Sri Lanka Institute of Information Technology

Database Design, Implementation and Security

Group Assignment

Group NO -07

IE2042 – Database Management Systems for Security.

Submitted by:

Student Registration Number	Student Name
IT22353184	Rathnayaka R.M.C.A
IT22199508	Athapaththu A.M.M.I.P
IT22921512	Jayawardhana S.I.B
IT22298508	Wanasinghe W.M.K.R

Date of submission

2023.10.21

Table of Contents

Abstract

This report provides an overview of the Database Design, Implementation, and Security assignment, focusing on the role of buyers and sellers in the system. Buyers list items, while sellers bid for them, with the highest bidder purchasing the items.

After identifying the highest bidder, a transaction occurs between the buyer and seller, resulting in account changes. A logical model is created for the ER diagram, which is refined by eliminating redundancies and errors to achieve the third normalized form.

The SQL queries were used to create a table and insert values, adding triggers to ensure the buyer's bid price is always higher than the current bid. Another trigger was added to update accounts and check transaction feasibility. Different users were examined and access control measures were created for them. There were six questions related to procedures and the answers are provided below.

After end of part 1, The members conducted a thorough research on common database vulnerabilities, identifying SQLI and XXS as the most prevalent. They discussed their impact on the database and users, the techniques used, and discussed mitigations and countermeasures to help users reduce and eliminate these types of attacks.

1. Introduction

The assignment focused on an Online Auction database system scenario, requiring the creation of an ER diagram and a logical model to illustrate relationships within the given scenario.

SQL queries were written to create tables associated with the database system and input demo values, enabling the creation of views, triggers, indexes, and procedures based on the assumptions made.

Triggers were created to check for specific conditions before table insertion, and views were created by identifying user roles to control access levels, and procedures were written to answer assignment questions.

Databases, which contain valuable data, are often targeted by attackers for obtaining private and confidential information. Leaked data can be used for selling or further attacks. Many data breaches occur due to poor database design, making organizations vulnerable to potential breaches.

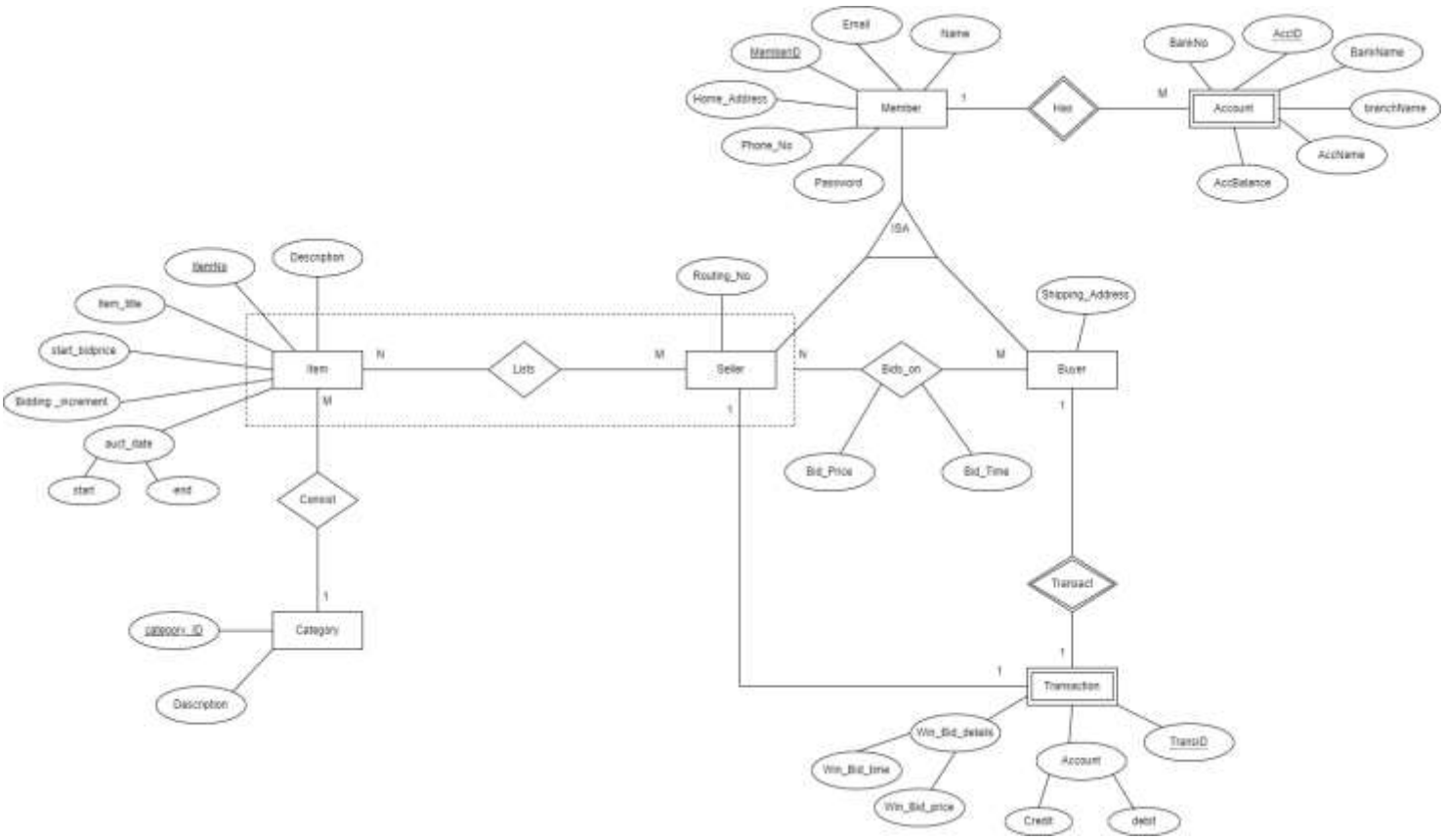
This assignment discusses SQL injection and cross-site scripting, two primary types of database attacks used by threat actors, their execution methods, and the benefits for attackers. It also details the benefits of these attacks.

The text highlights preventive measures against various attack types, highlighting how these security practices can enhance database security and reduce vulnerability to attacks.

