Problem statement for E-commerce Application on IBM Cloud Foundry

Project Title: E-commerce App

Cloud Application Development Phase 1: Problem Definition and Design Thinking

Problem Statement:

Build an artisanal e-commerce platform using IBM Cloud Foundry. Connect skilled artisans with a global audience. Showcase handmade products, from exquisite jewelry to artistic home decor. Implement secure shopping carts, smooth payment gateways, and an intuitive checkout process. Nurture creativity and support small businesses through an artisan's dream marketplace!

Problem Definition:

The project is to build an artisanal e-commerce platform using IBM Cloud Foundry. The goal is to connect skilled artisans with a global audience, showcasing their handmade products and providing features like secure shopping carts, payment gateways, and an intuitive checkout process. This involves designing the e-commerce platform, implementing necessary features, and ensuring a seamless user experience.

Design Thinking:

Platform Design: Design the platform layout with sections for product categories, individual product pages, shopping cart, checkout, and payment.

Product Showcase: Create a database to store product information such as images, descriptions, prices, and categories.

User Authentication: Implement user registration and authentication features to enable artisans and customers to access the platform.

Shopping Cart and Checkout: Design and develop the shopping cart functionality and a smooth checkout process.

Payment Integration: Integrate secure payment gateways to facilitate transactions.

User Experience: Focus on providing an intuitive and visually appealing user experience for both artisans and customers.

Key findings:

- **There is a need to include a clear CTA button at the checkout page that allows the users to checkout as a guest.**

- **Participants really appreciated the customisation process on the website.**
- **Participants would really appreciate if there was a dedicated section for "offers/sale" on the website .**
- **Participants really appreciated the detailed information about the clothing item that they like.**

Problems of E-Commerce Website Design:

- **Making the Consumer Trust Your Website.**
- **Ensuring a Hassle-Free Payment System.**
- **Keeping Track Of Your Stock.**
- **Ensuring Smooth Shipping Experience.**
- **A Simple UI/UX Design Layout.**
- **Making Sure that the Products look visually appealing.**

Innovation
Phase 2

E commerce application on IBM Cloud Foundry

1.0 INTRODUCTION:

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.

The objective of this project is to develop a general purpose e-commerce store where product like clothes can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online shopping for clothes.

An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as credit card number. An e-mail notification is sent to the customer as soon as the order is placed.

2.0 OVERALL DESCRIPTION:

2.1Description:

- ➢ Any member can register and view available products.
- ➢ Only registered member can purchase multiple products regardless of quantity.
- ➢ ContactUs page is available to contact Admin for queries.
- ➢ There are three roles available: Visitor, User and Admin.
  - Visitor can view available products.
  - User can view and purchase products.
  - An Admin has some extra privilege including all privilege of visitor and user.
    - ✓ Admin can add products, edit product information and add/remove product.
    - ✓ Admin can add user, edit user information and can remove user.
    - ✓ Admin can ship order to user based on order placed by sending confirmation mail.

2.2Using the code:

1. Attach the database in your "SQL Server Management Studio Express".
2. Run the application on Microsoft Visual Studio as web site.
3. Locate the database.

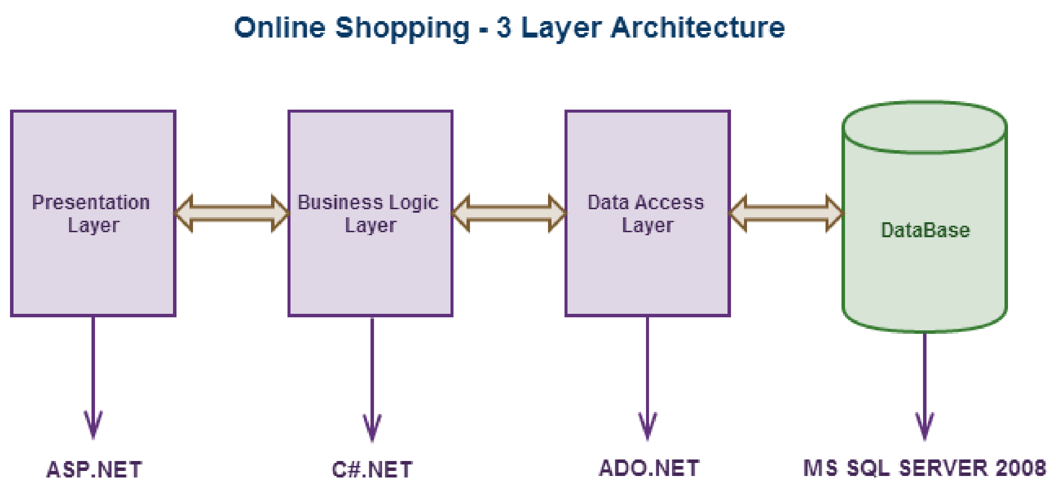
2.3MasterPage details:

- OnlineShopping Master Page (Similar MasterPage for Visitor, User and Admin)

2.4Web Pages details:

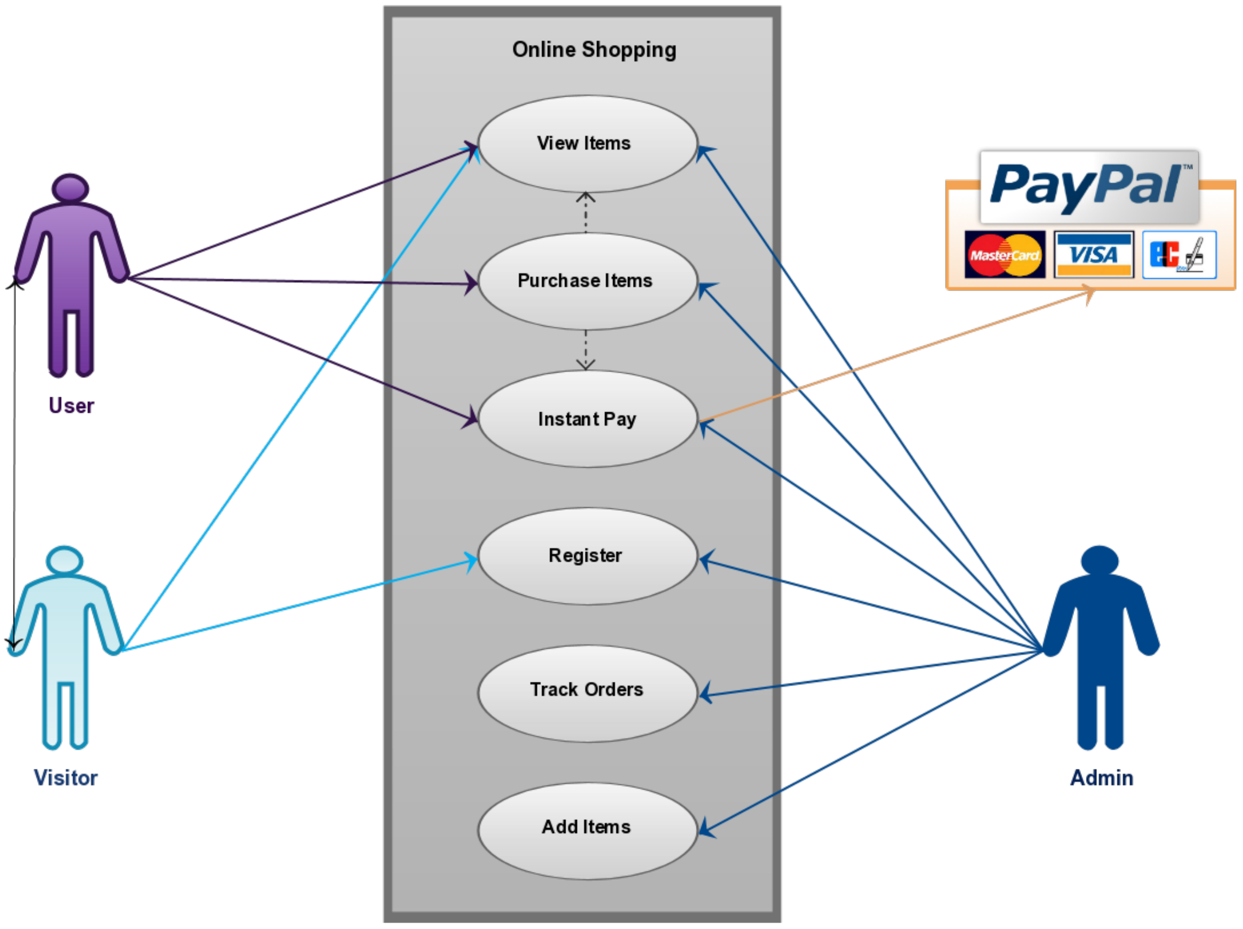
- Home Page
- AboutUs Page
- Clothing Page
- OrderUs Page
- ContactUs Page
- Admin Page
- Login Page
- Register Page
- Track

2.5Project Detail:



3.0 SYSTEM REQUREMENTS:

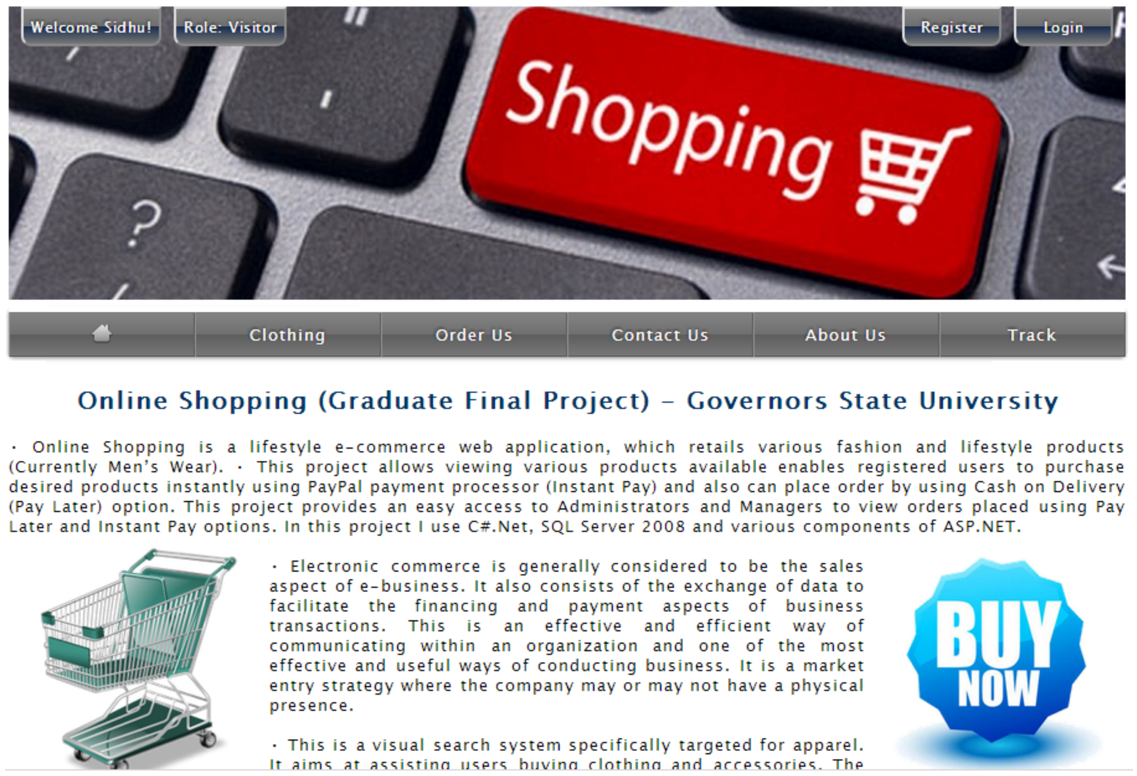
3.1 USE-CASE DIAGRAM:



4.0 ONLINE SHOPPING APPLICATION:

Anyone can view Online Shopping portal and available products, but every user must login by his/her Username and password in order to purchase or order products. Unregistered members can register by navigating to registration page. Only Admin will have access to modify roles, by default developer can only be an 'Admin'. Once user register site, his default role will be 'User'.

4.1 HOMEPAGE: The Home Screen will consist of screen were one can browse through the products which we have on our website



Online Shopping (Graduate Final Project) – Governors State University

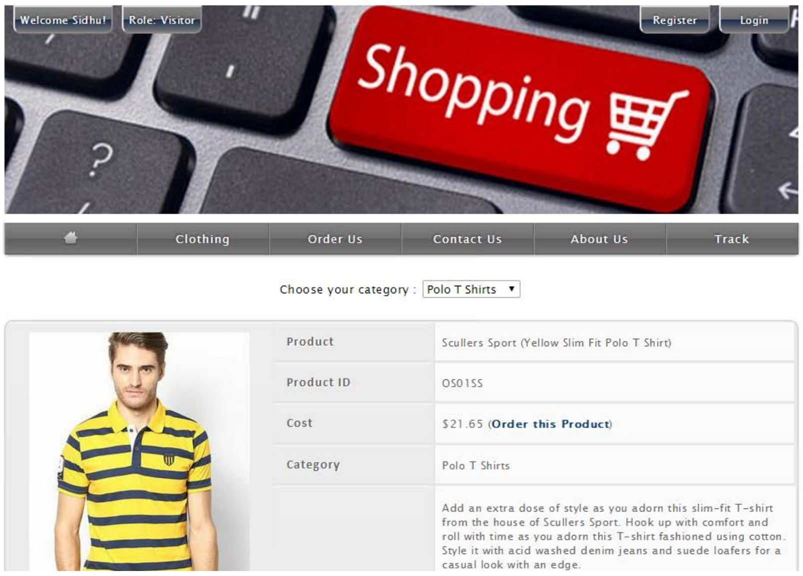
· Online Shopping is a lifestyle e-commerce web application, which retails various fashion and lifestyle products (Currently Men's Wear). · This project allows viewing various products available enables registered users to purchase desired products instantly using PayPal payment processor (Instant Pay) and also can place order by using Cash on Delivery (Pay Later) option. This project provides an easy access to Administrators and Managers to view orders placed using Pay Later and Instant Pay options. In this project I use C#.Net, SQL Server 2008 and various components of ASP.NET.

· Electronic commerce is generally considered to be the sales aspect of e-business. It also consists of the exchange of data to facilitate the financing and payment aspects of business transactions. This is an effective and efficient way of communicating within an organization and one of the most effective and useful ways of conducting business. It is a market entry strategy where the company may or may not have a physical presence.

· This is a visual search system specifically targeted for apparel. It aims at assisting users buying clothing and accessories. The

Figure1: Home Page

4.2. CLOTHING PAGE (PRODUCTS): This page consists of product details. This page appears same for both visitors and users.



