



**VNR Vignana Jyothi Institute of Engineering and Technology
(Affiliated to J.N.T.U, Hyderabad)**

Bachupally(v), Hyderabad, Telangana, India.

QR Payments System.

A course based project submitted in complete requirements for the award of the degree of

BACHELORS OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING (AI&ML).

Submitted By:

Name	Roll No
K. Santhosh Kumar	20071A6620
M. Sai Ritish Reddy	20071A6626
M. Uday Krishna	20071A6627
M. Harikesh	20071A6629

Under the guidance of

Mrs. Sayeeda Khanum Pathan

Assistant Professor

Department of Computer Science & Engineering (AIML & IoT)



VNR Vignana Jyothi Institute of Engineering and Technology
(Affiliated to J.N.T.U, Hyderabad)

Bachupally(v), Hyderabad, Telangana, India.

CERTIFICATE

This is to certify that Mr. K. Santhosh Kumar (20071A6620), Mr. M. Sai Ritish Reddy (20071A6626), Mr. M. Uday Krishna (20071A6627), Mr. M. Harikesh (20071A6629) have completed their course project work at Department of Computer Science & Engineering (AIML & IoT) of VNR VJIET, Hyderabad entitled "**QR PAYMENTS SYSTEM**" in complete fulfilment of the requirements for the award of B. Tech degree during the academic year 2021-2022. This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

Dr. Sayeeda Khanum Pathan
Assistant Professor

Department of CSE (AIML & IoT)

VNR VJIET

DECLARATION

This is to certify that our project report titled “**QR PAYMENTS SYSTEM**” submitted to Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology in complete fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Engineering (AI&ML) is a bonafide report to the work carried out by us under the guidance and supervision of Mrs.Sayeeda Khanum Pathan Assistant Professor, Department of Computer Science and Engineering (AIML & IoT), Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology. To the best of our knowledge, this has not been submitted in any form to other university or institution for the award of any degree or diploma.

K. Santhosh Kumar
20071A6620
AIML

M. Sai Ritish Reddy
20071A6626
AIML

M. Uday Krishna
20071A6627
AIML

M. Harikesh
20071A6629
AIML

ACKNOWLEDGEMENT

Over a span of Two years, VNR VJIET has helped us transform ourselves from mere amateurs in the field of Computer Science into skilled engineers capable of handling any given situation in real time. We are highly indebted to the institute for everything that it has given us. We would like to express our gratitude towards the principal of our institute, **Dr. Challa Dhanunjaya Naidu** and the Head of the Computer Science & Engineering (AIML & IoT) Department, **Dr. N. Sandhya** for their kind co-operation and encouragement which helped us complete the project in the stipulated time. We thank her for her guidance, constant supervision and for providing necessary information to complete this project. Our thanks and appreciations also go to all the faculty members, staff members of VNR VJIET, and all our friends who have helped us put this project together.

ABSTRACT

In an initiation to establish digitalised India one of the vastly used inventions came is online payment systems referred as Unified Payments Interface (UPI) which created cashless digitalised transactions. Payments through UPI are being done with the usage of internet and one often fails to make payments due to an improper internet connectivity at the places they wish to pay. This could be even more useful if these payment systems work without an internet connectivity i.e., development of an offline payment system.

The QR Payments system can be considered as a mini offline payment system which is developed to digitalise the transactions made in the canteen of VNR VJIET which works without an internet connectivity. Students often face the problem of internet connectivity in the college because of which they fail to make their UPI payments to the canteen, the QR Payments system resolves this problem as it helps students by not asking any internet connectivity to make payment and payments are done just by scanning a QR code generated which operates the database that contain details of user, stalls, food items and prices.

Students are required to deposit money with the college management, the payments made by the students will be appended in the respective databases designed for college management, student and food stalls.

INDEX

Contents	Page No
1. Introduction	1
2. Literature	3
3. Requirements	8
4. Architecture	10
5. Modules	11
6. Implementation	15
7. Conclusion	31
8. ER Diagrams	32
9. References	33

Introduction

This project “**QR Payments System**” is made with the motive of enabling user to make digital payments when he is offline. This project is restricted to the premises of our college canteen area. We are motivated to solve this problem when we faced the problem in making our UPI payments while ordering food in college canteen. Due to lack in internet services making a UPI payment is very hard.

The survival of this project is with few assumptions like:

- The college maintains student account where user deposit his pocket money.
- The food stall owners were provided with internet connectivity (Wi-Fi) by the college.
- Student has an all time updated QR Generation app on his device to make payments successful.

This project has two distinct products and their implementation. The first product is a **QR Generator** used by students. The second product is a **QR Scanner** used by food sellers.

How students use QR Generator?

Students use the QR Generator app (as of now is a hosted web application) to generate QR codes. This QR codes represent the user's food orders. User while having the internet connectivity gets himself logged in to this application. And even when he is offline user can generate QR codes. This stores data regarding food stalls, food items and it's prices. User can select his food quantity and enter pin while generating the QR code. This pin will be encrypted before encoded in the QR code. Thus user can make his food order as a QR code. Username (his roll no) is included in the QR code as he is logged into the app. This is made with the intention to replace any order taking touch panels to decrease equipment cost.

How sellers use QR Scanner?

The food sellers will have the QR Scanners installed at their desk. This helps in taking the student food orders. This QR Scanner will decode the QR code and understand the user food order and extract the user credentials and validate the user. After validating the user, the system validates the QR code based on it's time of generation and try to re-use status. Upon successful validation of QR code on checking for sufficient funds in user account transaction will be completed.

Literature

At this section we understand the development segments of QR code, QR code generator and QR scanner in deep. And the modules which were used in developing the different features of this project are as follows:

► **QR code:**

- pyqrcode (python module to generate a QR code).

► **QR Generator Application:**

► **Front-End:**

- HTML5.
- Bootstrap.
- CSS.
- JavaScript.

► **Server-APIs:**

- Flask.

► **QR Scanner Interface:**

- pyzbar (a python module to scan QR codes).
- opencv-python.

► **Database & application connectivity:**

- MySQL (a R-DBMS).
- Python-sql-connector (to connect to database).

► **pyqr code:**

1. This is a python module which is used to make QR codes.
2. It provides decent inbuilt functions to encode data inside a QR code.
3. The create function of this module creates a QR code by encoding the given data.
4. And the generated QR code is obtained as a png image with desired name.

► **HTML5:**

1. HTML5 is a markup language which is used to create the content that can be rendered as web pages.
2. All the required web pages for the QR Generator were built using the HTML5 markup language.

► **Bootstrap:**

1. Bootstrap is a framework which provides a vast area of builtin styles.
2. It is useful in developing dynamic styles easily.
3. It helps by providing many useful pre-designed styles.

► Cascading style sheets:

1. Cascading style sheets is a styling language which is used to add styles to the content created by HTML.
2. It makes the web content user attractive.
3. It is used to make custom styles apart of those provided by bootstrap.

► JavaScript:

1. JavaScript is the only programming language used at the user side.
2. It is used in developing dynamic nature to the web content.
2. It is used in performing DOM (Document Object Manipulation).

► Flask:

1. Flask is a python micro framework which is used to create server APIs in python.
2. Using flask we can perform serving of web pages based on user request.
3. By using flask Jinja templating data is dynamically rendered onto web pages.
4. This helps in making an integration between the python code and web pages.

► **pyzbar:**

1. pyzbar is a python module used in reading the QR codes and barcodes.
2. It is built on top of the zbar module.
3. zbar module is developed in c++ but pyzbar is built on top of the zbar to support python.

