FACULTY OF COMPUTING

SESSION 2023/2024

SEMESTER 1

SECD2523-03 DATABASE (PANGKALAN DATA)

PHASE 3 - DATABASE LOGICAL DESIGN

LECTURER: DR. IZYAN IZZATI BINTI KAMSANI

NO	NAME	MATRIC NO
1	NUR HAFIZAH BINTI JAFRI	A22EC5022
2	FARAH HAZIRAH NISHA BINTI ABD LATIF	A22EC0159
3	NURSYUHADA BINTI BADREN	A22EC0253
4	SARAH SOFEA BINTI ANUAR	A22EC0104
5	SALINI RAVINTHIRAN	A22EC0267

TABLE OF CONTENTS

1.0 Introduction

The goal of this project is to provide a user-friendly ticketing system that streamlines the event registration, ticket selection, ticket purchase, and event production processes, allowing organizers and registered users to create their own events. Users will be able to choose events of interest, subscribe to them with ease, select tickets, and complete the purchasing process with ease due to the system. The development of a full-service and accessible ticketing system would greatly improve the event management procedure, offering a simplified and convenient experience to event organizers, ticket buyers, and attendees. The system will help make the event environment more productive and well-organized, which will benefit events of all kinds.

2.0 Overview of Project

NexScholar is a social platform for students to connect whose main purpose is to organize an event. A "social platform" is an online service or website that facilitates online communication and connections between users. NexScholar gives users access to a virtual environment to make profiles and view events. Features that NexScholar provides are communication, where users can exchange messages, chat, and engage in conversation between them through text. Next is networking, which enables people to make connections with others through mutual acquaintances, affiliations, or shared interests. Besides, there is a collaboration in which students, either undergraduate or graduate, use NexScholar for professional networking and collaboration to interact with clients and colleagues, exchange work-related information, and work together on projects. Other users can discover new information, trends, and content by following other users or topics of interest.

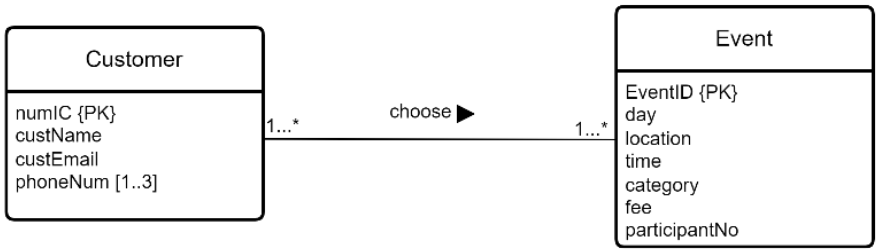
3.0 Database Conceptual Design

3.1 Update Business Rule

1. **Customer choose events**

   Customers can choose one or many event while Event might be chosen by one or many customers

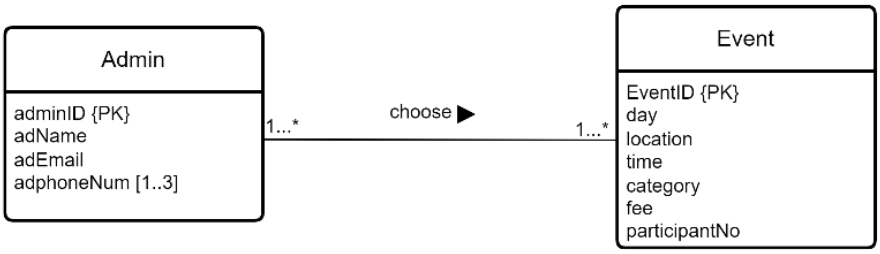
   'Choose' events are **many-to-many (*:*)** relationship



2. **Admin creates an event**

   Admin can create one or many events while Events can be created by one or many admins

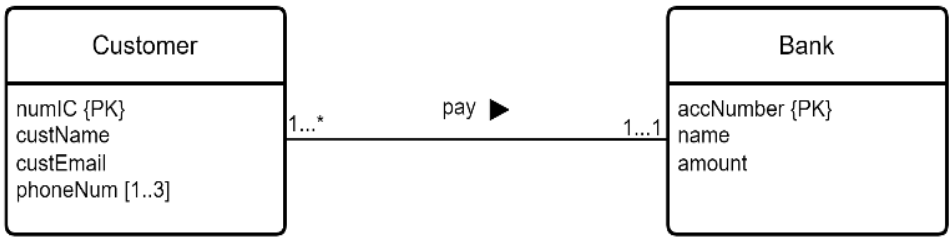
   'Create' an event is a **many-to-many (*:*)** relationship



3. **Customer makes a payment**

   Customer can pay to only one bank while Bank can receive payments from many customers

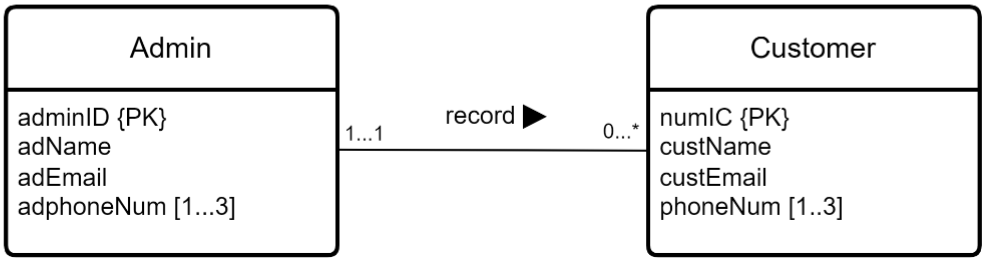
   'Pay' is a **many-to-one** (***:1**) relationship



4. **Admin records customer's data**

   Admin can record none or many customer's information while Customer's information can be recorded by only one admin

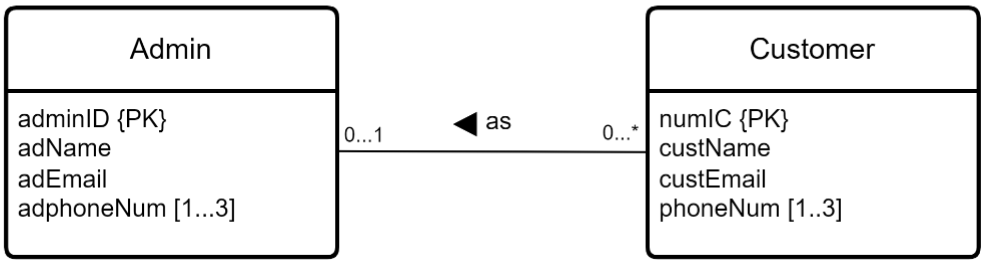
   'Record' is a **one-to-many (1:*)** relationship