Product	Scullers Sport (Yellow Slim Fit Polo T Shirt)
Product ID	OS01SS
Cost	$21.65 (**Order this Product**)
Category	Polo T Shirts
	Add an extra dose of style as you adorn this slim-fit T-shirt from the house of Scullers Sport. Hook up with comfort and roll with time as you adorn this T-shirt fashioned using cotton. Style it with acid washed denim jeans and suede loafers for a casual look with an edge.

Figure 2: Clothing Page

4.3 Order Us Page: Registered users can order desired products from here.

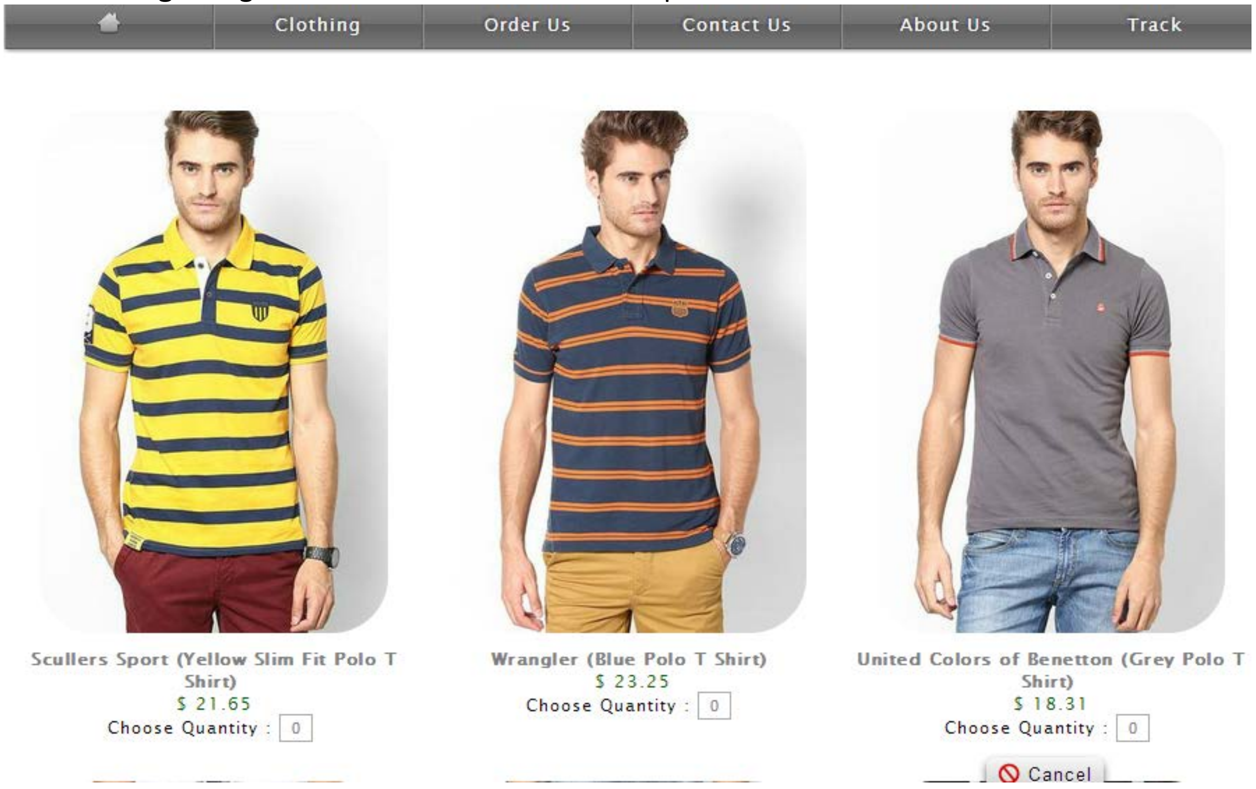


Figure 3: Order Us Page

4.4 Contact Us Page: Visitors and Registered users can contact website owners or administrators from here

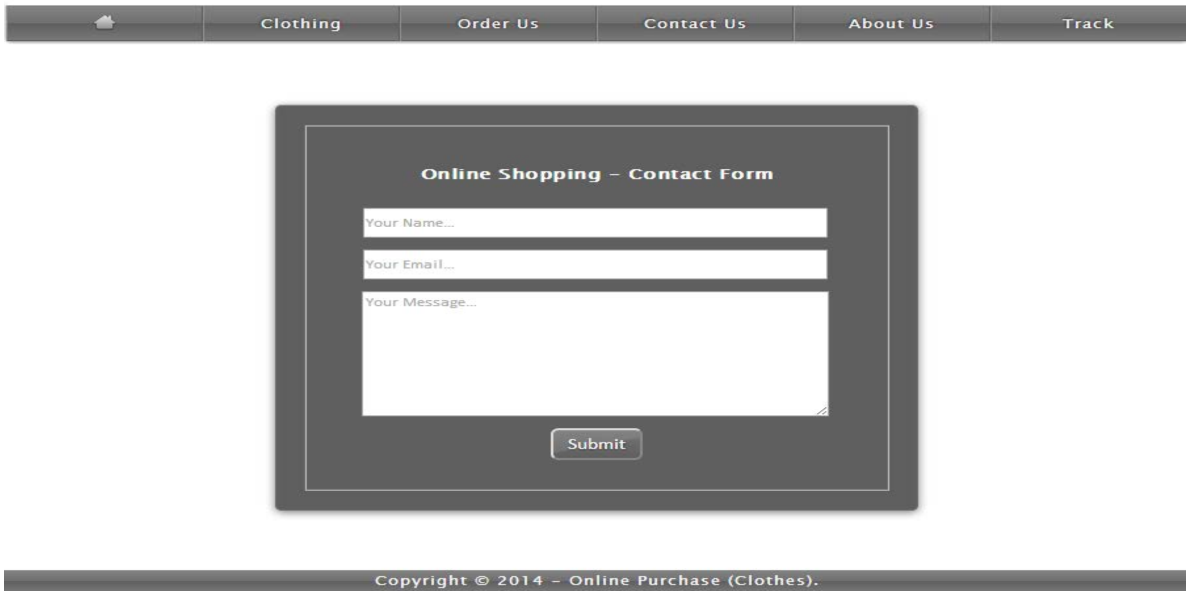
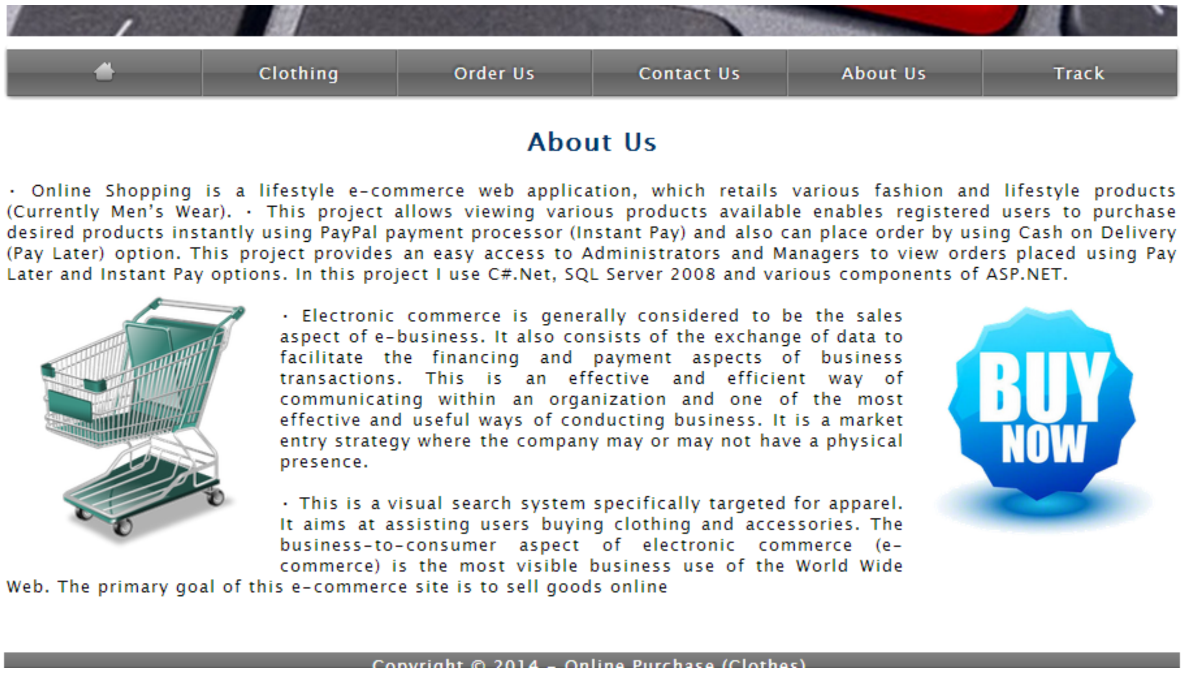


Figure 4: Contact Us Page

4.5 ABOUT US PAGE: This page describes about website and owners



About Us

· Online Shopping is a lifestyle e-commerce web application, which retails various fashion and lifestyle products (Currently Men's Wear). · This project allows viewing various products available enables registered users to purchase desired products instantly using PayPal payment processor (Instant Pay) and also can place order by using Cash on Delivery (Pay Later) option. This project provides an easy access to Administrators and Managers to view orders placed using Pay Later and Instant Pay options. In this project I use C#.Net, SQL Server 2008 and various components of ASP.NET.

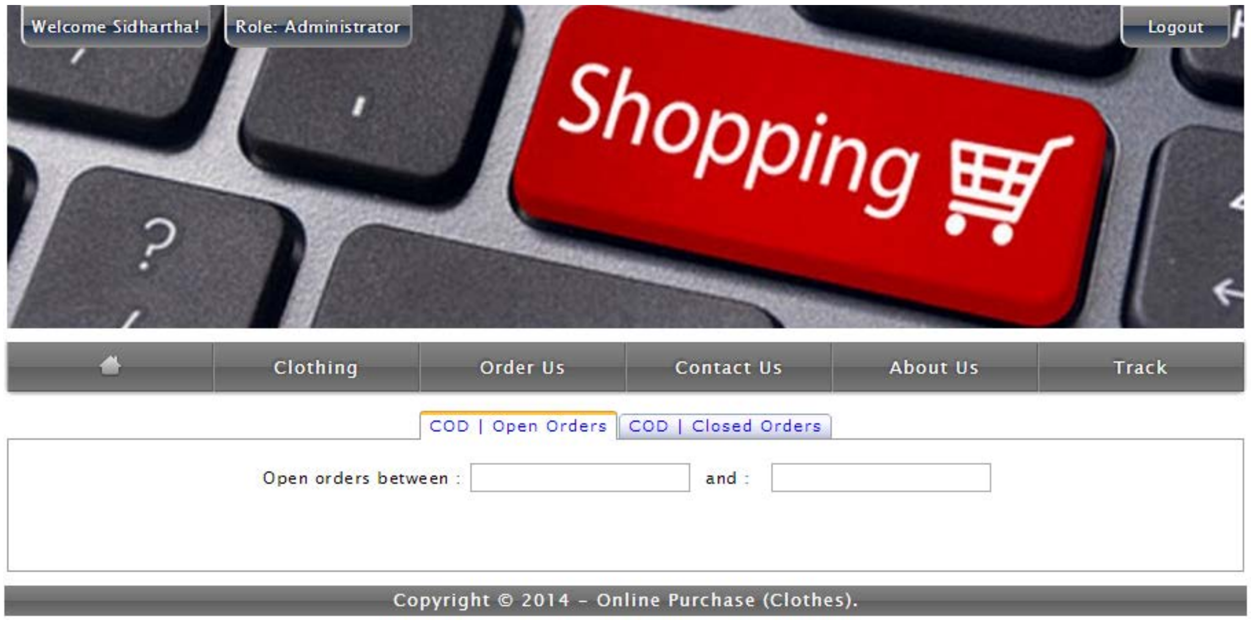
· Electronic commerce is generally considered to be the sales aspect of e-business. It also consists of the exchange of data to facilitate the financing and payment aspects of business transactions. This is an effective and efficient way of communicating within an organization and one of the most effective and useful ways of conducting business. It is a market entry strategy where the company may or may not have a physical presence.

· This is a visual search system specifically targeted for apparel. It aims at assisting users buying clothing and accessories. The business-to-consumer aspect of electronic commerce (e-commerce) is the most visible business use of the World Wide Web. The primary goal of this e-commerce site is to sell goods online

Copyright © 2014 – Online Purchase (Clothes).

Figure 5: About us Page

4.6 Track For Admin Page: Website Administrators can track and ship orders here.

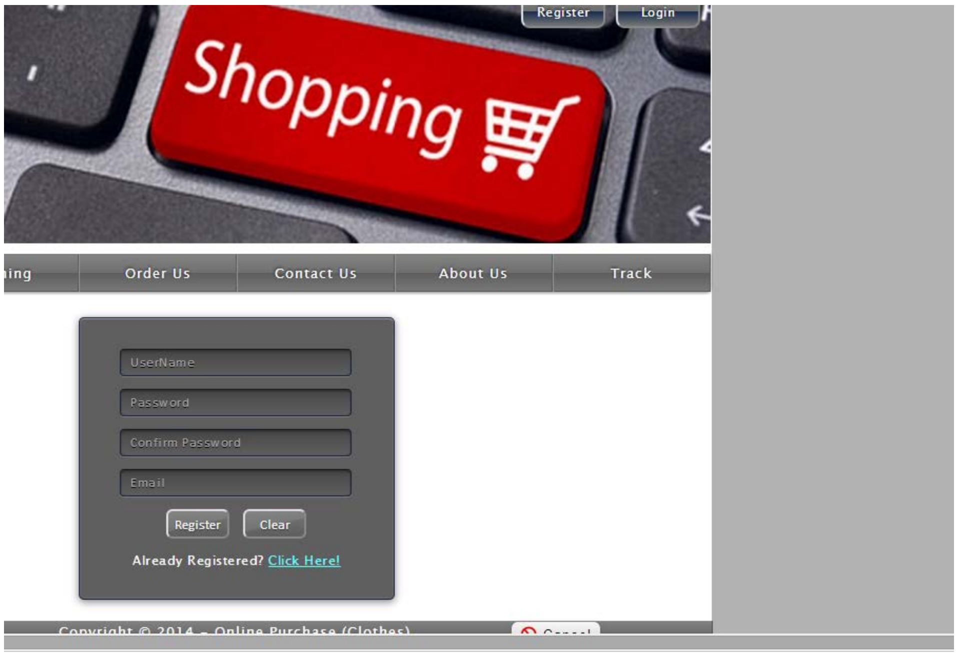


Open orders between : and :

Copyright © 2014 – Online Purchase (Clothes).

