Online Medical Catalogue

By:

Saptarshi Ghosh 160002052

Somesh Patil 160001056

The Project was made on MySQL (5.7) and the website deployed on IIS Server (Windows IIS 10.0) with PHP 7.1. The Frontend is made in HTML5, Javascript with JQuery(Ajax) linking the frontend to backend which is handled via PHP.



PHP includes in-built support to query to MySQL database.

<?php

$server = 'localhost:3306';

$username = 'root';

$password = 'rootpassword'; //Replace with actual root password

$database = 'medicine'; //Name of Database to connect

$db = mysqli\_connect($server, $username, $password, $database)

or die('<b>Error Connecting to MySQL Server or Else DB Not Found!</b>');

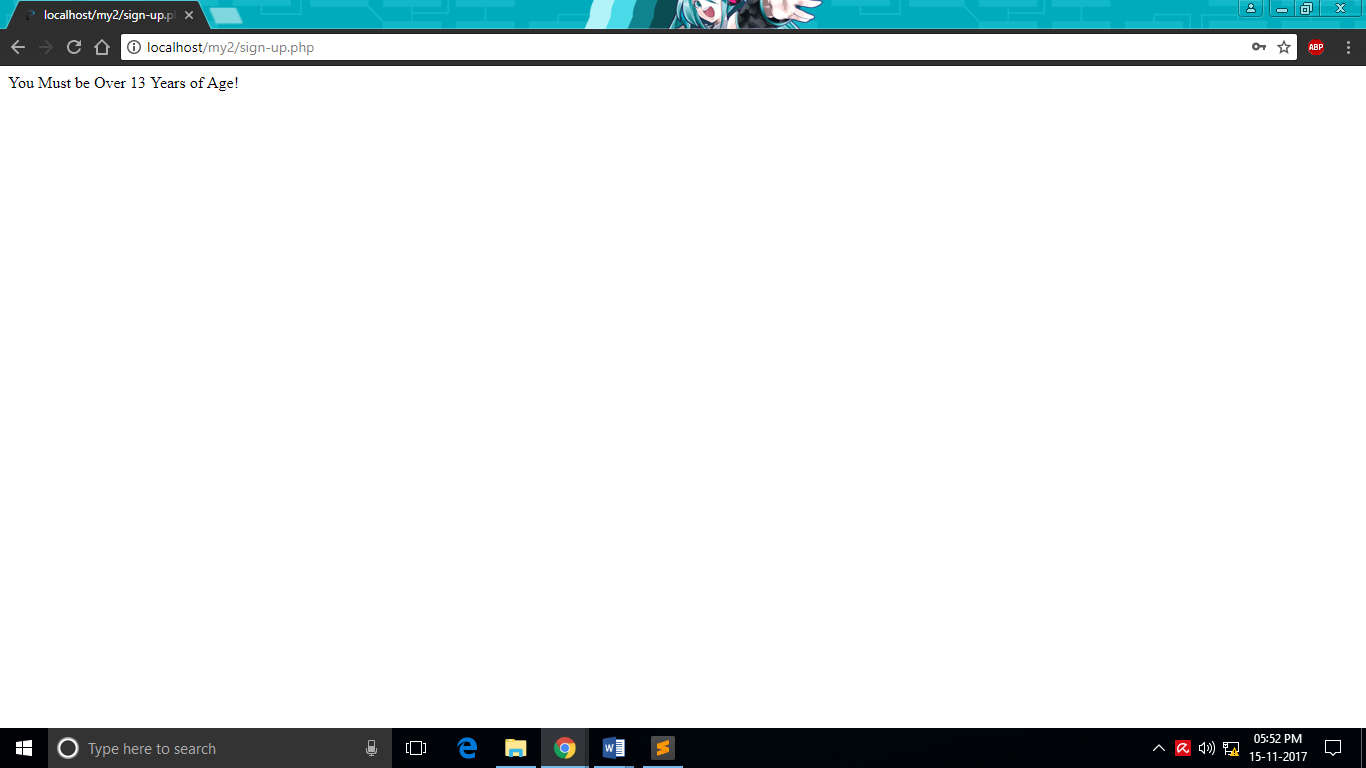
$query = “SELECT \* from medicine;”;

$result = mysqli\_query($db, $query); //Stores the Result Table

?>

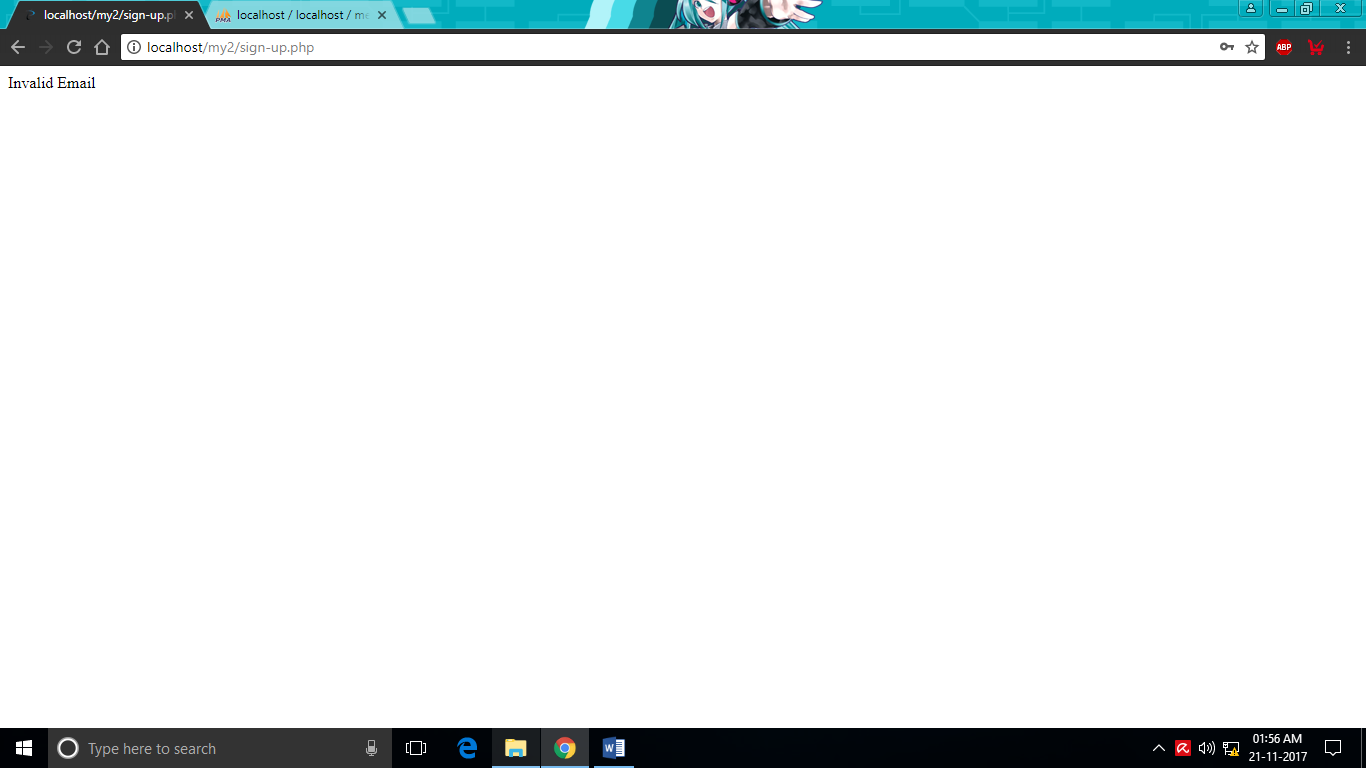
All Basic Functionalities as mentioned in “Problem Statement.docx” are modelled in this website.

