for VM\_WWW and VM\_DB

The two virtual machines VM\_WWW and VM\_DB are hosted using Oracle VM virtual box manager. A new host only network is created with provided network prefix (192.168.100.33/24) is added in the host network manager. Both VM\_WWW and VM\_DB are configured in the network preferences with newly added host only network adapter along with the NAT connection with host. The IP addresses of VM\_WWW and VM\_DB are 192.168.100.35 and 192.168.100.36 respectively. The frontend VM\_WWW is exposed to host via port forwarding with NAT.

## Database connection between VM\_WWW and VM\_DB

The backend MySQL database is configured to provide its remote access to the VM\_WWW. The MySQL is configured in the following ways for the remote access.

- 1 A user with the IP address of the client is added to the mysql.user table.
- 2 Grant all privileges to created user which wants to access this database remotely is performed.
- 3 Flush privileges and flush hosts.
- 4 The bind address is bound to the ip address of the remote device in the MySQL configuration file and the service is restarted.

## VM\_DB Database

The backend database is implemented using MySQL database. There are three tables inserted for storing products, cart items and user accounts. A remote MySQL connection is established between the frontend VM\_WWW and VM\_DB. For a client program to be able to connect to the MySQL server, it must use the proper connection parameters, such as the name of the host where the server is running and the user name and password of MySQL account.

## VM\_WWW Frontend

The frontend web server application is implemented using PHP and HTML files. PHP: Hypertext Pre-processor is a server-side scripting language designed for web development but also used as a general-purpose programming language. Hypertext Mark-up Language (HTML) is the standard mark-up language for creating web pages and web applications.

The main page is implemented with to provide d with signup, sign in, admin login tabs for user registration, user login and admin login respectively. The created user accounts are inserted into database. Then when the user logs, the product page fetches data from the database and lists the items in the music store. And when the user adds items to the cart, the cart items are inserted to the database. When the user deletes item from the cart then the item is deleted from database. When the user performs the purchase then the quantity is updated in the database. The admin can edit user accounts and edit product information in the edit user and edit product page and the action is performed in the database. Thus, the database is continuously

fetches, inserts, updates and deletes is implemented in PHP source code.

## Implementation of Rest API functions

The Rest API functions are implemented using PHP. Whenever the client performs the action towards the web application Rest API functions GET, POST, PUT and Delete are used to retrieve, post, update and delete the information. The whole application is implemented following the functional decomposition diagram. The PHP server sends the response when the client performs some action.

### GET Operation:

When the client performs get information after sign in. The list of products information is received from the server, and the http response is sent back to the client. The signed in user information is always carried using the GET operation and the response is continuously sent back to the client. The example URL for GET method is listed below.

GET URL: <http://127.0.0.1:2222/products.php?id=16>

### POST Operation:

Whenever the client provides the username and password the POST operation is performed, and the information is posted to the server and the response is sent back to the client. The example URL for POST method is listed below.

POST URL: <http://127.0.0.1:2222/signin.php>

## PUT Operation:

Whenever the client wants to edit the information in the server the PUT operation is performed and the information is updated in the server and the response is sent back to the client. The example URL for the PUT operation is listed below.

PUT URL: <http://127.0.0.1:2222/edit.php?id=4>

## DELETE Operation:

Whenever the client wants to delete information in the server the DELETE operation is performed, and the information is deleted in the server and the response is sent back to the client. The example URL for the DELETE operation is listed below.

DELETE URL: <http://127.0.0.1:2222/delete.php?id=2&user=16>

## Testing

Test T1:

Purpose	To test implementation of virtual machines with suitable connections.
Requirement	R1
Operation	The virtual machines are communicating with Host only network and the frontend VM provides its access to client via port forwarding.