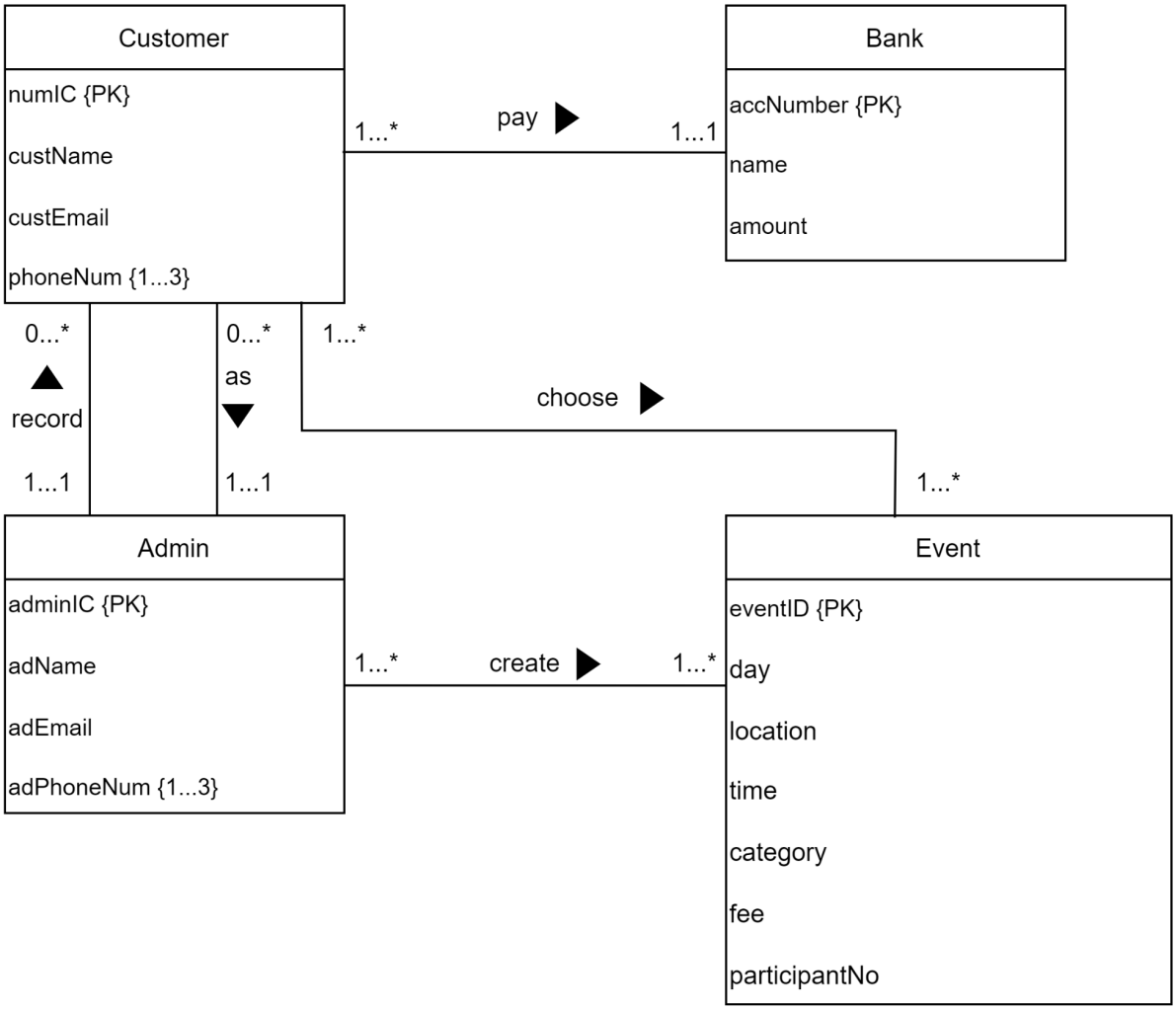
5. Customer act as Admin

Customer can act as zero or an Admin while Admin can be acted by one or many customers.

'As' is a **many-to-one (*:1)** relationship

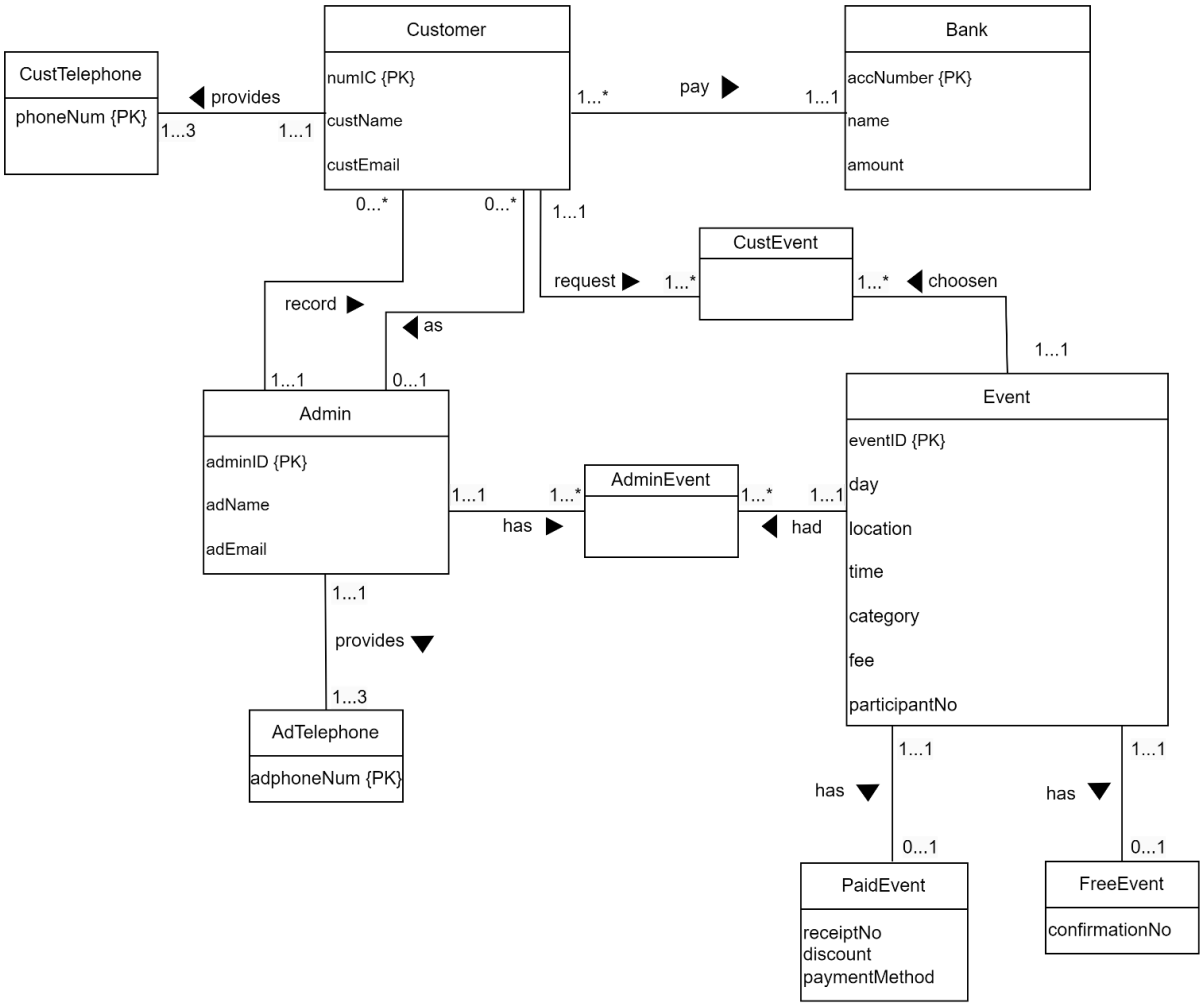


3.2 Conceptual ERD



4.0 DB Logical Design

4.1 Logical ERD



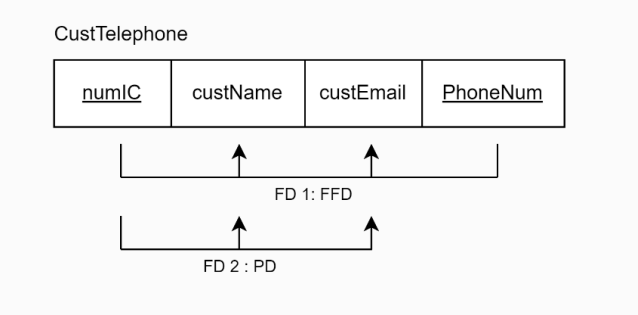
4.2 Updated Data Dictionary

Entity Name	Attributes	Description	Data Type & Length	Nullity
Admin	adminID {PK}	Unique ID for the admin	Varchar2 (6)	No
	adName	Name of the admin	Varchar2 (50)	No
	adEmail	Admin's email	Varchar2 (50)	No
Customer	numIC {PK}	Unique ID for the customer	Varchar2 (15)	No
	custName	Name of the customer	Varchar2 (20)	No
	custEmail	Customer's email	Varchar2 (50)	No
	adminID	Unique ID for the admin	Varchar2 (6)	No
	accNum	Unique ID for the bank	Varchar2 (15)	No
Event	eventID {PK}	Unique ID for the event	Varchar2 (6)	No
	day	Day of the event	Varchar2 (10)	No
	location	Location of the event	Varchar2 (20)	No
	time	Time of the event	Date	No
	category	Category of the event	Varchar2 (10)	No
	fee	Fee of the event	Number (10)	Yes
	participantNo	Participant number of the event	Varchar2 (4)	No
Bank	accNum {PK}	Unique ID for the bank	Varchar2 (15)	No
	name	Name of the cardholder	Varchar2 (20)	No
	amount	Price of the event	Number (10 , 2)	No
PaidEvent	eventID {PK}	Unique ID for the event	Varchar2 (6)	No
	day	Day of the event	Varchar2 (10)	No
	location	Location of the event	Varchar2 (20)	No
	time	Time of the event	Date	No
	category	Category of the event	Varchar2 (10)	No