Some Administrator only tasks can only be performed via “Admin CP” which requires the site to be opened on the server itself or remotely connected to server. (i.e. Admin CP will not be accessible to normal user)

Triggers mentioned in “Triggers, Transactions and Users.docx” are all modelled in the medicine schema which ensure consistency and integrity of the database. Triggers prevent inconsistent data in the database. If the conditions mentioned in the triggers are violated while inserting or updatating to the database, the connection via php is broken and error message outputs to the webpage. An Example of Trigger returning an error 

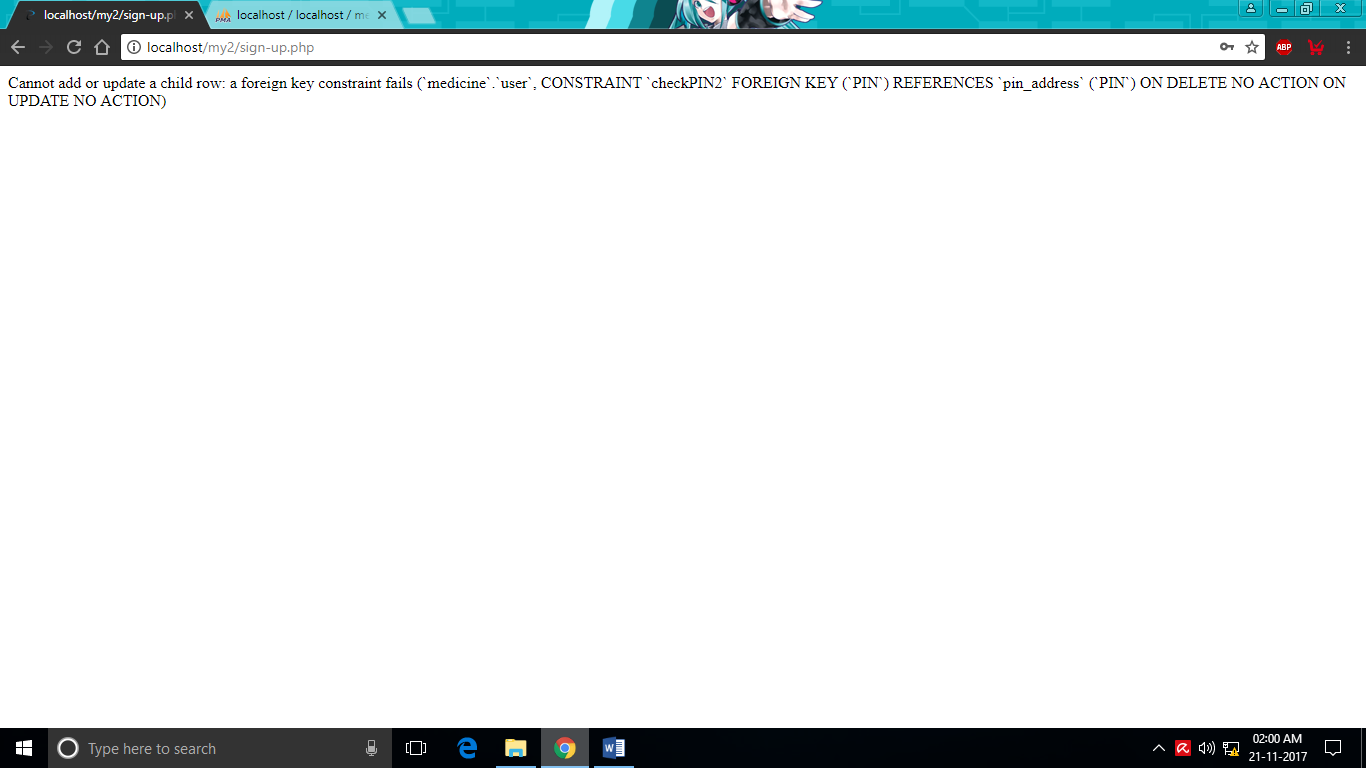
The Sign Up Page mandates that user be atleast be over 13 years of age. (The Entered DOB was 10/10/2017 and the following Error shows up when trying to create the user profile) which shows the trigger working correctly.

Another case where invalid email id : “xyz#abc” was entered in sign-up page.



Another example where invalid PIN was entered. (Assumption : Our list of PIN Codes is exhaustive and includes all valid PIN). Any PIN Code not in our database is therefore not a valid PIN.

Test Case #: PIN entered (123126) which is not in our database.



All possible test cases were tried and shows correct output (“Errors” in this case) for testing other Triggers.

Although some of the triggers are handled in PHP and HTML itself. For eg. Email must of form “ %@%.domain “. The HTML email input tag automatically checks for this format although the Database rechecks for this while inserting the user record.

In Another Case, Gender can of 3 types ‘Male’, ‘Female’ and ‘Others’. In the Signup Page only these 3 options were provided, but still the correctness of the gender is checked in the database too.

All the Triggers are modelled such that it prevents wrong data from entering into the Database in the first place thus ensuring consistency and integrity. Thorough testing was done to make sure that database is error-free.

