UTA GORIDE SUMMARY REPORT

IS 6420
GROUP 3
PROJECT

Report By
Aadhithya VijayaKumar
Chris Taulbee
Hasitha Josyula
Meghana Nerusu

CONTENTS

EXECUTIVE SUMMARY

Utah Transit Authority, or UTA, is a government operated public transportation service that provides mobility solutions to the Wasatch region. Access to mobility is important for fostering opportunities for success, regardless of socio-economic status. Additionally, the carbon footprint of each individual has an impact on the environment, so it is the mission of UTA to provide a transportation solution that is affordable, reliable and environmentally conscious.

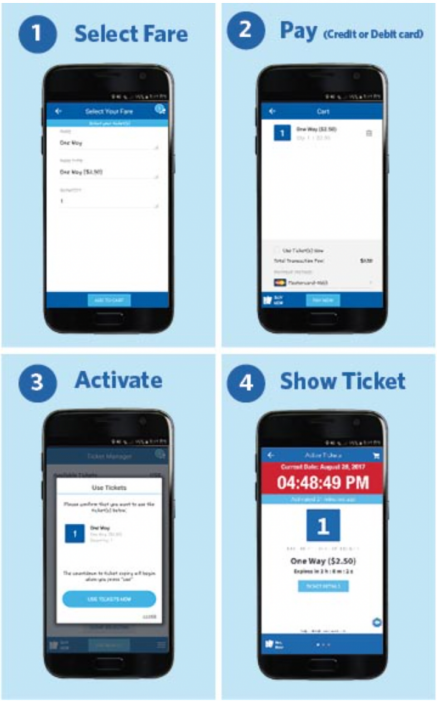
As the world becomes increasingly more connected digitally it is paramount that public transportation providers create ways for users to interact with their platforms that are familiar and convenient. The creation of the UTA GoRide mobile phone app is the solution. With the UTA GoRide mobile app, riders can purchase, manage, and use their tickets all in one place. The goal of this project is to design a database that mimics the functionality of the UTA GoRide mobile app. The features that this database seeks to fulfill are not fully inclusive of all of the UTA GoRide mobile app features, but include the process of creating an account, filling out profile information, adding payment details, purchasing and activating a ticket, and viewing ticket history.

ABOUT UTAGORIDE

Customers can use the mobile ticketing app UTA GoRide to plan their trip and purchase fare products for specific UTA services. Riders can purchase single-ride fares for passengers through the app. Once passengers are prepared to board their bus or train, they need to activate their fares.UTA is one of the latest agencies to roll out a mobile ticketing app.

How to use the app:

1. Download the free UTA GoRide mobile app
2. Add payment information
3. Purchase your fare
4. Show the transit operator your active fare ticket or store for later use



VISION & OBJECTIVES

Vision

To provide an integrated system of innovative, accessible and efficient public transportation services that increase access to opportunities and contribute to a healthy environment for the people of the Wasatch region.

Objectives

Utah Transit Authority provides integrated mobility solutions to service life's connections, improve public health and enhance quality of life.

Enhance alliances with regional corporate and nonprofit agencies to achieve shared transportation goals and to jointly lead out and identify new services and funding sources.

Utilize technologically advanced analytics and planning tools to design and implement an optimized, total transit network that connects people to their communities.

PROUCTS/ SERVICES

BUS

For more than 45 years, Wasatch Front residents have relied on UTA buses to get them where they need to go.With UTA's fleet of more than 400 buses, passengers can now travel to destinations in Box Elder, Weber, Davis, Tooele, Salt Lake, Summit, and Utah counties.More than 120 bus routes are available to passengers throughout the 1,400-mile UTA service area.Local bus service, as well as ski service and express service specifically designed for commuters to winter resorts, are provided by UTA to community destinations, schools, and hospitals.

TRAX

UTA's light rail system, TRAX, connects community destinations, shopping centers, schools and universities, bus hubs, and Park & Ride lots.TRAX operates seven days a week and operates every 15 minutes during peak times.Through a pantograph that connects the train to overhead catenary wires, TRAX is powered electrically.On December 4, 1999, the first section of TRAX opened, connecting riders from Salt Lake City to Sandy.

HOW TRANSACTIONAL DATABASES ARE USED TO SUPORT OPERATIONS

Within the UTA GoRide mobile app there is a transactional database that handles all of the operations that can be executed. When a new user creates an account, an associated record must be created that saves the information about that user so that the next time the user wants to use the app, all of their preferences are saved. Each time a ticket is purchased, a record is generated and must be properly associated with the customer that purchased it, and when it was purchased. A user could purchase all of the tickets they will use for the month and the transactional database would save the information about each of those tickets so that when the user is ready to use them, they are readily available. The transactional database is also used for storing the different types of tickets that the users could purchase, as well as saving the payment info for the users so that users don't have to input that information each time they go to purchase a ticket. In short, the transactional database is what makes this application possible, and what allows for the UTA GoRide mobile app to be a tool that makes it easier for users to ride with UTA.