Figure 6: Tracking Page for Admin.

4.7 REGISTER PAGE: New users can register here

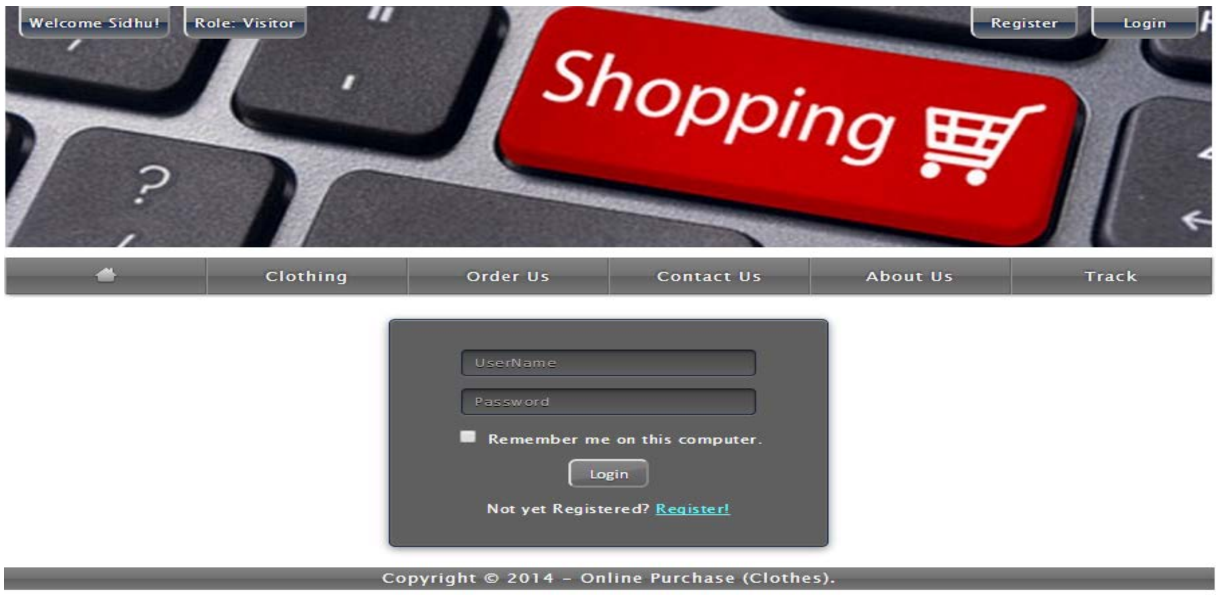


Already Registered? Click Here!

Copyright © 2014 – Online Purchase (Clothes)

Figure 7: Register Page

4.8 LOGIN PAGE: Login page for both users and administrators.



4.9 Admin Page: Only difference you see in this page is Role: Admin. User and Admin role will be checked once the page was login and Session [“role”] will be either Admin or User. If credentials belong to Admin then role will be Admin and if credentials belong to User then role will be User.

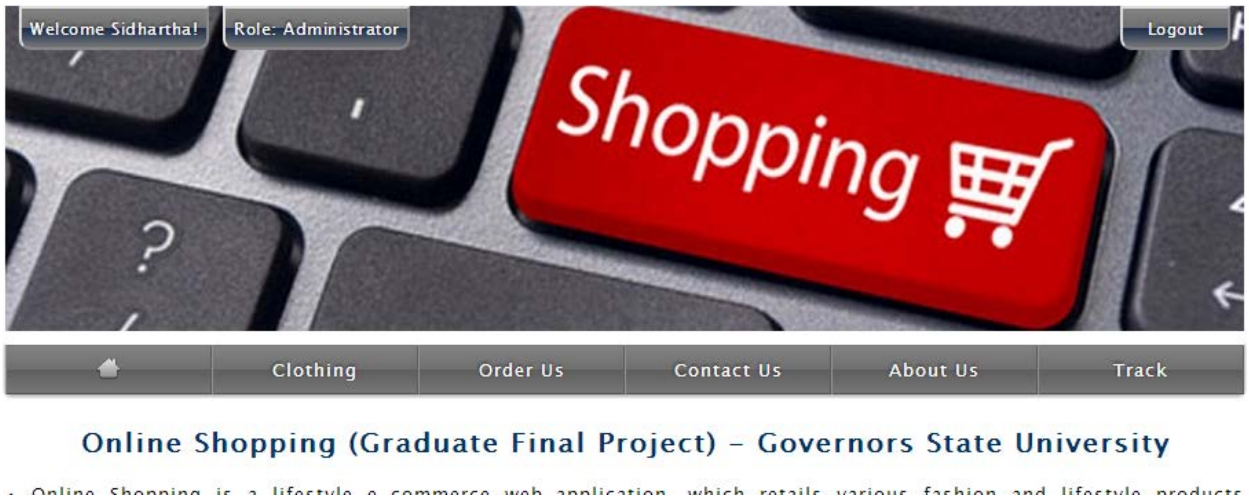
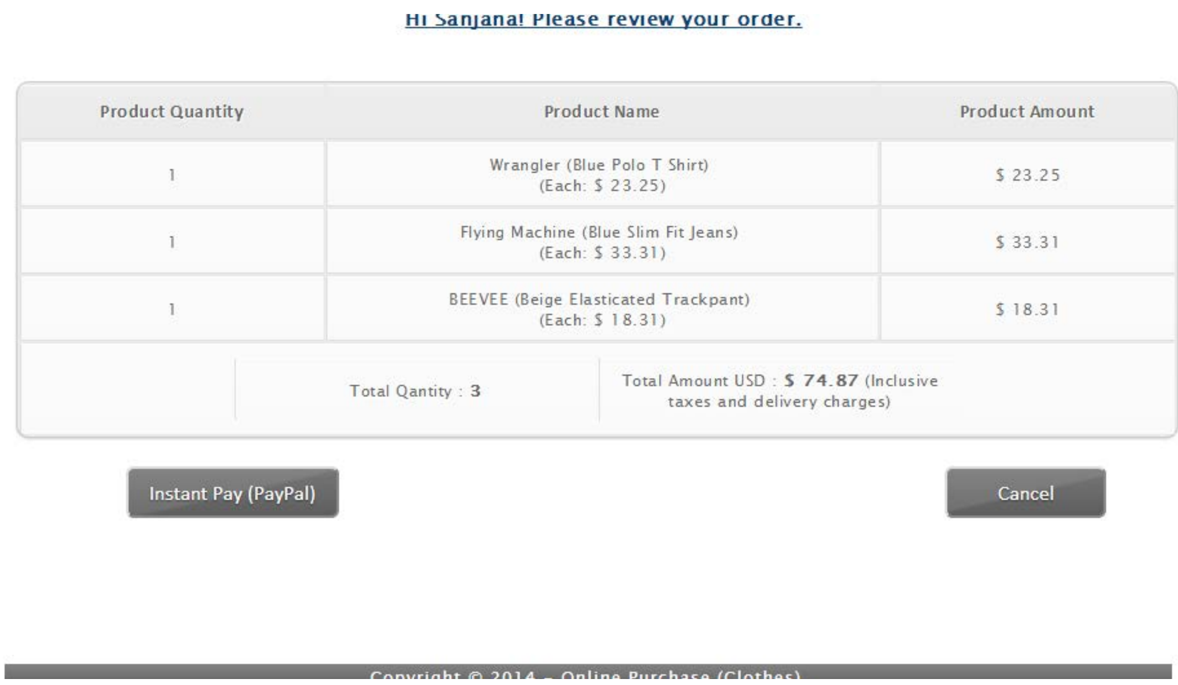


Figure 9: Admin Page

4.10 ORDER VIEW FOR USER: Once users order item they are able to see ordered products and grand total.

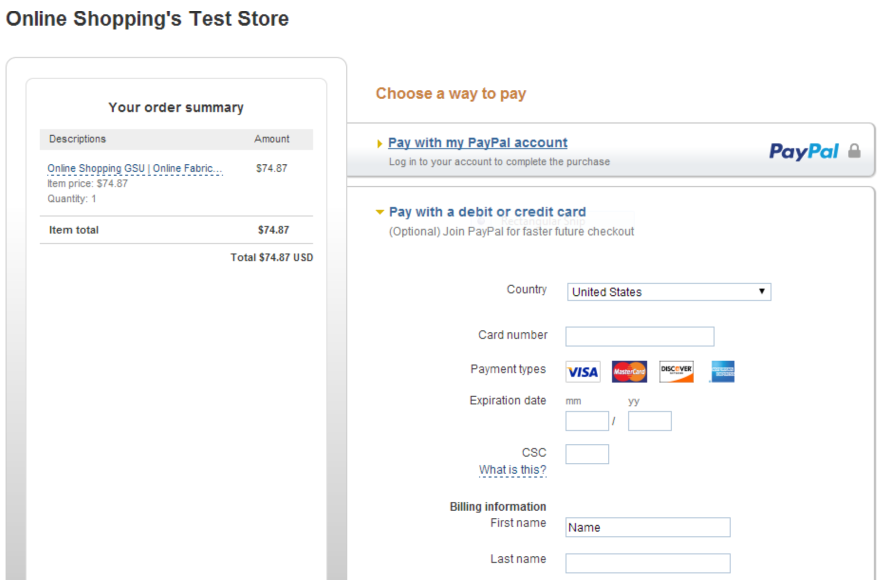


Hi Sanjana! Please review your order.

Product Quantity	Product Name	Product Amount
1	Wrangler (Blue Polo T Shirt) (Each: $ 23.25)	$ 23.25
1	Flying Machine (Blue Slim Fit Jeans) (Each: $ 33.31)	$ 33.31
1	BEEVEE (Beige Elasticated Trackpant) (Each: $ 18.31)	$ 18.31
	Total Qantity : 3	Total Amount USD : $ 74.87 (Inclusive taxes and delivery charges)

Figure 10: Order View for User

4.11 PAYPAL FOR PAYMENT: Once users orders products they are redirected to payment page.



Online Shopping's Test Store

Your order summary

Descriptions	Amount
Online Shopping GSU | Online Fabric... Item price: $74.87 Quantity: 1	$74.87
Item total	$74.87
	Total $74.87 USD

Choose a way to pay

Pay with my PayPal account
Log in to your account to complete the purchase

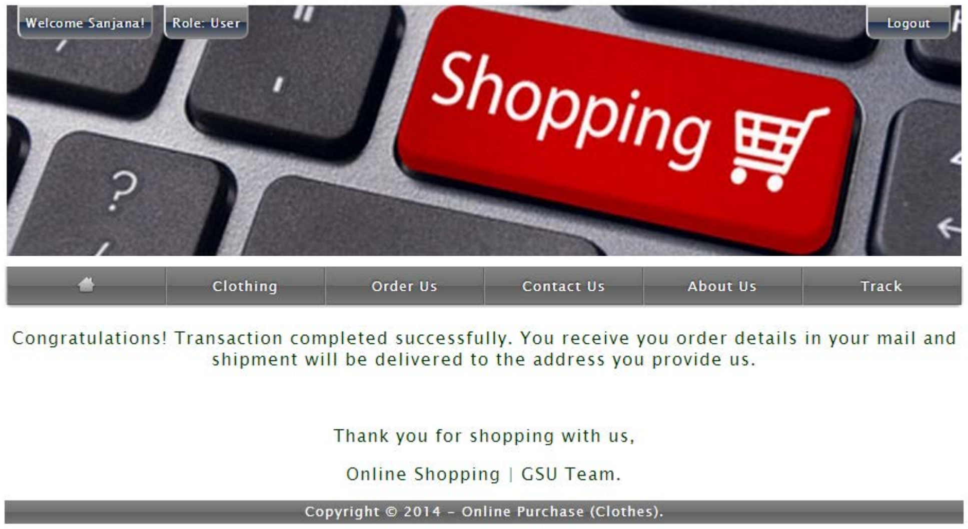
Pay with a debit or credit card
(Optional) Join PayPal for faster future checkout

Country	United States
Card number	
Payment types	
Expiration date	mm / yy
CSC What is this?	
Billing information First name	Name
Last name	

L

Figure 11: PayPal Page

Figure 4.12: Success URL



Congratulations! Transaction completed successfully. You receive you order details in your mail and shipment will be delivered to the address you provide us.

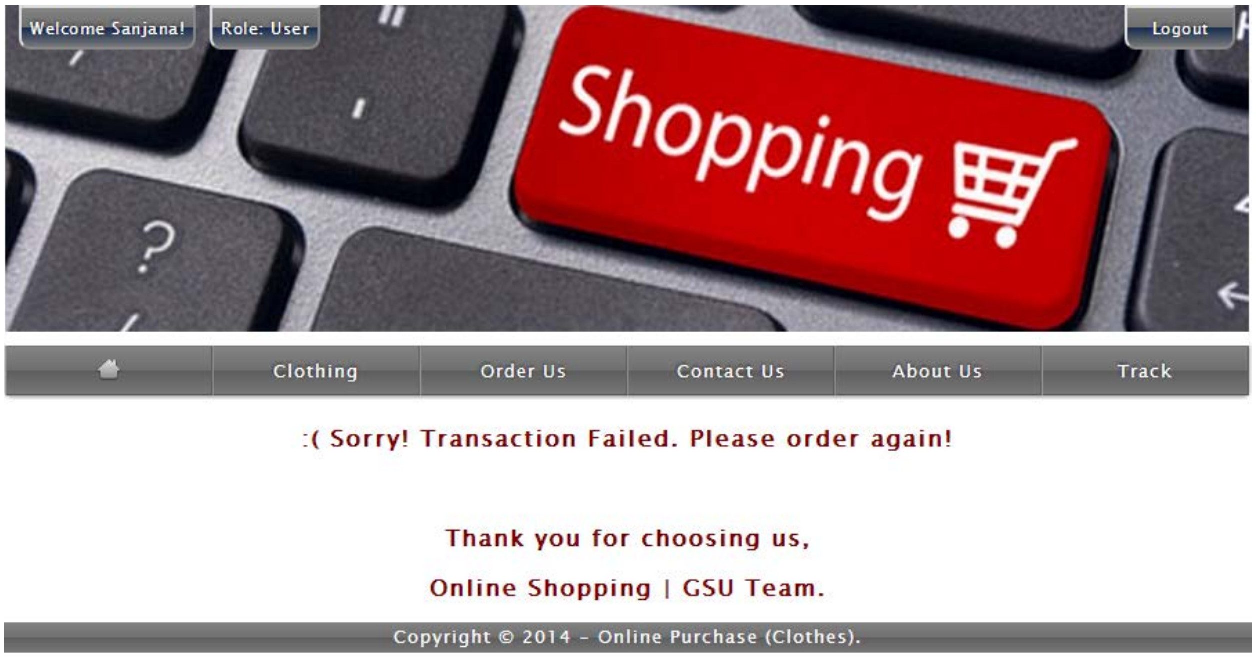
Thank you for shopping with us,

Online Shopping | GSU Team.