**Transactions**

Transactions are required when a task (inserting records into a table) either must be fully carried out or not at all. At the end either it must be there in the table without any errors or not at all.

Transactions were used to insert records into the Order\_Medicine Table which stores information about the medicines ordered by the user and later is used to generate the bill.

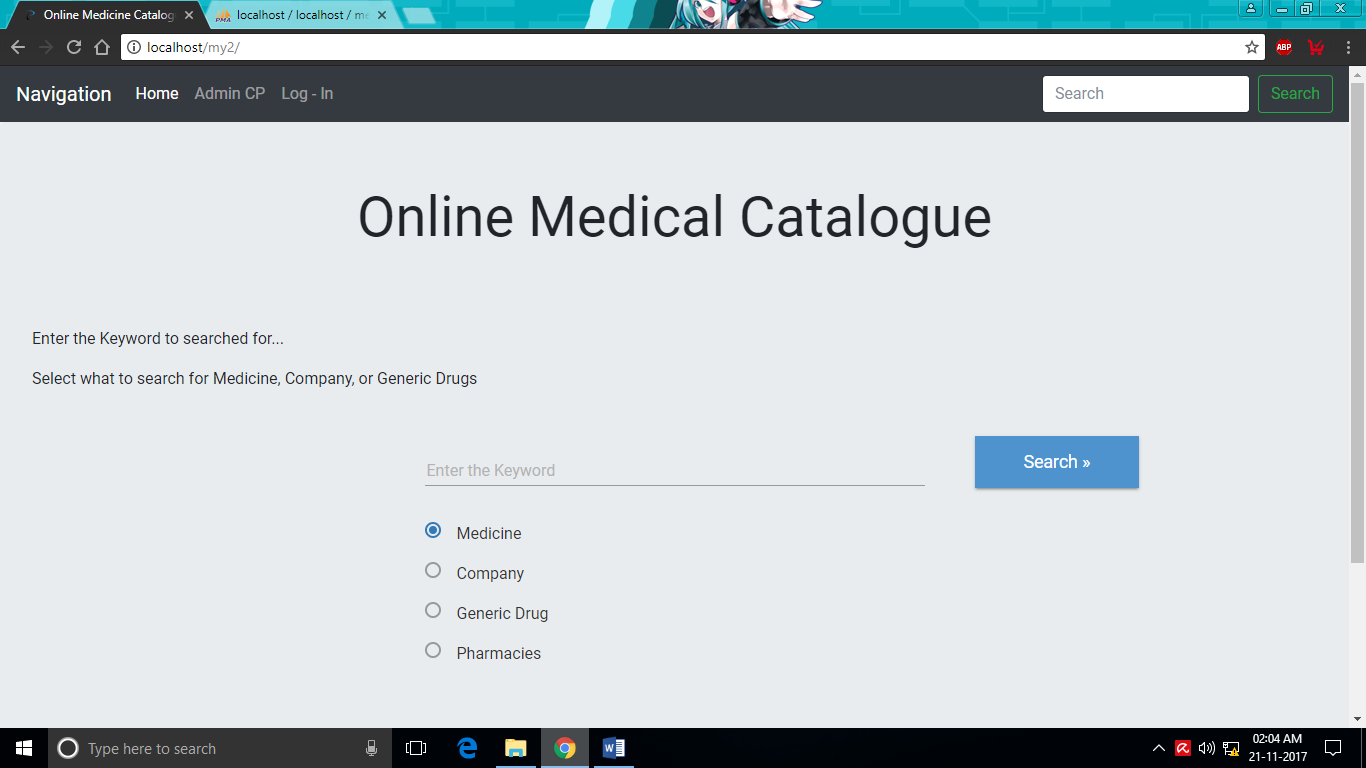
Transactions are handled by MySQL using “START TRANSACTION”, and “COMMIT” and “ROLLBACK” commands which automatically ensure that statements within start transaction and commit are correctly inserted into the table or else rollback (explicit handler was defined to handle rollbacks).

Summarizing, the project incorporates all the functionalities mentioned earlier. The Website is lucid and takes care of exceptions.

Final View of the Website

1. Main Home Page

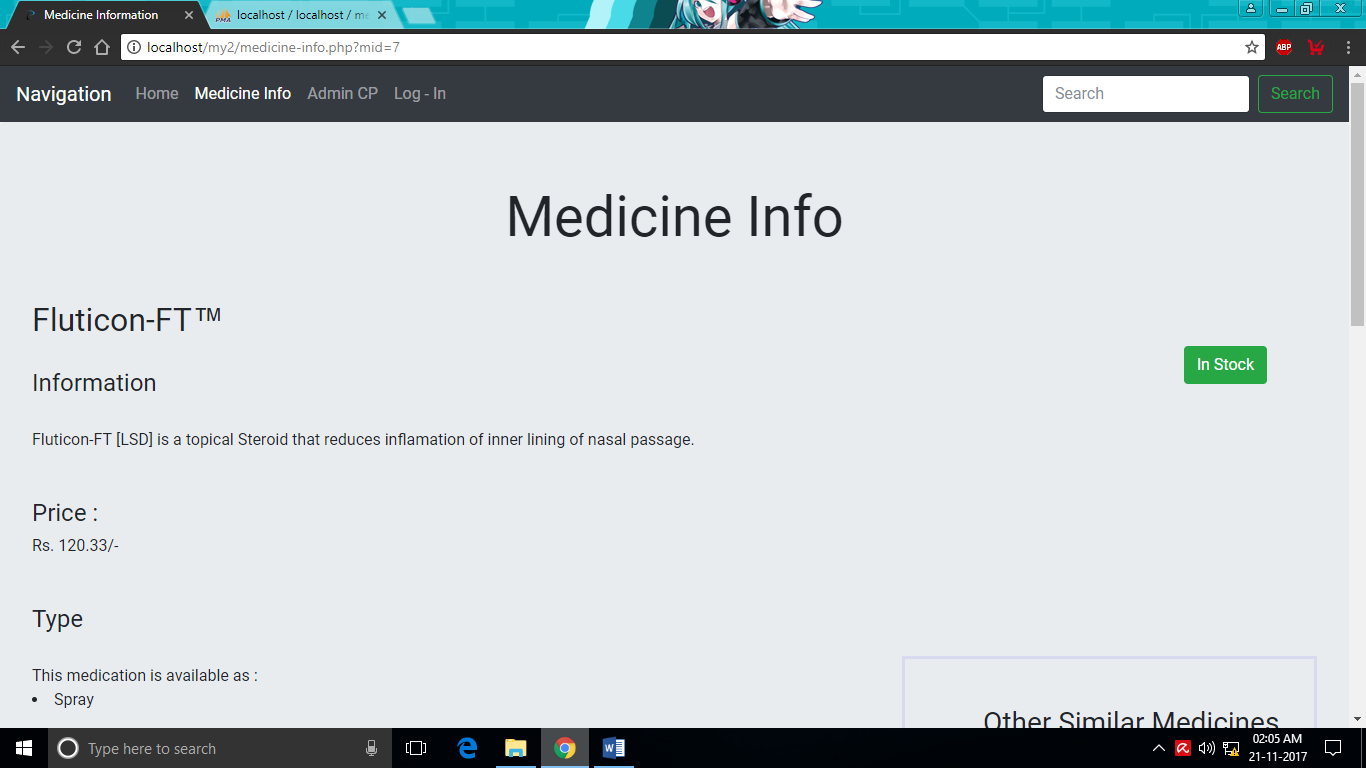
This is where user can perform search to find all Medicines / Pharmacies / Companies and Generic Drugs. User can search by keyword as well as leave it blank. Leaving it blank returns top 25 Medicines. Clicking on a Medicine redirects to its info page.