INITIAL REQUIREMENTS

This project entails creating a database that tracks how riders use the UTA Go Ride Mobile Application to book one-way tickets and how they activate them.

Before booking a ticket, new customers must create an account on the mobile application. To sign up, customers must include their phone number and later populate their first and last names. They can also include their email address to receive email receipts. Once sign up is complete, a unique Customer ID is automatically generated for the customer.

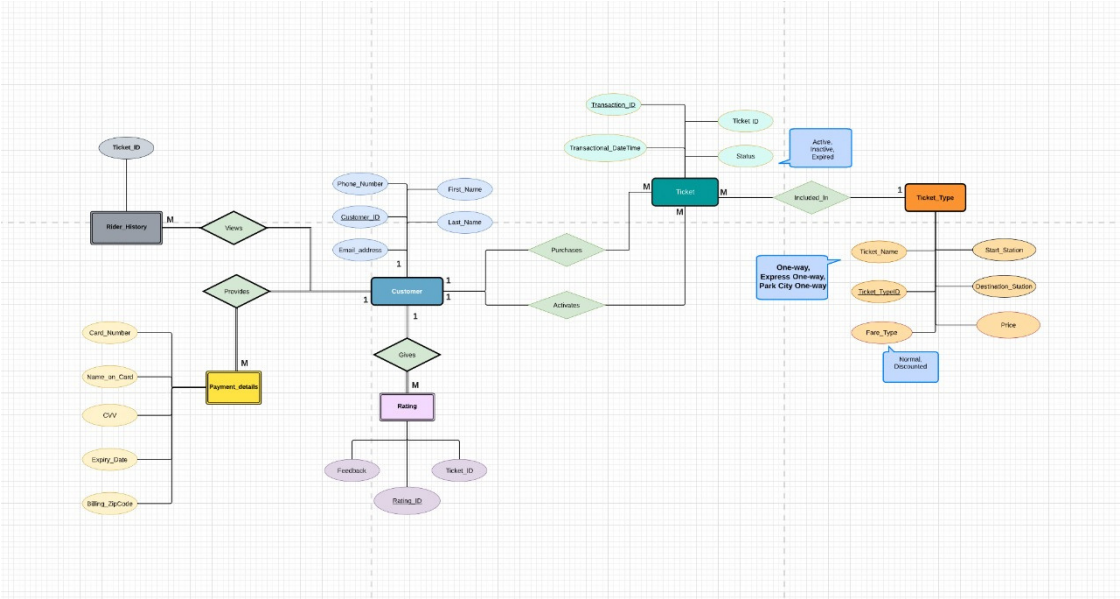
Customers can choose and purchase three types of one-way tickets: One Way, Express, and Park City. Each type defines a route which includes details of the start and destination station. Although start and destination stations are defined in the ticket type, tickets are valid for use at any station along a route, so users do not need to define where they plan on boarding and off-boarding. This ticket type is uniquely identified by a ticket type ID. While booking a ticket, customers are required to provide information such as type of one-way ticket, type of fare. The fare of each type of one-way ticket is based on which category they fall into Normal: $2.5/5, and Discounted: $1.25/2.5. The customers eligible for a discounted fare are 65+, Youth 6-17Y, Disabled, and Medicare Card Holders. They are also required to present proof to the operator when using discounted fare tickets.

For payments, customers must use a credit or debit card that includes their Card number, Expiry Month/Year, Billing Zipcode, CVV, and Card name. They can also save their debit or credit card information within their account for future ticket purchases.

In order to use the purchased ticket, customer need to activate it. A unique QR code is generated when a customer flips the ticket over to activate it, and this can be done only once. The ticket is only active for 2.5 hours after it has been scanned. Tickets that haven't been used, have a validity of 90 days from the date of purchase. They become inactive in case they are not activated during this time or the tickets exceed the 2.5hour time limit. Once the customer crosses the validity period, they no longer can activate and it gets expired. Customers can check the status of tickets if they are Active, Inactive or Expired in the Status section on the app. A unique Transaction ID and an associated Transaction DateTime is created everytime the user either purchases or activates the ticket. Each purchase has a unique Ticket ID, Transactional ID, Transactional DateTime, and Status.

Customers will also be able to track previous rides in the Rider history section using their ticket ID. Additionally, the app also requests for a rating on five and feedback from customer. A unique Rating ID is created for each customer who provides a rating. This will help the UTA team to track the user experience while interacting with the application.