Expected Result	The Host only adapter settings for the VMs is tested and verified in the network preferences. The NAT configuration with port forwarding is tested and verified in the network preferences. The IP configuration is tested inside the two VMs.
-----------------	--

### Test T2:

Purpose	To test the Database connection between frontend and backend.
Requirement	R2
Operation	The frontend should not keep any store information. It should continuously fetch, insert, update and delete information remotely from backend.
Expected Result	The MySQL configuration file is tested and bind address is verified. The insert operation is performed, and the backend database is verified.

### Test T3:

Purpose	To test the User registration
Requirement	R3



Operation	The user registration should perform creation of account for the user if the user provides valid id and matching passwords.
Expected Result	By creating the user account and signing with the created account the user registration is verified. The mismatched passwords are provided, and the error message is verified.

#### Test T4:

Purpose	To test the user login.
Requirement	R4
Operation	The users having successful registration can login to the music store.
Expected Result	By signing in to the music store with the created account the user registration is verified. The wrong password and unknown username are provided, and the error message is verified.

#### Test T5:

Purpose	To test the admin login.
Requirement	R5

Operation	The admin should have a account for managing users and products information.
Expected Result	By signing in to the admin page with valid credentials is tested and verified. The wrong password and unknown username are provided, and the error message is verified.

#### Test T6:

Purpose	To test the listing of items and properties in the music store after successful login.
Requirement	R6
Operation	The users can view the list of available products after successful sign in.
Expected Result	By providing the correct credentials, the list of products is verified.

#### Test T7:

Purpose	To test adding items to the cart
Requirement	R7
Operation	When ever the user performs the add to car option then the item should be added to the cart.

Expected Result	Add an item to the cart using the add button and the item added is verified in the cart page.
-----------------	---

#### Test T8:

Purpose	To test the limited quantity error.
Requirement	R8
Operation	Whenever the user tries to add the item which is having zero quantity, then the error message should be displayed.
Expected Result	Add an item to the cart which is having 0 quantity and the error message is verified.

#### Test T9:

Purpose	To test listing items in the cart.
Requirement	R9
Operation	The cart page should list the items and their properties which are added by the user.
Expected Result	Add some items to the cart and cart page is visited to verify if the cart items are displayed.

#### Test T10:

Purpose	To test removing item from the cart.
Requirement	R10
Operation	Whenever the user performs remove option beside individual item in the cart, the item should be deleted from the cart and added back to the products.
Expected Result	By deleting the cart item and visiting cart page and products page, the quantity of that product is verified.

#### Test T11:

Purpose	To test the total amount of the purchase
Requirement	R11
Operation	The sum of the prices of the cart items is displayed beside the total price tab.
Expected Result	Add some items to the cart and count the prices of the items in the cart page manually and the total amount is verified.

#### Test T12:

Purpose	To test the purchase of products
Requirement	R12

Operation	Whenever the user performs purchase option then the success message should be displayed and the list of items in the cart page should be deleted.
Expected Result	Make the purchase and visit the cart page to verify.

#### Test T13:

Purpose	To test delete my account
Requirement	R13
Operation	Whenever the user deete my account option in the products page, then the user account should be deleted.
Expected Result	Perform delete my account, then the users database is verified if the user account is deleted.

#### Test T14:

Purpose	To test the admin page for editing users.
Requirement	R14
Operation	The admin page can manage user accounts in the edit users page. The admin can delete any user account.

Expected Result	<b>Try deleting an account form the edit users page and the user database is visited to verify if the user account is deleted.</b>
-----------------	--

#### Test T15:

Purpose	To test the admin page for editing products.
Requirement	R15
Operation	The admin can information of the new product. The admin can edit the information of the existing product and the admin can delete the information of any product.
Expected Result	Try editing the information of any product and check the products database to verify if the information is updated. Try deleting any product and check the products database to verify if the item is deleted from it. Try adding new product and check the products database to verify if the new information is added.

#### Test T16:

Purpose	To test the information of the products table in My SQL database in backend.
Requirement	R16

Operation	The My SQL database in the backend VM should consists of a table which has list of products.
Expected Result	Login to the MY SQL database and verify the products table. .