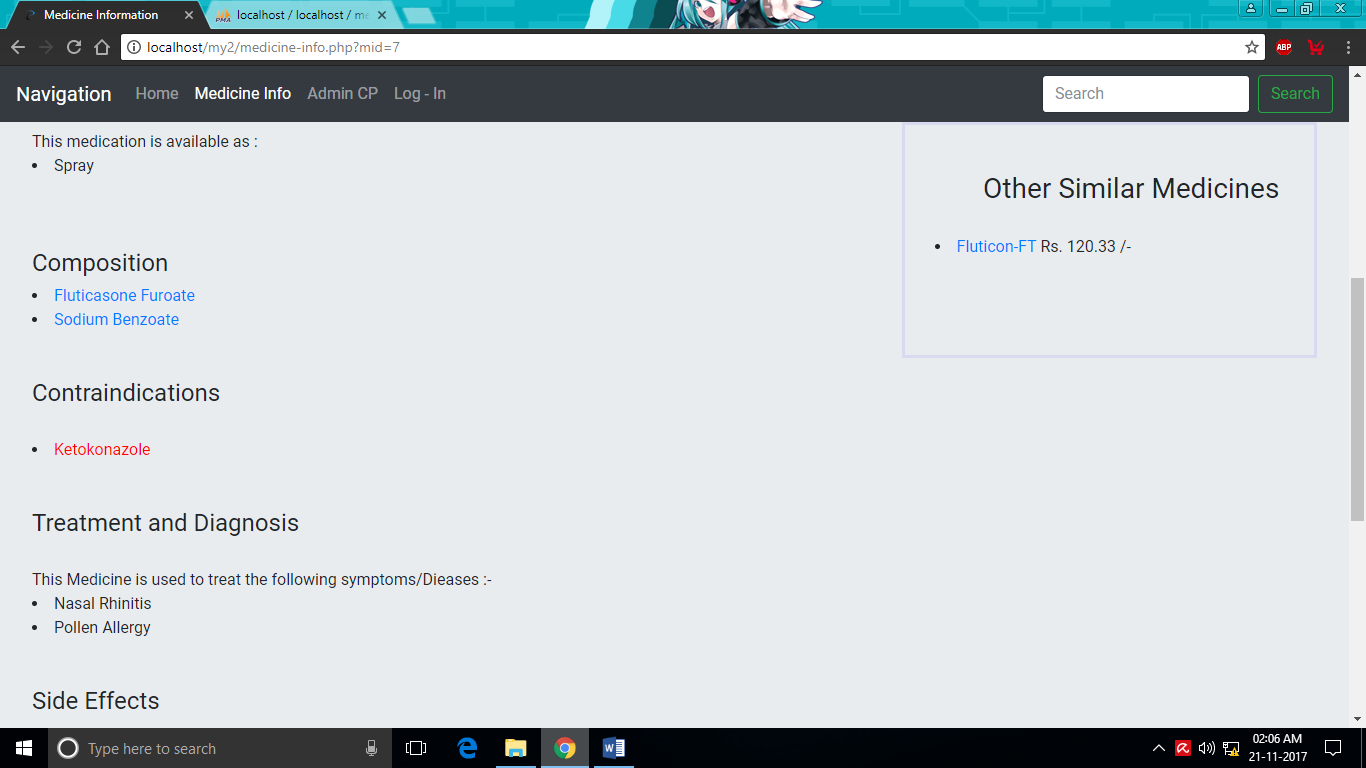




1. Medicine Info Page

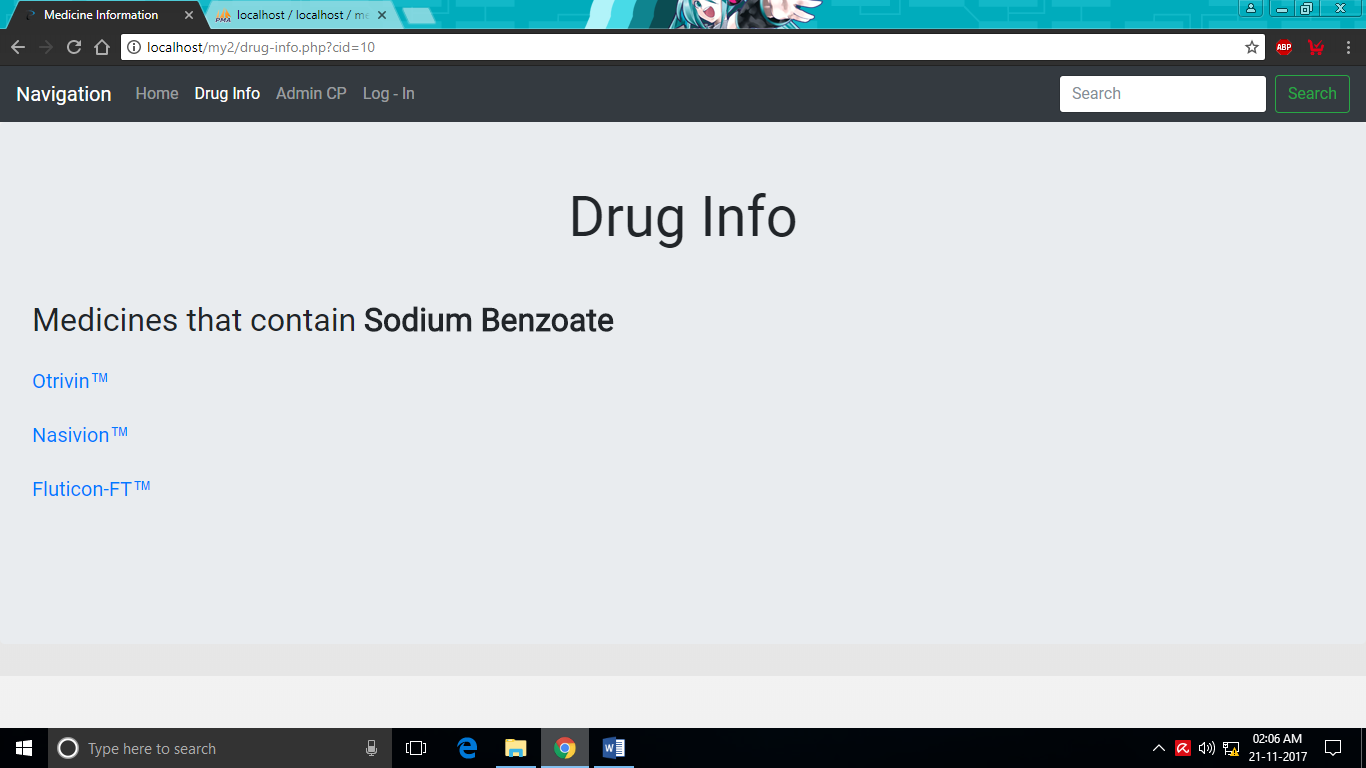
This is where user can get all information about a single medicine. All Details such as Name, Price, Composition, Dosage, Other Similar Medicines, Allergy Information, Treatments, Side Effects, Company are mentioned.