CONCEPTUAL MODEL



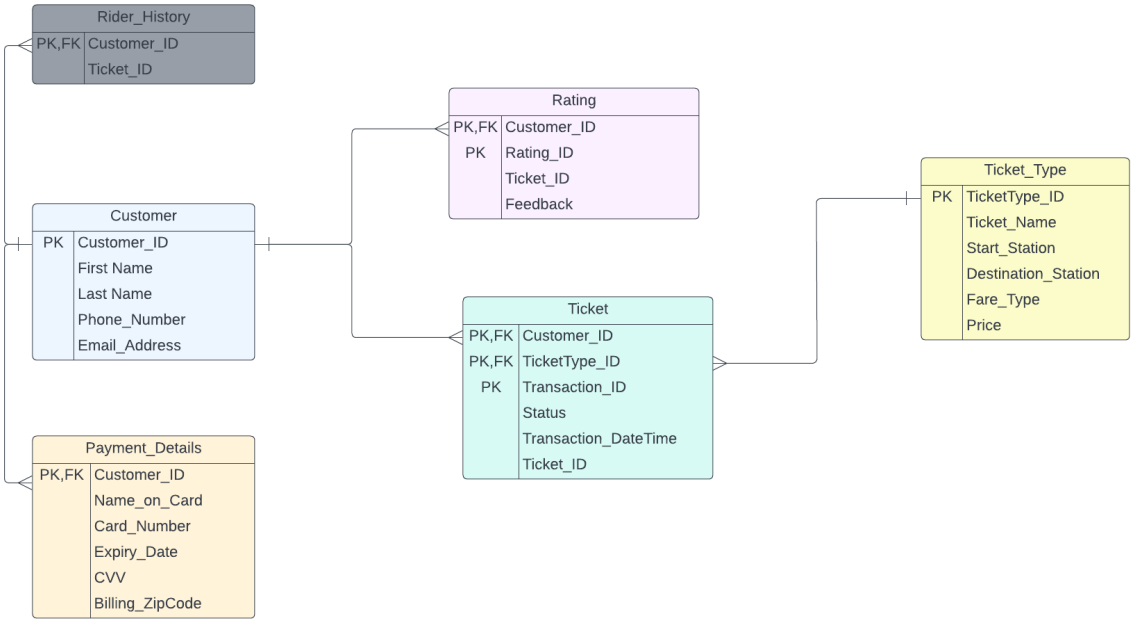
The conceptual model consists of three main entities Customer, Ticket, Ticket_Type and two weak entities Payment details and Rating. Customer ID and Ticket ID are identifier attributes because they are unique for each customer and purchased ticket per customer. However, the Transaction_ID attribute uniquely identifies the entire Ticket table. As shown above, almost every entity has a relationship with customer. This is because the central focus that business is based around is customer.

Our model consists of three types of one-ways: One-way, Express and Park City. To track these various types, we have a Ticket_type ID which is unique for each type.

Another interesting component of our model is the various cardinality constraints around the customer. We can see that there is a 1 to many relationship between Customer and Ticket since a customer can buy multiple tickets, but those unique Ticket ID's will be only associated to one customer.

Additionally, we have weak entities which are Payment Details and Rating which cannot exist without the customer. There is a one to many relationship from customers to payment details since a customer can have multiple payment methods and they can be only associated with a single customer. The customer can also provide multiple ratings which can be used to track the performance of the app from customer perspective. This information is useful for the UTA team to make the app even more better!

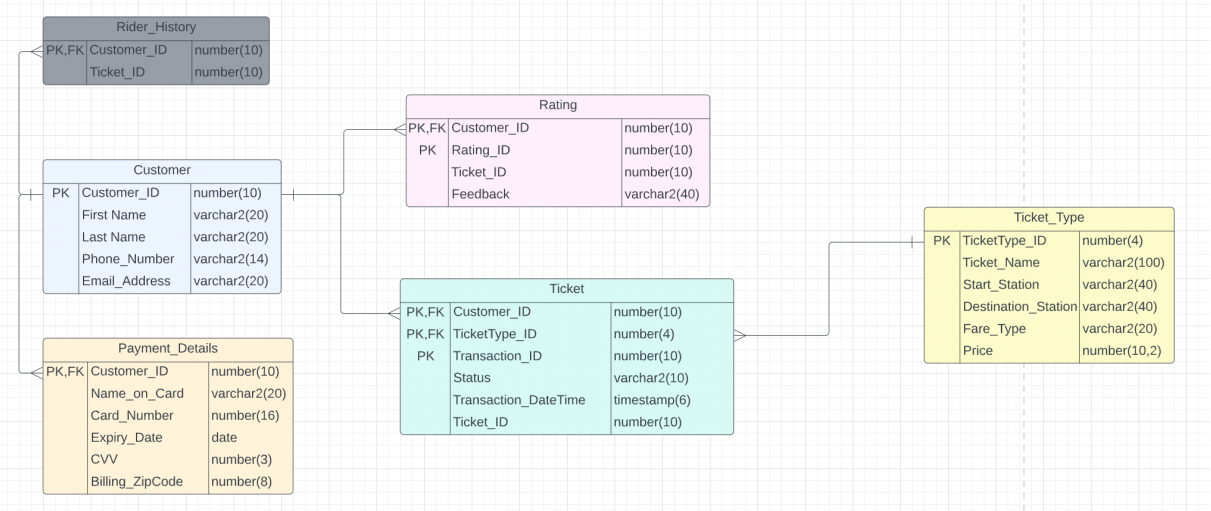
LOGICAL MODEL



The above diagram represents our logical model. The ticket table consists of two foreign keys Customer_Id and TicketType_ID from Customer table and Ticket_Type table and this is useful to easily associate the Customer Details and the Ticket Type Details with the purchased ticket. The Ticket table has the most attributes since, apart from providing the functionality to track the customer details associated with the purchased ticket, we are further able to monitor the status and datetime details of updation of the ticket status.