► **opencv-python:**

1. opencv-python is a python module which helps in performing the operations between python code and system camera.
2. To scan the QR code we use system camera using opencv-python module.

► **MySQL:**

1. SQL has high performance.
2. Light and fast comparatively with other Databases.
3. Open Source.
4. On-Demand Scalability.

► **python-sql-connector:**

1. This is a python module used for making the python code communicate with the database built on SQL.
2. This is useful for storing the data got from web page into the database and fetching data that to be rendered onto the web page.
3. It helps in writing the SQL queries for all database related operations with in python.

Requirements

Functional Requirements:

1. Login.
2. Food stall data.
3. Pin encryption.
4. Single time usage of a QR code.
5. QR must be valid for limited time.
6. Download QR code.
7. Validate User.
8. Check for sufficient funds from user.
9. Complete transaction only if all database transactions are done without fail.
10. Generate Bill.

Non-Functional Requirements:

1. Usability:

1. we get the response within seconds.
2. The software must have a simple, user friendly interface so customers can save time and confusion.

2. Reliability:

1. The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

3. Supportability:

1. The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform which is having python interpreter built into the system.

4. Implementation:

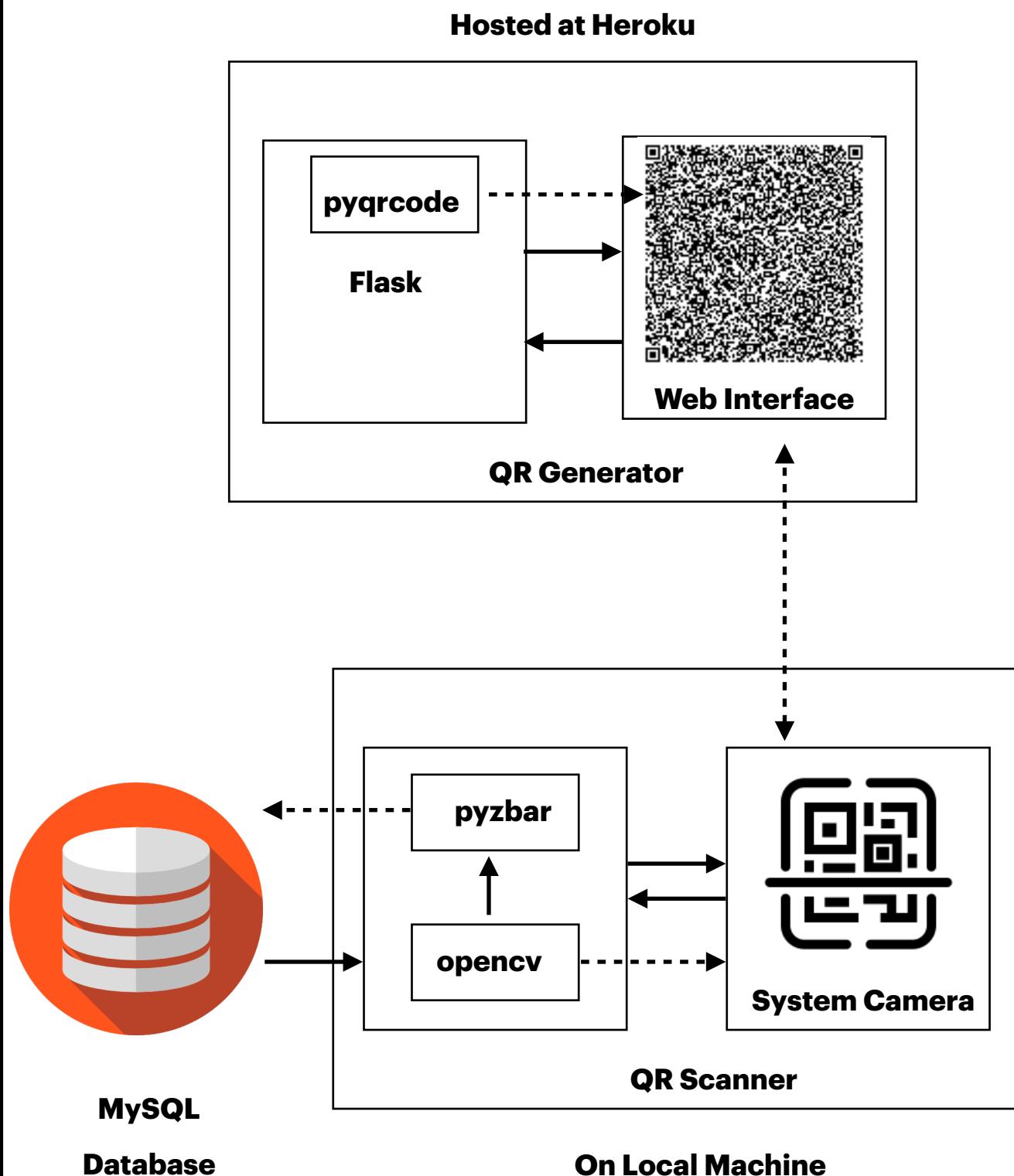
1. The system is implemented in a cloud based hosting service called heroku.

5. Interface:

1. The user interfaces are the web pages which are developed using the HTML and other styling frameworks.
2. The interface design is aimed at a flexible front-end communication to provide the user with clear information in navigating a user-friendly interface is planned.

Architecture

Representation of Control-Flow:



Modules

Modules of QR Generation Section:

This section has four modules:

1. View Stalls.
2. Order Food.
3. Enter pin.
4. View QR code.

1. View Stalls:

1. This section allows the user to see all the stalls available to order food.
2. User can select any of the stalls and view the food available in that stall through order food module.

2. Order Food:

1. This section provides the details of food available at a stall and their base price.
2. This section allows user to order food by modifying the quantity of food item user wants to order.
3. This section also displays a sub-total (per each selected food item) and grand total (for all selected items) as a bill.

3. Enter Pin:

1. This section allows user to enter his 6-digit security pin.
2. This pin is encrypted before encoded as QR code.

4. View QR code:

1. This section displays the QR code generated with the data:
 1. Username.
 2. Encrypted pin.
 3. Foodstall name.
 4. Food ordered.
 5. Food prices.
 6. QR generation day & time.
2. This also provides the feature to download the generated QR code.

Modules of QR Scanner Section:

This section has four modules:

1. QR code validation.
2. User validation.
3. Fund validation.
4. Complete Transaction.

1. QR code validation:

1. QR code validation is done based on 2 criteria:
 1. Single time usage.
 2. Generation time.
2. Every QR code on successful usage appended in QR records table, which helps in identifying it when user tries to use the same QR twice.
3. Every QR code has a time of generation in it, and from time of generation it is valid only for 15 minutes.

2. User validation:

1. User is validated by looking for username in users details table.
2. On finding a matching username password is decrypted and matched with password that is in database.

3. Fund validation:

1. By comparing the food prices in the QR and prices in the data base for selected food items we generate the bill.
2. Upon generating the bill we look for sufficient amount of funds in the user's account.

4. Complete Transaction:

1. Upon successful completion of all the above procedures we deduct the bill amount from user's account and credit in that stall account for that day.
2. Then we append this food order details in that respective stall orders table for future reference.
3. Then we append QR information in QR Records table to prevent multiple usage of same QR code.
4. All the transactions are committed only if all are completed successfully. Otherwise, we rollback and display corresponding problem.
5. On committing the transactions we display bill complete transaction.