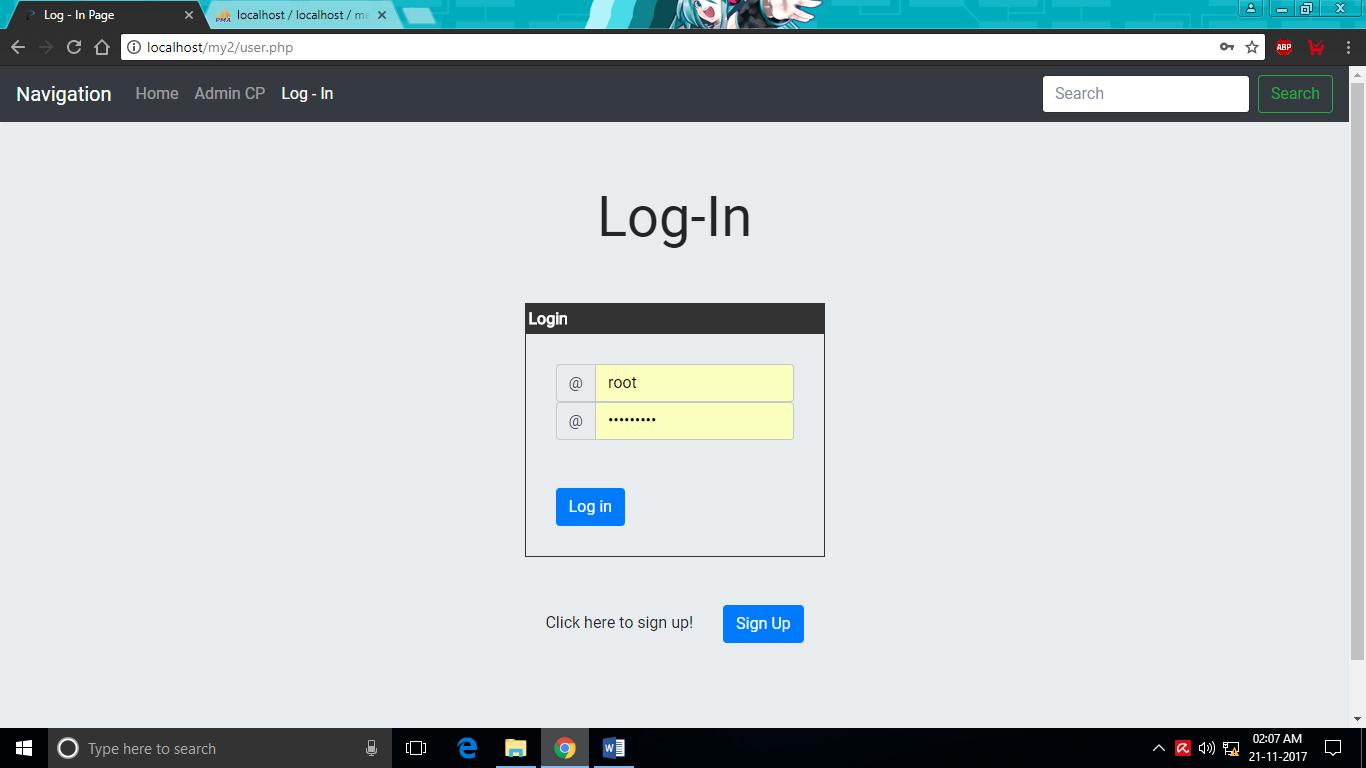


1. Drug Info Page

This is where user can get all drugs that have this generic drug. If a user is allergic to some component then he/she will want to avoid these drugs. It is a good way to know about medicines that might cause more harm than good.