	fee participantNo receiptNo discount paymentMethod	Fee of the event Participant number of the event Receipt number of the payment Discount of the event fee Payment method for the event	Number (10) Varchar2 (4) Varchar2 (10) Number (2) Varchar2 (20)	Yes No No Yes No
FreeEvent	eventID {PK} day location time category fee participantNo confirmationNo	Unique ID for the event Day of the event Location of the event Time of the event Category of the event Fee of the event Participant number of the event Confirmation number for the event	Varchar2 (6) Varchar2 (10) Varchar2 (20) Date Varchar2 (10) Number (10) Varchar2 (4) Varchar2 (10)	No No No No No Yes No No
CustTelephone	phoneNum {PK} numIC	Customer's phone number Unique ID for the customer	Varchar2 (15) Varchar2 (15)	No No
AdTelephone	adphoneNum {PK} adminID	Admin's phone number Unique ID for the admin	Varchar2 (15) Varchar2(15)	No No
AdminEvent	adminID {PK} eventID {PK}	Unique ID for the admin Unique ID for the event	Varchar2 (6) Varchar2 (6)	No No
CustEvent	numIC {PK} eventID {PK}	Unique ID for the admin Unique ID for the event	Varchar2 (15) Varchar2 (6)	No No

4.3 Normalization

4.3.1 Normalization of CustTelephone Relation



FD1 : Full Functional Dependency

numIC, phoneNum → custName, custEmail

FD2 : Partial Dependency

numIC → custName, custEmail

1st Normalization Form

- CustTelephone (numIC, phoneNum, custName, custEmail)
  PK : numIC,phoneNum

2nd Normalization Form

- Customer (numIC, custName, custEmail)
  PK : numIC

- CustTelephone (numIC, phoneNum)
  PK : numIC,phoneNum
  FK : numIC references Customer (numIC)

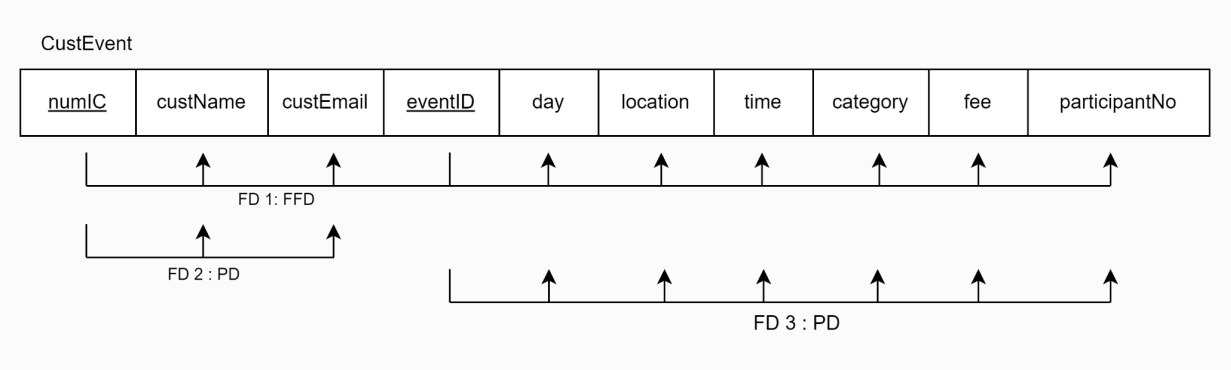
3rd Normalization Form

Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form.

- Customer (numIC, custName, custEmail)
  PK : numIC

- CustTelephone (numIC, phoneNum)
  PK : numIC,phoneNum
  FK : numIC references Customer (numIC)

4.3.2 Normalization of CustEvent Relation



FD1 : Full Functional Dependency

numIC, eventID → custName, custEmail, day, location, time, category, fee, participantNo

FD2 : Partial Dependency

numIC → custName, custEmail

FD3 : Partial Dependency

eventID → day, location, time, category, fee, participantNo

1st Normalization Form

- CustEvent (numIC, eventID, custName, custEmail, day, location, time, category, fee, participantNo)
  PK : numIC, eventID

2nd Normalization Form

- Customer (numIC, custName, custEmail)
  PK : numIC

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- CustEvent (numIC, eventID)
  PK : numIC, eventID
  FK : numIC references Customer (numIC)
  FK : eventID references Event (eventID)

3rd Normalization Form

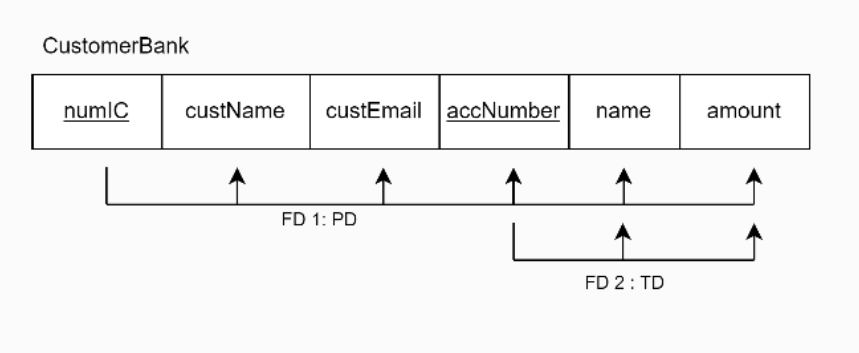
Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form

- Customer (numIC, custName, custEmail)
  PK : numIC

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- CustEvent (numIC, eventID)
  PK : numIC, eventID
  FK : numIC references Customer (numIC)
  FK : eventID references Event (eventID)

4.3.3 Normalization of CustomerBank Relation



FD1 : Partial Dependency

numIC→ custName, custEmail, accNumber, name, amount

FD2 : Transitive Dependency

accNumber → name, amount

1st Normalization Form

- CustomerBank (numIC, accNumber, custName, name, amount)
  PK : numIC, accNumber

2nd Normalization Form

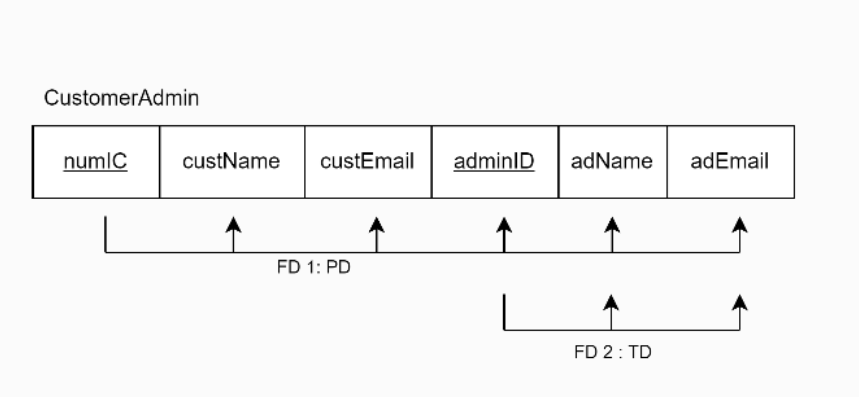
- CustomerBank (numIC, custName, accNumber, name, amount)
  PK : numIC

3rd Normalization Form

- Bank (accNumber, name, amount)
  PK : accNumber

- Customer (numIC, custName, custEmail, accNumber)
  PK : numIC
  FK : accNumber references Bank (accNumber)