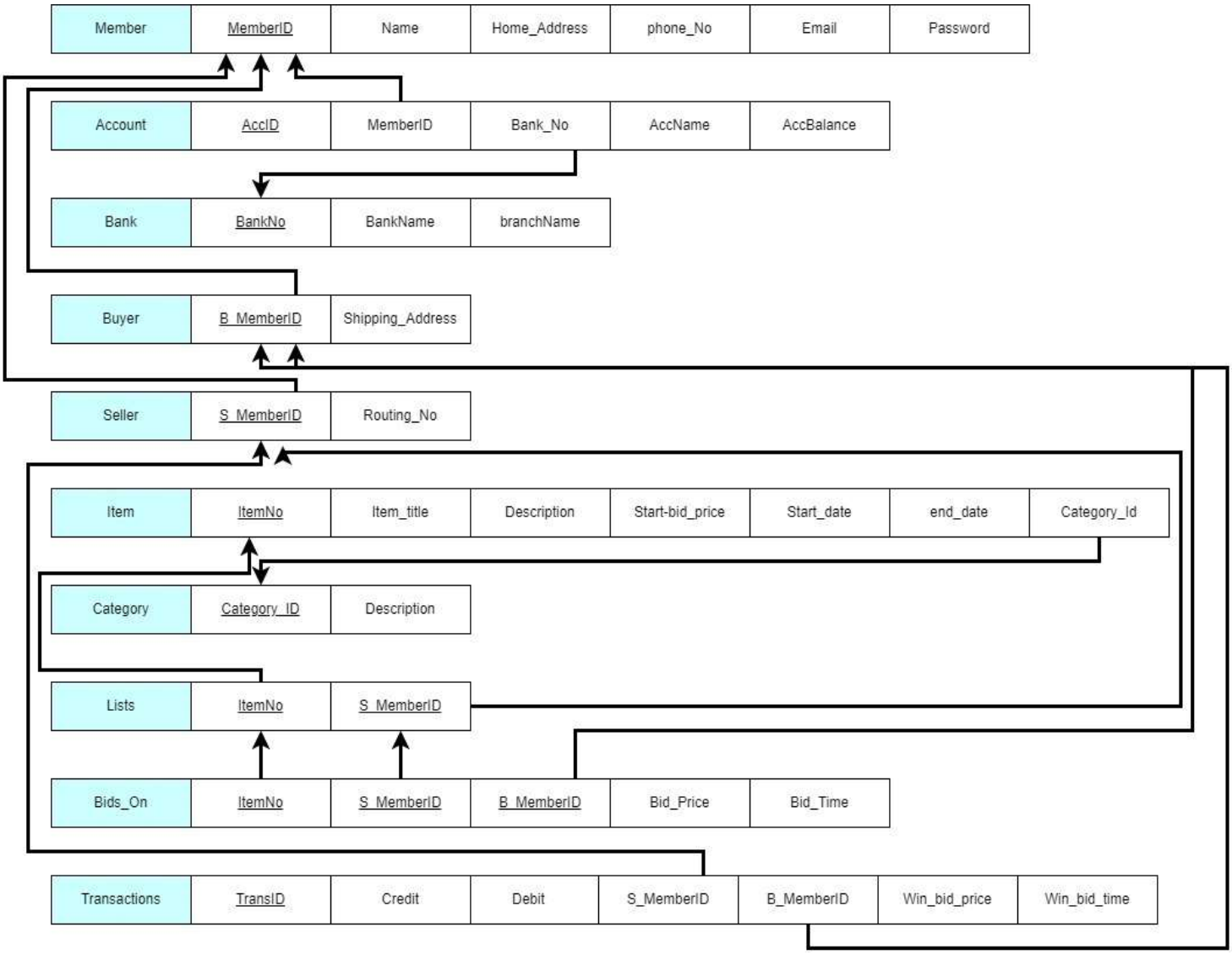
2. Assumptions

- A member can only enter one phone number.
- Account is recognized as a weak entity so it cannot exist if a Member does not exist.
- Both Buyer and Seller entities does not cover the Member entity (Ex: Administrator).
- The Buyer and Seller entities are disjoint.
- Aggregation refers that seller lists an item and the buyer bids on the item.
- Transaction occurs between buyer and seller entities as the account entity is connected through a weak entity.

3. ER Diagram


Email
Name
MemberID
Home_Address
Phone_No
Password
Member
1
Has
M
BankNo
AcctID
BankName
Account
branchName
AcctName
AccBalance
ISA
Routing_No
Shipping_Address
Item_title
start_bidprice
Bidding _increment
auct_date
start
end
Item
N
Lists
M
Seller
N
Bids_on
M
Buyer
Bid_Price
Bid_Time
Consist
1
M
1
1
1
Transact
1
1
Transaction
Win_Bid_details
Win_Bid_time
Win_Bid_price
Account
Credit
debit
TransID
category_ID
Category
Description