Copyright © 2014 – Online Purchase (Clothes).

Figure 12: Success URL

FIGURE 4.12: Failed URL



:(Sorry! Transaction Failed. Please order again!

Thank you for choosing us,

Online Shopping | GSU Team.

Copyright © 2014 – Online Purchase (Clothes).

Figure 12: Failed URL

5.0 Data Management

5.1 Data Description

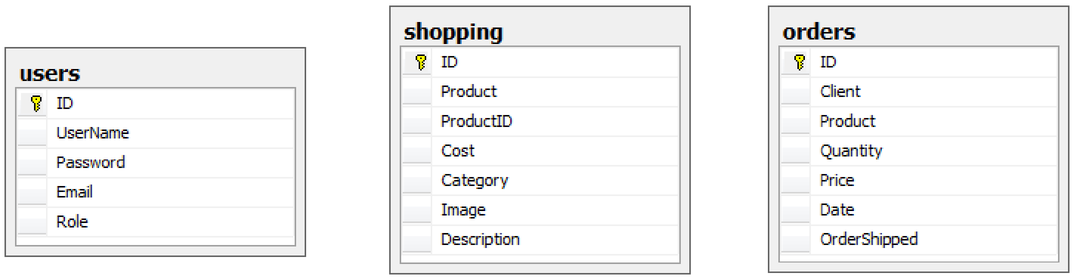
This database consists of

- Users: User and Admin information is added to database with Unique ID based on their roles.
- Shopping: Complete products information is stored in this table.
- Orders: Customer ordered products, status and delivery information is stored in this table.

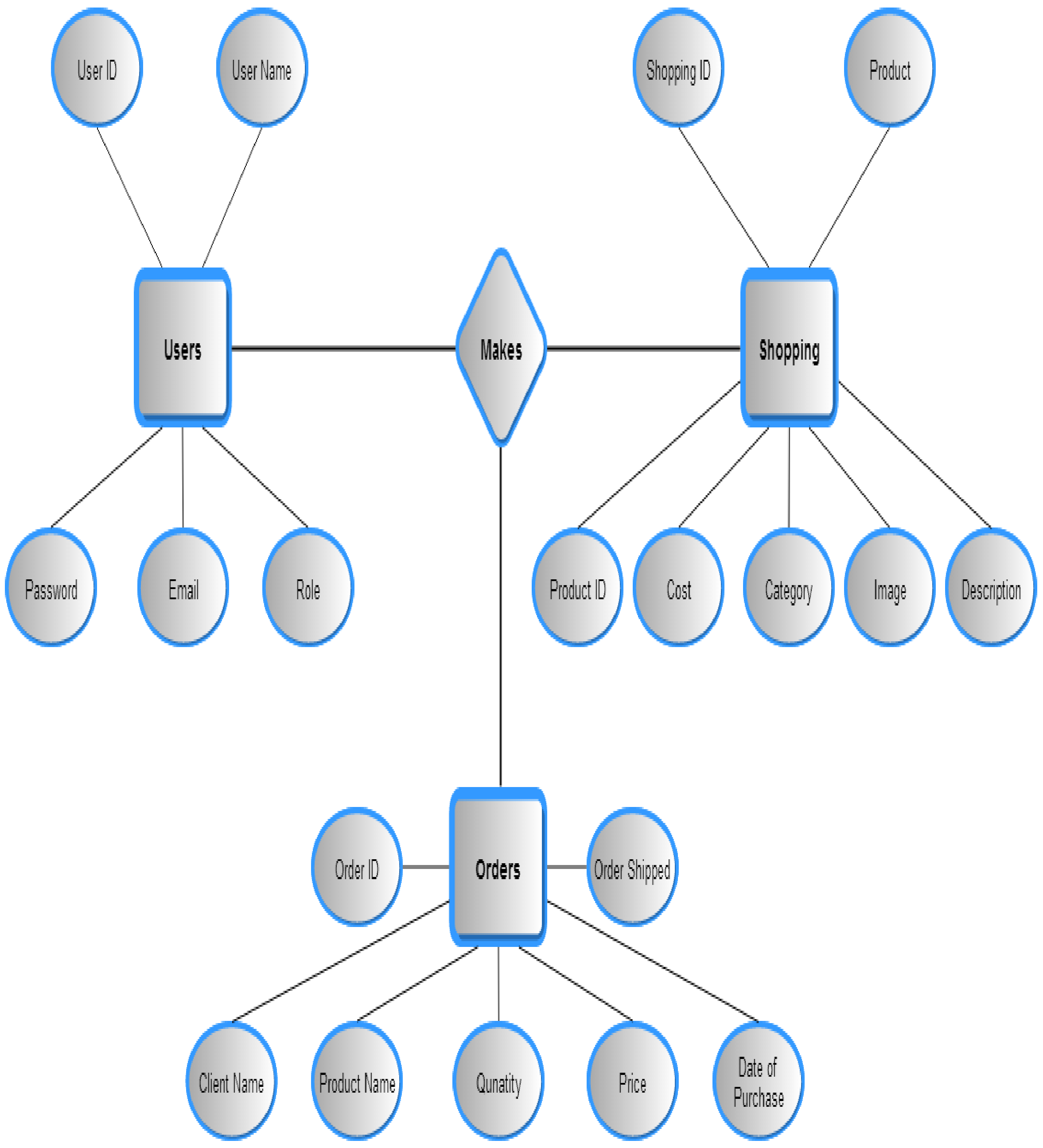
5.2 Data Objects

- User: ID, UserName, Password, Email, Role
- Shopping: ID, Product, Product ID, Cost, Category, Image, Description
- Orders: ID, Client, Product, Quantity, Price, Date, OrderShipped

5.3 Database Table Diagram



5.4 Relationships:



Phase 3

Cloud computing:

cloud computing has attracted a lot of attention. Cloud computing has developed quickly from a theoretical concept to the real applications in the past few years. More and more businesses and research agencies are striving to release cloud computing strategies and business models, perfect cloud computing technology, and propose the related applications of cloud computing. It enables the dynamic computing capacity, storage capacity, network exchanging capability and information service capability.Cloud computing serves the users as “pay-as-service”, which supplies and delivers the end users with IT services based on their demand. It departs the IT service processes and transfers them to the cloud platform, which leads to the new service modes such as IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). As a new information means and mode, cloud computing is being applied to many industries creatively. E-commerce is a typical industry which is being influenced inevitably by the features of cloud computing. This paper discusses the impacts of cloud computing on the traditional E-commerce respectively from the perspective of technology, service and industry chain and presents the necessary suggestions on the development of E-commerce businesses in the cloud era.

E-commerce technical architecture:

E-commerce is the exchange of products and services via Internet. From the perspective of system, it is composed of two layers: one layer is the technical architecture made up of hardware and software; the other layer is the business transactions based on the technical architecture. According to Laudon [6], the technical architecture is the base of E-commerce. And only on the base of the technical architecture, can the E-commerce business modes and marketing strategies be realized. In addition, the security and stability of technical architecture are the premise of online products and services exchange. Cloud computing, a new computing mode, is going to make a significant impact on E-commerce technical architecture.

cloud computing benefits the building and implementing of E-commerce technical architecture, the problems of system security and stability with it will be a problem non-neglectable. When all the IT resources such as hardware, software, data and network applications are stored on the cloud platform as services, users will unavoidably concern about the security and stability of the platform. Once the cloud platform is attacked, the important data of E-commerce transactions will be lost. Besides, customers’ privacy may become an obstacle for the cloud applications of E-commerce.

E-commerce backend service mode:

The new service mode offered by cloud computing differentiates it from the traditional IT services. Firstly, all the IT resources such as hardware, software, data and infrastructure are offered to the E-commerce enterprises as service by virtue of the cloud platform [18,19]. Secondly, just like the utility services (e.g. electricity), an E-commerce company is allowed to access to the IT resources on the cloud platform and pay for them as services [8].

It does not require the high expenses on devices purchase and each firm is able to choose the appropriate IT resources through renting. In another word, the emergence of cloud computing brings the new service philosophy and mode which enables the lower cost and changes the traditional IT licensing mode.

E-commerce business strategies:

Since the emergence of cloud computing, a lot of Ecommerce firms begin to expand their business to cloud computing. Some famous E-commerce businesses such as Amazon, Google and Alibaba have involved cloud computing in their long-term strategies.Several driving forces lead to the migration of cloud computing into E-commerce strategies: 1) Demand. With the rapid development of information technology, E-commerce services are improving—the services with higher efficiency, lower cost, more flexibility and diversity are needed. For instance, Alibaba, the biggest B2B E-commerce enterprise offers the online loan services by virtue of cloud computing. Alibaba loans the small and medium businesses its own idle capital from B2B transactions. When evaluating a customer's creditability, Alibaba implements the quick data analysis with cloud computing, which ensure the efficiency and effectiveness of creditability evaluation; 2) Efficiency. The efficiency advantages of cloud computing lies in two aspects: on the one hand, the huge data storage is becoming a problem with the growth of E-commerce firms. Establishing the data center is unaffordable for medium and small E-commerce enterprises.

E-commerce industry chain structure:

Cloud computing may influence the traditional E-commerce industry chain and lead to the chain restructuringTraditionally, the E-commerce industry chain is composed of the hardware supplier, software developerInternet service provider, system integrating provider, service supplier, E-commerce enterprise and customer (Figure 1). Each member of the industry chain fulfills its own functionalities. The hardware supplier, software developer, Internet service provider, system integration provider, service supplier exist as the backend of the E-commerce enterprise and offers it the technical support.When cloud computing is migrated into E-commerce industry, one cloud service provider can supply almost all the necessary products and services to an E-commerce website. As a result, the structure of E-commerce industry chain will be changed (Figure 2).

Figure 1:

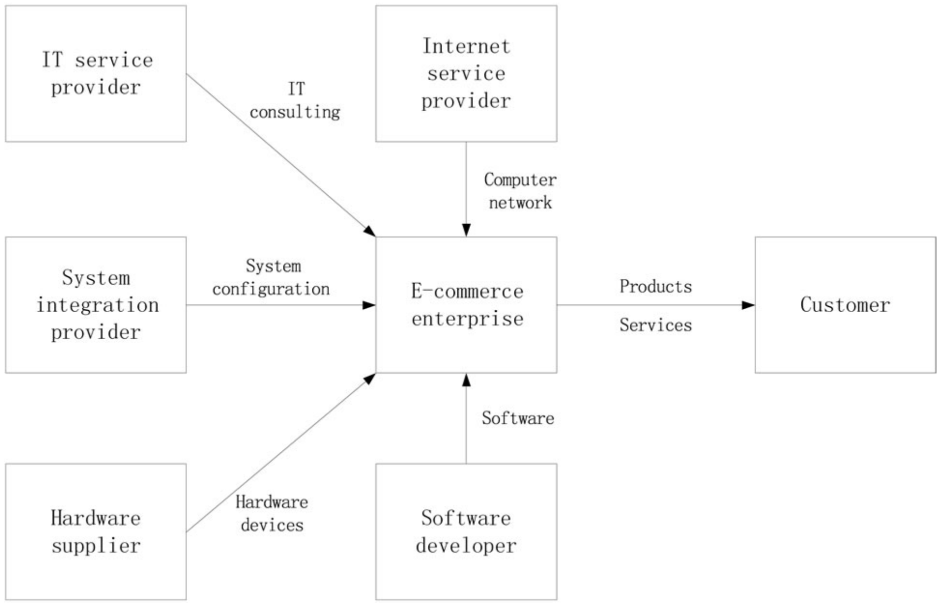
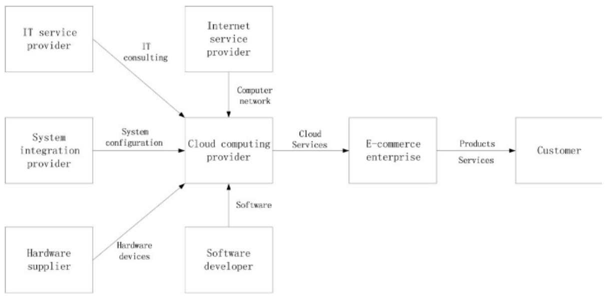

IT service provider
Internet service provider
IT consulting
Computer network
System integration provider
System configuration
E-commerce enterprise
Products
Services
Customer
Software
Hardware supplier
Hardware devices
Software developer

Figure 2:


IT service provider
IT consulting
Internet service provider
Computer network
System integration provider
System configuration
Cloud computing provider
Cloud Services
E-commerce enterprise
Products
Services
Customer
Software
Hardware supplier
Hardware devices
Software developer

A simple Registration and Login backend using NodeJS and MySQL

PHASE 4

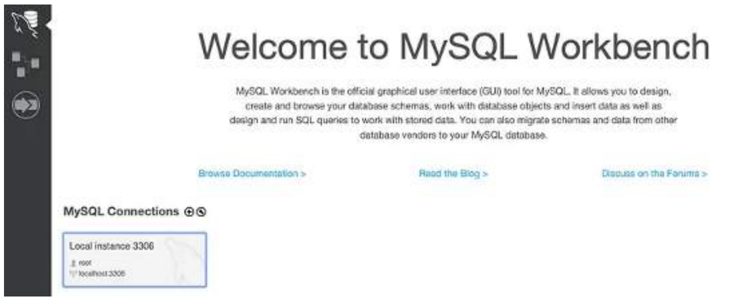
(a) Install MySQL

(b) Install mySQL Workbench

This is a GUI tool that allow you to easily work with mySQL.
Download **MySQL Workbench** from
https://downloads.mysql.com/archives/workbench/

Note: Since I'm using Mac OS Catalina 10.15, I downloaded
MySQL Workbench 8.0.19 (compatible with Catalina 10.15)

After you download your mySQL Workbench, it should automatically be connected to your "root" mySQL database.