4.3.4 Normalization of CustomerAdmin Relation



FD1 : PartialDependency

numIC→ custName, custEmail, adminID, adName, adEmail

FD2 : Transitive Dependency

adminID → adName, adEmail

1st Normalization Form

- CustomerAdmin (numIC, custName, custEmail, adminID, adName, adEmail)
  PK : numIC, adminID

2nd Normalization Form

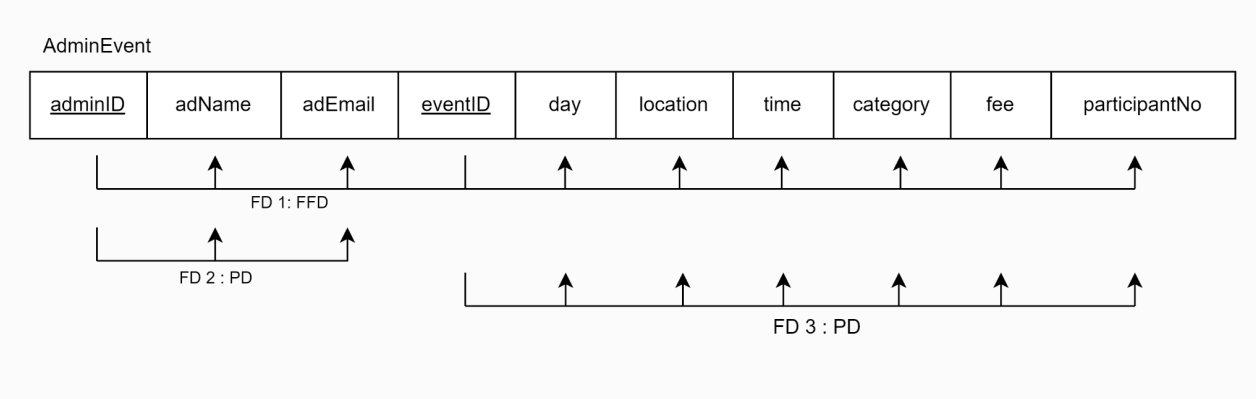
- CustomerAdmin (numIC, custName, custEmail, adminID, adName, adEmail)
  PK : numIC

3rd Normalization Form

- Admin (adminID, adName, adEmail)
  PK : adminID

- CustomerAdmin (numIC, custName custEmail, adminID)
  PK : numIC
  FK : adminID references Admin (adminID)

4.3.5 Normalization of AdminEvent Relation



FD1 : Full Functional Dependency

adminID, eventID → adName, adEmail, day, location, time, category, fee, participantNo

FD2 : Partial Dependency

adminID → adName, adEmail

FD3 : Partial Dependency

eventID → day, location, time, category, fee, participantNo

1st Normalization Form

- AdminEvent (adminID, eventID, adName, adEmail, day, location, time, category, fee, participantNo)
  PK : adminID, eventID

2nd Normalization Form

- Admin (adminID, adName, adEmail)
  PK : adminID

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- AdminEvent (adminID, eventID)
  PK : adminID, eventID
  FK : adminID references Admin (adminID)
  FK : eventID references Event (eventID)

3rd Normalization Form

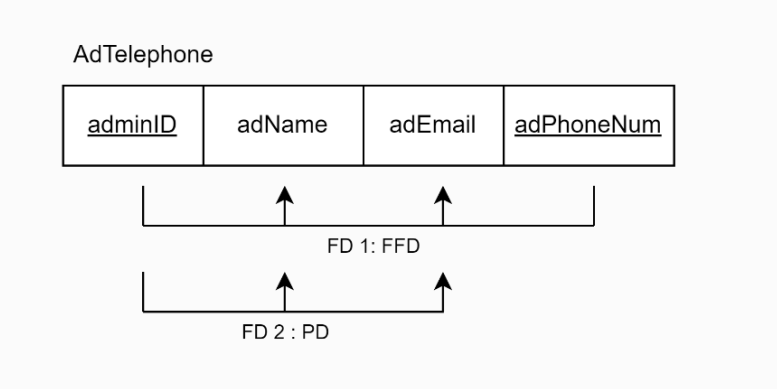
Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form

- Admin (adminID, adName, adEmail)
  PK : adminID

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- AdminEvent (adminID, eventID)
  PK : adminID, eventID
  FK : adminID references Admin (adminID)
  FK : eventID references Event (eventID)

4.3.6 Normalization of AdTelephone Relation



FD1 : Full Functional Dependency

adminID, phoneNum → adName, adEmail

FD2 : Partial Dependency

adminID → adName, adEmail

1st Normalization Form

- AdTelephone (numIC, phoneNum, adName, adEmail)
  PK : adminID,phoneNum

2nd Normalization Form

- Admin (adminID, adName, adEmail)
  PK : adminID

- AdTelephone (adminID, phoneNum)
  PK : adminID, phoneNum
  FK : adminID references Admin (adminID)

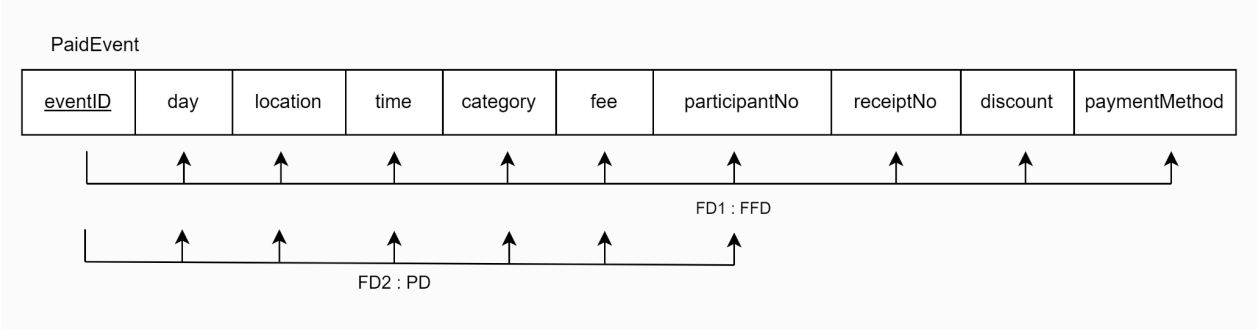
3rd Normalization Form

Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form.

- Admin (adminID, adName, adEmail)
  PK : adminID

- AdTelephone (adminID, phoneNum)
  PK : adminID, phoneNum
  FK : adminID references Admin (adminID)

4.3.7 Normalization of PaidEvent Relation



FD1 : Full Functional Dependency

eventID → day, location, time, category, fee, participantNo, receiptNo, discount, paymentMethod

FD2 : Partial Dependency

eventID → day, location, time, category, fee, participantNo

1st Normalization Form

- PaidEvent (eventID, day, location, time, category, fee, participantNo, receiptNo, discount, paymentMethod)
  PK : eventID

2nd Normalization Form

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- PaidEvent (eventID, receiptNo, discount, paymentMethod)
  PK : eventID
  FK : eventID references Event (eventID)

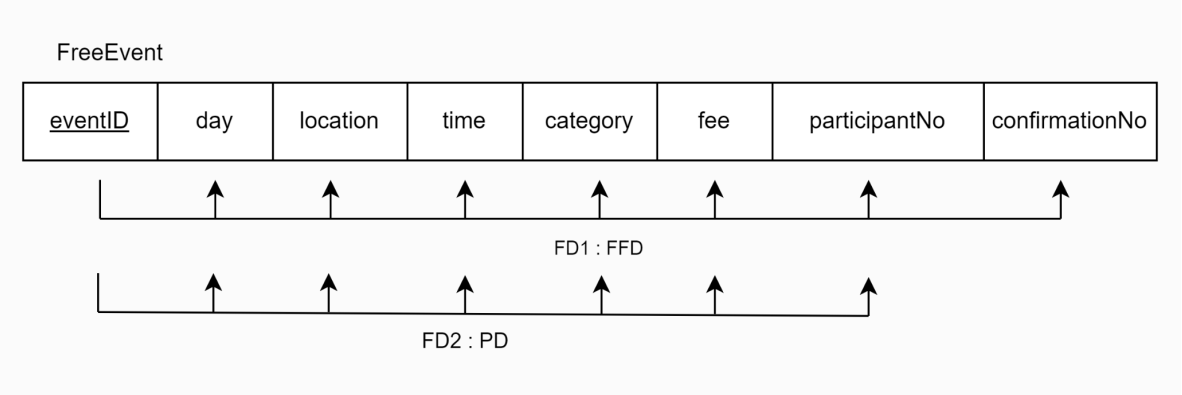
3rd Normalization Form

Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form.