4. Logical Model



5. SQL Queries

5.1) Creating Tables

```
CREATE DATABASE ONLINEAUCTION;

--creation of Bank table--
CREATE TABLE Bank(
Bank_No char(5),
BankName varchar(20) NOT NULL,
branchName varchar(20),
CONSTRAINT pk_Bank PRIMARY KEY (Bank_No)
);

--creation of Member table--
CREATE TABLE Member(
 MemberID varchar(10),
 Name varchar(10),
 Home_Address varchar(50),
 phone_No nchar(10),
 Email varchar(50),
 Password varchar(10),
 CONSTRAINT pk_Member PRIMARY KEY (MemberID),
 CONSTRAINT me_chk1 check(Email like '%@%'),
 CONSTRAINT me_chk2 check(Password like '%[0-9]%' and Password like '%[A-Z]%' and
 Password like '%[!@#$%a^&*()-_+=.,;:"`~]%' )
);

--creation of Account table--
CREATE TABLE Account(
 AccID varchar(18),
 MemberID varchar(10),
 Bank_No char(5),
 AccName varchar(10),
 AccBalance real,
 CONSTRAINT ac_pk PRIMARY KEY(MemberID,AccID),
 CONSTRAINT ac_fk1 FOREIGN KEY(MemberID) references Member(MemberID),
 CONSTRAINT ac_fk2 FOREIGN KEY(Bank_No) references Bank(Bank_No)
);
```

```sql
--creation of Buyer table--
CREATE TABLE Buyer(
 B_MemberID varchar(10),
 Shipping_Address varchar(50),
 CONSTRAINT buy_pk PRIMARY KEY(B_MemberID),
 CONSTRAINT buy_fk FOREIGN KEY(B_MemberID) references Member(MemberID)
);

--creation of Seller table--
CREATE TABLE Seller(
 S_MemberID varchar(10),
 Routine_No varchar(50),
 CONSTRAINT sell_pk PRIMARY KEY(S_MemberID),
 CONSTRAINT sell_fk FOREIGN KEY(S_MemberID) references Member(MemberID)
);

--creation of Category table--
CREATE TABLE Category(
 Category_ID varchar(10),
 Description varchar(50),
 CONSTRAINT cat_pk PRIMARY KEY(Category_ID)
);
```

```sql
--creation of Item table--
CREATE TABLE Item(
 ItemNo varchar (10),
 Item_title varchar(10),
 Description varchar(50),
 start_bidprice real,
 start_date date,
 end_date date,
 Bidding_increasement real,
 Category_ID varchar(10),
  CONSTRAINT it_pk PRIMARY KEY(ItemNo),
 CONSTRAINT it_fk FOREIGN KEY(Category_ID) references Category(Category_ID)
);

--creation of Lists table--
CREATE TABLE Lists(
 ItemNo varchar(10),
 S_MemberID varchar(10),
 CONSTRAINT li_pk PRIMARY KEY(ItemNo,S_MemberID),
 CONSTRAINT li_fk FOREIGN KEY(S_MemberID) references Seller(S_MemberID)
);

--creation of Bids_on table--
CREATE TABLE Bids_On(
 ItemNo varchar(10),
 S_MemberID varchar(10),
 B_MemberID varchar(10),
 Bid_Price real,
 Bid_Time datetime DEFAULT(getdate()),
 constraint bid_pk primary key(ItemNo,S_MemberID,B_MemberID),
 constraint bid_fk1 foreign key(ItemNo) references Item(ItemNo),
 constraint bid_fk2 foreign key(B_MemberID) references Buyer(B_MemberID),
 constraint bid_fk3 foreign key(ItemNo,S_MemberID) references Lists(ItemNo,S_MemberID)
);
```

```
--creation of Transations table--
CREATE TABLE Transactions(
 TransID varchar(10),
 Credit varchar(10),
 Debit varchar(10),
 S_MemberID varchar(10),
 B_MemberID varchar(10),
 Win_bid_price real,
 Win_bid_time datetime DEFAULT(getdate()),
 constraint tr_fk1 foreign key(B_MemberID) references Buyer(B_MemberID),
 constraint tr_fk2 foreign key(S_MemberID) references Seller(S_MemberID)
);
```

5.2) Inserting Data into Tables

```
--Data insertion of Bank table--
INSERT INTO Bank VALUES ('1','Sampath Bank','colombo');
INSERT INTO Bank VALUES ('2','Bank of Ceylon','kurunegala');
INSERT INTO Bank VALUES ('3','Peoples Bank','negambo');
INSERT INTO Bank VALUES ('4','NDB bank','galle');
INSERT INTO Bank VALUES ('5','Commercial Bank','malabe');


--Data insertion of Member table--
INSERT INTO Member VALUES ('001', 'Saman','No 432, Colombo','0715224578',
'saman@gmail.com','aaA46@&$');
INSERT INTO Member VALUES ('002', 'perera','No 22, kurunegala','0725224568',
'perera@gmail.com.com','aBrg44$$');
INSERT INTO Member VALUES ('003', 'supun','No 35, negombo','0765234578',
'supun@gmail.com','mMlL33@$');
INSERT INTO Member VALUES ('004', 'namal','No 75, fort','0755233578',
'namal@gmail.com','aaAA44$$');
INSERT INTO Member VALUES ('005', 'Yasas','No 865, malabe town','0755233578',
'Yasas@gmail.com.com','gdAB55@@');


--Data insertion of Account table--
INSERT INTO Account VALUES ('11111','001','1','Saving', 100000.00);
INSERT INTO Account VALUES ('22222','002','2','Saving', 60000.00);
INSERT INTO Account VALUES ('33333','003','3','Current', 30000.00);
INSERT INTO Account VALUES ('44444','004','4','Saving', 21000.00);
INSERT INTO Account VALUES ('55555','005','5','Current', 110000.00);


--Data insertion of Buyer table--
INSERT INTO Buyer VALUES ('001','No 432, Colombo');
INSERT INTO Buyer VALUES ('002','No 22, kurunegala');
INSERT INTO Buyer VALUES ('003','No 35, negombo');
```

```
--Data insertion of Seller table--
INSERT INTO Seller VALUES ('004','No 75, fort');
INSERT INTO Seller VALUES ('005','No 865, malabe town');


--Data insertion of Category table--
INSERT INTO Category VALUES('0001', 'Electronic');
INSERT INTO Category VALUES('0002', 'vehicle');
INSERT INTO Category VALUES('0003', 'antics');
INSERT INTO Category VALUES('0004', 'dress');
INSERT INTO Category VALUES('0005', 'household');


--Data insertion of Item table--
INSERT INTO Item VALUES ('01','Laptop', 'Laptop Description', 100000.00, '2023/10/26',
'2023/11/10',101000.00,'0001');
INSERT INTO Item VALUES ('02','picture', 'picture Description', 50000.00, '2023/09/28',
'2023/10/05',52000.00, '0003');
INSERT INTO Item VALUES ('03','cab', 'Cab Description', 5000000.00, '2023/10/06',
'2023/11/20',5025000.00, '0002');
INSERT INTO Item VALUES ('04','Table', 'Table Description', 10000.00, '2023/10/17',
'2023/10/30',11000.00, '0005');
INSERT INTO Item VALUES ('05','T-shirt', 'T-shirt Description', 4000.00, '2023/10/21',
'2023/10/28',4500.00, '0004');


--Data insertion of Lists table--
INSERT INTO Lists VALUES ('01', '004');
INSERT INTO Lists VALUES ('02', '005');
INSERT INTO Lists VALUES ('03', '004');
INSERT INTO Lists VALUES ('04', '004');
INSERT INTO Lists VALUES ('05', '005');
```

```
--Data insertion of Bids_on table--
INSERT INTO Bids_On VALUES ('01', '004', '001', 140000.00, '2023/10/28 07:45:10');
INSERT INTO Bids_On VALUES ('01', '004', '002', 160000.00, '2023/10/28 15:50:10');
INSERT INTO Bids_On VALUES ('02', '005', '001', 130000.00, '2023/09/30 02:45:10');
INSERT INTO Bids_On VALUES ('02', '005', '003', 140000.00, '2023/10/01 04:45:10');
INSERT INTO Bids_On VALUES ('04', '004', '002', 15500.00, '2023/10/18 08:45:10');
INSERT INTO Bids_On VALUES ('04', '004', '001', 16000.00, '2023/10/19 11:45:10');


--Data insertion of Transation table--
INSERT INTO Transactions VALUES ('T1','222222', '444444',
'004', '002', 160000.00, '2023/10/28 15:50:10');
INSERT INTO Transactions VALUES ('T2','333333', '555555',
'005', '003', 140000.00, '2023/10/01 04:45:10');
INSERT INTO Transactions VALUES ('T3','111111', '444444',
'004', '001', 16000.00, '2023/10/19 11:45:10');
```