Implementation

This project has two different products to be implemented. The QR Generator is hosted at heroku cloud hosting services and can be accessed using the link generated. The flask server APIs are the control flows between different request url generated by the user.

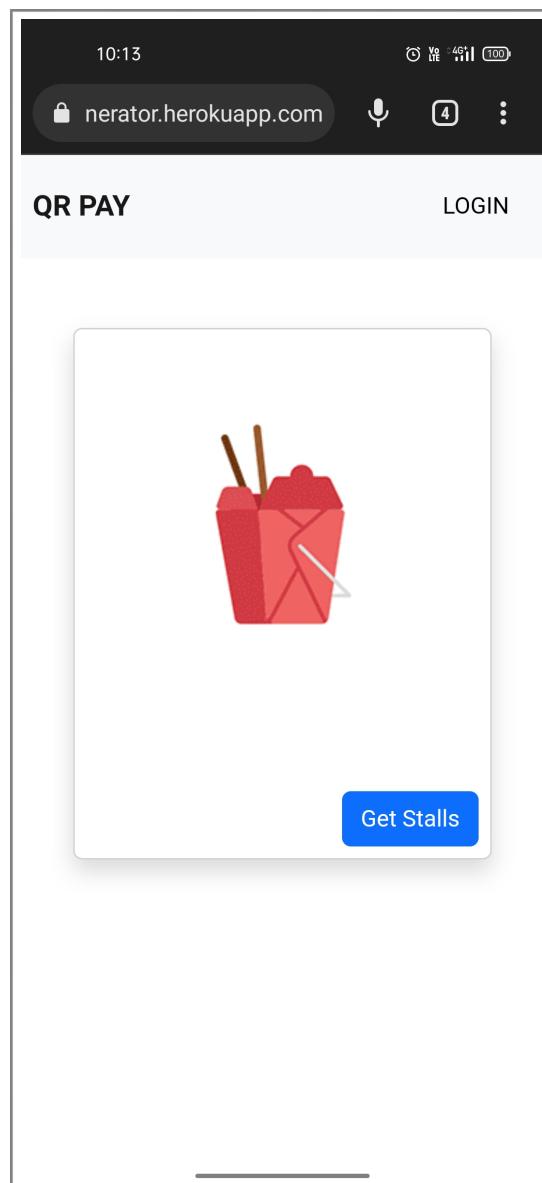
While the QR Scanner is executed on a remote laptop as we chose to use a Relational Database Management System like MySQL we had it locally to perform database operations.

The below are the rendered web pages of QR Generator:

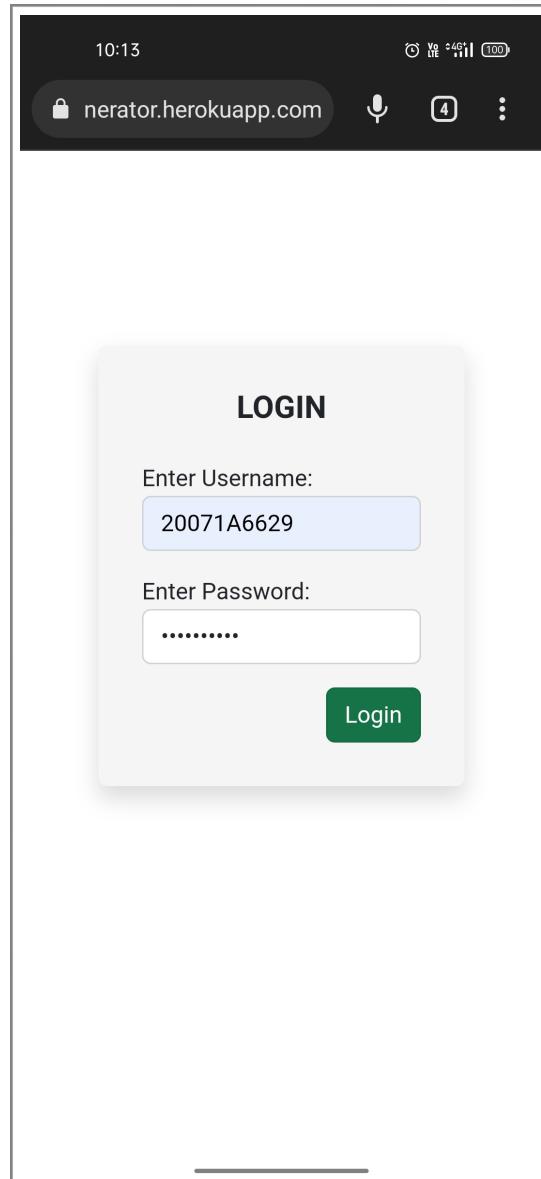
Hosted link: <https://qrps-qr-generator.herokuapp.com/>

The QR Generator App:

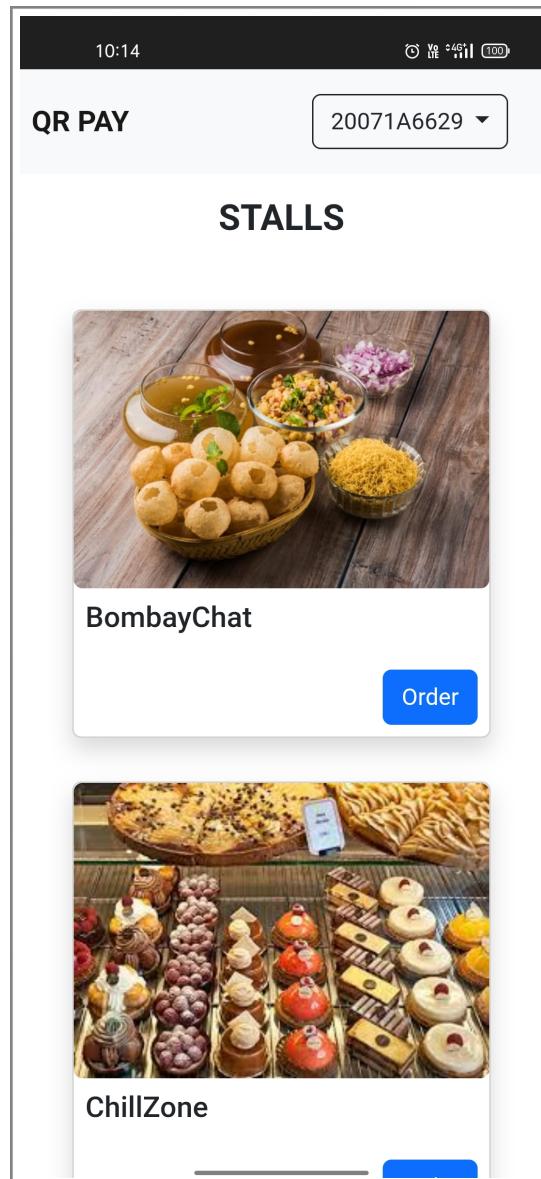
Home page:

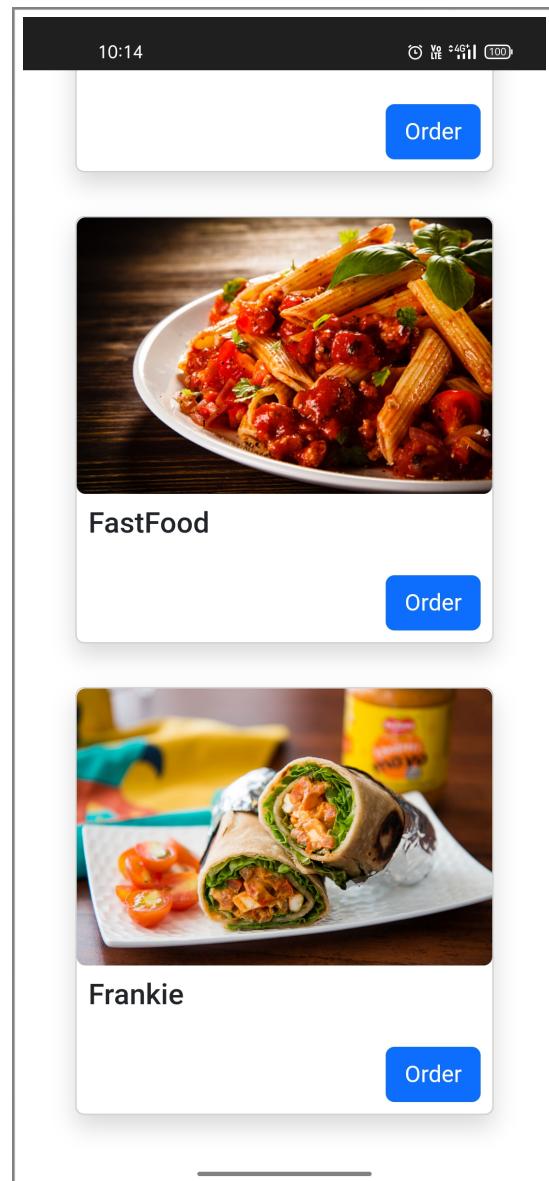


Login Page:



Stalls Page:





Order Food Page:

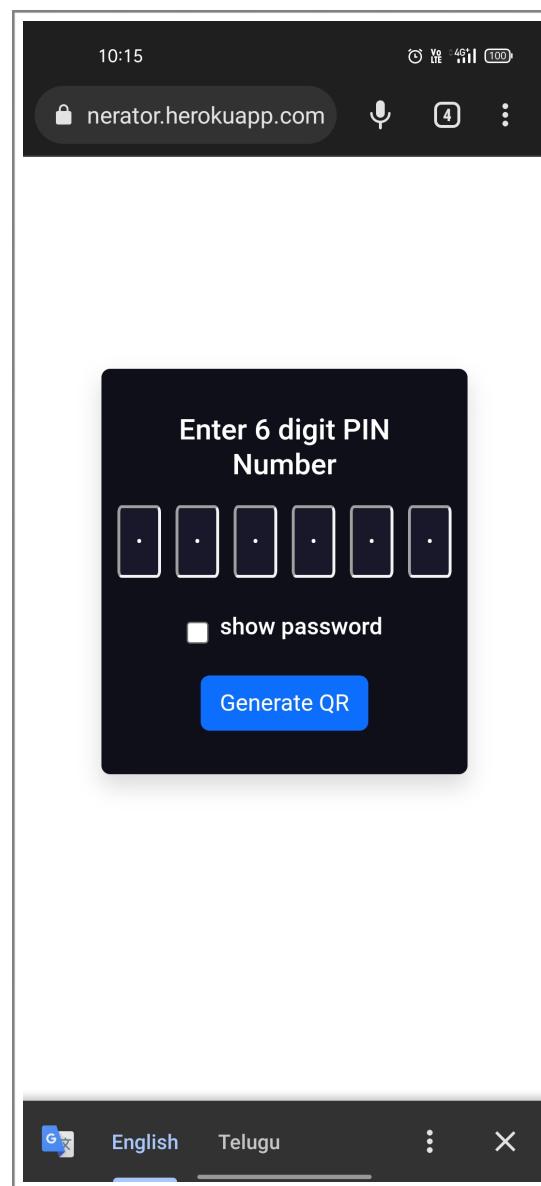
10:15

20071A6629

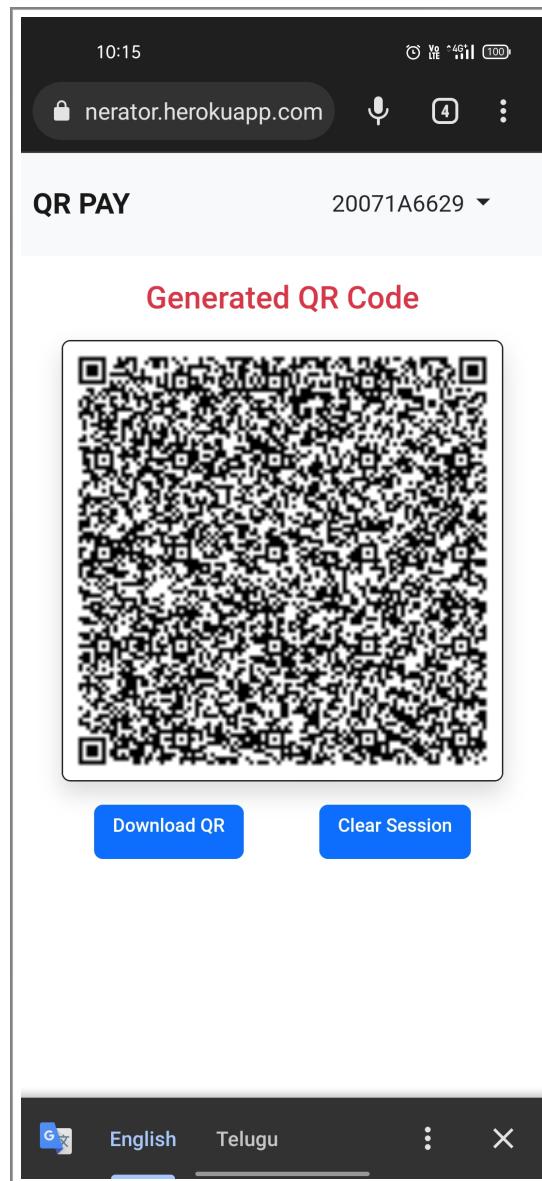
Item	Price	Quantity	Sub Total
Pani Puri	20	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	20
Ragada Chat	30	<input type="button" value="-"/> <input type="button" value="0"/> <input type="button" value="+"/>	0
Samosa Chat	30	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	30
Pav Bhaji	60	<input type="button" value="-"/> <input type="button" value="0"/> <input type="button" value="+"/>	0
Masala Puri	30	<input type="button" value="-"/> <input type="button" value="0"/> <input type="button" value="+"/>	0
Bhel Puri	30	<input type="button" value="-"/> <input type="button" value="0"/> <input type="button" value="+"/>	0
Grand Total			50

VERIFY

Enter Pin Page:

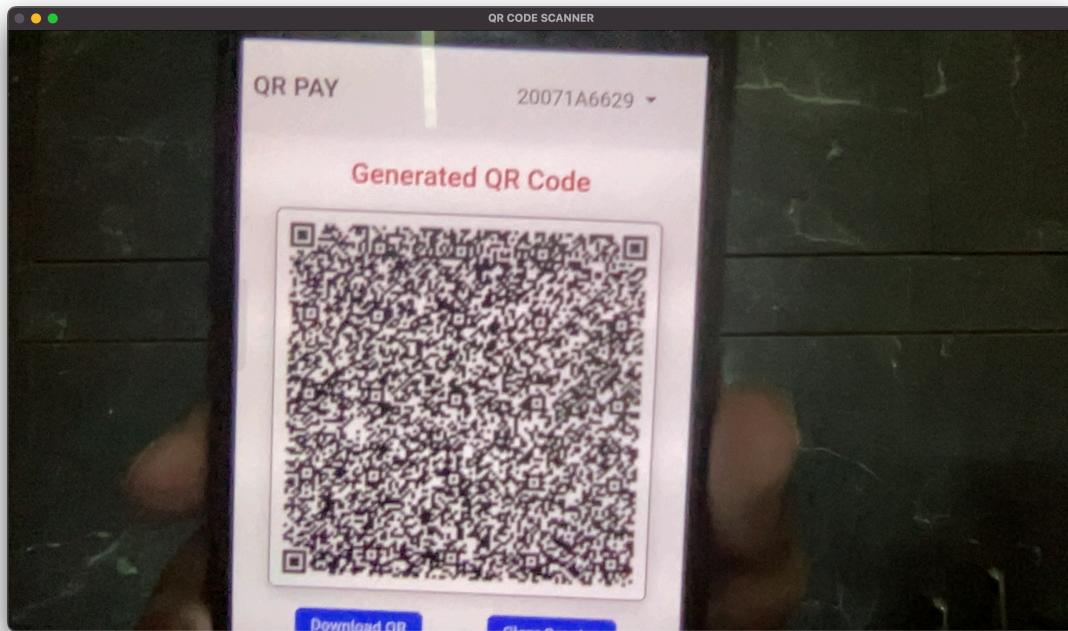


QR Display Page:



The QR Scanner Module:

QR Scanner:



QR Scanner Result (scanning QR for first time):

A screenshot of a code editor showing the "app.py" file for the QR Code Scanner. The code is written in Python and includes functions for validating QR codes and users. It prints a message indicating a successful transaction. The code editor interface shows the file path, code snippets, and a terminal tab.

QR Scanner Result (scanning QR for second time):

```
app.py > home
  32     print(data, end='\n')
  33
  34     if isinstance(data, dict):
  35
  36         # initialize data components
  37         initialize_data(data)
  38
  39         # validate qr code
  40         qr_status, message = is_qr_valid(userData, metaData)
  41
  42         print(message)
  43         valid_user=False
  44
  45         if qr_status:
  46
  47             # validate user
  48
  49             valid_user = is_user_valid(userData)
  50
  51         if valid_user:
  52             print('Valid User')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
zsh
Python
Ln 36, Col 24 Spaces: 4 UTF-8 LF Python 3.9.13 ('venv': venv) Go Live
```

{'userData': {'username': '20071A6629', 'encryptedPassword': 'D_JG?'}, 'stallData': {'stallName': 'BombayChat'}, 'metaData': {'generatedDay': '11/07/2022', 'generatedTime': '10:48:47'}, 'foodData': {'Pani Puri': 1, 'Ragada Chat': 0, 'Samosa Chat': 1, 'Pav Bhaji': 0, 'Masala Puri': 0, 'Bhel Puri': 0}, 'foodPrices': {'Pani Puri': 20, 'Ragada Chat': 30, 'Samosa Chat': 30, 'Pav Bhaji': 60, 'Masala Puri': 30, 'Bhel Puri': 30}}
Generate a new QR Code (as QR Code has been already used)
(venv) mancha.laharikesh@manchala-MacBook-Air QR Code Scanner %

Database Schemas:

Customer Details Table:

MySQL Workbench - Local instance 3306

Administration Schemas Query 3 ChillZone_Orders FastFood_Orders Frankie_Orders FastFood STALLS_CREDITS

SCHEMAS
Filter objects
BombayChat...
ChillZone
ChillZone_Or...
CUSTOMER_...
FastFood
FastFood_Or...
Frankie
Frankie_Ord...
QR_RECORDS
STALLS_CRE...
Views
Stored Proced...
Functions
sys

Object Info Session No object selected

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types

1 • SELECT * FROM `QR-PAYMENTS-SYSTEM`.CUSTOMER_DETAILS;

NAME	USERNAME	PIN	BALANCE
Santhosh Kumar	20071A6620	111111	940
Sai Rithish	20071A6626	654321	1000
Uday Krishna	20071A6627	999999	1000
Harikesh	20071A6629	123456	740
HULL	NULL	NULL	NULL

CUSTOMER_DETAILS 1 10:35:49 SELECT * FROM `QR-PAYMENTS-SYSTEM...` 4 row(s) returned 0.0035 sec / 0.00000...

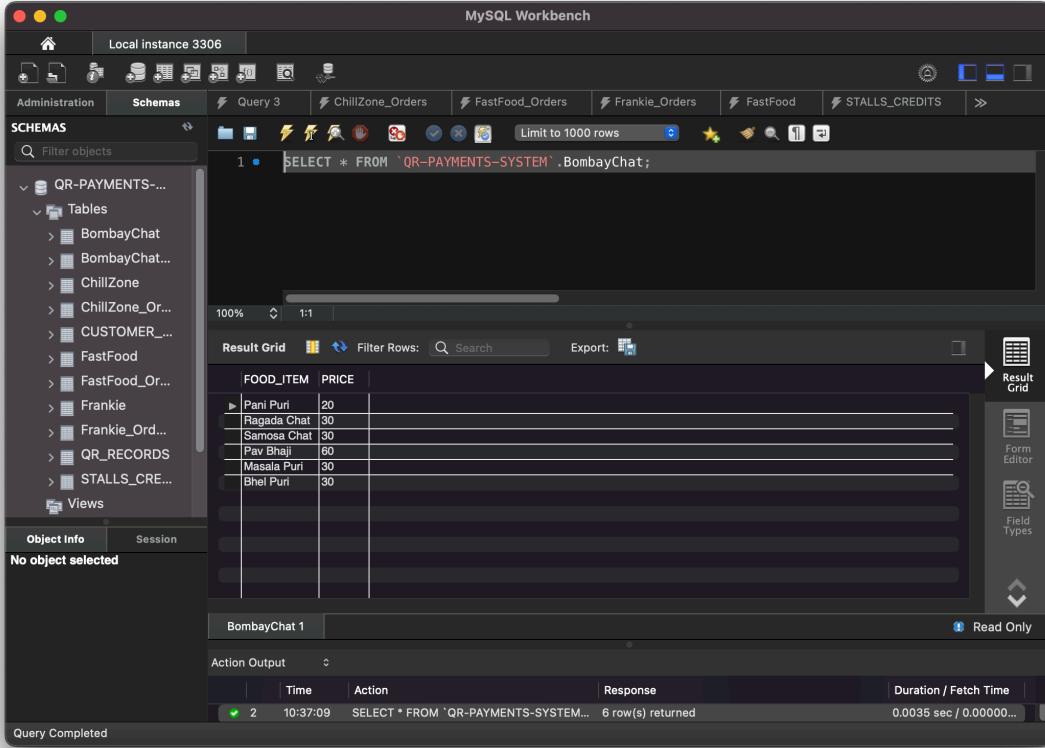
Action Output

Time Action Response Duration / Fetch Time

Query Completed

Food Stalls Tables:

Bombay Chat:



The screenshot shows the MySQL Workbench interface with a dark theme. The left sidebar displays the 'SCHEMAS' tree, which includes the 'QR-PAYMENTS--' schema containing tables like 'BombayChat', 'ChillZone', etc. The main query editor window contains the following SQL query:

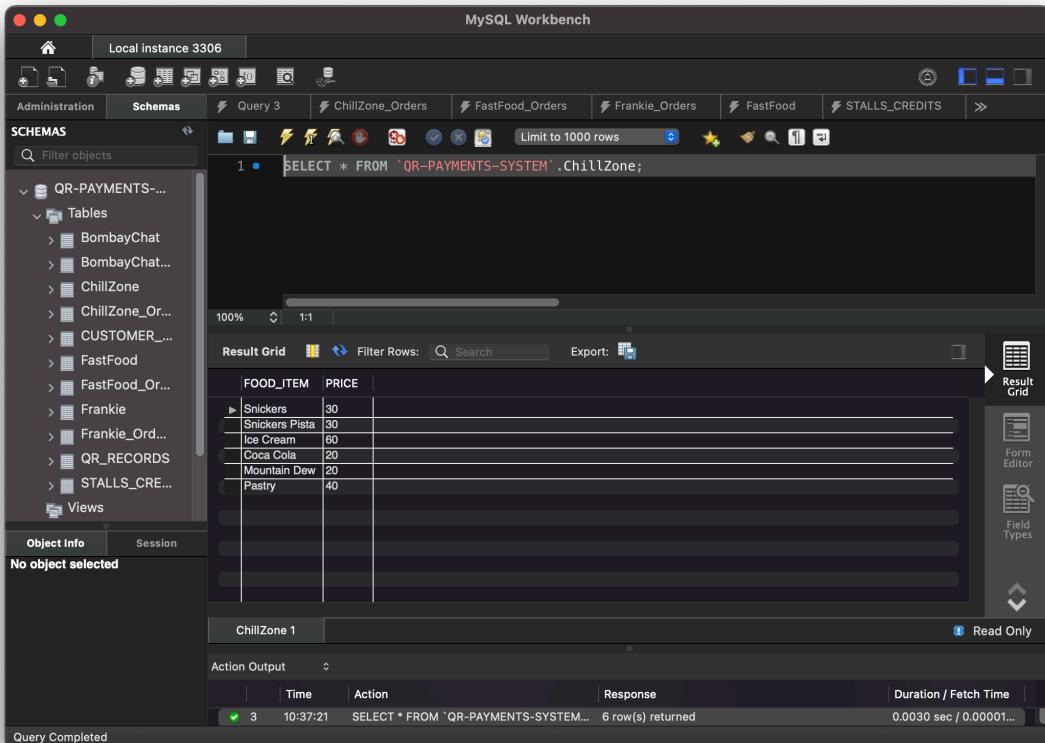
```
1 • SELECT * FROM `QR-PAYMENTS-SYSTEM`.BombayChat;
```

The result grid shows the following data:

FOOD_ITEM	PRICE
Pani Puri	20
Ragada Chat	30
Samosa Chat	30
Pav Bhaji	60
Masala Puri	30
Bhel Puri	30

Below the result grid, the status bar indicates "Query Completed".

ChillZone:



The screenshot shows the MySQL Workbench interface with a dark theme. The left sidebar displays the 'SCHEMAS' tree, which includes the 'QR-PAYMENTS--' schema containing tables like 'ChillZone', 'FastFood', etc. The main query editor window contains the following SQL query:

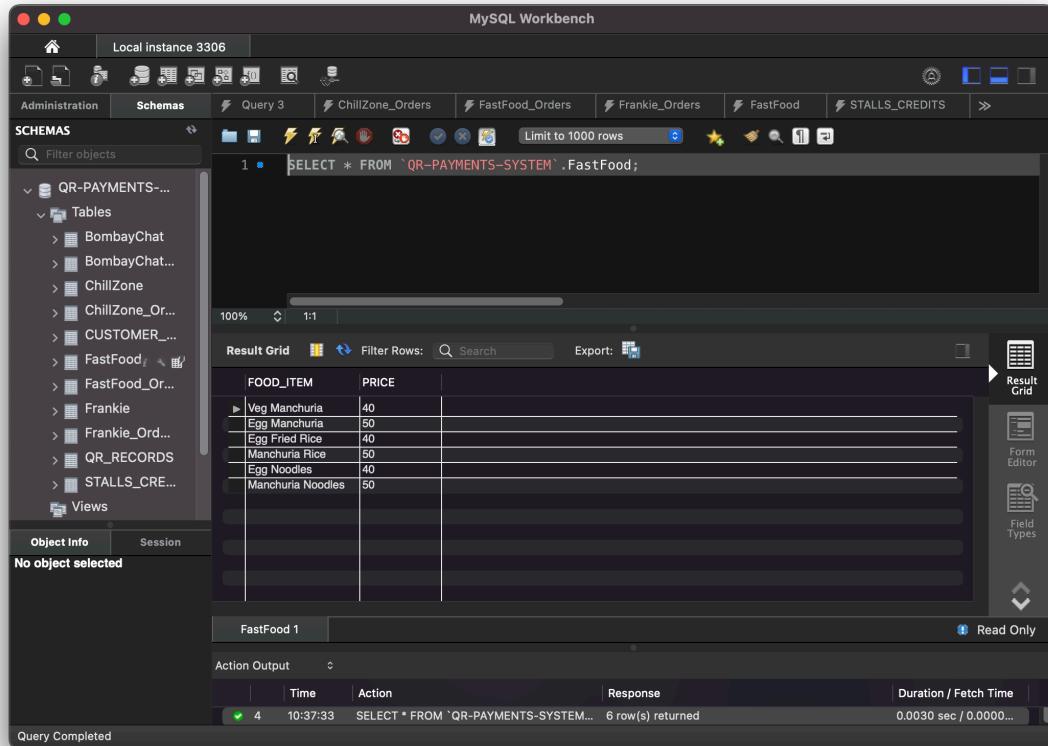
```
1 • SELECT * FROM `QR-PAYMENTS-SYSTEM`.ChillZone;
```

The result grid shows the following data:

FOOD_ITEM	PRICE
Snickers	30
Snickers Pista	30
Ice Cream	60
Coca Cola	20
Mountain Dew	20
Pastry	40

Below the result grid, the status bar indicates "Query Completed".

Fast Food:

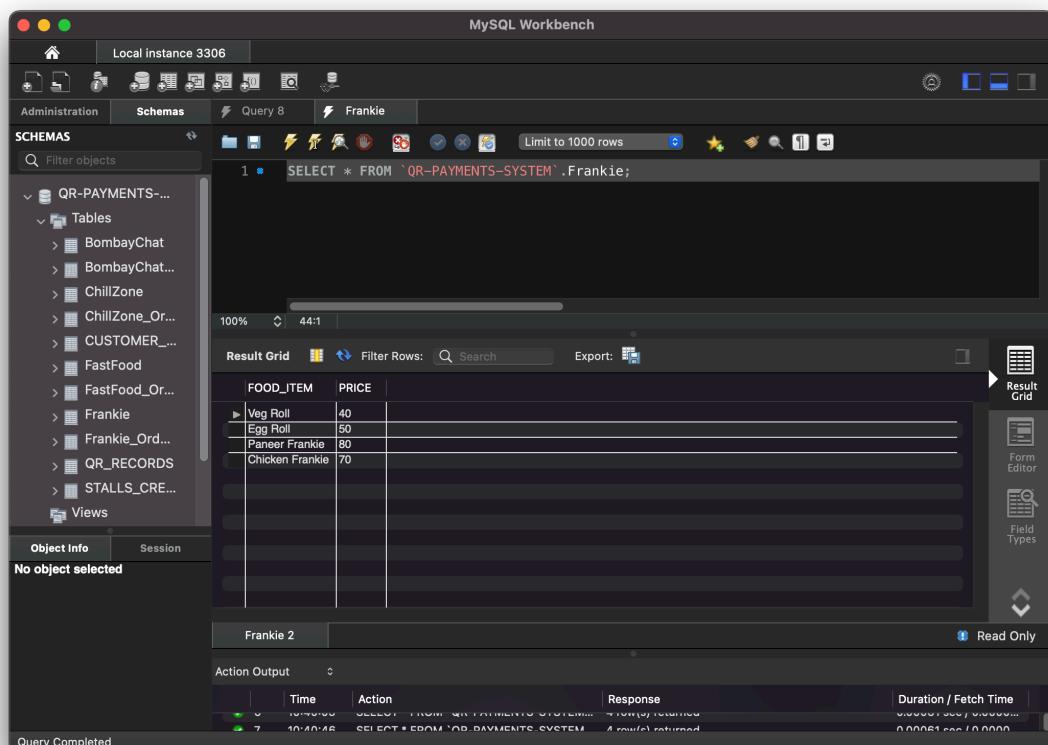


The screenshot shows the MySQL Workbench interface with the 'Local instance 3306' connection selected. The 'Schemas' tab is open, displaying the 'QR-PAYMENTS-...' schema. Under the 'Tables' section, the 'FastFood' table is selected. A SQL query window shows the command: 'SELECT * FROM `QR-PAYMENTS-SYSTEM`.FastFood;'. The result grid displays the following data:

FOOD_ITEM	PRICE
Veg Manchuria	40
Egg Manchuria	50
Egg Fried Rice	40
Manchuria Rice	50
Egg Noodles	40
Manchuria Noodles	50

The status bar at the bottom indicates 'Query Completed'.

Frankie:



The screenshot shows the MySQL Workbench interface with the 'Local instance 3306' connection selected. The 'Schemas' tab is open, displaying the 'QR-PAYMENTS-...' schema. Under the 'Tables' section, the 'Frankie' table is selected. A SQL query window shows the command: 'SELECT * FROM `QR-PAYMENTS-SYSTEM`.Frankie;'. The result grid displays the following data:

FOOD_ITEM	PRICE
Veg Roll	40
Egg Roll	50
Paneer Frankie	80
Chicken Frankie	70

The status bar at the bottom indicates 'Query Completed'.

Bombay Chat Orders Table:

This table stores the details of order details of the Bombay chat food stall.

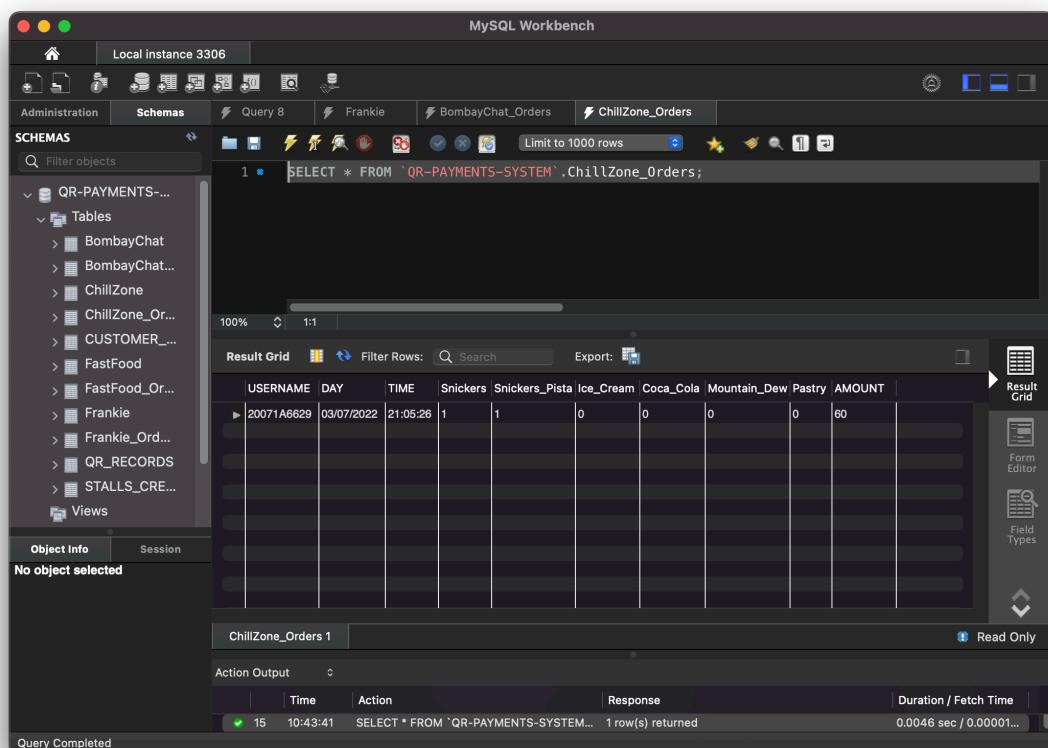
The screenshot shows the MySQL Workbench interface with the following details:

- Session:** Local instance 3306
- Schemas:** QR-PAYMENTS-... (selected)
- Tables:** Shows tables like BombayChat, ChillZone, ChillZone_Or..., CUSTOMER_..., FastFood, FastFood_Or..., Frankie, Frankie_Ord..., QR_RECORDS, and STALLS_CRE...
- Query Editor:** Displays the SQL query: `SELECT * FROM `QR-PAYMENTS-SYSTEM`.BombayChat_Orders;`
- Result Grid:** Shows the results of the query in a grid format. The columns are: USERNAME, DAY, TIME, Pani_Puri, Ragada_Chat, Samosa_Chat, Pav_Bhaji, Masala_Puri, Bhel_Puri, and AMOUNT. The data returned is:

USERNAME	DAY	TIME	Pani_Puri	Ragada_Chat	Samosa_Chat	Pav_Bhaji	Masala_Puri	Bhel_Puri	AMOUNT
20071A6629	03/07/2022	20:57:46	2	1	0	0	0	0	70
20071A6629	05/07/2022	21:56:27	1	1	0	0	0	0	50
20071A6629	06/07/2022	19:40:29	2	0	0	0	0	0	40
20071A6629	09/07/2022	14:18:13	2	0	0	0	0	0	40
20071A6620	09/07/2022	14:22:25	0	2	0	0	0	0	60
- Action Output:** Shows the execution details: Time (10:43:32), Action (SELECT * FROM `QR-PAYMENTS-SYSTEM`...), Response (5 row(s) returned), and Duration / Fetch Time (0.0090 sec / 0.00001...).
- Status:** Query Completed

ChillZone Orders Table:

This table stores the details of order details of the ChillZone Food stall.



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** QR-PAYMENTS-... is selected.
- Tables:** ChillZone_Orders is selected.
- Query Editor:** The query `SELECT * FROM `QR-PAYMENTS-SYSTEM`.ChillZone_Orders;` is run, resulting in 1 row returned.
- Result Grid:** The table structure is as follows:

USERNAME	DAY	TIME	Snickers	Snickers_Pista	Ice_Cream	Coca_Cola	Mountain_Dew	Pastry	AMOUNT
20071A6629	03/07/2022	21:05:26	1	1	0	0	0	0	60

- Action Output:** Shows the query was run at 10:43:41 and took 0.0046 sec / 0.00001...

QR Records Table:

This table stores the records of all valid QR codes.