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- PaidEvent (eventID, receiptNo, discount, paymentMethod)
  PK : eventID
  FK : eventID references Event (eventID)

4.3.8 Normalization of FreeEvent Relation



FD1 : Full Functional Dependency

eventID → day, location, time, category, fee, participantNo, receiptNo, discount, paymentMethod

FD2 : Partial Dependency

eventID → day, location, time, category, fee, participantNo

1st Normalization Form

- FreeEvent (eventID, day, location, time, category, fee, participantNo, confirmationNo)
  PK : eventID

2nd Normalization Form

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- FreeEvent (eventID, confirmationNo)
  PK : eventID
  FK : eventID references Event (eventID)

3rd Normalization Form

Since there is no transitive dependency on this relation, hence the third normalization form will be the same as the second normalization form.

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- FreeEvent (eventID, confirmationNo)
  PK : eventID
  FK : eventID references Event (eventID)

5.0 Relational DB Schemas (after normalization)

- Admin (adminID, adName, adEmail)
  PK : adminID

- AdTelephone (adminID, phoneNum)
  PK : adminID, phoneNum
  FK : adminID references Admin (adminID)

- Customer (numIC, custName, custEmail)
  PK : numIC

- CustTelephone (numIC, phoneNum)
  PK : numIC,phoneNum
  FK : numIC references Customer (numIC)

- Bank (accNumber, name, amount)
  PK : accNumber

- Event (eventID, day, location, time, category, fee, participantNo)
  PK : eventID

- PaidEvent (eventID, receiptNo, discount, paymentMethod)
  PK : eventID
  FK : eventID references Event (eventID)

- FreeEvent (eventID, confirmationNo)
  PK : eventID
  FK : eventID references Event (eventID)

- AdminEvent (adminID, eventID)

PK : adminID, eventID

FK : adminID references Admin (adminID)

FK : eventID references Event (eventID)

- CustEvent (numIC, eventID)

  PK : numIC, eventID

  FK : numIC references Customer (numIC)

  FK : eventID references Event (eventID)

- CustomerAdmin (numIC, custName custEmail, adminID)

  PK : numIC

  FK : adminID references Admin (adminID)

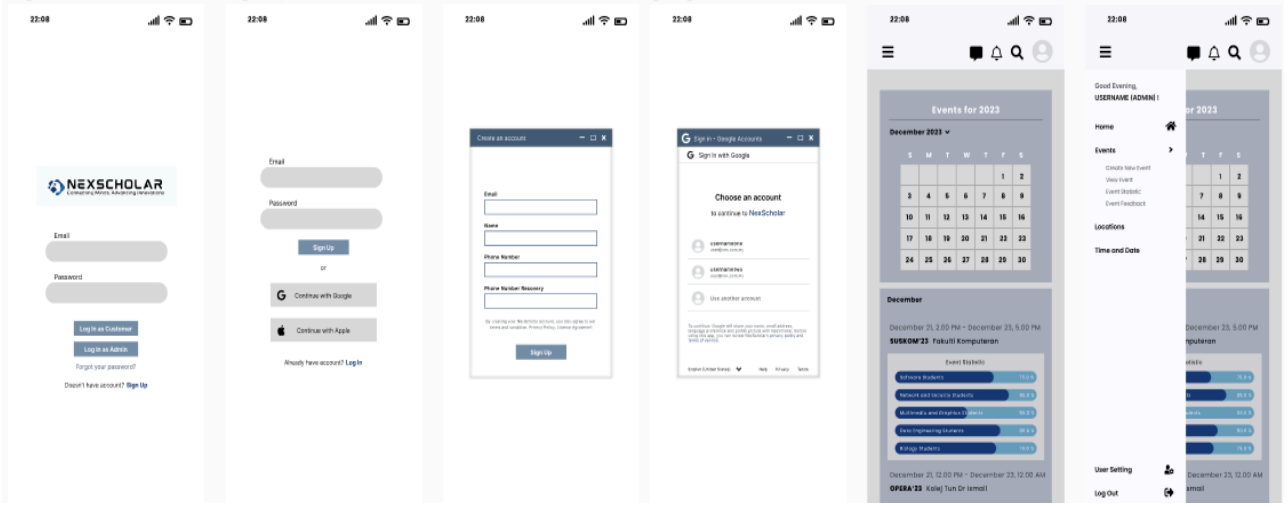
- CustomerBank (numIC, custName, custEmail, accNumber)

  PK : numIC

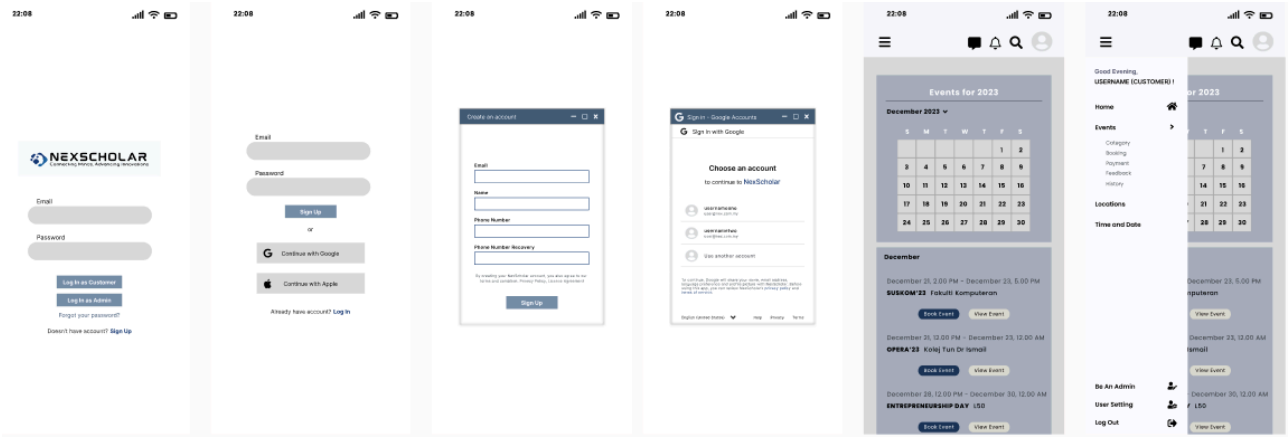
  FK : accNumber references Bank (accNumber)