5.3) Triggers

```
--01)–
```

This trigger is implemented to check whether the Account balance in Account table is lower than 25000.00 rupees, if so, display an error message.

```
CREATE TRIGGER Account_Balance_Check
ON Account
AFTER INSERT
AS
BEGIN
 IF EXISTS(SELECT 1 FROM inserted WHERE AccBalance<=25000.00)
BEGIN
 RAISERROR ('Bank Balance cannot be less than 25000 Rupees',16,1);
 ROLLBACK
 RETURN
END
END
```

```
--02)–
```

This trigger is implemented so once the buyer with the highest bid price needs to do the transaction to the seller, first the account balance of the buyer will be checked to see if it is sufficient else it will be rolled back. If the balance is sufficient updates on the buyer and seller accounts will take place.

```
CREATE TRIGGER CheckTransaction
ON Transactions
FOR INSERT
AS
BEGIN
 DECLARE @bidPrice real, @bAccNo varchar(18), @sAccNo varchar(18)
 SELECT @bidPrice=Win_bid_price, @bAccNo=Credit, @sAccNo=Debit
 FROM Transactions;
 DECLARE @bAccBal real, @sAccBal real
 SELECT @bAccBal=AccBalance
 FROM Account
 WHERE AccID = @bAccNo;
 SELECT @sAccBal=AccBalance
 FROM Account
 WHERE AccID = @sAccNo;
 IF @bAccBal < @bidPrice
 ROLLBACK TRANSACTION
 ELSE

  BEGIN
   UPDATE Account
   SET AccBalance = @bAccBal - @bidPrice
   WHERE AccID = @bAccNo;
   UPDATE Account
   SET AccBalance = @sAccBal + @bidPrice
   WHERE AccID = @sAccNo;
END
END
```

5.4) Views

```
--01–
```

This view can be used by the **Administrator** to check for all the member details.

```
CREATE VIEW [Member Details]
AS
SELECT
M.MemberID,M.Name,M.Email,M.Home_Address,M.phone_No,B.BankName,B.branchName,A.AccID
AS [Account No],A.AccBalance AS [Account Blance]
FROM Account A, Member M, Bank B
WHERE M.MemberID=A.MemberID AND A.Bank_No=B.Bank_No;

Select * from [Member Details]
```

Results | Messages

	MemberID	Name	Home_Address	phone_No	Email	Password
1	001	Saman	No 432, Colombo	0715224578	saman@gmail.com	aaA46@&$
2	002	perera	No 22, kurunegala	0725224568	perera@gmail.com.com	aBrg44$$
3	003	supun	No 35, negombo	0765234578	supun@gmail.com	mMIL33@$
4	004	namal	No 75, fort	0755233578	namal@gmail.com	aaAA44$$
5	005	Yasas	No 865, malabe town	0755233578	Yasas@gmail.com.com	gdAB55@@

```
--02–
```

This view can be used in searches by **buyers** or **sellers** to find for any items that are being sold currently and information regarding them.

```
CREATE VIEW showItem
AS
SELECT mem.Name, itm.Item_title, itm.Description AS [Item Description], cat.Description AS
[Category]
FROM Item itm, Lists lst, Member mem, Category cat
WHERE itm.ItemNo = lst.ItemNo AND lst.S_MemberID = mem.MemberID AND itm.Category_ID = cat.Category_ID;

select * from showItem
```

Results | Messages

	Name	Item_title	Item Description	Category
1	namal	Laptop	Laptop Description	Electronic
2	Yasas	picture	picture Description	antics
3	namal	cab	Cab Description	vehicle
4	namal	Table	Table Description	household
5	Yasas	T-shirt	T-shirt Description	dress

5.5) Indexes

```
--Indexes--
--implement an index on Accbalance column in Account table--

CREATE INDEX INX_Acc_Balance
ON Account (AccBalance);

--implement an index on Bid_Price column in Bids_on--

CREATE INDEX INX_Bid_on
ON Bids_On (Bid_Price);
```

5.6) Procedures

```
--question 01--
CREATE PROCEDURE Member_Bank
AS
BEGIN
SELECT M.Name,M.Home_Address
FROM Member M, Account A, Bank B
WHERE M.MemberID=A.MemberID AND A.Bank_No=B.Bank_No AND B.BankName= 'Sampath Bank'
END
EXEC Member_Bank
```

Results | Messages

	Name	Home_Address
1	Saman	No 432, Colombo

```sql
--question 02--
CREATE PROCEDURE Member_Bid
AS
BEGIN
SELECT M.Name,M.Email,BN.Bid_Price
FROM Member M, Bids_On BN, Buyer Bu, Lists L, Item I
WHERE BN.B_MemberID=Bu.B_MemberID AND Bu.B_MemberID = M.MemberID AND
BN.ItemNo=L.ItemNo AND L.ItemNo = I.ItemNo AND I.Item_title='Laptop'
ORDER BY BN.Bid_Price
END

EXEC Member_Bid
```