In Customer table, we are tracking all the demographic information like First Name, Last Name, Phone Number and Email address from to customer. This information is useful when generating a Customer ID.

PHYSICAL MODEL



In the physical model shown above, we are able to discern the breakdown of the logical model into meaningful data about the tables present in our database. We can notice the Transaction_DateTime attribute is of a timestamp datatype that facilitates the tracking of the exact time the status of the ticket was updated. The Price Attribute of the Ticket_Type table is of a Numeric type defined with the size allocation for decimal values inorder to include both the dollar and cent value for the ticket price.

SQL CODE- CREATE TABLES

```
--DROP TABLE

DROP TABLE RATING;
DROP TABLE RIDER_HISTORY;
DROP TABLE PAYMENT_DETAILS;
DROP TABLE TICKET;
DROP TABLE TICKET_TYPE;
DROP TABLE CUSTOMER;

--CREATE TABLES

CREATE TABLE CUSTOMER (
  CUSTOMER_ID   NUMBER(10) GENERATED BY DEFAULT AS IDENTITY
   (START WITH 101 INCREMENT BY 1),
  FIRST_NAME    VARCHAR2(20) NOT NULL,
  LAST_NAME     VARCHAR2(20) NOT NULL,
  PHONE_NUMBER  VARCHAR2(14) NOT NULL UNIQUE,
  EMAIL_ADDRESS VARCHAR2(20),
  PRIMARY KEY ( CUSTOMER_ID )
);

CREATE TABLE TICKET_TYPE (
 TICKETTYPE_ID NUMBER(4) GENERATED BY DEFAULT AS IDENTITY
 (START WITH 1 INCREMENT BY 1),
 TICKET_NAME VARCHAR2(100) NOT NULL,
 START_STATION VARCHAR2(40) NOT NULL,
 DESTINATION_STATION VARCHAR2(40) NOT NULL,
 FARE_TYPE VARCHAR2(20) NOT NULL,
 PRICE NUMBER(10,2) NOT NULL,
 PRIMARY KEY ( TICKETTYPE_ID )
);
```

SQL CODE- CREATE TABLES

```
CREATE TABLE TICKET (
TRANSACTION_ID NUMBER(10) GENERATED BY DEFAULT AS IDENTITY
 (START WITH 1001 INCREMENT BY 1),
CUSTOMER_ID       NUMBER(10),
TICKETTYPE_ID     NUMBER(4),
TICKET_ID         NUMBER(10),
TRANSACTION_DATETIME TIMESTAMP NOT NULL,
 STATUS           VARCHAR2(10) NOT NULL,
PRIMARY KEY ( TRANSACTION_ID ),
 CONSTRAINT "FK_Ticket.Customer_ID" FOREIGN KEY ( CUSTOMER_ID )
 REFERENCES CUSTOMER ( CUSTOMER_ID ),
 CONSTRAINT "FK_Ticket.TicketType_ID" FOREIGN KEY ( TICKETTYPE_ID )
  REFERENCES TICKET_TYPE ( TICKETTYPE_ID )
);

CREATE TABLE PAYMENT_DETAILS (
  CUSTOMER_ID     NUMBER(10),
  NAME_ON_CARD    VARCHAR2(20) NOT NULL,
  CARD_NUMBER     NUMBER(16) NOT NULL,
  EXPIRY_DATE     DATE NOT NULL,
  CVV             NUMBER(3) NOT NULL,
  BILLING_ZIPCODE NUMBER(8) NOT NULL,
  PRIMARY KEY ( CUSTOMER_ID ),
 CONSTRAINT "FK_Payment_Details.Customer_ID" FOREIGN KEY (
CUSTOMER_ID)
 REFERENCES CUSTOMER ( CUSTOMER_ID )
);
```