6.0 Interface Design

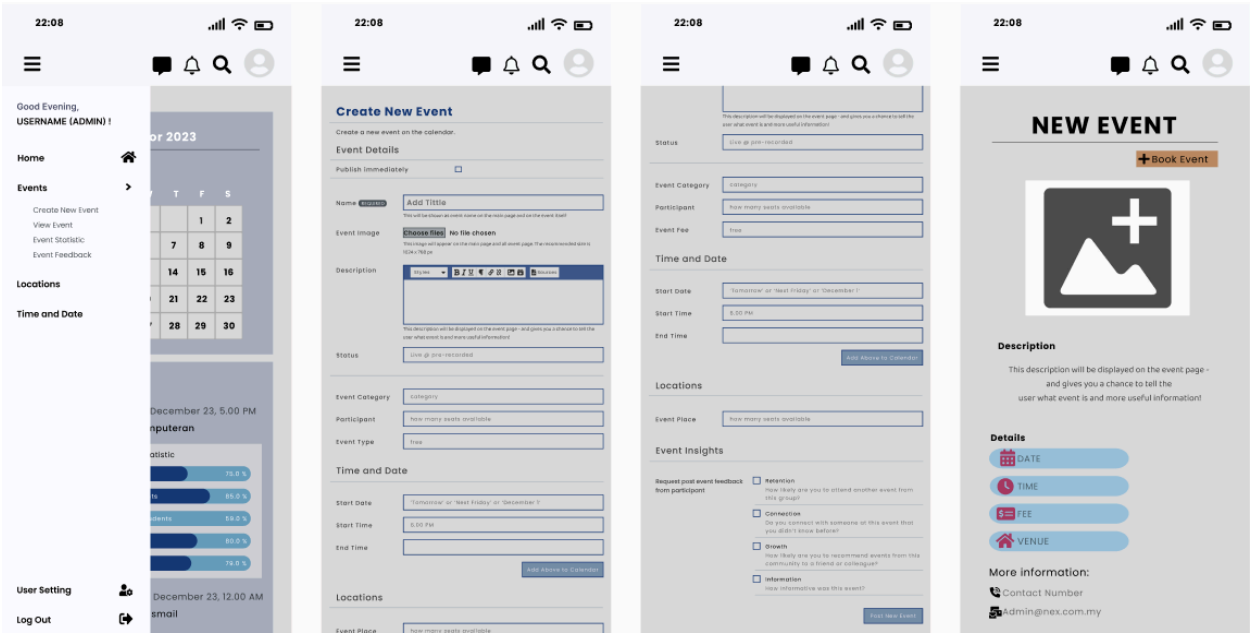
Process 1: Log In (Admin)



Process 1: Log In (Customer)

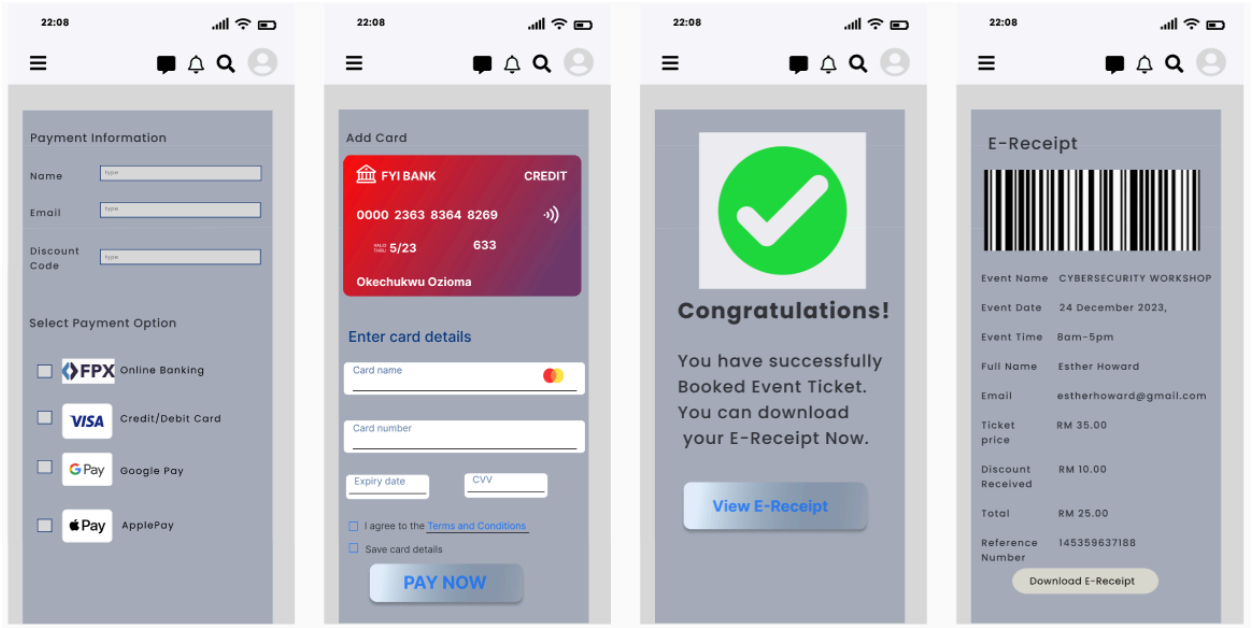


Process 2: Create New Event



Process 3: View Event

Process 4: Make Payment



7.0 SQL Statements (DDL & DML)

7.1 Creating Table

```
/* Creating table named Admin with adminID as primary key */
CREATE TABLE Admin (
   adminID VARCHAR2(6) NOT NULL,
   adName VARCHAR2(50) NOT NULL,
   adEmail VARCHAR2(50) NOT NULL,
   CONSTRAINT admin_pk PRIMARY KEY (adminID),
   CONSTRAINT ad_email_uk UNIQUE (adEmail)
);

/* Creating table named AdTelephone with adPhoneNum as primary key and
adminID as foreign key with references to Admin (adminID)  */
CREATE TABLE AdTelephone (
   adPhoneNum VARCHAR2(15) NOT NULL,
   adminID VARCHAR2(15) NOT NULL,
   CONSTRAINT ad_tel_pk PRIMARY KEY (adPhoneNum),
   CONSTRAINT ad_tel_fk FOREIGN KEY (adminID) REFERENCES Admin
(adminID)
);

/* Creating table named Customer with numIC as primary key */
CREATE TABLE Customer (
   numIC VARCHAR2(15) NOT NULL,
   custName VARCHAR2(50) NOT NULL,
   custEmail VARCHAR2(50) NOT NULL,
   CONSTRAINT cust_pk PRIMARY KEY (numIC)
);
```

/* Creating table named CustTelephone with phoneNum as primary key and numIC as foreign key with references to Customer (numIC) */

```
CREATE TABLE CustTelephone (
   phoneNum VARCHAR2(15) NOT NULL,
   numIC VARCHAR2(15) NOT NULL,
   CONSTRAINT cust_tel_pk PRIMARY KEY (phoneNum),
   CONSTRAINT cust_tel_fk FOREIGN KEY (numIC) REFERENCES Customer
(numIC)
);
```

/* Creating table named Bank with accNumber as primary key */

```
CREATE TABLE Bank (
   accNumber VARCHAR2 (15) NOT NULL,
   name VARCHAR2(20) NOT NULL,
   amount NUMBER(10,2) NOT NULL,
   CONSTRAINT bank_pk PRIMARY KEY (accNumber)
);
```

/* Creating table named Event with eventID as primary key */

```
CREATE TABLE Event (
   eventID VARCHAR2(6) NOT NULL,
   day_ VARCHAR2(10) NOT NULL,
   location VARCHAR2(20) NOT NULL,
   time_ DATE NOT NULL,
   category VARCHAR2(10) NOT NULL,
   fee NUMBER(10,2),
   participantNo NUMBER(4) NOT NULL,
   CONSTRAINT event_pk PRIMARY KEY (eventID)
);
```

/* Creating table named PaidEvent with eventID as primary key and eventID as foreign key with references to Event (eventID) */

```
CREATE TABLE PaidEvent (
   eventID VARCHAR2(6) NOT NULL,
   receiptNo VARCHAR2(10) NOT NULL,
   discount NUMBER(2),
   paymentMethod VARCHAR2(20) NOT NULL,
   CONSTRAINT paid_event_pk PRIMARY KEY (eventID),
   CONSTRAINT paid_event_fk FOREIGN KEY (eventID) REFERENCES Event
(eventID)
);
```

/* Creating table named FreeEvent with eventID as primary key and eventID as foreign key with references to Event (eventID) */

```
CREATE TABLE FreeEvent (
   eventID VARCHAR2(6) NOT NULL,
   confirmationNo VARCHAR2(10) NOT NULL,
   CONSTRAINT free_event_pk PRIMARY KEY (eventID),
   CONSTRAINT free_event_fk FOREIGN KEY (eventID) REFERENCES Event
(eventID)
);
```