Results Messages

	Name	Email	Bid_Price
1	perera	perera@gmail.com.com	160000
2	Saman	saman@gmail.com	161000

```sql
--question 03--
CREATE PROCEDURE Seller_Sum
AS
BEGIN
SELECT M.Name,SUM(I.start_bidprice)
FROM Item I, Lists L, Seller S, Member M
WHERE I.ItemNo=L.ItemNo AND S.S_MemberID=L.S_MemberID AND
S.S_MemberID=M.MemberID
GROUP BY M.Name
HAVING SUM(I.start_bidprice)>30000
END

EXEC Seller_Sum
```

Results Messages

	Name	(No column name)
1	namal	5110000
2	Yasas	54000

```sql
--question 04--
CREATE PROCEDURE bidIncrease
AS
BEGIN
DECLARE @currentBid real, @ItmNo varchar(10), @sMemNo varchar(10),
@bMemNo varchar(10), @bidCount int, @i int
SET @i = 0
SELECT @bidCount=COUNT(BN.B_MemberID)
FROM Bids_On BN,Buyer Bu, Member M
WHERE BN.B_MemberID=Bu.B_MemberID AND Bu.B_MemberID=M.MemberID AND
M.Name='Saman'
WHILE @i < @bidCount
BEGIN
SELECT @currentBid = BN.Bid_Price, @ItmNo = BN.ItemNo,
@sMemNo = BN.S_MemberID, @bMemNo =BN.B_MemberID
FROM Bids_On BN,Buyer Bu, Member M
WHERE BN.B_MemberID=Bu.B_MemberID AND
Bu.B_MemberID=M.MemberID AND M.Name='Saman'
ORDER BY Bid_Time DESC
OFFSET @i ROWS FETCH FIRST 1 ROWS ONLY
DECLARE @increaseBy real
SET @increaseBy = (@currentBid * 15) / 100
UPDATE Bids_On
SET Bid_Price = Bid_Price + @increaseBy
WHERE ItemNo = @ItmNo AND S_MemberID = @sMemNo AND
B_MemberID = @bMemNo;
SET @i = @i + 1
END
END

EXEC bidIncrease
```

Results | Messages

	ItemNo	S_MemberID	B_MemberID	Bid_Price	Bid_Time
1	01	004	001	244860.9	2023-10-28 07:45:10.000
2	01	004	002	160000	2023-10-28 15:50:10.000
3	02	005	001	227370.8	2023-09-30 02:45:10.000
4	02	005	003	140000	2023-10-01 04:45:10.000
5	04	004	001	27984.1	2023-10-19 11:45:10.000
6	04	004	002	15500	2023-10-18 08:45:10.000

6. SQL Code Scripts

SQL codes in text format for easy access

SQL codes for creating tables

```
CREATE DATABASE ONLINEAUCTION;


--creation of Bank table--
CREATE TABLE Bank(
Bank_No char(5),
BankName varchar(20) NOT NULL,
branchName varchar(20),
CONSTRAINT pk_Bank PRIMARY KEY (Bank_No)
);
--creation of Member table--
CREATE TABLE Member(
 MemberID varchar(10),
 Name varchar(10),
 Home_Address varchar(50),
 phone_No nchar(10),
 Email varchar(50),
 Password varchar(10),
 CONSTRAINT pk_Member PRIMARY KEY (MemberID),
 CONSTRAINT me_chk1 check(Email like '%@%'),
 CONSTRAINT me_chk2 check(Password like '%[0-9]%' and Password like '%[A-Z]%' and
 Password like '%[!@#$%a^&*()-_+=.,;:"`~]%' )
);
```

```
--creation of Account table--
CREATE TABLE Account(
 AccID varchar(18),
 MemberID varchar(10),
 Bank_No char(5),
 AccName varchar(10),
 AccBalance real,
 CONSTRAINT ac_pk PRIMARY KEY(MemberID,AccID),
 CONSTRAINT ac_fk1 FOREIGN KEY(MemberID) references Member(MemberID),
 CONSTRAINT ac_fk2 FOREIGN KEY(Bank_No) references Bank(Bank_No)
);

--creation of Buyer table--
CREATE TABLE Buyer(
 B_MemberID varchar(10),
 Shipping_Address varchar(50),
 CONSTRAINT buy_pk PRIMARY KEY(B_MemberID),
 CONSTRAINT buy_fk FOREIGN KEY(B_MemberID) references Member(MemberID)
);

--creation of Seller table--
CREATE TABLE Seller(
 S_MemberID varchar(10),
 Routine_No varchar(50),
 CONSTRAINT sell_pk PRIMARY KEY(S_MemberID),
 CONSTRAINT sell_fk FOREIGN KEY(S_MemberID) references Member(MemberID)
);
```

```
--creation of Category table--
CREATE TABLE Category(
 Category_ID varchar(10),
 Description varchar(50),
 CONSTRAINT cat_pk PRIMARY KEY(Category_ID)
);

--creation of Item table--
CREATE TABLE Item(
 ItemNo varchar (10),
 Item_title varchar(10),
 Description varchar(50),
 start_bidprice real,
 start_date date,
 end_date date,
 Bidding_increasement real,
 Category_ID varchar(10),
  CONSTRAINT it_pk PRIMARY KEY(ItemNo),
 CONSTRAINT it_fk FOREIGN KEY(Category_ID) references Category(Category_ID)
);

--creation of Lists table--
CREATE TABLE Lists(
 ItemNo varchar(10),
 S_MemberID varchar(10),
 CONSTRAINT li_pk PRIMARY KEY(ItemNo,S_MemberID),
 CONSTRAINT li_fk FOREIGN KEY(S_MemberID) references Seller(S_MemberID)
);
```

```
--creation of Bids_on table--
CREATE TABLE Bids_On(
 ItemNo varchar(10),
 S_MemberID varchar(10),
 B_MemberID varchar(10),
 Bid_Price real,
 Bid_Time datetime DEFAULT(getdate()),
 constraint bid_pk primary key(ItemNo,S_MemberID,B_MemberID),
 constraint bid_fk1 foreign key(ItemNo) references Item(ItemNo),
 constraint bid_fk2 foreign key(B_MemberID) references Buyer(B_MemberID),
 constraint bid_fk3 foreign key(ItemNo,S_MemberID) references Lists(ItemNo,S_MemberID)
);

--creation of Transations table--
CREATE TABLE Transactions(
 TransID varchar(10),
 Credit varchar(10),
 Debit varchar(10),
 S_MemberID varchar(10),
 B_MemberID varchar(10),
 Win_bid_price real,
 Win_bid_time datetime DEFAULT(getdate()),
 constraint tr_fk1 foreign key(B_MemberID) references Buyer(B_MemberID),
 constraint tr_fk2 foreign key(S_MemberID) references Seller(S_MemberID)
);
```