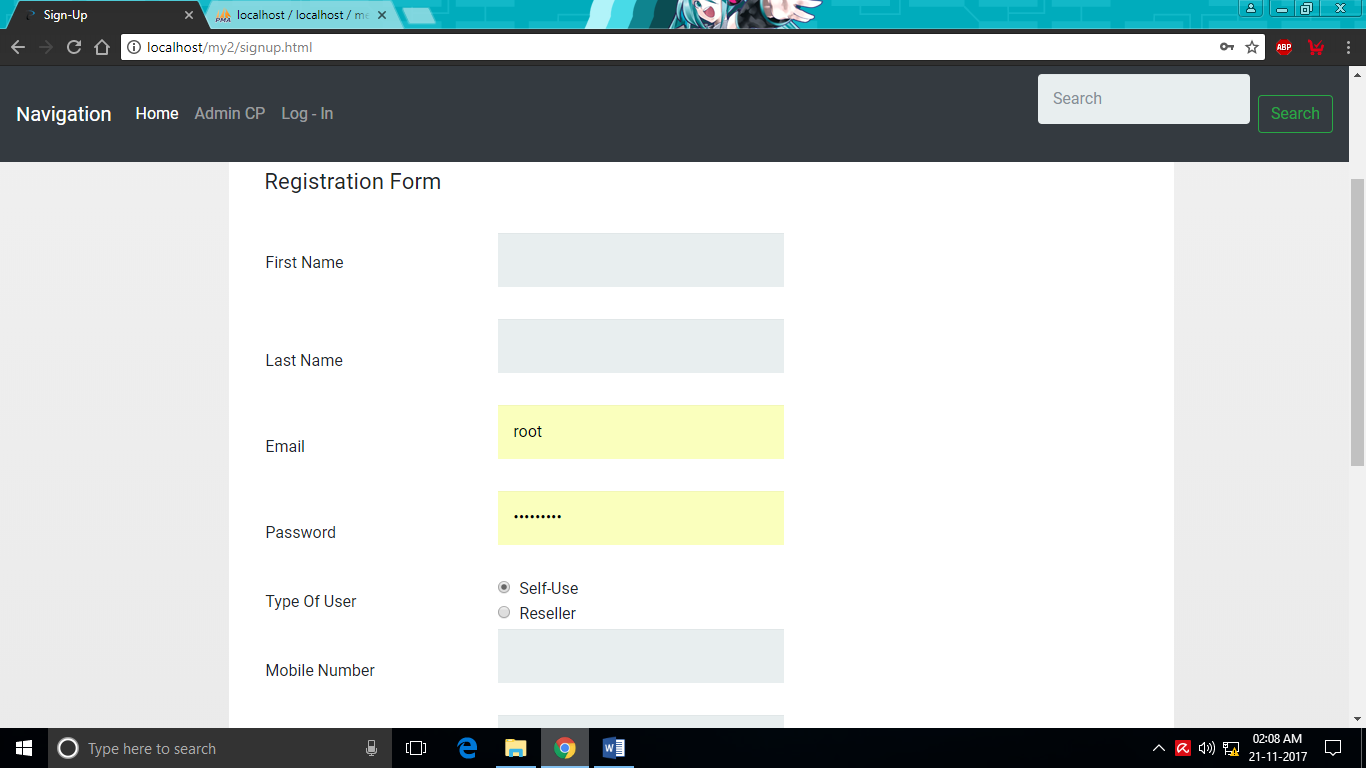


1. Log-In Page



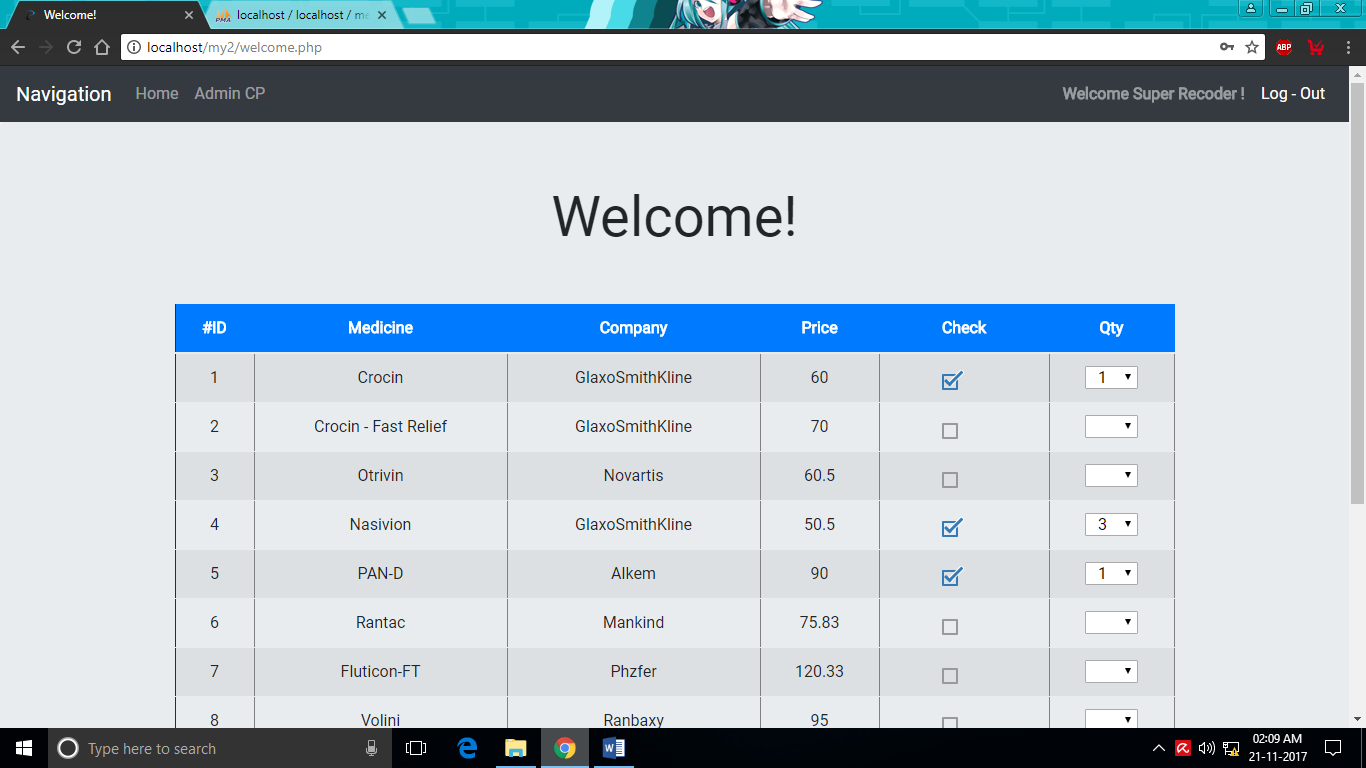
1. Sign Up Page

User is mandated to fill all required fields such as Name, Email, Phone No, Address, Password etc. On Successful creation of user, he/she is redirected to log in page.