/* Creating table named AdminEvent with adminID, eventID as primary key and adminID as foreign key with references to Admin (adminID) also eventID as foreign key with references to Event (eventID) */

```
CREATE TABLE AdminEvent (
   adminID VARCHAR2(6),
   eventID VARCHAR2 (6),
   CONSTRAINT adminevent_pk PRIMARY KEY (adminID,eventID),
   CONSTRAINT admin_ID_fk FOREIGN KEY (adminID) REFERENCES Admin
(adminID),
```

```
   CONSTRAINT event_ID_admin_fk FOREIGN KEY (eventID) REFERENCES Event
(eventID)
);
/* Creating table named CustEvent with numIC, eventID as primary key and
numIC as foreign key with references to Customer (numIC) also eventID as foreign
key with references to Event (eventID) */
CREATE TABLE CustEvent (
   numIC VARCHAR2(15),
   eventID VARCHAR2 (6),
   CONSTRAINT custevent_pk PRIMARY KEY (numIC,eventID),
   CONSTRAINT num_IC_fk FOREIGN KEY (numIC) REFERENCES Customer
(numIC),
   CONSTRAINT event_ID_cust_fk FOREIGN KEY (eventID) REFERENCES Event
(eventID)
);

/* Creating table named CustomerBank with numIC as primary key and
accNumber as foreign key with references to Bank (accNumber) */
CREATE TABLE CustomerBank (
   numIC VARCHAR2(15),
   custName VARCHAR2(50) NOT NULL,
   custEmail VARCHAR2(50) NOT NULL,
   accNumber VARCHAR2 (15) NOT NULL,
   CONSTRAINT cust_bank_pk PRIMARY KEY (numIC),
   CONSTRAINT cust_bank_fk FOREIGN KEY (accNumber) REFERENCES Bank
(accNumber)
);

/* Creating table named CustomerAdmin with numIC as primary key and adminID
as foreign key with references to Admin (adminID) */
CREATE TABLE CustomerAdmin (
```

```
    numIC VARCHAR2(15),
    custName VARCHAR2(50) NOT NULL,
    custEmail VARCHAR2(50) NOT NULL,
    adminID VARCHAR2 (6) NOT NULL,
    CONSTRAINT cust_admin_pk PRIMARY KEY (numIC),
    CONSTRAINT cust_admin_fk FOREIGN KEY (adminID) REFERENCES Admin
(adminID)
);
```

7.2 Inserting data

```
/* Inserting data into Admin table */
INSERT INTO Admin
VALUES ('AD0001', 'Nur Hafizah Jafri', 'hafizah@nex.com.my');

INSERT INTO Admin
VALUES ('AD0002', 'Farah Hazirah Nisha', 'farah@nex.com.my');

INSERT INTO Admin
VALUES ('AD0003', 'Nursyuhada Badren', 'syuhada@nex.com.my');

INSERT INTO Admin
VALUES ('AD0004', 'Sarah Sofea Anuar ', 'sarah@nex.com.my');

INSERT INTO Admin
VALUES ('AD0005', 'Salini Ravinthiran', 'salini@nex.com.my');

INSERT INTO Admin
VALUES ('AD0006', 'Wan Muhammad Faris', 'faris@nex.com.my');
```

	ADMINID	ADNAME	ADEMAIL
1	AD0001	Nur Hafizah Jafri	hafizah@nex.com.my
2	AD0002	Farah Hazirah Nisha	farah@nex.com.my
3	AD0003	Nursyuhada Badren	syuhada@nex.com.my
4	AD0004	Sarah Sofea Anuar	sarah@nex.com.my
5	AD0005	Salini Ravinthiran	salini@nex.com.my
6	AD0006	Wan Muhammad Faris	faris@nex.com.my

/* Inserting data into AdTelephone table */

```
INSERT INTO AdTelephone
VALUES ('0132826430', 'AD0001');

INSERT INTO AdTelephone
VALUES ('01110833455', 'AD0002');

INSERT INTO AdTelephone
VALUES ('0173750797', 'AD0003');

INSERT INTO AdTelephone
VALUES ('0134107723', 'AD0004');

INSERT INTO AdTelephone
VALUES ('0173903650', 'AD0005');

INSERT INTO AdTelephone
VALUES ('019547632', 'AD0005');
```

	ADPHONENUM	ADMINID
1	0132826430	AD0001
2	01110833455	AD0002
3	0173750797	AD0003
4	0134107723	AD0004
5	0173903650	AD0005
6	019547632	AD0005

/* Inserting data into Bank table */

```
INSERT INTO Bank
VALUES ('87654321', 'CIK AZRA ATHIRAH', 30.00);

INSERT INTO Bank
VALUES ('56789765', 'ENCIK HARRAZ', 10.00);
```

	ADPHONENUM	ADMINID
1	0132826430	AD0001
2	01110833455	AD0002
3	0173750797	AD0003
4	0134107723	AD0004
5	0173903650	AD0005

/* Inserting data into Event table */

```
INSERT INTO Event
VALUES ('EF1001', 'Friday', 'DSR', TO_DATE('10-09-2023', 'DD-MM-YYYY'),
'Academic', NULL, 100);

INSERT INTO Event
VALUES ('EF1002', 'Monday', 'DSI', TO_DATE('13-11-2023', 'DD-MM-YYYY'),
'Academic', NULL, 600);

INSERT INTO Event
VALUES ('EP1003', 'Wednesday', 'Tasek Ilmu', TO_DATE('22-11-2023',
'DD-MM-YYYY'), 'Sports', 30.00, 300);
```

INSERT INTO Event
VALUES ('EF1004', 'Thursday', 'L50', TO_DATE('23-11-2023', 'DD-MM-YYYY'),
'Volunteer', NULL, 300);

INSERT INTO Event
VALUES ('EP1005', 'Monday', 'L50', TO_DATE('04-12-2023', 'DD-MM-YYYY'),
'Career', 10.00, 300);

	EVENTID	DAY_	LOCATION	TIME_	CATEGORY	FEE	PARTICIPANTNO
1	EF1001	Friday	DSR	10/09/2023	Academic	(null)	100
2	EF1002	Monday	DSI	13/11/2023	Academic	(null)	600
3	EP1003	Wednesday	Tasek Ilmu	22/11/2023	Sports	30	300
4	EF1004	Thursday	L50	23/11/2023	Volunteer	(null)	300
5	EP1005	Monday	L50	04/12/2023	Career	10	300

/* Inserting data into PaidEvent table */
INSERT INTO PaidEvent
VALUES ('EP1003', 'ABC130000', NULL, 'ONLINE BANKING');

INSERT INTO PaidEvent
VALUES ('EP1005', 'DEF140000', 10, 'DEBIT CARD');

	EVENTID	RECEIPTNO	DISCOUNT	PAYMENTMETHOD
1	EP1003	ABC130000	(null)	ONLINE BANKING
2	EP1005	DEF140000	10	DEBIT CARD

/* Inserting data into FreeEvent table */
INSERT INTO FreeEvent
VALUES ('EF1001', 'GHI110000');

INSERT INTO FreeEvent
VALUES ('EF1002', 'JKL120000');

INSERT INTO FreeEvent

VALUES ('EF1004', 'MNO140000');

	EVENTID	CONFIRMATIONNO
1	EF1001	GHI110000
2	EF1002	JKL120000
3	EF1004	MNO140000

/* Inserting data into Customer table */

INSERT INTO Customer

VALUES ('010203045565', 'Wan Mohammad Faris', 'faris@nex.com.my');

INSERT INTO Customer

VALUES ('010203045566', 'Azra Athirah Azahari', 'azra@nex.com.my');

INSERT INTO Customer

VALUES ('010203045568', 'Hanis Batrisya', 'hanis@nex.com.my');

INSERT INTO Customer

VALUES ('010203045563', 'Muhammad Manul', 'manul@nex.com.my');

INSERT INTO Customer

VALUES ('010203045567', 'Harraz', 'harraz@nex.com.my');

	NUMIC	CUSTNAME	CUSTEMAIL
1	010203045565	Wan Mohammad Faris	faris@nex.com.my
2	010203045566	Azra Athirah Azahari	azra@nex.com.my
3	010203045568	Hanis Batrisya	hanis@nex.com.my
4	010203045563	Muhammad Manul	manul@nex.com.my
5	010203045567	Harraz	harraz@nex.com.my