SQL codes for inserting data into tables

```
--Data insertion of Bank table--
INSERT INTO Bank VALUES ('1','Sampath Bank','colombo');
INSERT INTO Bank VALUES ('2','Bank of Ceylon','kurunegala');
INSERT INTO Bank VALUES ('3','Peoples Bank','negambo');
INSERT INTO Bank VALUES ('4','NDB bank','galle');
INSERT INTO Bank VALUES ('5','Commercial Bank','malabe');


--Data insertion of Member table--
INSERT INTO Member VALUES ('001', 'Saman','No 432, Colombo','0715224578',
'saman@gmail.com','aaA46@&$');
INSERT INTO Member VALUES ('002', 'perera','No 22, kurunegala','0725224568',
'perera@gmail.com.com','aBrg44$$');
INSERT INTO Member VALUES ('003', 'supun','No 35, negombo','0765234578',
'supun@gmail.com','mMlL33@$');
INSERT INTO Member VALUES ('004', 'namal','No 75, fort','0755233578',
'namal@gmail.com','aaAA44$$');
INSERT INTO Member VALUES ('005', 'Yasas','No 865, malabe town','0755233578',
'Yasas@gmail.com.com','gdAB55@@');


--Data insertion of Account table--
INSERT INTO Account VALUES ('11111','001','1','Saving', 100000.00);
INSERT INTO Account VALUES ('22222','002','2','Saving', 60000.00);
INSERT INTO Account VALUES ('33333','003','3','Current', 30000.00);
INSERT INTO Account VALUES ('44444','004','4','Saving', 21000.00);
```

```
INSERT INTO Account VALUES ('55555','005','5','Current', 110000.00);


--Data insertion of Buyer table--
INSERT INTO Buyer VALUES ('001','No 432, Colombo');
INSERT INTO Buyer VALUES ('002','No 22, kurunegala');
INSERT INTO Buyer VALUES ('003','No 35, negombo');


--Data insertion of Seller table--
INSERT INTO Seller VALUES ('004','No 75, fort');
INSERT INTO Seller VALUES ('005','No 865, malabe town');


--Data insertion of Category table--
INSERT INTO Category VALUES('0001', 'Electronic');
INSERT INTO Category VALUES('0002', 'vehicle');
INSERT INTO Category VALUES('0003', 'antics');
INSERT INTO Category VALUES('0004', 'dress');
INSERT INTO Category VALUES('0005', 'household');


--Data insertion of Item table--
INSERT INTO Item VALUES ('01','Laptop', 'Laptop Description', 100000.00, '2023/10/26',
'2023/11/10',101000.00,'0001');
INSERT INTO Item VALUES ('02','picture', 'picture Description', 50000.00, '2023/09/28',
'2023/10/05',52000.00, '0003');
INSERT INTO Item VALUES ('03','cab', 'Cab Description', 5000000.00, '2023/10/06',
'2023/11/20',5025000.00, '0002');
```

```
INSERT INTO Item VALUES ('04','Table', 'Table Description', 10000.00, '2023/10/17',
'2023/10/30',11000.00, '0005');
INSERT INTO Item VALUES ('05','T-shirt', 'T-shirt Description', 4000.00, '2023/10/21',
'2023/10/28',4500.00, '0004');

--Data insertion of Lists table--
INSERT INTO Lists VALUES ('01', '004');
INSERT INTO Lists VALUES ('02', '005');
INSERT INTO Lists VALUES ('03', '004');
INSERT INTO Lists VALUES ('04', '004');
INSERT INTO Lists VALUES ('05', '005');

--Data insertion of Bids_on table--
INSERT INTO Bids_On VALUES ('01', '004', '001', 140000.00, '2023/10/28 07:45:10');
INSERT INTO Bids_On VALUES ('01', '004', '002', 160000.00, '2023/10/28 15:50:10');
INSERT INTO Bids_On VALUES ('02', '005', '001', 130000.00, '2023/09/30 02:45:10');
INSERT INTO Bids_On VALUES ('02', '005', '003', 140000.00, '2023/10/01 04:45:10');
INSERT INTO Bids_On VALUES ('04', '004', '002', 15500.00, '2023/10/18 08:45:10');
INSERT INTO Bids_On VALUES ('04', '004', '001', 16000.00, '2023/10/19 11:45:10');

--Data insertion of Transation table--
INSERT INTO Transactions VALUES ('T1','222222', '444444',
'004', '002', 160000.00, '2023/10/28 15:50:10');
INSERT INTO Transactions VALUES ('T2','333333', '555555',
'005', '003', 140000.00, '2023/10/01 04:45:10');
INSERT INTO Transactions VALUES ('T3','111111', '444444',
'004', '001', 16000.00, '2023/10/19 11:45:10');
```

SQL codes for triggers used in tables

Trigger 1

```
create Trigger Account_Balance_Check
ON Account
AFTER INSERT
AS
BEGIN
 IF EXISTS(SELECT 1 FROM inserted WHERE AccBalance<=25000.00)
BEGIN
 RAISERROR ('Bank Balance cannot be less than 25000 Rupees',16,1);
 ROLLBACK
 RETURN
END
END
```

Trigger 2

```
CREATE TRIGGER CheckTransaction
ON Transactions
FOR INSERT
AS
BEGIN
 DECLARE @bidPrice real, @bAccNo varchar(18), @sAccNo varchar(18)
 SELECT @bidPrice=Win_bid_price, @bAccNo=Credit, @sAccNo=Debit
 FROM Transactions;
 DECLARE @bAccBal real, @sAccBal real
```

```
SELECT @bAccBal=AccBalance
FROM Account
WHERE AccID = @bAccNo;
SELECT @sAccBal=AccBalance
FROM Account
WHERE AccID = @sAccNo;
IF @bAccBal < @bidPrice
ROLLBACK TRANSACTION
ELSE

 BEGIN
  UPDATE Account
  SET AccBalance = @bAccBal - @bidPrice
  WHERE AccID = @bAccNo;
  UPDATE Account
  SET AccBalance = @sAccBal + @bidPrice
  WHERE AccID = @sAccNo;
END
END
```