SQL CODE- CREATE TABLES

```
CREATE TABLE RATING (
 CUSTOMER_ID NUMBER(10),
 RATING_ID NUMBER(10) GENERATED BY DEFAULT AS IDENTITY
 (START WITH 4101 INCREMENT BY 1),
 TICKET_ID NUMBER(10) NOT NULL,
 FEEDBACK VARCHAR2(40),
 PRIMARY KEY ( CUSTOMER_ID, RATING_ID ),
 CONSTRAINT "FK_Rating.Customer_ID" FOREIGN KEY ( CUSTOMER_ID )
 REFERENCES CUSTOMER ( CUSTOMER_ID )
);

  CREATE TABLE RIDER_HISTORY (
  CUSTOMER_ID NUMBER(10),
  TICKET_ID   NUMBER(10),
  PRIMARY KEY ( CUSTOMER_ID,
    TICKET_ID ),
   CONSTRAINT "FK_Rider_History.Customer_ID" FOREIGN KEY ( CUSTOMER_ID )
    REFERENCES CUSTOMER ( CUSTOMER_ID )
);
```

SQL CODE- INSERT DATA

--CUSTOMER

```
INSERT INTO CUSTOMER
(FIRST_NAME,LAST_NAME,PHONE_NUMBER,EMAIL_ADDRESS)
VALUES ('SAM','RYDER','4056589084','SRYDER@GMAIL.COM);

INSERT INTO CUSTOMER
(FIRST_NAME,LAST_NAME,PHONE_NUMBER,EMAIL_ADDRESS)
VALUES ('GEORGE','MILLER','5068541205','GMILLER@GMAIL.COM');

INSERT INTO CUSTOMER (FIRST_NAME,LAST_NAME,PHONE_NUMBER,
EMAIL_ADDRESS)
VALUES ('JAMIE','ANDERS','6048204265','JANDERS@GMAIL.COM');

INSERT INTO CUSTOMER
(FIRST_NAME,LAST_NAME,PHONE_NUMBER,EMAIL_ADDRESS)
VALUES ('ASHLEY','SUMMERS','3855162905','ASUMMERS@GMAIL.COM');
```

SQL CODE- INSERT DATA

--TICKET_TYPE

```
INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('ONE WAY','WEST VALLEY
CENTRAL','AIRPORT','NORMAL',2.5);

INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('ONE WAY','WEST VALLEY
CENTRAL','AIRPORT','DISCOUNTED',1.25);

INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('EXPRESS ONE WAY','KOAMS DR','OGDEN
STATION','NORMAL',5);

INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('EXPRESS ONE WAY','KOAMS DR','OGDEN
STATION','DISCOUNTED',2.5);

INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('PARK CITY ONE WAY','MEADOWBROOK','SALT LAKE
CENTRAL','NORMAL',5);

INSERT INTO
TICKET_TYPE(TICKET_NAME,START_STATION,DESTINATION_STATION,
FARE_TYPE,PRICE) VALUES ('PARK CITY ONE WAY','MEADOWBROOK','SALT LAKE
CENTRAL','DISCOUNTED',5);

SELECT * FROM TICKET_TYPE;
```

SQL CODE- INSERT DATA

--TICKET

```
INSERT INTO TICKET (CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (101,1, 201, '20-JAN-2022 12:41:24.284', 'ACTIVE');


INSERT INTO TICKET (CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (101,1, 201,'20-JAN-2022 03:11:24.284 PM','EXPIRED');

INSERT INTO TICKET (CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (104,4, 202,'16-APR-2022 11:29:14.162','EXPIRED');

INSERT INTO TICKET (CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (102,5,203,'25-JUN-2022 12:50:16.544','INACTIVE');

INSERT INTO TICKET (CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (102,5,203,'28-AUG-2022 9:20:24.801','ACTIVE');

INSERT INTO TICKET(CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (102,5,203,'28-AUG-2022 11:50:24.801','EXPIRED');

INSERT INTO TICKET(CUSTOMER_ID,
TicketType_ID,TICKET_ID,TRANSACTION_DATETIME,STATUS)
VALUES (101,5, 204,'20-OCT-2022 05:04:28.940 PM','INACTIVE');

SELECT * FROM TICKET;
```

SQL CODE- INSERT DATA

--PAYMENT DETAILS

```
NSERT INTO
PAYMENT_DETAILS(CUSTOMER_ID,NAME_ON_CARD,CARD_NUMBER,
EXPIRY_DATE,CVV,BILLING_ZIPCODE)
VALUES (101,'SAM RYDER',1004298065829540,'14-JUN-2026',264,84100);

INSERT INTO
PAYMENT_DETAILS(CUSTOMER_ID,NAME_ON_CARD,CARD_NUMBER,
EXPIRY_DATE,CVV,BILLING_ZIPCODE)
VALUES (102,'GEORRGE MILLER',6254091426848604,'22-OCT-
2028',802,84200);

INSERT INTO
PAYMENT_DETAILS(CUSTOMER_ID,NAME_ON_CARD,CARD_NUMBER,
EXPIRY_DATE,CVV,BILLING_ZIPCODE)
VALUES (103,'JAMIE ANDERS',5264892155820026,'18-FEB-2030',081,84140);

INSERT INTO
PAYMENT_DETAILS(CUSTOMER_ID,NAME_ON_CARD,CARD_NUMBER,
EXPIRY_DATE,CVV,BILLING_ZIPCODE)
VALUES (104,'ASHLEY SUMMERS 1204896106241645,'14-DEC-
2032',648,84501);

SELECT * FROM PAYMENT_DETAILS;
```