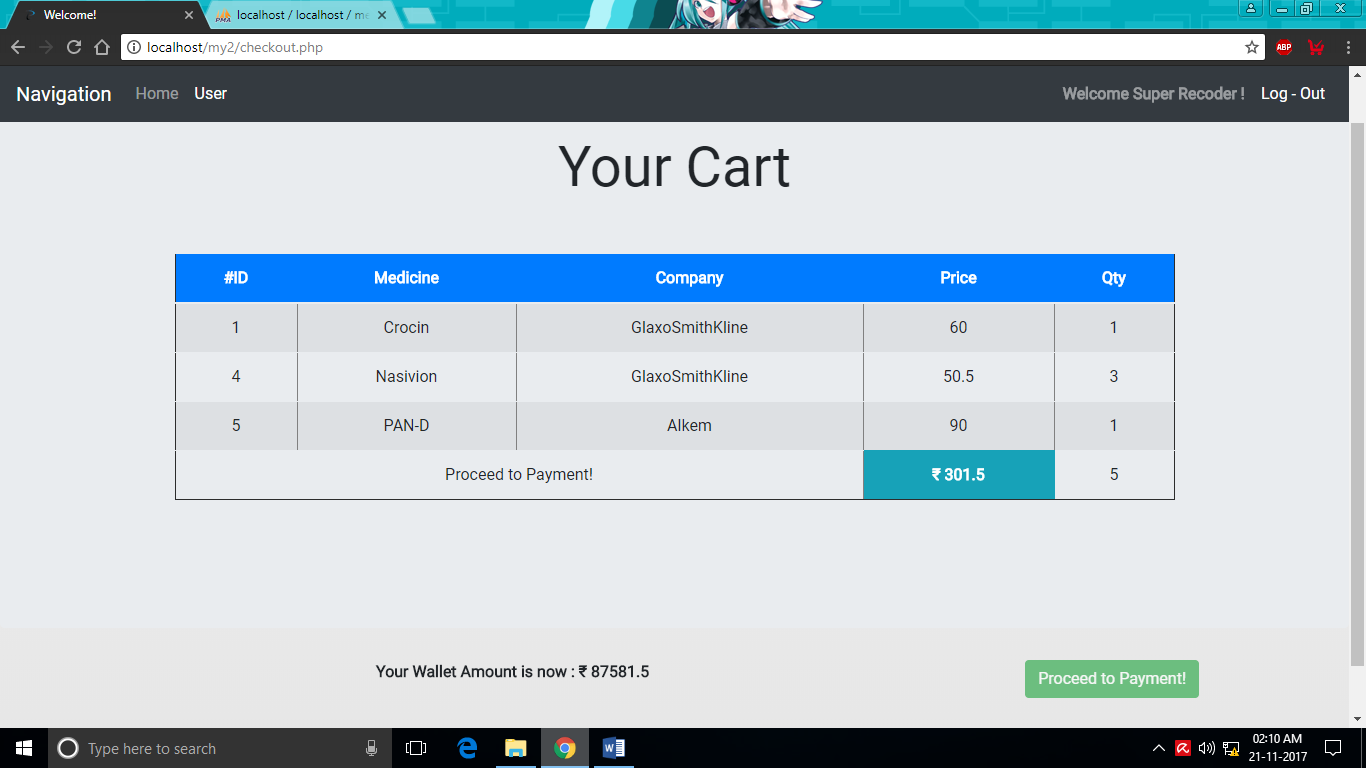
1. Welcome Page

This is where user can select all medicines he/she needs and place order for the same. It allows to select all medicines along with quantity. (Quantity is limited to 5 only for safety reasons)



1. Checkout Page

This is the final checkout page. It shows the currently placed order along with total price. User can pay via online Wallet (as of now). The money is deducted from his/her balance. After clicking “Proceed to Payment”, it shows the final wallet amount left. The Order is placed. He/She may proceed to place further orders or log-out.



Code Snippets

The Below mentioned Code’s are just a few of the codes that were used while creating the Tables / Triggers / Transactions. For the comprehensive code, refer to “Medicine\_SQL.sql”.

1. **Tables**

* Medicine Table

CREATE TABLE `medicine` (

`M\_ID` int(11) NOT NULL,

`Name` varchar(45) NOT NULL,

`Dosage` varchar(45) DEFAULT NULL,

`Price` float DEFAULT NULL,

`Info` varchar(500) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `medicine`

ADD PRIMARY KEY (`M\_ID`),

ADD UNIQUE KEY `M\_ID\_UNIQUE` (`M\_ID`);

ALTER TABLE `medicine`

MODIFY `M\_ID` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=11;

* Order Table

CREATE TABLE `order` (

`O\_ID` int(11) NOT NULL,

`T\_ID` int(11) NOT NULL,

`time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`status` varchar(45) NOT NULL DEFAULT 'pending'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `order`

ADD PRIMARY KEY (`O\_ID`),

ADD UNIQUE KEY `O\_ID\_UNIQUE` (`O\_ID`),

ADD UNIQUE KEY `T\_ID\_UNIQUE` (`T\_ID`);

ALTER TABLE `order`

MODIFY `O\_ID` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=65;

* Medicine Type

CREATE TABLE `medicine`.`med\_type` (

`M\_ID` INT NOT NULL,

`type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`M\_ID`, `type`),

CONSTRAINT `M\_ID`

FOREIGN KEY (`M\_ID`)

REFERENCES `medicine`.`medicine` (`M\_ID`)

ON DELETE CASCADE

ON UPDATE CASCADE);

* Order User

CREATE TABLE `medicine`.`order\_user` (

`O\_ID` INT NOT NULL,

`U\_ID` INT NOT NULL,

`transaction\_id` INT NOT NULL,

`time\_transact` TIMESTAMP NULL,

`mode\_pay` VARCHAR(45) NOT NULL,

PRIMARY KEY (`U\_ID`, `O\_ID`),

UNIQUE INDEX `U\_ID\_UNIQUE` (`U\_ID` ASC),

UNIQUE INDEX `O\_ID\_UNIQUE` (`O\_ID` ASC),

UNIQUE INDEX `transaction\_id\_UNIQUE` (`transaction\_id` ASC),

CONSTRAINT `checkOID`

FOREIGN KEY (`O\_ID`)

REFERENCES `medicine`.`order` (`O\_ID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `checkUID3`