SQL codes for views made

View 1

```
CREATE VIEW [Member Details]
AS
SELECT
M.MemberID,M.Name,M.Email,M.Home_Address,M.phone_No,B.BankName,B.branchName,A.AccID
AS [Account No],A.AccBalance AS [Account Blance]
```

```
FROM Account A, Member M, Bank B

WHERE M.MemberID=A.MemberID AND A.Bank_No=B.Bank_No;


Select * from [Member Details]
```

View 2

```
CREATE VIEW showItem

AS

SELECT mem.Name, itm.Item_title, itm.Description AS [Item Description], cat.Description AS

[Category]

FROM Item itm, Lists lst, Member mem, Category cat

WHERE itm.ItemNo = lst.ItemNo AND lst.S_MemberID = mem.MemberID AND itm.Category_ID =
cat.Category_ID;


select * from showItem
```

SQL codes for indexes

```
--implement an index on Accbalance column in Account table--


CREATE INDEX INX_Acc_Balance

ON Account (AccBalance);


--implement an index on Bid_Price column in Bids_on--


CREATE INDEX INX_Bid_on

ON Bids_On (Bid_Price);
```

SQL codes for stored procedures (Q1 -Q4)

Question 1

```
CREATE PROCEDURE Member_Bank
AS
BEGIN
SELECT M.Name,M.Home_Address
FROM Member M, Account A, Bank B
WHERE M.MemberID=A.MemberID AND A.Bank_No=B.Bank_No AND B.BankName= 'Sampath Bank'
END
EXEC Member_Bank
```

Question 2

```
CREATE PROCEDURE Member_Bid
AS
BEGIN
SELECT M.Name,M.Email,BN.Bid_Price
FROM Member M, Bids_On BN, Buyer Bu, Lists L, Item I
WHERE BN.B_MemberID=Bu.B_MemberID AND Bu.B_MemberID = M.MemberID AND
BN.ItemNo=L.ItemNo AND L.ItemNo = I.ItemNo AND I.Item_title='Laptop'
ORDER BY BN.Bid_Price
END

EXEC Member_Bid
```

Question 3

```
CREATE PROCEDURE Seller_Sum
AS
BEGIN
SELECT M.Name,SUM(I.start_bidprice)
FROM Item I, Lists L, Seller S, Member M
WHERE I.ItemNo=L.ItemNo AND S.S_MemberID=L.S_MemberID AND
S.S_MemberID=M.MemberID
GROUP BY M.Name
HAVING SUM(I.start_bidprice)>30000
END

EXEC Seller_Sum
```

Question 4

```
CREATE PROCEDURE bidIncrease
AS
BEGIN
DECLARE @currentBid real, @ItmNo varchar(10), @sMemNo varchar(10),
@bMemNo varchar(10), @bidCount int, @i int
SET @i = 0
SELECT @bidCount=COUNT(BN.B_MemberID)
FROM Bids_On BN,Buyer Bu, Member M
WHERE BN.B_MemberID=Bu.B_MemberID AND Bu.B_MemberID=M.MemberID AND
M.Name='Saman'
WHILE @i < @bidCount
BEGIN
```

```
SELECT @currentBid = BN.Bid_Price, @ItmNo = BN.ItemNo,
@sMemNo = BN.S_MemberID, @bMemNo =BN.B_MemberID
FROM Bids_On BN,Buyer Bu, Member M
WHERE BN.B_MemberID=Bu.B_MemberID AND
Bu.B_MemberID=M.MemberID AND M.Name='Saman'
ORDER BY Bid_Time DESC
OFFSET @i ROWS FETCH FIRST 1 ROWS ONLY
DECLARE @increaseBy real
SET @increaseBy = (@currentBid * 15) / 100
UPDATE Bids_On
SET Bid_Price = Bid_Price + @increaseBy
WHERE ItemNo = @ItmNo AND S_MemberID = @sMemNo AND
B_MemberID = @bMemNo;
SET @i = @i + 1
END
END

EXEC bidIncrease
```

7. Common Database Vulnerabilities

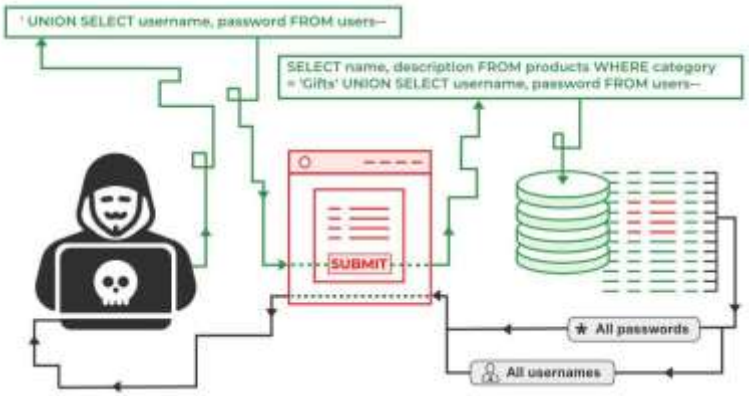
Databases are most commonly targeted by hackers or attackers, because they contain sensitive information. When the information is breached and falls into the hands of the attacker, the attacker can make good amount of money by threatening the database owner as it contains sensitive information about the organization.

Attackers mostly target databases which contains low security measures. When the designers of the database fails to create the system with perfect security, it becomes prone to various attacks and assaults. That makes the database vulnerable to most common attacks ,Some of the main attacks which can happen when the database is vulnerable are listed below.

SQL injection

Structured Query Language was created specifically for managing and modifying relational databases. The user can Create, edit , read and delete data using sql's standard interface for databases. SQL injection or SQLi is a malicious technique where a SQL query is manipulated by inserting malicious SQL codes by the attacker. The database of the application then runs this code ,it enables unauthorized data access, modification or deletion. Web applications that improperly check and filter user inputs are mainly prone to attacks like SQL injection.

Targeting these databases, SQL injection (SQLi) is a type of cybers attack that use specifically crafted SQL statements to trick the systems into performing unwanted and undesirable actions.