SQL CODE- INSERT DATA

--RIDER_HISTORY

```
INSERT INTO RIDER_HISTORY (CUSTOMER_ID,TICKET_ID)
VALUES (101,201);

INSERT INTO RIDER_HISTORY (CUSTOMER_ID,TICKET_ID)
VALUES (104, 202);

INSERT INTO RIDER_HISTORY (CUSTOMER_ID,TICKET_ID)
VALUES (102,203);

INSERT INTO RIDER_HISTORY (CUSTOMER_ID,TICKET_ID)
VALUES (101, 204);

SELECT * FROM RIDER_HISTORY;
```

--RATING

```
INSERT INTO RATING (CUSTOMER_ID,RATING_ID,TICKET_ID, FEEDBACK)
VALUES(101,4101,201,'Good experience');

INSERT INTO RATING (CUSTOMER_ID,RATING_ID,TICKET_ID, FEEDBACK)
VALUES(104,4102,202,'Hard to navigate');

INSERT INTO RATING (CUSTOMER_ID,RATING_ID,TICKET_ID, FEEDBACK)
VALUES(102,4103,203,'Easy options to chose from');

INSERT INTO RATING (CUSTOMER_ID,RATING_ID,TICKET_ID, FEEDBACK)
VALUES(101,4104, 204, '');
```

REQUIREMENTS REVIEW

Category	Description	Status
Onboarding	App should be able to add new Customers	Done
Onboarding	Users should be able to add their Name & Email	Done
Purchasing	Users can Pick between multiple different Routes	Done
Purchasing	Users can pick between different Fare Types	Done
Purchasing	Users can select a start and end station	Future Work
Payments	Users can add their payment method	Done
Payments	Users can add multiple payment methods	Done
Activate	Users can activate their tickets	Done
Activate	Users can activate multiple tickets	Done
Ticket Management	Users can view the status' of all of their tickets	Done

REQUIREMENTS REVIEW

Ticket Management	Users can view the history of their tickets	Done
Ticket Management	System should Automatically expire unused tickets after 90 days	Future Work
Ticket Management	System should Automatically Expire Tickets 150 minutes after they're activated	Future Work
Feedback	Users should be able to give feedback after they purchase a ticket	Done

FEATURE 1 DEMONSTRATION

```
SELECT TICKET_ID, TICKET_NAME, PRICE,STATUS,
TRANSACTION_DATETIME
FROM TICKET
INNER JOIN TICKET_TYPE ON TICKET.TICKETTYPE_ID =
TICKET_TYPE.TICKETTYPE_ID
WHERE TRANSACTION_ID IN (SELECT MAX(TRANSACTION_ID) FROM
TICKET WHERE
CUSTOMER_ID = '101'
GROUP BY TICKET_ID);
```

Output-

	TICKET_ID	TICKET_NAME	PRICE	STATUS	TRANSACTION_DATETIME
1	201	ONE WAY	2.5	EXPIRED	2022-01-20T15:11:24.284Z
2	204	PARK CITY ONE WAY	5	INACTIVE	2022-10-20T17:04:28.940Z

In Feature 1, the design allows the customers to view the latest status of each ticket purchased by them. This feature implements this functionality by selecting the latest inserted record for each ticket based on the DateTime attribute. This is accomplished by running a SELECT query on the result grouped by TICKET ID after performing a join operation on the TICKET TYPE AND TICKET table.

FEATURE 2 DEMONSTRATION

```
UPDATE CUSTOMER SET EMAIL_ADDRESS = 'ASHSUMMERS@GMAIL.COM'
WHERE CUSTOMER_ID = 104;
```

Output before updation-

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL_ADDRESS
1	104	ASHLEY	SUMMERS	3855162905	ASUMMERS@GMAIL.COM

Output after updation-

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL_ADDRESS
1	104	ASHLEY	SUMMERS	3855162905	ASHSUMMERS@GMAIL.COM

The second feature provides the functionality of the customer being able to update their email id. This is a vital functionality that allows customers to seamlessly update their details since this is a common occurrence in many scenarios. This feature is executed by performing an UPDATE query on the CUSTOMER table and updating the record based on the CUSTOMER_ID.