The screenshot shows the MySQL Workbench interface with the following details:

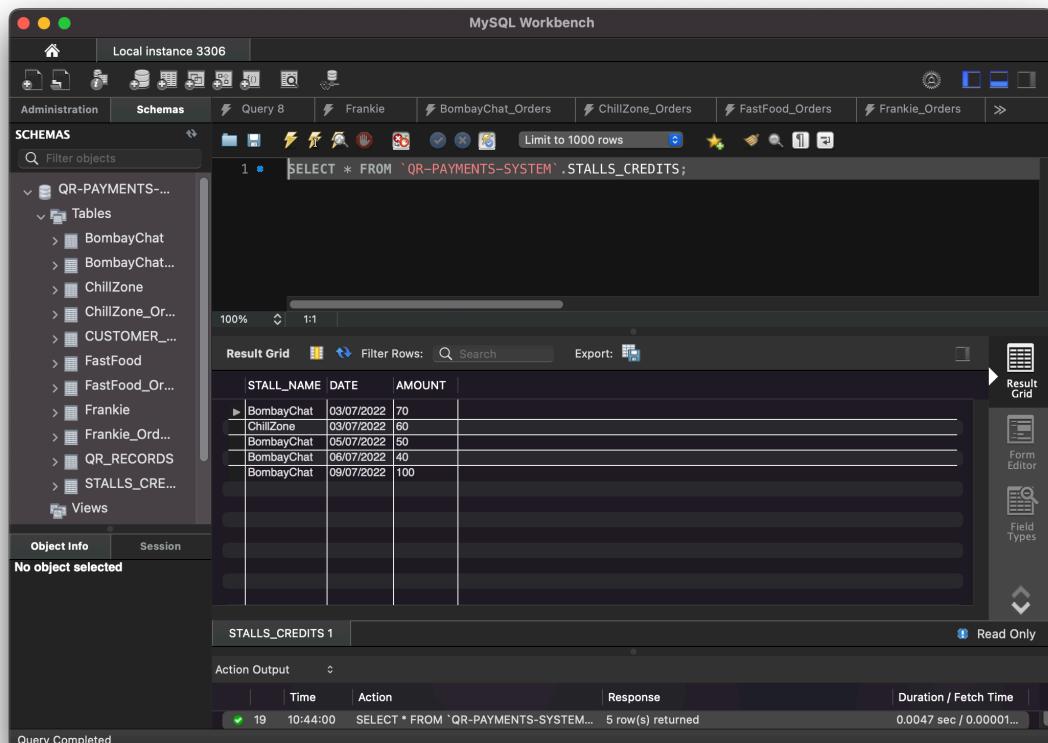
- Schemas:** QR-PAYMENTS--> Tables
- Table:** QR_RECORDS
- Query:** SELECT * FROM `QR-PAYMENTS-SYSTEM` . QR_RECORDS;
- Result Grid:** Displays 6 rows of data:

	USERNAME	STALL_NAME	GENERATED_DAY	GENERATED_TIME	AMOUNT
▶	20071A6629	BombayChat	03/07/2022	20:57:46	70
	20071A6629	ChillZone	03/07/2022	21:05:26	60
	20071A6629	BombayChat	05/07/2022	21:56:27	50
	20071A6629	BombayChat	06/07/2022	19:40:29	40
	20071A6629	BombayChat	09/07/2022	14:18:13	40
	20071A6620	BombayChat	09/07/2022	14:22:25	60

- Action Output:** Shows the query executed and its duration.
- Object Info:** No object selected.

Stall Credits Table:

This table stores the amount credited by student food orders with respective to each day for a particular food stall.



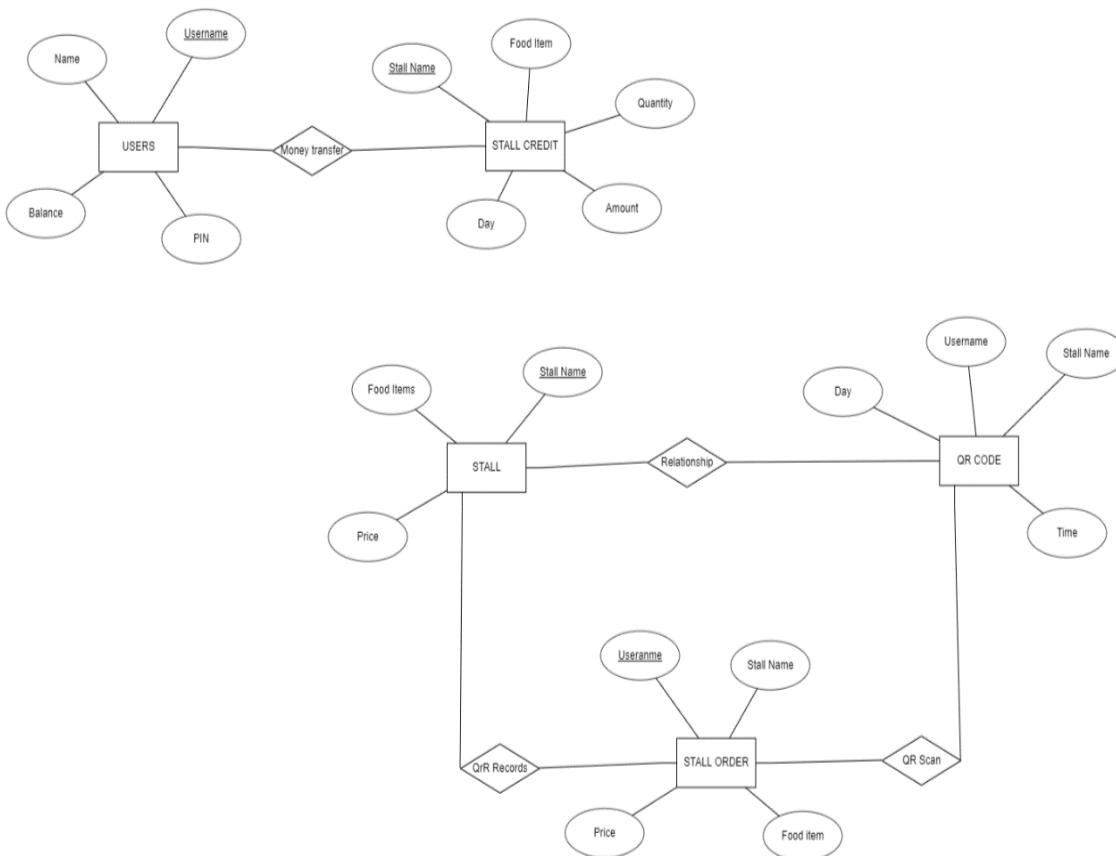
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** QR-PAYMENTS-... is selected.
- Tables:** STALLS_CREDITS is selected.
- Query Editor:** The query `SELECT * FROM `QR-PAYMENTS-SYSTEM` .STALLS_CREDITS;` is run, returning 5 rows.
- Result Grid:** The data is displayed in a grid:

STALL_NAME	DATE	AMOUNT
BombayChat	03/07/2022	70
ChillZone	03/07/2022	60
BombayChat	05/07/2022	50
BombayChat	06/07/2022	40
BombayChat	09/07/2022	100

- Action Output:** Shows the query was run at 10:44:00, returned 5 rows, and took 0.0047 sec / 0.00001...

ER Diagrams



Conclusion

With the working of this QR Payments system we can conclude that this idea can be developed and implemented to carry out any type of transactions within the college or the present UPI payments can be made to work with this similar mechanism, a UPI system that works without an internet connectivity can be developed.

As of now, this QR Payments system is a hosted web application and can be deployed into a mobile application.

References

1. <https://flask.palletsprojects.com/en/2.1.x/quickstart/>
2. https://www.w3schools.com/python/python_mysql_getstarted.asp
3. <https://www.geeksforgeeks.org/python-generate-qr-code-using-pyqrcode-module/>
4. <https://learnopencv.com/barcode-and-qr-code-scanner-using-zbar-and-opencv/>