#### Test T17:

Purpose	To test the information of the users table in My SQL database in backend.
Requirement	R17
Operation	The My SQL database in the backend VM should consists of a table which has list of users.
Expected Result	Login to the MY SQL database and verify the users table. .

#### Test T18:

Purpose	To test the information of the cart table in My SQL database in backend.
Requirement	R18
Operation	The My SQL database in the backend VM should consists of a table which has list of cart items.

Expected Result	Login to the MY SQL database and verify the cart table. .
-----------------	---

#### Test T19:

Purpose	To test the rest api methods.
Requirement	R19
Operation	The rest api methods GET, PUT, POST and DELETE are performed to send retrieve, post, update and delete requests to the front end from the client.
Expected Result	The wireshark is used to verify the GET, PUT , POST and DELETE metods between the client ad the frontend server.

## Developer Documentation:

### Communication settings for VM\_WWW and VM\_DB:

In oracle virtual box, a new host only adaptor is added with the network prefix 192.168.100.33/24 inside the host network manager. The network settings of the two VMs are edited and the two VMs are attached to the created host only network adaptor. The frontend VM's network configuration is edit and the second interface is attached to the NAT network for external communication and port forwarding is enabled providing the ports and IPs of host and VM.



## Database connection between VM\_WWW and VM\_DB

The remote access configuration of My SQL databases involves the following commands in the backend My SQL.

```
Create USER 'vmwww' @ '192.168.100.35' Identified by '<password>;'
```

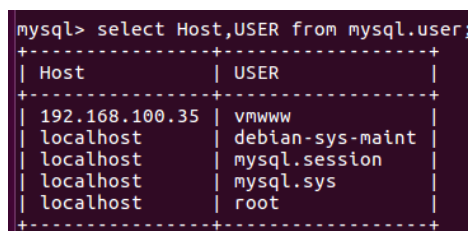
```
GRANT ALL PRIVILEGES ON *.* TO 'vmwww' @ '192.168.100.35' with GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

```
FLUSH HOSTS;
```

Now, verify the added user using the following command.