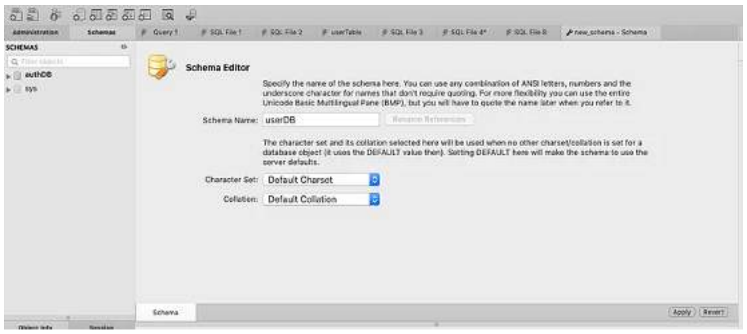


Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

(c) Create a new Database

Click on the "**Create new schema**" icon, on the mySQL Workbench toolbar, and name your new schema "**userDB**", then click on "Apply" on the lower right side.



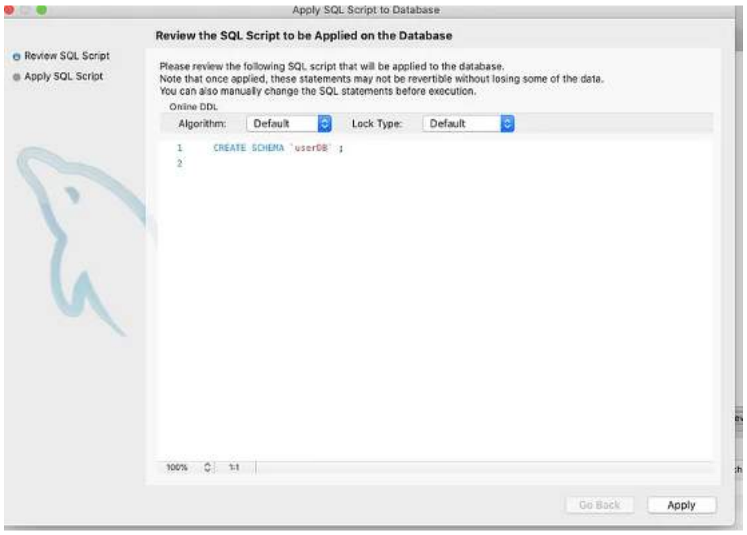
Schema Editor

Specify the name of the schema here. You can use any combination of ANSI letters, numbers and the underscore character for names that don't require quoting. For more flexibility you can use the entire Unicode Basic Multilingual Pane (BMP), but you will have to quote the name later when you refer to it.

Schema Name: userDB

The character set and its collation selected here will be used when no other charset/collation is set for a database object (it uses the DEFAULT value then). Setting DEFAULT here will make the schema to use the server defaults.

Character Set: Default Charset
Collation: Default Collation



Apply SQL Script to Database

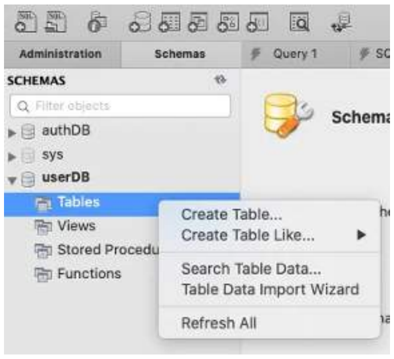
Review the SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database.
Note that once applied, these statements may not be revertible without losing some of the data.
You can also manually change the SQL statements before execution.

Online DDL
Algorithm: Default   Lock Type: Default

```
CREATE SCHEMA `userDB` ;
```

You will now see the “userDB” database on the left side bar. Click on it to expand the “userDB”, then mouse over to the userDB → Tables, and right click to “Create Table”

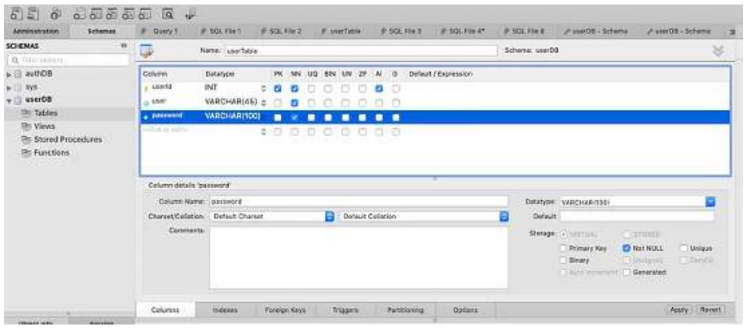


(d) Create a userTable in the userDB

Create a table with the following columns

- **userId:** INTEGER — *Primary Key (PK), Not Null (NN), AutoIncrement (AI)*
- **user:** VARCHAR(45) — *Not Null (NN)*
- **password:** VARCHAR (100) — *Not Null (NN)*

Click on “Apply” on the lower right, and create this table.

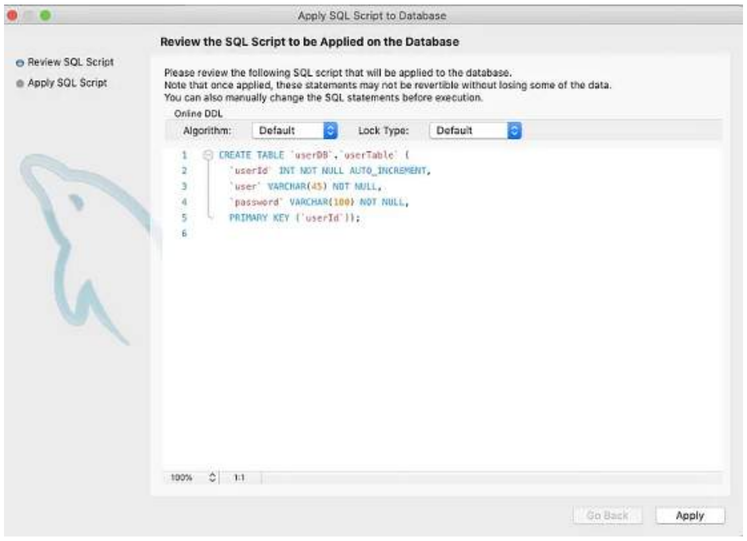


Name: userTable   Schema: userDB

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
userId	INT	✓	✓					✓		
user	VARCHAR(45)		✓							
password	VARCHAR(100)		✓							

Column details 'password'

Column Name: password
Charset/Collation: Default Charset   Default Collation
Datatype: VARCHAR(100)
Not NULL ✓



Apply SQL Script to Database

Review the SQL Script to be Applied on the Database

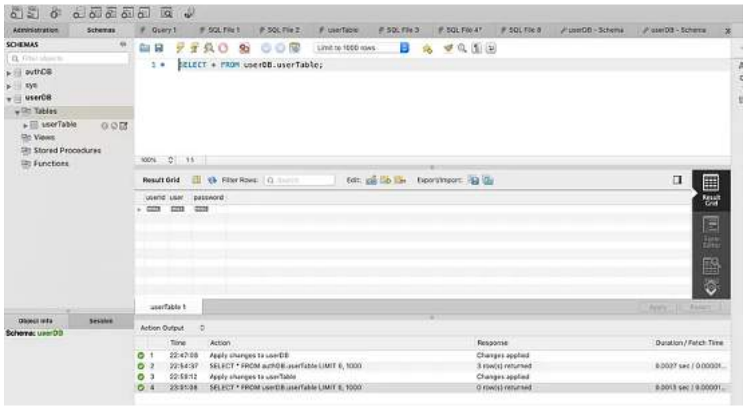
Please review the following SQL script that will be applied to the database.
Note that once applied, these statements may not be revertible without losing some of the data.
You can also manually change the SQL statements before execution.

Online DDL
Algorithm: Default   Lock Type: Default

```
CREATE TABLE `userDB`.`userTable` (
  `userId` INT NOT NULL AUTO_INCREMENT,
  `user` VARCHAR(45) NOT NULL,
  `password` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`userId`));
```

CONGRATULATIONS !!!

You have finished all the mySQL setup that you need. You can see the “userTable” under the “userDB”



```
SELECT * FROM userDB.userTable;
```

Action Output

	Time	Action	Response	Duration / Fetch Time
1	22:47:08	Apply changes to userDB	Changes applied	
2	22:54:37	SELECT * FROM authDB.userTable LIMIT 0, 1000	3 row(s) returned	0.0027 sec / 0.00001...
3	22:59:12	Apply changes to userTable	Changes applied	
4	23:01:08	SELECT * FROM userDB.userTable LIMIT 0, 1000	0 row(s) returned	0.0013 sec / 0.00001...

. . .

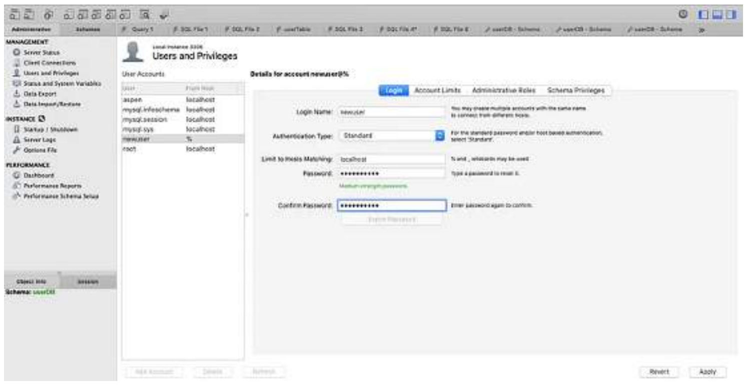
(e) (Optional) Create a new user in mySQL

While you can use the "root" user to connect into your database via Node JS, a better practice is to create a new user, and assign it the limited privileges that include READ, INSERT, DELETE rows in your DB. You do not want to assign Database Administrator privileges to the new user.

To create a new user, click on the "Administration" tab on the top left, and go to "Users and Privileges", and then click on (ADD ACCOUNT) button on the bottom.

```
Login Name: newuser
Authentication Type: Standard
Limit to Hosts Matching: localhost
Password: password1#
```

> *NOTE: We will use these credentials in the NodeJS code, to connect to the mySQL DB.*



Users and Privileges

User	From Host
aspen	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	%
root	localhost

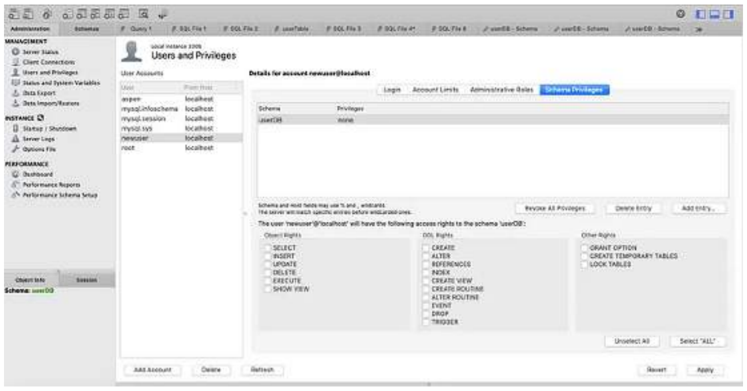
Details for account newuser@%

Login Name: newuser — You may create multiple accounts with the same name to connect from different hosts.
Authentication Type: Standard — For the standard password and/or host based authentication, select 'Standard'.
Limit to Hosts Matching: localhost — % and _ wildcards may be used
Password: •••••••••• — Type a password to reset it.
Medium strength password
Confirm Password: •••••••••• — Enter password again to confirm.

Click on APPLY on the lower right to create new user.

. . .

Next go to the SCHEMA PRIVILEGES tab and select (ADD ENTRY)



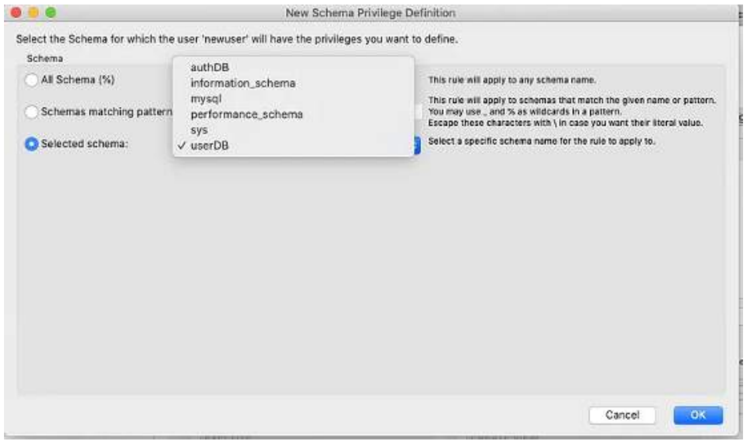
Users and Privileges

Details for account newuser@localhost

User	From Host
aspen	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	localhost
root	localhost

Schema	Privileges
userDB	none

The user 'newuser'@'localhost' will have the following access rights to the schema 'userDB':



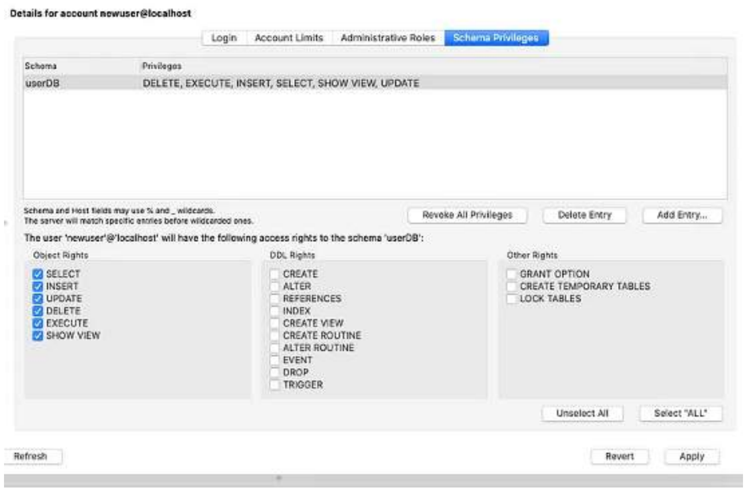
New Schema Privilege Definition

Select the Schema for which the user 'newuser' will have the privileges you want to define.

Schema

- All Schema (%) — This rule will apply to any schema name.
- Schemas matching pattern — This rule will apply to schemas that match the given name or pattern. You may use _ and % as wildcards in a pattern. Escape these characters with \ in case you want their literal value.
- Selected schema: userDB — Select a specific schema name for the rule to apply to.