FEATURE 3 DEMONSTRATION

INSERT INTO
TICKET(TRANSACTION_ID,CUSTOMER_ID,TicketType_ID,TICKET_ID,
TRANSACTION_DATETIME,STATUS)
VALUES (1010,101,5,204,SYSDATE,'ACTIVE');

Output before Insertion-

	TRANSACTION_ID	CUSTOMER_ID	TICKETTYPE_ID	TICKET_ID	TRANSACTION_DA	STATUS
1	1001	101	1	201	2022-01-10T08:00:4	INACTIVE
2	1002	104	4	202	2022-01-16T11:29:1	INACTIVE
3	1003	101	1	201	2022-01-20T12:41:2	ACTIVE
4	1004	101	1	201	2022-01-20T15:11:2	EXPIRED
5	1005	104	4	202	2022-04-16T11:29:1	EXPIRED
6	1006	102	5	203	2022-06-25T12:50:'	INACTIVE
7	1007	102	5	203	2022-08-28T09:20:	ACTIVE
8	1008	102	5	203	2022-08-28T11:50:2	EXPIRED
9	1009	101	5	204	2022-10-20T17:04:2	INACTIVE

Output after Insertion-

	TRANSACTION_ID	CUSTOMER_ID	TICKETTYPE_ID	TICKET_ID	TRANSACTION_DA	STATUS
1	1010	101	5	204	2022-10-23T09:34:'	ACTIVE
2	1001	101	1	201	2022-01-10T08:00:4	INACTIVE
3	1002	104	4	202	2022-01-16T11:29:1	INACTIVE
4	1003	101	1	201	2022-01-20T12:41:2	ACTIVE
5	1004	101	1	201	2022-01-20T15:11:2	EXPIRED
6	1005	104	4	202	2022-04-16T11:29:1	EXPIRED
7	1006	102	5	203	2022-06-25T12:50:'	INACTIVE
8	1007	102	5	203	2022-08-28T09:20:	ACTIVE
9	1008	102	5	203	2022-08-28T11:50:2	EXPIRED
10	1009	101	5	204	2022-10-20T17:04:2	INACTIVE

FEATURE 3 DEMONSTRATION

The final feature has been designed to insert the new status of the ticket when the customer activates the ticket. Aside from the status, this feature also captures the ticket's precise date and time of activation. This functionality provides an opportunity to view the exact time the ticket was activated so that the customer knows how long it remains usable before expiring. The above concept is implemented by performing an INSERT statement with the new 'ACTIVE' status and current date and time on the TICKET table.

REFERENCES

https://www.rideuta.com/Fares-And-Passes/UTA-GoRide-Mobile-Ticketing

https://www.rideuta.com/Services

APPENDIX

CUSTOMERS

CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL_ADDRESS
101	SAM	RYDER	4056589084	SRYDER@GMAIL.COM
102	GEORGE	MILLER	5068541205	GMILLER@GMAIL.CO
103	JAMIE	ANDERS	6048204265	JANDERS@GMAIL.CO
104	ASHLEY	SUMMERS	3855162905	ASHSUMMERS@GMA

The customers table stores all of the information about the customers as they sign in. Customers create their accounts by providing their phone numbers, but can later on add their first and last names, and their email address. The system is automatically generating a unique ID for the primary key of this table.

PAYMENT_DETAILS

CUSTOMER_ID	NAME_ON_CARD	CARD_NUMBER	EXPIRY_DATE	CVV	BILLING_ZIPCODE
101	SAM RYDER	1004298065829540	6/14/2026, 12:00:00 A	264	84100
102	GEORRGE MILLER	6254091426848604	10/22/2028, 12:00:00	802	84200
103	JAMIE ANDERS	5264892155820026	2/18/2030, 12:00:00 A	81	84140
104	ASHLEY SUMMERS	1204896106241645	12/14/2032, 12:00:00	648	84501

When users are ready to purchase tickets, they must first save the card details of the card they want to use to pay for their tickets on the app. Each customer can have multiple payment methods linked to their account.

APPENDIX

RATINGS

CUSTOMER_ID	RATING_ID	TICKET_ID	FEEDBACK
101	4101	201	Good experience
104	4102	202	Hard to navigate
102	4103	203	Easy options to chose
101	4104	204	(null)

After a purchase, customers are prompted to leave feedback about their experience using the app. Information about who submitted the response and what ticket they had just purchased is collected so that if the customer had a negative experience, the team can examine what could have been the reason.

TICKETS

TRANSACTION_ID	CUSTOMER_ID	TICKETTYPE_ID	TICKET_ID	TRANSACTION_DAT	STATUS
1001	101	1	201	2022-01-10T08:00:44.	INACTIVE
1002	104	4	202	2022-01-16T11:29:14.	INACTIVE
1003	101	1	201	2022-01-20T12:41:24.	ACTIVE
1004	101	1	201	2022-01-20T15:11:24.	EXPIRED
1005	104	4	202	2022-04-16T11:29:14.	EXPIRED
1006	102	5	203	2022-06-25T12:50:16.	INACTIVE

APPENDIX

This is the table that stores all of the transactions about the tickets. When a ticket is purchased, a new line is created in this table that tracks which customer purchased it, the type of ticket it is, and the ticket ID. Information about when it was created is also collected. Additional records are created when a user activates a ticket, a ticket expires after it is activated, or If a ticket that was never used becomes older than 90 days. This table is then used when a customer wants to view their active tickets, or if they want to see the whole history of their ticket transactions.