Techniques:

Few techniques used in SQL injection are mentioned below,

1.Malicious Inputs:

In this method, the attacker insert the SQL code into input fields where user input is not sanitized or checked properly (Such as login forms or search bars).

2.Union based attacks:

In Union based attacks, the attacker may extract sensitive information by joining the results from many database tables using UNION operators.

3.Time-based blind attacks:

- Time-based blind attacks are executed by taking advantage of database response time lags, which makes detection challenging.
- Attackers utilize reaction time lags to deduce information.

4.Tautology-Based attacks:

This method is done by injecting 'OR'1'='1, the attackers always make the query true, bypassing the authentication checking and gaining unauthorized access.

Impact:

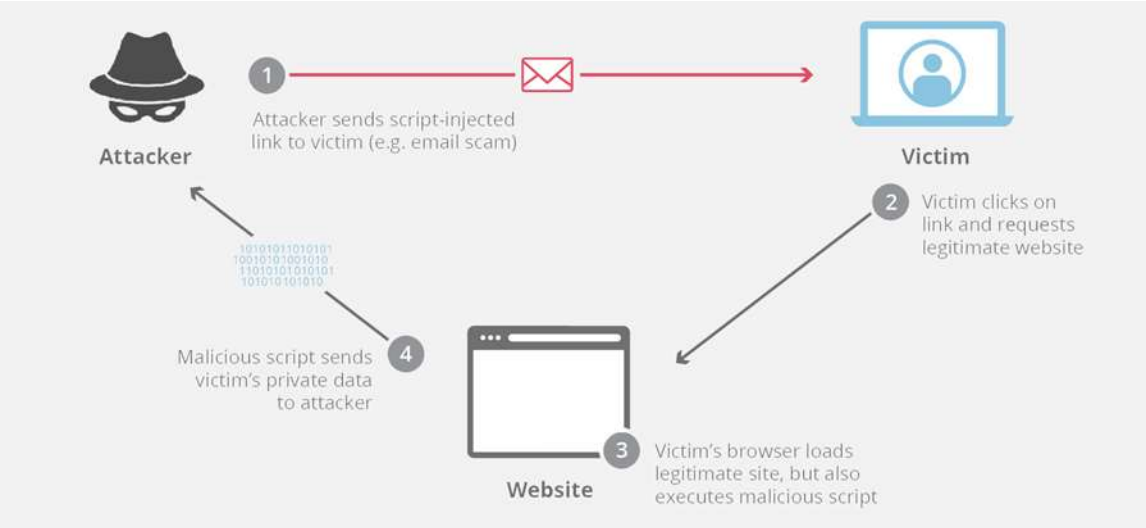
- Access, Modification and Deletion of data - The attacker can read, update and erase sensitive data.
- Authentication Bypass - gaining access through compromising security.
- Data disclosure - leading to privacy breaches

Cross-Site Scripting (XSS)

A common web security flaw known as cross-site scripting (XSS) happens when an attacker inserts harmful scripts into web content that is then served to users. These scripts run within the context of a user's browser, giving attackers the ability to steal confidential data, carry out tasks on the user's behalf, or alter websites.

Although XSS primarily impacts the client-side, it can have serious implications for the connected databases specially when sensitive data is involved.

potentially compromising the underlying database while also having an impact on the presentation layer. A thorough explanation of Cross-Site Scripting in relation to database vulnerabilities is given below,



Techniques:

1.Stored XSS:

When visitors browse a specific page, malicious scripts are saved on a server and sent to the users.

2.Reflected XSS:

Often through email or phishing attempts, malicious scripts are injected in URLs and sent from a web server to a user's browser.

3.DOM-based XSS:

Scripts change the behavior and structure of a web page by modifying the Document Object Model (DOM) in the user's browser

Impact:

- Data theft - Attackers may take sensitive user data, login passwords, or session cookies.
- Session Hijacking - Attackers could assume the identity of users, act in their place, or modify account settings.
- Malware Distribution - Users' computers can become infected by malicious programs that drive them to websites hosting malware.
- Defacement - Attackers could alter a website's appearance, harming its reputation.

8. Mitigations and Countermeasures

Preventing SQL injection

Parameterized Queries

- Use parameterized queries to keep data distinct from SQL statements and avoid direct injection by using prepared statements and parameterized queries.

Keeping the database up to date with recent fixes

- Attackers won't be able to exploit known vulnerabilities in earlier versions.

Input Validation

- Validate and sanitize user inputs to make sure they match expected performance.

Web application Firewall

- Implement a WAF to block and detect SQL injections in real-time.

Don't rely on dynamic SQL

- Prepared statements and parameterized queries are safer alternatives.

Avoid leaving sensitive information in plaintext.

- Keep confidential data on the database encrypted.
- The encrypted hashes should be salted

Stored Procedures

- Increase security by encapsulating SQL functionality in stored procedures.

Preventing Cross site scripting (XSS)

Enter Validations and Proper sanitization

- To exclude or neutralize potentially dangerous characters from user input filter and sanitize it.

Encode user generated Material

- To stop the injected scripts from running, encode user-generated material.Use the necessary encoding functions related to the situation.
- Use HTML encoding when inserting data into HTML elements.

CSP (Content security policy)

- Using CSP headers to limit the sources of executable code by indicating where resources such as scripts can be loaded.

Secure Session Management

- To prohibit JavaScript from accessing cookies and to ensure that they are only transmitted over HTTPS, use the HttpOnly and Secure options.

Educate developers

- Make the developers aware of XSS risks and educate them with coding practices to prevent vulnerabilities.

Browse Security features

- Encourage users to keep their browsers up-to-date, as modern browsers include security features to mitigate XSS attacks.

9. Conclusion

This report discusses a database system used in an online auction system where sellers list items and buyers bid. The highest bid price wins, and a transaction occurs between the buyer and seller. The bank accounts are updated after the transaction. The report presents an ER diagram, entities, attributes, and relationships, and a logical model is derived, normalized to 3NF form using schema refinement techniques.

The assignment includes images related to SQL queries, codes for creating tables, inserting data, applying constraints, and optimizing triggers. Procedures questions and answers are provided as codes, and users in the system are identified and their views are published. Triggers are applied to the database with indexes for optimization.

The final part listed two common cybercriminal attacks, their attack chains, techniques, and impact on databases and user privacy, while also introducing mitigations to prevent and reduce these types of attacks.