Now add privileges ONLY for row CRUD operations



Details for account newuser@localhost

Schema	Privileges
userDB	DELETE, EXECUTE, INSERT, SELECT, SHOW VIEW, UPDATE

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

The user 'newuser'@'localhost' will have the following access rights to the schema 'userDB':

Object Rights: ✓ SELECT, ✓ INSERT, ✓ UPDATE, ✓ DELETE, ✓ EXECUTE, ✓ SHOW VIEW

DDL Rights: CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, TRIGGER

Other Rights: GRANT OPTION, CREATE TEMPORARY TABLES, LOCK TABLES

Click on “Apply”

Your new user is now setup, and we will access the mySQL DB using this new user!

. . .

Step 2: Connect your NodeJS App with mySQL DB

(a) Create a new folder and initialize your NodeJS App

```
$ mkdir db-practice1
$ cd db-practice1
$ npm init --y

//next we will install some node packages

$ npm i express mysql
$ npm i nodemon dotenv --save-dev

//We installed "express" and "mysql", and "nodemon" and "dotenv"
as devDependencies
```

(b) Connect your NodeJS App to your mySQL DB

Create a *dbServer.js* file as follow,

```
const express = require("express")
const app = express()

const mysql = require("mysql")

const db = mysql.createPool({
   connectionLimit: 100,
   host: "127.0.0.1",       //This is your localhost IP
   user: "newuser",         // "newuser" created in Step 1(e)
   password: "password1#",  // password for the new user
   database: "userDB",      // Database name
   port: "3306"             // port name, "3306" by default
})

db.getConnection( (err, connection)=> {

   if (err) throw (err)
   console.log ("DB connected successful: " + connection.threadId)

})
```

Now run the *dbServer.js* file,

```
$ nodemon dbServer
```

If all your credentials are correct you see a “DB connected successful” message.

AWESOME!! You are now connected to your mySQL DB!!!

Note: that we use *mysql.createPool(),* instead of *mysql.createConnection(),* since we want our application to be PRODUCTION grade.

. . .

- ***Troubleshooting:** In case all your login credentials are fine and you get a connection error, run the following in your **mySQL Workbench**

```
ALTER USER 'newuser'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'password1#';

FLUSH PRIVILEGES;
```

- https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server

. . .

(c) Create a .env file and hide your DB credentials in there

Create a **.env** file as follows

```
DB_HOST = 127.0.0.1
DB_USER = newuser
DB_PASSWORD = password1#
DB_DATABASE = userDB
DB_PORT = 3306

PORT = 3000
```

In your ***dbServer.js*** file use “dotenv” to import credentials from the .env file.

```
require("dotenv").config()

const DB_HOST = process.env.DB_HOST
const DB_USER = process.env.DB_USER
const DB_PASSWORD = process.env.DB_PASSWORD
const DB_DATABASE = process.env.DB_DATABASE
const DB_PORT = process.env.DB_PORT

const db = mysql.createPool({
   connectionLimit: 100,
   host: DB_HOST,
   user: DB_USER,
   password: DB_PASSWORD,
   database: DB_DATABASE,
   port: DB_PORT
})

//remember to include .env in .gitignore file
```

Step 3 — Setup your NodeJS App and Routes

(a) Get the NodeJS server running

Add the following to ***dbServer.js*** file

```
const port = process.env.PORT

app.listen(port,
()=> console.log(`Server Started on port ${port}...`))
```

Once you save your file, nodemon will refresh and will get your Express JS server up and running!

(b) Register a new user (save in mySQL DB)

Now you are ready to create routes to Register a new user. Note that we will be using **bcrypt** to store hashed passwords in our DB.

So let’s first npm install bcrypt.

```
$ npm i bcrypt
```

Now in your dbServer.js let’s add a route to “createUser”

```
const bcrypt = require("bcrypt")

app.use(express.json())
//middleware to read req.body.<params>

//CREATE USER
app.post("/createUser", async (req,res) => {

const user = req.body.name;
const hashedPassword = await bcrypt.hash(req.body.password,10);

db.getConnection( async (err, connection) => {

 if (err) throw (err)
 const sqlSearch = "SELECT * FROM userTable WHERE user = ?"
 const search_query = mysql.format(sqlSearch,[user])

 const sqlInsert = "INSERT INTO userTable VALUES (0,?,?)"
 const insert_query = mysql.format(sqlInsert,[user,
hashedPassword])
 // ? will be replaced by values
 // ?? will be replaced by string

 await connection.query (search_query, async (err, result) => {

  if (err) throw (err)
  console.log("------> Search Results")
  console.log(result.length)

  if (result.length != 0) {
   connection.release()
   console.log("------> User already exists")
   res.sendStatus(409)
  }
  else {
   await connection.query (insert_query, (err, result)=> {

   connection.release()

   if (err) throw (err)
   console.log ("--------> Created new User")
   console.log(result.insertId)
   res.sendStatus(201)
  })
 }

}) //end of connection.query()

}) //end of db.getConnection()

}) //end of app.post()
```

While the above code may look complicated here is what we are doing,

```
1. We first store the "req.body.name" in "user"

2. We then use the bcrypt.hash() function to hash the "password"

NOTE: that the bcrypt.hash() function may take a while to generate
the hash, and so we use the "await" keyword in front of it.

Since we are using the "await" keyword within the function, we
need to add "async" keyword in front of the function.

"async" and "await" are basically "syntactical sugar", or a neater
way to write promises in Javascript.

Ideally we want to include the "await" part in a "try/catch" block
that represents the "resolve/reject" parts of the Promise. However
we will forego this, to keep our code simple and readable for the
purposes of this tutorial.

3. We then use the db.getConnection() function to get a new
connection. This function may have 2 outcomes, either an "error"
or a "connection". i.e. db.getConnection ( (err, connection) )

4. In case we get the connection, we can then QUERY the
connection, using connection.query(). Note that since the query()
function may take some time to respond, we use the keyword "await"
in front of it. Accordingly we need to include the "async" keyword
in front of the parent function.
i.e. db.getConnection ( async (err, connection) => {
           await connection.query(<query>)
})

5. The construction of the query strings are particularly
interesting,

const sqlSearch = "SELECT * FROM userTable WHERE user = ?"
const search_query = mysql.format(sqlSearch,[user])

const sqlInsert = "INSERT INTO userTable VALUES (0,?,?)"
const insert_query = mysql.format(sqlInsert,[user,
hashedPassword])

NOTE: Basically the ? will get replaced by the values in the []

Also, notice that in the INSERT query we are using (0,?,?), this
is because the first column in our userDB is an AUTOINCREMENT
column. So we pass either a "0" or "null", and the mySQL database
will assign the next autoincrement value from its side
i.e. we do not need to pass a value in the query, and we want
mySQL DB to assign an autoincremented value.

6. The reason we have a "search_query" and a "insert_query", is
because, we want to check to see if the "user" already exists in
our MySQL DB.
- In case they do, we do not add them again ("User already
exists"). - In case they do NOT exist, we add them to the DB.
```

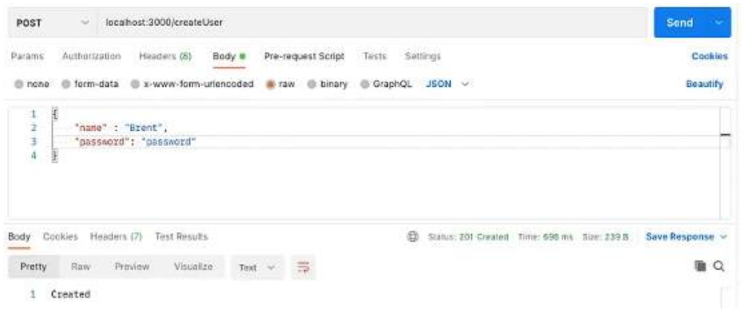
```
7. Note that after we make the query, we no longer need the
connection and we must call a **connection.release()**

8. The **connection.query()** function will either have an error OR a
result. i.e. connection.query( (err, result) )

9. **The "results" returns each ROW as an object in an array.**
Thus if the "search_query" results.length==0, then no ROWs were
returned indicating that the user does not exist in the DB
```

Now that we understand the code, let’s save it up and let nodemon restart the server.

YOUR REGISTRATION BACKEND IS NOW COMPLETE!!!

. . .

In order to test our Registration backend, we can use POSTMAN to add a new user as follows,



POST localhost:3000/createUser

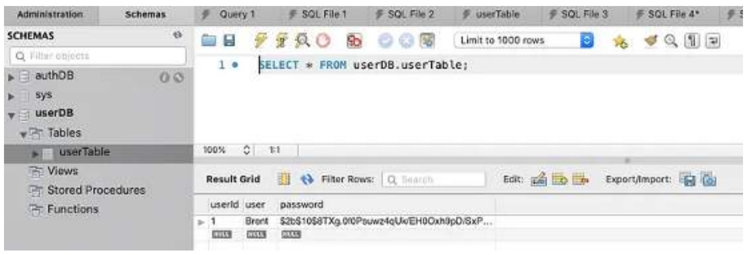
```
{
    "name" : "Brent",
    "password": "password"
}
```

Status: 201 Created Time: 698 ms Size: 239 B

```
Created
```

Your user is successfully created and saved in the mySQL DB!!

In your mySQL DB you will now see that the new user is saved, along with the (hashed) password.



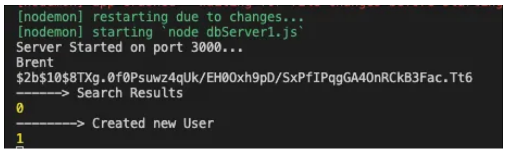
```
SELECT * FROM userDB.userTable;
```

userId	user	password
1	Brent	$2b$10$8TXg.0f0Psuwz4qUk/EH0Oxh9pD/SxP...
NULL	NULL	NULL

Also, notice that in the mySQL DB the “userId” shows **1**, even though we did not explicitly specify this in our “insert_query”. The reason for this is that when we created the DB in Step 1(d), we specified that “userId” will be “AUTO INCREMENT” and in the INSERT query, we left this field “0” or “null”.

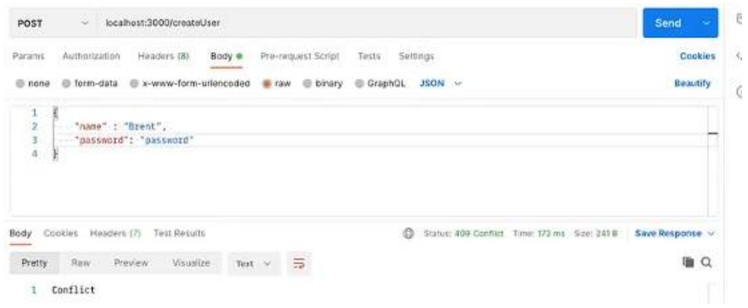
So mySQL DB automatically assigns a unique autoincrement value to this column when we insert a new record into the DB.

NOTE: Your console logs will show that the app checked to see if “Brent” exists in the DB, and since that user did not, “Brent” and his “hashedPassword” was added into the mySQL DB.



```
[nodemon] restarting due to changes...
[nodemon] starting `node dbServer1.js`
Server Started on port 3000...
Brent
$2b$10$8TXg.0f0Psuwz4qUk/EH0Oxh9pD/SxPfIPqgGA4OnRCkB3Fac.Tt6
------> Search Results
0
--------> Created new User
1
```

NOTE: If you attempt to add "Brent" again you will see the following,



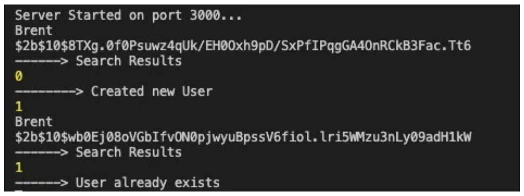
POST localhost:3000/createUser

```
{
    "name" : "Brent",
    "password": "password"
}
```

Status: 409 Conflict Time: 172 ms Size: 241 B

```
Conflict
```

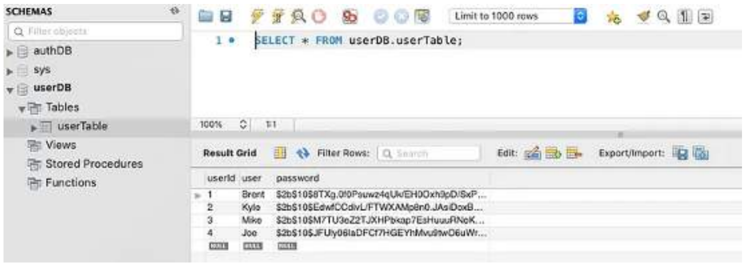
And your logs will show the following



```
Server Started on port 3000...
Brent
$2b$10$8TXg.0f0Psuwz4qUk/EH0Oxh9pD/SxPfIPqgGA4OnRCkB3Fac.Tt6
------> Search Results
0
--------> Created new User
1
Brent
$2b$10$wb0Ej08oVGbIfvON0pjwyuBpssV6fiol.lri5WMzu3nLy09adH1kW
------> Search Results
1
------> User already exists
```

. . .

Very Cool !!
Now, using POSTMAN let's add in a few more unique users to our mySQL DB. I added "Kyle", "Mike", "Joe".



```
SELECT * FROM userDB.userTable;
```

userId	user	password
1	Brent	$2b$10$8TXg.0f0Psuwz4qUk/EH0Oxh9pD/SxP...
2	Kyle	$2b$10$EdwfCCdivL/FTWXAMp8n0.JAsiDoxB...
3	Mike	$2b$10$M7TU3eZ2TJXHPbkap7EsHuuuRNoK...
4	Joe	$2b$10$JFUly06laDFCf7HGEYhMvu9twD6uWr...
NULL	NULL	NULL

> *Note that since we are storing hashedPasswords, our system is quite secure since even if the userDB is compromised, the hacker cannot get the passwords directly from the userTable.*

. . .