TICKET_TYPES

	TICKETTYPE_ID	TICKET_NAME	START_STATION	DESTINATION_STATION	FARE_TYPE	PRICE
1	1	ONE WAY	WEST VALLEY CENTRAL	AIRPORT	NORMAL	2.5
2	2	ONE WAY	WEST VALLEY CENTRAL	AIRPORT	DISCOUNTED	1.25
3	3	EXPRESS ONE WAY	KOMAS DR	OGDEN STATION	NORMAL	5
4	4	EXPRESS ONE WAY	KOMAS DR	OGDEN STATION	DISCOUNTED	2.5
5	5	PARK CITY ONE WAY	MEADOWBROOK	SALT LAKE CENTRAL	NORMAL	5
6	6	PARK CITY ONE WAY	MEADOWBROOK	SALT LAKE CENTRAL	DISCOUNTED	2.5

This table defines the types of tickets that customers can purchase and the details of those tickets. There are multiple entries per route because each ticket can also be segmented by the type of fare.

APPENDIX

RIDER HISTORY

CUSTOMER_ID	TICKET_ID
101	201
101	204
102	203
104	202

This is a header table for the tickets table that shows the ticket_ids that are associated with customers.

TIME TRACKING DETAILS

DATE	TEAM MEMBER	HOURS SPENT	DESCRIPTION OF WORK
9/17/2022	Aadhithya Vijayakumar, Chris Taulbee, Hasitha Josyula, Meghana Nerusu	2	Met to discuss about the proposal of the project
9/19/2022	Hasitha Josyula	0.25	Submitted Proposal
9/20/2022	Aadhithya Vijayakumar, Chris Taulbee, Hasitha Josyula, Meghana Nerusu	1.5	Discussed & finalized the main features of the project and requirement, Started Conceptual model
10/23/2022	Hasitha Josyula	2.5	Worked on Conceptual Model
10/24/2022	Aadhithya Vijayakumar, Chris Taulbee, Hasitha Josyula, Meghana Nerusu	0.5	Discussed and finalized conceptual model
10/11/2022	Meghana Nerusu	2.5	Worked on Logical Model
10/12/2022	Aadhithya Vijayakumar, Chris Taulbee, Hasitha Josyula, Meghana Nerusu	1	Discussed and finalized logical model
10/23/2022	Aadhithya Vijayakumar, Chris Taulbee, Hasitha Josyula, Meghana Nerusu	1	Met to discuss the work done so far and made required changes, discussed the final tasks to be done
10/24/2022	Meghana Nerusu, Hasitha Josyula	1	Worked and Finalized Physical Model
10/29/2022	Aadhithya Vijayakumar	3	Research and modelling the DB
10/30/2022	Aadhithya Vijayakumar	2	Creating and Inserting tables
11/1/2022	Chris Taulbee	1	Cleaned up all the models

TIME TRACKING DETAILS

11/4/2022	Meghana Nerusu	2.25	Started Filling in the summary report, added Vision and objectives, Product Services
11/8/2022	Chris Taulbee	2	Added Executive summary and How transactional databases are used to support operations for organisation to summary report
11/11/2022	Hasitha Josyula	3	Added Initial Requirements, Conceptual & Logical screenshots and their brief explanatory text/ highlights to summary report
11/14/2022	Aadhithya Vijayakumar	0.5	Added Physical Model & brief explanatory text/highlights to summary report
11/15/2022	Chris Taulbee	2	Worked and added Final Requirements to the report
11/16/2022	Aadhithya Vijayakumar, Chris Taulbee	2.5	Worked on Feature Demonstrations
11/18/2022	Meghana Nerusu	2	Added SQL queries (half) and output screenshots to the report
11/20/2022	Aadhithya Vijayakumar	1	Added Feature Demonstration & Description to the report
11/21/2022	Hasitha Josyula	2	Added Colors to Conceptual, logical,Physical Model and added remaining SQL tables & screenshots to the report, Time-tracking details to Final report
11/22/2022	Chris Taulbee	2.5	Worked and added Final Requirements to table, Database Sample Data Screenshots & Brief explanation
11/23/2022	Meghana Nerusu	1.5	Added cover page & page numbers, updated table of contents and all the final changes to the summary report
11/24/2022	Aadhithya Vijayakumar, Hasitha Josyula, Meghana Nerusu	1	Worked on PPT
11/25/2022	Aadhithya Vijayakumar, Hasitha Josyula, Meghana Nerusu	1	Pratice Presentation

TIME-TRACKING SUMMARY

TEAM MEMBER	TOTAL HOURS FOR PROJECT	ADDITIONAL COMMENTS
Aadhithya Vijayakumar	17	NA
Chris Taulbee	16	NA
Hasitha Josyula	16.75	NA
Meghana Nerusu	16.75	NA