/* Inserting data into CustTelephone table */

```
INSERT INTO CustTelephone
VALUES ('0189681089', '010203045565');

INSERT INTO CustTelephone
VALUES ('0108237656', '010203045566');

INSERT INTO CustTelephone
VALUES ('01118969315', '010203045568');

INSERT INTO CustTelephone
VALUES ('0194492959', '010203045563');

INSERT INTO CustTelephone
VALUES ('0189681087', '010203045567');
```

	PHONENUM	NUMIC
1	0189681089	010203045565
2	0108237656	010203045566
3	01118969315	010203045568
4	0194492959	010203045563
5	0189681087	010203045567

/* Inserting data into CustomerBank table */

```
INSERT INTO CustomerBank
VALUES ('010203045566', 'Azra Athirah Azahari', 'azra@nex.com.my','87654321');

INSERT INTO CustomerBank
VALUES ('010203045567', 'Harraz', 'harraz@nex.com.my','56789765');
```

	NUMIC	CUSTNAME	CUSTEMAIL	ACCNUMBER
1	010203045566	Azra Athirah Azahari	azra@nex.com.my	87654321
2	010203045567	Harraz	harraz@nex.com.my	56789765

/* Inserting data into CustomerAdmin table */

```
INSERT INTO CustomerAdmin
VALUES ('010203045565', 'Wan Mohammad Faris', 'faris@nex.com.my','AD0006');
```

	NUMIC	CUSTNAME	CUSTEMAIL	ADMINID
1	010203045565	Wan Mohammad Faris	faris@nex.com.my	AD0006

/* Inserting data into AdminEvent table */

```
INSERT INTO AdminEvent
VALUES ('AD0001', 'EF1001');

INSERT INTO AdminEvent
VALUES ('AD0002', 'EF1002');

INSERT INTO AdminEvent
VALUES ('AD0003', 'EP1003');

INSERT INTO AdminEvent
VALUES ('AD0004', 'EF1004');

INSERT INTO AdminEvent
VALUES ('AD0006', 'EP1005');
```

	ADMINID	EVENTID
1	AD0001	EF1001
2	AD0002	EF1002
3	AD0003	EP1003
4	AD0004	EF1004
5	AD0006	EP1005

/* Inserting data into CustEvent table */

```
INSERT INTO CustEvent
VALUES ('010203045565', 'EF1001');

INSERT INTO CustEvent
VALUES ('010203045568', 'EF1002');

INSERT INTO CustEvent
VALUES ('010203045566', 'EP1003');

INSERT INTO CustEvent
VALUES ('010203045563', 'EF1004');

INSERT INTO CustEvent
VALUES ('010203045567', 'EP1005');

INSERT INTO CustEvent
VALUES ('010203045567', 'EF1004');
```

	NUMIC	EVENTID
1	010203045565	EF1001
2	010203045568	EF1002
3	010203045566	EP1003
4	010203045563	EF1004
5	010203045567	EP1005
6	010203045567	EF1004

7.3 Appropriate Queries DML skills

/* Delete data from AdTelephone table where adPhoneNum = '019547632' */

```
DELETE FROM AdTelephone
WHERE adPhoneNum = '019547632';
```

	ADPHONENUM	ADMINID
1	0132826430	AD0001
2	01110833455	AD0002
3	0173750797	AD0003
4	0134107723	AD0004
5	0173903650	AD0005

/* Update data in Event table location = 'Tasek Ilmu' -> 'Tasik Ilmu' where eventID = 'EP1003' */

```
UPDATE EVENT
SET location = 'Tasik Ilmu'
WHERE eventID = 'EP1003';
```

	EVENTID	DAY_	LOCATION	TIME_	CATEGORY	FEE	PARTICIPANTNO
1	EF1001	Friday	DSR	10/09/2023	Academic	(null)	100
2	EF1002	Monday	DSI	13/11/2023	Academic	(null)	600
3	EP1003	Wednesday	Tasik Ilmu	22/11/2023	Sports	30	300
4	EF1004	Thursday	L50	23/11/2023	Volunteer	(null)	300
5	EP1005	Monday	L50	04/12/2023	Career	10	300

/* Add column custAddress at Customer table */

```
ALTER TABLE Customer
ADD custAddress VARCHAR2(50);
```

BEFORE:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	NUMIC	VARCHAR2(15 BYTE)	No	(null)	1	(null)
2	CUSTNAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	CUSTEMAIL	VARCHAR2(50 BYTE)	No	(null)	3	(null)

AFTER:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	NUMIC	VARCHAR2(15 BYTE)	No	(null)	1	(null)
2	CUSTNAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	CUSTEMAIL	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	CUSTADDRESS	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)

/*Sorting data in Admin table as decreasing based on adName*/

```
SELECT * FROM Admin
ORDER BY adName DESC;
```

	ADMINID	ADNAME	ADEMAIL
1	AD0006	Wan Muhammad Faris	faris@nex.com.my
2	AD0004	Sarah Sofea Anuar	sarah@nex.com.my
3	AD0005	Salini Ravinthiran	salini@nex.com.my
4	AD0003	Nursyuhada Badren	syuhada@nex.com.my
5	AD0001	Nur Hafizah Jafri	hafizah@nex.com.my
6	AD0002	Farah Hazirah Nisha	farah@nex.com.my

/* Display data from Admin and AdminEvent by using left outer join ON (a.adminID = e.adminID) */

```
SELECT a.adName, a.adEmail, e.eventID
FROM Admin a LEFT OUTER JOIN AdminEvent e
ON (a.adminID = e.adminID);
```

	ADNAME	ADEMAIL	EVENTID
1	Nur Hafizah Jafri	hafizah@nex.com.my	EF1001
2	Farah Hazirah Nisha	farah@nex.com.my	EF1002
3	Nursyuhada Badren	syuhada@nex.com.my	EP1003
4	Sarah Sofea Anuar	sarah@nex.com.my	EF1004
5	Wan Muhammad Faris	faris@nex.com.my	EP1005
6	Salini Ravinthiran	salini@nex.com.my	(null)

/* Display data by custName that contain ‘a’ at second letter in their name */

```
SELECT custName, custEmail
FROM Customer
WHERE custName LIKE '_a%';
```

	CUSTNAME	CUSTEMAIL
1	Wan Mohammad Faris	faris@nex.com.my
2	Hanis Batrisya	hanis@nex.com.my
3	Harraz	harraz@nex.com.my

/* Display data with NULL value from Event table and replace column day_ name with DAY*/

SELECT eventID, day_ “DAY”, location, fee

FROM event

WHERE fee IS NULL;

	EVENTID	DAY	LOCATION	FEE
1	EF1001	Friday	DSR	(null)
2	EF1002	Monday	DSI	(null)
3	EF1004	Thursday	L50	(null)

8.0 Summary

The primary objective of this project is to facilitate and streamline the event creation process for NexScholar. The existing system only allows the admin to create events, not the users, which are customers. Our goal is to empower users to create their own events, ensuring the system is reliable and fully equipped for event creation.

With this goal, we concentrated on developing software and a database that would be more beneficial to the users, who are the customers in this scenario. Considering their needs and desires to create their own events, this proposed system enables users to do the same as an admin. This system is particularly useful for users who wish to organize their own events.

The proposed system allows customers to customize their own events, including setting their own time, date, and location. It also allows every participant to pay for the event or the organizer, which is the customer, to make it a free event. This software is designed to assist computer science students in learning about databases, system applications, and how to build a complete work application. One of the benefits of this project is that it allows for the creation of more events that can be beneficial to people. For instance, a user might want to create a cyber security talk event, which could provide valuable knowledge to many computer science students.

Through various analysis processes on our project, we gathered a wealth of information and gained an understanding of NexScholar's business requirements and processes. We then identified some weaknesses in the current NexScholar system. By using a method to gather all the information about the current NexScholar system, we successfully collected some key points for our project. We primarily used the interview method to gather all the information about the current NexScholar system. We also realized the importance of understanding the system flow before implementation to ensure smooth deployment without any issues. Our team members demonstrated excellent teamwork in completing this project, each with their own tasks to complete. As

we approached the completion of Phase 2, we understood the importance of non-functional and functional requirements in project implementation.