(d) Authenticate a Registered user (i.e. Login)

Ok, now that we've created a Registration system and are able to create and save new users/passwords in the mySQL database, let's create a Login system that will authenticate users that attempt to login with the correct username and password.

```
The authentication steps will be as follows:
```

```
1. Read the "name" and "password" from the req.body

2. Search to see if the user "name" exists in the DB,
- If does not exist, return a 404 error
- If exists, then get the "hashedPassword" stored in the DB

3. Compare the "password" with the "hashedPassword" using
bcrypt.compare(<password>, <hashedPassword>)

4. if bcrypt.compare() returns true, passwords match and the user
is AUTHENTICATED !!
```

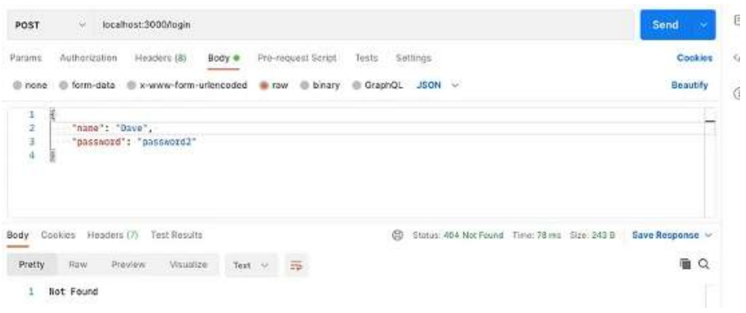
The code for this is as follows,

```
//LOGIN (AUTHENTICATE USER)
app.post("/login", (req, res)=> {

const user = req.body.name
const password = req.body.password

db.getConnection ( async (err, connection)=> {

 if (err) throw (err)
 const sqlSearch = "Select * from userTable where user = ?"
 const search_query = mysql.format(sqlSearch,[user])

 await connection.query (search_query, async (err, result) => {

  connection.release()

  if (err) throw (err)

  if (result.length == 0) {
   console.log("--------> User does not exist")
   res.sendStatus(404)
  }
  else {
     const hashedPassword = result[0].password
     //get the hashedPassword from result

    if (await bcrypt.compare(password, hashedPassword)) {
    console.log("---------> Login Successful")
    res.send(`${user} is logged in!`)
    }
    else {
    console.log("---------> Password Incorrect")
    res.send("Password incorrect!")
    } //end of bcrypt.compare()

  }//end of User exists i.e. results.length==0

 }) //end of connection.query()

}) //end of db.connection()

}) //end of app.post()
```

. . .

Let's test this using Postman

- Let's use a user that does not exist



POST localhost:3000/login

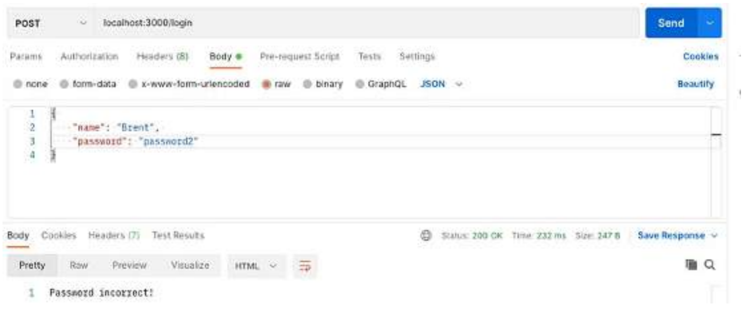
```
{
    "name": "Dave",
    "password": "password2"
}
```

Status: 404 Not Found Time: 78 ms Size: 243 B

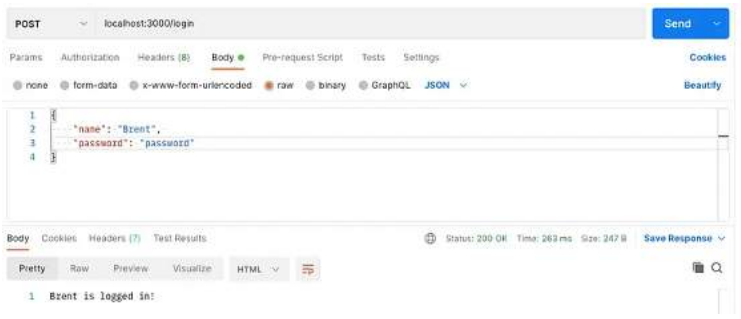
```
Not Found
```

User does not exist

- Let's now use a user that exists, but an incorrect password


```
POST localhost:3000/login
{
    "name": "Brent",
    "password": "password2"
}
```
Status: 200 OK Time: 232 ms Size: 247 B
```
Password incorrect!
```
User exists, but password incorrect

- Let's now use a user the exists and enter the correct password


```
POST localhost:3000/login
{
    "name": "Brent",
    "password": "password"
}
```
Status: 200 OK Time: 263 ms Size: 247 B
```
Brent is logged in!
```
Successful Authentication!

YOUR LOGIN BACKEND IS NOW COMPLETE!!!

. . .

(e) (Optional) On Login return an accessToken

In real world systems, once the user is authenticated, we typically want to maintain sessions for the user. One approach of doing this is by using "stateless" session management using JWT tokens.

You can read more about how to maintain sessions using accessToken and refreshTokens in my post here: https://medium.com/@prashantramnyc/authenticate-rest-apis-in-node-js-using-jwt-json-web-tokens-f0e97669aad3

We can enhance our AUTHENTICATION system, and return an accessToken to authenticated users once they successfully login.

```
const generateAccessToken = require("./generateAccessToken")
//import the generateAccessToken function

//LOGIN (AUTHENTICATE USER, and return accessToken)
app.post("/login", (req, res)=> {

const user = req.body.name
const password = req.body.password

db.getConnection ( async (err, connection)=> {

if (err) throw (err)
 const sqlSearch = "Select * from userTable where user = ?"
 const search_query = mysql.format(sqlSearch,[user])

await connection.query (search_query, async (err, result) => {

connection.release()

  if (err) throw (err)

if (result.length == 0) {
   console.log("--------> User does not exist")
   res.sendStatus(404)
  }
```

```
    else {
     const hashedPassword = result[0].password
     //get the hashedPassword from result

  if (await bcrypt.compare(password, hashedPassword)) {
      console.log("---------> Login Successful")
      console.log("---------> Generating accessToken")
      const token = generateAccessToken({user: user})
      console.log(token)
      res.json({accessToken: token})
     } else {
      res.send("Password incorrect!")
     } //end of Password incorrect

  }//end of User exists

  }) //end of connection.query()

  }) //end of db.connection()

  }) //end of app.post()
```

To generate the accessToken, you can add the following function in a **generateAccessToken.js** file and import it into your **dbServer.js**.

Note that we will be using “jsonwebtoken” library to generate the jwt tokens, so let’s first install it.

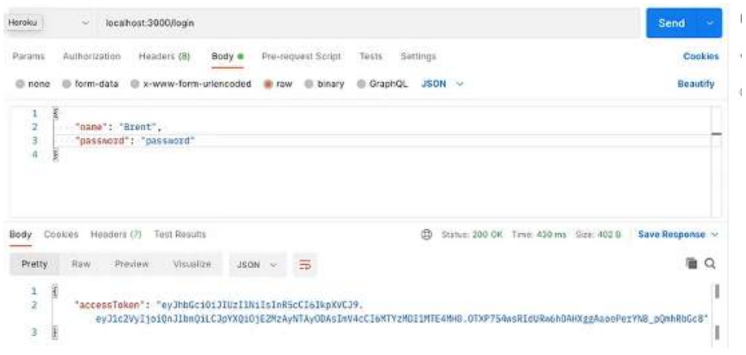
```
$ npm i jsonwebtoken
```

Your generateAccessToken.js file will be as follows

```
const jwt = require("jsonwebtoken")

function generateAccessToken (user) {

return
jwt.sign(user, process.env.ACCESS_TOKEN_SECRET, {expiresIn:
"15m"})

}

module.exports=generateAccessToken
```

. . .

Now when you login using the correct user and password, the App will return an ***accessToken***, that can be used to access authorized resources after login.



```
{
    "name": "Brent",
    "password": "password"
}
```

```
{
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyIjoiQnJlbnQiLCJpYXQiOjE2MzAyNTAyODAsImV4cCI6MTYzMDI1MTE4MH0.OTXP754wsRIdURw6h0AHXggAaaoPezYN8_pQmhRbGc8"
}
```

Return accessToken on successful authentication

. . .

And that’s it!

Shopping cart:

This is the simple implementation of shopping cart.

Source code:

```
Source Code for ShoppingCartItem.java
// This class contains data for an individual item in a
// shopping cart.

import java.net.URL;

public class ShoppingCartItem implements Cloneable
{
 public String itemName;
 public int itemCost;
 public int quantity;
 public URL descriptionURL;

 public ShoppingCartItem()
 {
 }

 public ShoppingCartItem(String itemName, int itemCost,
  int quantity, URL descriptionURL)
 {
  this.itemName = itemName;
  this.itemCost = itemCost;
  this.quantity = quantity;
  this.descriptionURL = descriptionURL;
 }

// The add method is a quick method for combining two similar
// items. It doesn't perform any checks to insure that they are
// similar, however. You use this method when adding items to a
// cart, rather than storing two instances of the same item, you
// add the quantities together.

 public void add(ShoppingCartItem otherItem)
 {
  this.quantity = this.quantity + otherItem.quantity;
 }

// The subtract method is similar to the add method, but it
// removes a certain quantity of items.

 public void subtract(ShoppingCartItem otherItem)
 {
  this.quantity = this.quantity - otherItem.quantity;
 }
// You can store items in a hash table if you implement hashCode. It's
// always a good idea to do this.

 public int hashCode()
 {
  return itemName.hashCode() + itemCost;
 }

// The equals method does something a little dirty here, it only
// compares the item names and item costs. Technically, this is
// not the way that equals was intended to work.

 public boolean equals(Object other)
 {
  if (this == other) return true;

  if (!(other instanceof ShoppingCartItem))
   return false;

  ShoppingCartItem otherItem =
   (ShoppingCartItem) other;

  return (itemName.equals(otherItem.itemName)) &&
   (itemCost == otherItem.itemCost);
 }

// Create a copy of this object

 public ShoppingCartItem copy()
 {
  return new ShoppingCartItem(itemName, itemCost,
   quantity, descriptionURL);
 }

// Create a printable version of this object

 public String toString()
 {
  return itemName+" cost: "+itemCost+" qty: "+quantity+" desc: "+
   descriptionURL;
 }
}
```

Designing user interface of web-based e-commerce application

Abstract. This study aims to provide guidance and examples in designing web interface of e-commerce applications that are good and easy to use by the users. This research used descriptive method by making a systematic description of the facts that refer to the object of research. Data collecting was carried out through literature study from journals, learning materials, and curriculum data. The result is that if users find it difficult to operate an application, the application will immediately be abandoned. In conclusion, this design has fulfilled the basic requirement of comfortable design and practibility.

1. Introduction

E-commerce is defined as activities and transactions done through social media environment and mostly on social networks [1]. Computers and other electronical devices today have used graphic interfaces where users can interact with another computer or devices [2]. E-commerce acts as a media that uses a device or computer, which certainly has a dependence on the interface [3]. Designing a visually good website has become the main goal for companies that wanted to maximize profits and promote their services or products in markets amidst intense competition [4]. User satisfaction in using the application is also important as a recommendation system to help users make better choices [5].

The qualities of recommendation are suitability, novelty, attractiveness, diversity, and context compatibility [6]. A compact display that has fewer controls is preferred by most users [7]. Standardization is very important for website design because people did not want to waste their time on the site. If the site is complicated, people will not use it [8]. Based on research done by Eid, 35% - 40% of the an e-commerce website income comes from visitors who regularly visit the site [9]. The success of a website is achieved through information and entertainment relationships, and it is better to fulfill both for positive evaluations [10].

This study aims to provide guidance and examples in designing web interface of e-commerce applications that are good and easy to use by users. This research used descriptive method by making a systematic description of the facts that refer to the object of research and collecting data carried out through journals, learning materials, and curriculum data.

2. Method

This research used descriptive method by making a systematic description of the facts that refer to the object of research and collected data. Primary data collection was obtained by interviewing student as the respondents and secondary data was obtained from journals, study materials, and curriculum data.

3. Results and Discussion

3.1 Case Assumptions

In the application that is going to be made, the user has access rights in changing personal information and using the features available in the application.

3.2 Problem Identification

Creating a web-based e-commerce application interface design to assist users in conducting transactions and other activities online by looking at problems that may occur when conducting transaction activities.

3.3 User Interface Design

The interface design consists of displaying each web page along with its functions and attributes. The draft web page is made up of the main page, entry page, registration page, product details page, profile page, basket page, wish list page, settings page, and checkout page. The following is the design of the view that has been made.

3.3.1 Main Page

The following figure shows the main page display with the display name T01 (See Figure1).

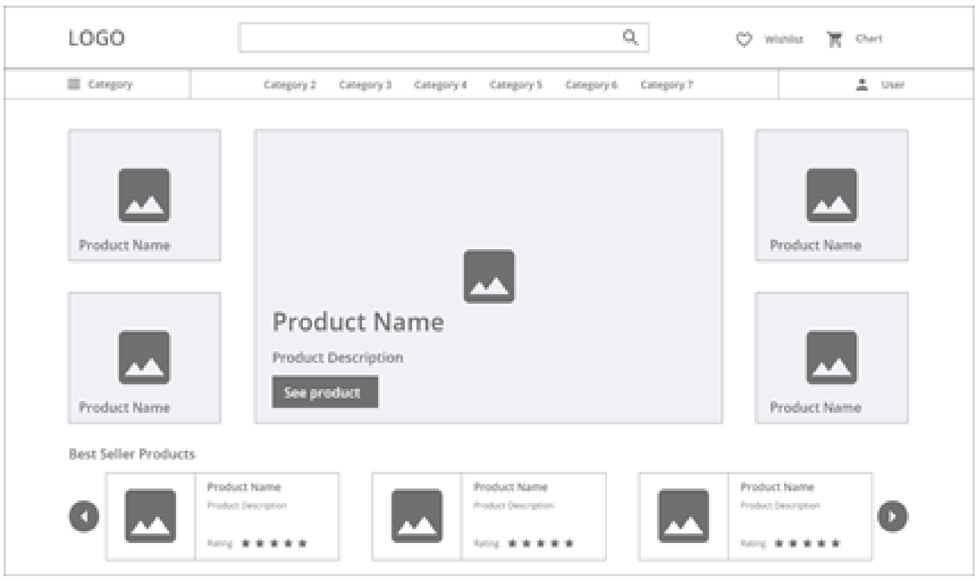


Figure 1. Display T01

Functions:

- Click the wishlist button to go to the T08 display.
- Click the basket button to go to the T07 view.
- Click the user button to go to T03 and T06 views.
- Click the product see button to go to T04 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T01.

3.3.2 Registration Page

The following figure shows the page display with the display name T02 (See Figure 2).