```
SELECT Host,USER from mysql.user;
```



```
mysql> select Host,USER from mysql.user;
```

Host	USER
192.168.100.35	vmwww
localhost	debian-sys-maint
localhost	mysql.session
localhost	mysql.sys
localhost	root

Add the following line in the mysql configuration file in /etc/mysql/mysql.conf.d/mysqld.cnf

Bind-adress = 0.0.0.0. (listens universal clients).

Restart the mysql service.

### config.php:

The PHP-MySQL connection is represented in config.php file. It can be included in all other files where the database connection is required.

### Main.html:

The html layout of main page of the music store is implemented here. The signup, sign in and sign in as admin are provided with html reference tag and their actions are forwarded to signup.html, signin.html, admin.html.

### Signup.html:

The html layout for sign up is implemented with three boxes for providing username, password, and confirm password. The submit button is provided and its action is specified to singup.php and the post method is used.

### Signup.php:

The username, password and confirm password which are posted earlier via post method are stored in variables. Now testing if the provided values are empty and the error message is displayed. If the passwords do not match, then the error message is displayed. Selecting the users in the users database and if the user already exists, then the error message is displayed. If there are no errors the credentials are stored in the users database using insert command.

### Signin.html:

The html layout for sign in is implemented with two boxes for providing username, and password. The submit button is provided and its action is specified to singin.php and the post method is used.

### Signin.php:

The username, and password which are posted earlier via post method are stored in variables. Now testing if the provided values are empty and the error message is displayed. Selecting the users in the user's database and matching the column with the password. If the password matches, then the id is carried to the products.php which represents list of products. If the password does not match, then the error message will be displayed.

### Products.php:

It carries the id of the user and views list of products. The list of products is selected from the database and displayed with html layout. And Add to cart box is created with its action is forwarded to add.php. A view cart tab is created with its action is forwarded to cart.php. A delete my account tab is created with its action is forwarded to acdel.php. A sign out tab is created, and its action is forwarded to del.php.

### Add.php:

An id of the selected item is carried to add.php. Here, the items quantity is selected from the database. The quantity is compared with zero. If the quantity is less than or equal to 0 then the error message will be displayed, and the item is not added to the car. Else the item is added to the cart and the success message is printed. A view cart tab is created with its action is forwarded to cart.php. A view products tab is created with its action is forwarded to products.php. When ever the item is added to the

cart the item is removed from the database i.e. the item quantity is reduced.

### Cart.php:

The id of the signed in user is carried to cart.php. In the cart page the items which are added to the cart are selected from the cart database and displayed using html layout. An 'add other items' tab is created with its action is forwarded to products.php. A 'delete' tab is created with its action is forwarded to delete.php. A purchase tab is created, and its action is forwarded to purchase.php. The sum of the prices in the price column of the cart database is displayed beside the total.

### Delete.php:

The id of the selected item is carried to the delete.php. The item is selected from the cart database and matched with the item in the products database and the quantity of the item is updated(incremented). Then the item is removed from the cart database. The http delete method is performed here. A view cart tab is created, and its action is forwarded to cart.php.

### Purchase.php:

The id of the signed in user is carried to the purchase.php. In purchase.php after the confirmation the success message is displayed, and the items are removed from the cart database. And the location header is set to products.php.

### Acdel.php:

The id of the signed in user is carried to Acdel.php. Then the row corresponding to the id is deleted from the user's database. The

http delete operation is performed here. The location header is set to main.html.

### Del.php:

The item is selected from the cart database and matched with the item in the products database and the quantity of the item is updated(incremented). Then the item is removed from the cart database. The http delete method is performed here. The location header is forwarded to main.html.

### Admin.html:

The html layout for sign in is implemented with two boxes for providing username, and password. The submit button is provided and its action is specified to admin.php and the post method is used.

### Admin.php:

The username, and password which are posted earlier via post method are stored in variables. Now testing if the provided values are empty and the error message is displayed. The username is matched with 'saaish' and the password is matched with 'saaish'. If the matches are successful then items edit users and edit products tabs are created and their actions are forwarded to users.php and list.php respectively.

### Users.php:

The user items are selected from the database and displayed in the edit users page. A delete tab is created beside every user account and its action is forwarded to minus.php. A sign out tab

is created, and its action is forwarded to main.html. An edit products tab is created, and its action is forwarded to list.php.

### Minus.php:

The id of the selected user is carried to minus.php. Then the row corresponding to the id is deleted from the user's database. The http delete operation is performed here. The location header is set to users.php.

### List.php:

The list of products is selected from the database and displayed with html layout. Add new item is created with its action is forwarded to plus.html. A sign out tab is created, and its action is forwarded to main.html. An edit tab is created beside every item and its action is forwarded to edit.php. Here when the edit action is performed, each column of the selected row of item are stored in the variables and forwarded to edit.php. A delete tab is created beside each item and its action is forwarded to remove.php.

### Plus.html:

The html layout for adding new item is implemented with six boxes for adding artist, album, producer, release date, price and quantity. The submit button is provided and its action is specified to plus.php and the post method is used.

### Plus.php:

The item properties which are posted earlier via post method are stored in variables. Now testing if the provided values are empty and the error message is displayed. If there are no errors the new

item properties are stored in the products database using insert command.

### Edit.php:

The items which are forwarded from the list.php are stored in the variables and displayed by six boxes and provided the admin to edit them using the html layout. After entering the information and after performing the update button. The new values are stored in new variables and verified if they are non-empty. If either of the any variable is empty, then the error message is displayed. Else the values are updated in the products database. Then the location header is forwarded to list.php.

### Remove.php:

The id of the selected item is carried to remove.php. The item corresponding to the carried id is deleted from the products database. The location header is forwarded to list.php.