FOREIGN KEY (`U\_ID`)

REFERENCES `medicine`.`user` (`U\_ID`)

ON DELETE CASCADE

ON UPDATE CASCADE);

1. **Triggers**

Note: Most Triggers have their BEFORE UPDATE counterpart as well. All the Triggers are mentioned in the “Triggers, Transaction, User.docx” and the exhaustive code can be found in “Medicine.sql”

* Make sure that medication does not contraindicates itself

DROP TRIGGER IF EXISTS `medicine`.`contraindication\_BEFORE\_INSERT`;

DELIMITER $$

USE `medicine`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `medicine`.`contraindication\_BEFORE\_INSERT`

BEFORE INSERT ON `contraindication`

FOR EACH ROW

BEGIN

IF (NEW.C\_ID1 = NEW.C\_ID2) THEN

SIGNAL sqlstate '45000' SET MESSAGE\_TEXT='Medication cannot contraindicate itself';

END IF;

END$$

DELIMITER ;

* Make Sure that Medicine don't cause same SideEffect as the Treatment

-- DROP TRIGGER IF EXISTS `medicine`.`side\_effects\_BEFORE\_INSERT`;

DELIMITER $$

USE `medicine`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `medicine`.`side\_effects\_BEFORE\_INSERT` BEFORE INSERT ON `side\_effects` FOR EACH ROW

BEGIN

IF (EXISTS(SELECT \* FROM treats t WHERE (t.M\_ID=NEW.M\_ID) AND (t.S\_ID = NEW.S\_ID))) THEN

SIGNAL sqlstate '45000' SET MESSAGE\_TEXT='Medicine cannot cause same Side Effect as the Treatment!';

END IF;

END$$

DELIMITER ;

* User Integrity

DROP TRIGGER IF EXISTS `medicine`.`user\_BEFORE\_INSERT`;

DELIMITER $$

USE `medicine`$$

CREATE DEFINER=`root`@`localhost` TRIGGER `medicine`.`user\_BEFORE\_INSERT` BEFORE INSERT ON `user` FOR EACH ROW

BEGIN

IF (timestampdiff(year, NEW.DOB, curdate()) < '13' ) THEN

Signal sqlstate '45000' set message\_text='You Must be Over 13 Years of Age!';

END IF;

IF (NEW.walletamt < 0) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='Wallet Amount cannot be less than 0';

END IF;

IF ((NEW.type != 'normal')AND(NEW.type != 'admin')AND(NEW.type != 'dbm')) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='Invalid User Type';

END IF;

IF ((NEW.gender != 'M')AND(NEW.gender != 'F')AND(NEW.gender != 'O')) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='Invalid Gender';

END IF;

IF (NEW.email NOT LIKE '\_%@\_%.com') THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='Invalid Email';

END IF;

END$$

DELIMITER ;

1. **Stored Procedures and Functions (Along with Transactions)**

* Add Cart Procedure

delimiter $$

CREATE PROCEDURE addCart(IN oid INT, IN M\_id\_x INT, IN q INT)

BEGIN

START TRANSACTION;

IF (EXISTS(SELECT \* FROM `order\_medicine` WHERE O\_ID = oid AND M\_ID = M\_id\_x)) THEN

UPDATE `order\_medicine` SET quantity = q WHERE O\_ID = oid AND M\_ID = M\_id\_x;

UPDATE `stock` SET quantity = quantity - q WHERE M\_ID = M\_id\_x;

ELSE

INSERT INTO `order\_medicine` VALUES (oid, M\_id\_x, q);

UPDATE `stock` SET quantity = quantity - q WHERE M\_ID = M\_id\_x;

END IF;

COMMIT;

END $$

delimiter ;

* Delete Cart Procedure

delimiter $$

CREATE PROCEDURE delCart(IN oid INT, IN M\_id\_x INT, IN q INT)

BEGIN

START TRANSACTION;

DELETE FROM `order\_medicine` WHERE O\_ID = oid AND M\_ID = M\_id\_x;

UPDATE `stock` SET quantity = quantity + q WHERE M\_ID = M\_id\_x;

COMMIT;

END $$

delimiter ;

* Get User’s Order ID

CREATE FUNCTION get\_User\_OrderId() RETURNS INT

BEGIN

DECLARE oid INT;

SET oid = (SELECT O\_ID FROM `order` ORDER BY `O\_ID` DESC LIMIT 1);

RETURN oid;

END $$

delimiter ;

* Insert User ID, Order ID and set Transaction ID

delimiter $$

CREATE PROCEDURE Insert\_User\_OrderId()

BEGIN

START TRANSACTION;

INSERT INTO `order` (T\_ID) VALUES ('0');

UPDATE `order` SET `T\_ID` = get\_User\_OrderId() WHERE `T\_ID` = '0';

COMMIT;

END $$

delimiter ;

CALL Insert\_User\_OrderId();

* Insert into Order Table

delimiter $$

CREATE PROCEDURE order\_insert(IN oid INT, IN uid INT)

BEGIN

START TRANSACTION;

INSERT INTO `order\_user` (O\_ID, U\_ID, transaction\_id) VALUES (oid, uid, oid);

COMMIT;

END $$

delimiter ;

1. **Queries used to fetch data**

* To retrieve data from ‘Medicine Contraindication’

SELECT cname FROM components WHERE components.C\_ID =

(SELECT C\_ID2 AS `cnamereq` FROM medicine m, composition cmp, components c, contraindication cin

WHERE m.M\_ID = 7 AND m.M\_ID = cmp.M\_ID AND cmp.C\_ID = c.C\_ID AND c.C\_ID = cin.C\_ID1)

UNION

SELECT cname FROM components WHERE components.C\_ID =

(SELECT C\_ID1 AS `cnamereq` FROM medicine m, composition cmp, components c, contraindication cin

WHERE m.M\_ID = 7 AND m.M\_ID = cmp.M\_ID AND cmp.C\_ID = c.C\_ID AND c.C\_ID = cin.C\_ID2)

* Query used for Search

SELECT m.M\_ID as `#ID`, m.Name as `Medicine`, c.Name as `Company`, m.Price

FROM medicine m, company c, manufacturer mf

WHERE

m.M\_ID = mf.M\_ID AND

mf.C\_ID = c.C\_ID WHERE m.Name LIKE '%keyword%' ORDER BY m.M\_ID LIMIT 25;

**END OF REPORT**