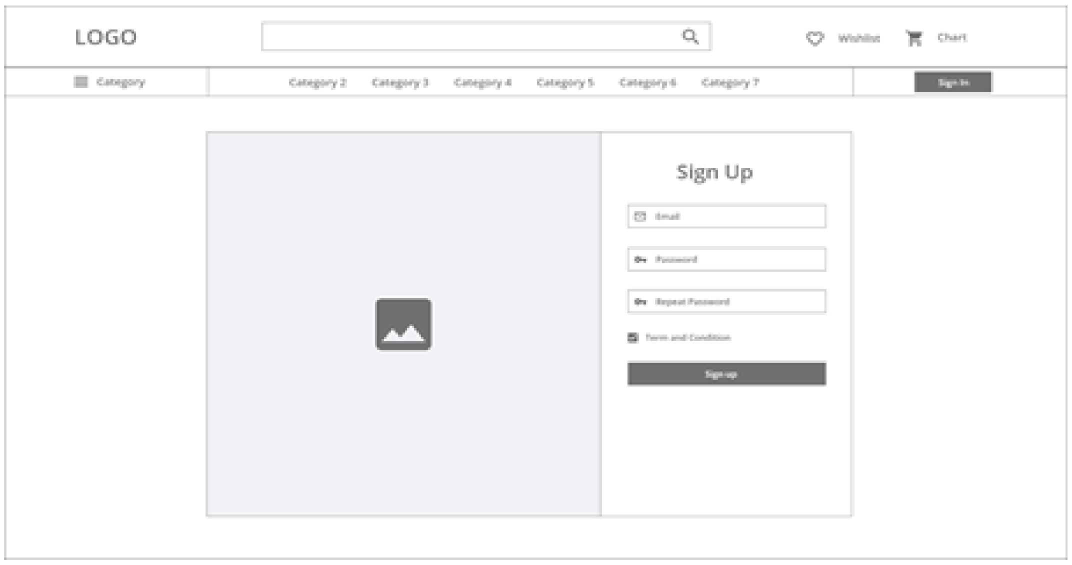


Figure 2. Display T02

Functions:

- Click the enter button to go to T03.
- Click or select the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T02.

3.3.3 Sign In Page

The following figure shows the page display with the display name T03 (See Figure 3).

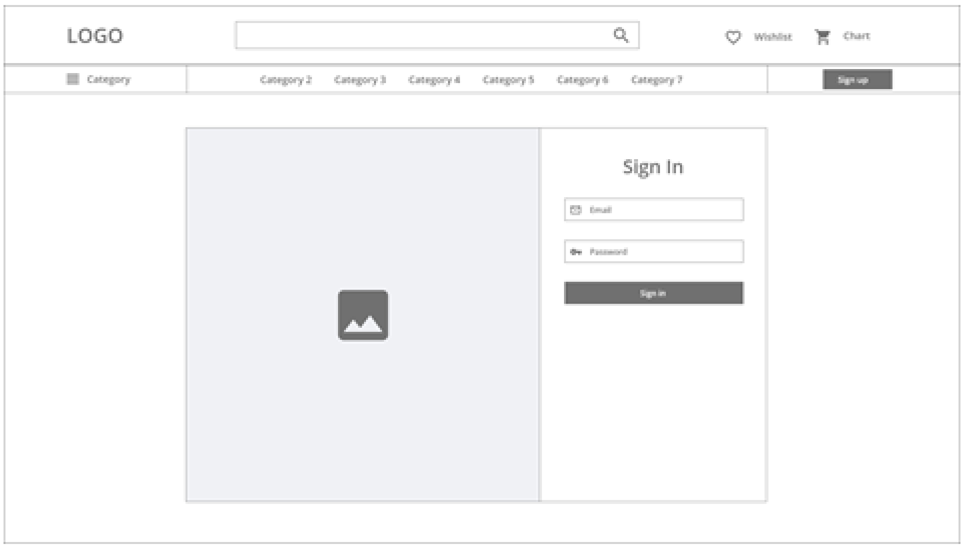


Figure 3. Display T03

Functions:

- Click the registration button to go to the T02 display.
- Click or select the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T03.

3.3.4 Product Details Page

The following figure shows the page display with the display name T04 (See Figure 4).

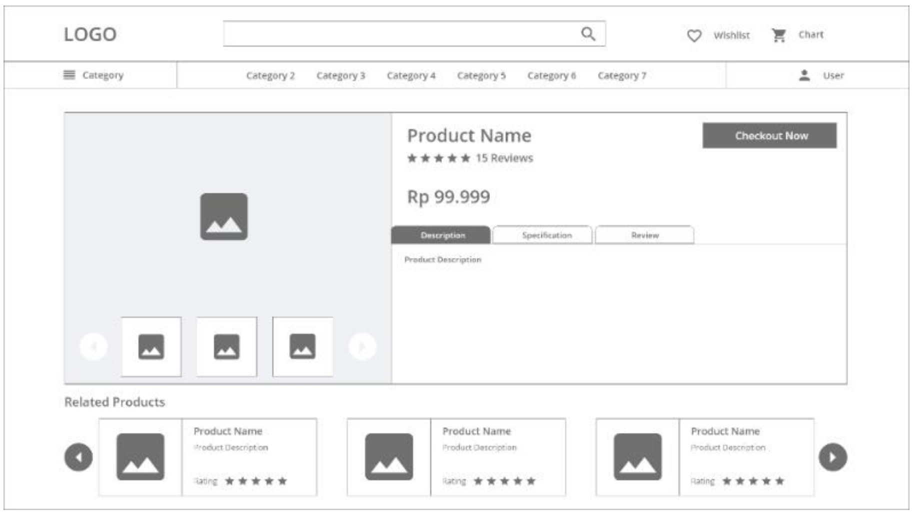


Figure 4. Display T04

Functions:

- Click the wishlist button to go to the T08 display.
- Click the basket button to go to the T07 view.
- Click the user button to go to T03 and T06 views.
- Click or select the logo button or text to go to T01 display.
- Click the buy button now to go to the T05 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T04.

3.3.5 Checkout Page

The following figure shows the page display with the display name T05 (See Figure 5).

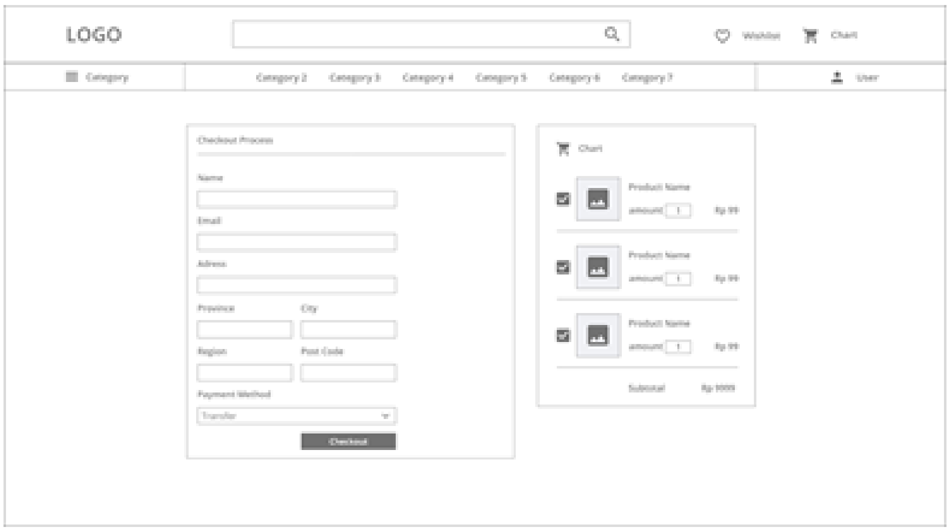


Figure 5. Display T05

Function:

- Click the wish list button to go to the T08 display.
- Click the basket button to go to the T07 view.
- Click the user button to go to T03 and T06 views.
- Click or select the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T05.

3.3.6 *Profile Page*

The following figure shows the page display with the display name T06 (See Figure 6).

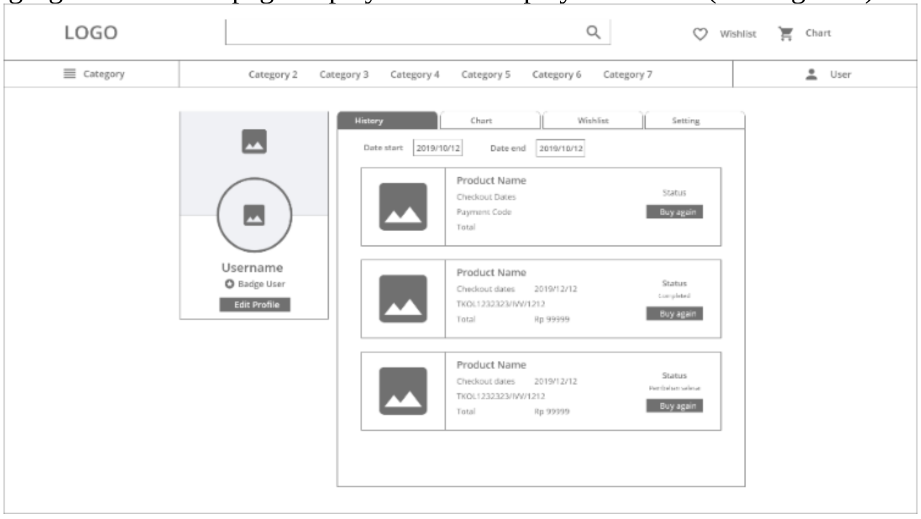


Figure 6. Display T06

Functions:

- Click the wishlist button to go to the T08 display.
- Click the basket button to go to the T07 view.
- Click the user button to go to T03.
- Click the settings button to go to T09 display.
- Click the buy button again to go to the T05 display.
- Click or press the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T06.

3.3.7 *Basket Page*

The following figure shows the page display with the display name T07 (See Figure 7).

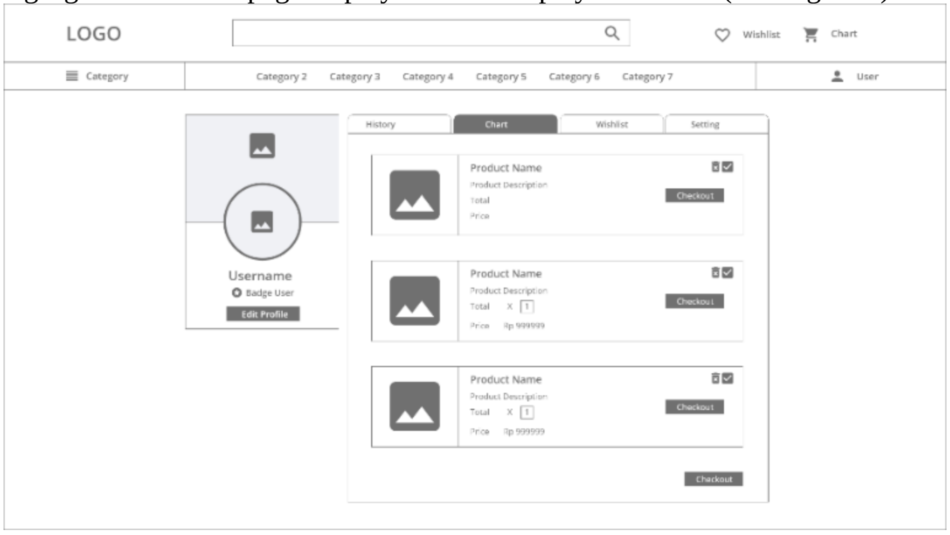


Figure 7. Display T07

Functions:

- Click the wishlist button to go to the T08 display.
- Click the transaction history button to go to the T06 view.
- Click the user button to go to T03.

- Click the settings button to go to T09 display.
- Click the checkout button to go to the T05 display.
- Click or select the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T07.

3.3.8 *Wish list Page*

The following figure shows the page display with the display name T08 (See Figure 8).

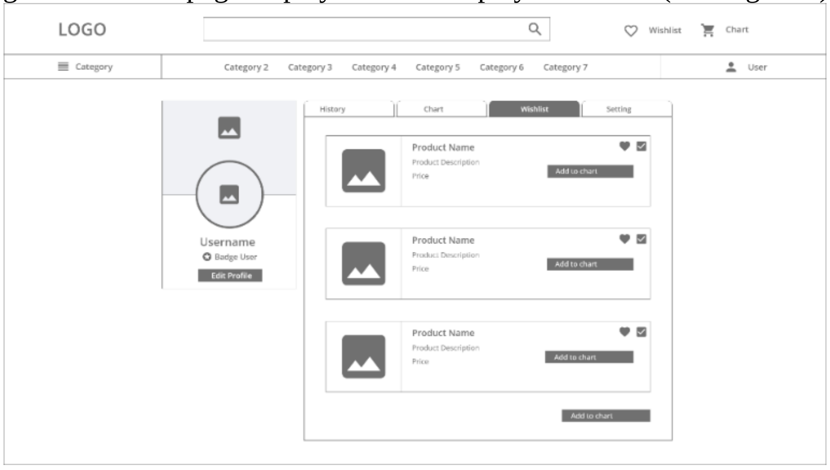


Figure 8. Display T08

Functions:

- Click the basket button to go to the T07 view.
- Click the transaction history button to go to the T06 view.
- Click the user button to go to T03.
- Click the settings button to go to T09 display.
- Click the checkout button to go to the T05 display.
- Click or select the logo button or text to go to T01 display.

Attribute: Size of 1920x1080, open-sans 12pt font, 14pt black, display name T08.

3.3.9 *Setting Page*

The following figure shows the page display with the display name T09 (See Figure 9).

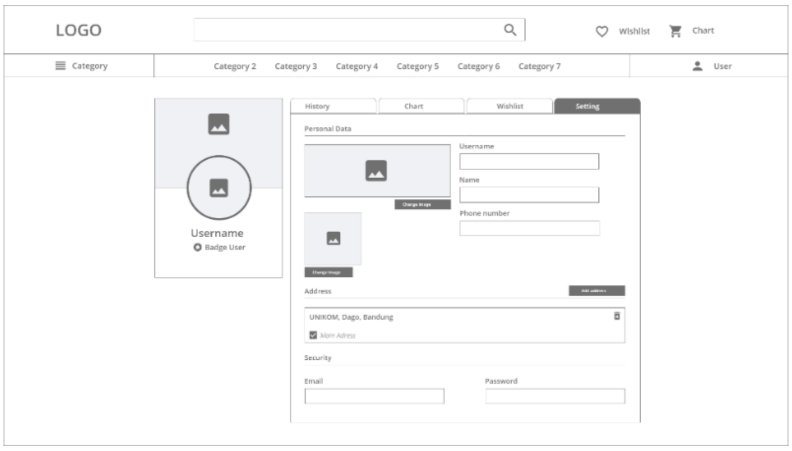


Figure 9. Display T09