

**CS6106**

**DATABASE MANAGEMENT SYSTEM**

**PROJECT**

**REPORT**

**BLOOD DONATION**

**MANAGEMENT SYSTEM**

JANANI A  
2022503502  
TEJESSHREE S  
2022503524

## **Scenario for dB design:**

- Blood Bank and Hospital are planning to implement a blood donation management system that supports both users' requirements and gives connectivity.
- Stores Donor's details, Patient's, Blood bank and Hospital details, and Nurse and Blood bank manager details.
- Keeps track of the availability of blood units.
- Keeps track of the donor-to-patient transaction path
- Communication logs

## **Tools Study**

**Tool used:** Microsoft SQL Server

### **Backend:**

#### **1. ASP.NET Core Web API:**

Framework: ASP.NET Core

Purpose: Provides the server-side logic and functionality for your application.

Controllers and Classes: Implemented to handle the logic for each table in your database (e.g., handling CRUD operations).

Operations: Includes GET, POST, PUT, and DELETE operations to manage data.

Database Connection: Uses Entity Framework Core to interact with Microsoft SQL Server.

#### **2. Microsoft SQL Server:**

Database: blooddon1

Tables: Contains structured data relevant to your application (e.g., donor information, blood types, donation records).

Function: Stores and retrieves data as requested by the ASP.NET Core Web API.

## **Frontend**

### **3. React.js:**

Library: React.js, a popular JavaScript library for building user interfaces.

Purpose: Provides the client-side logic and the user interface.

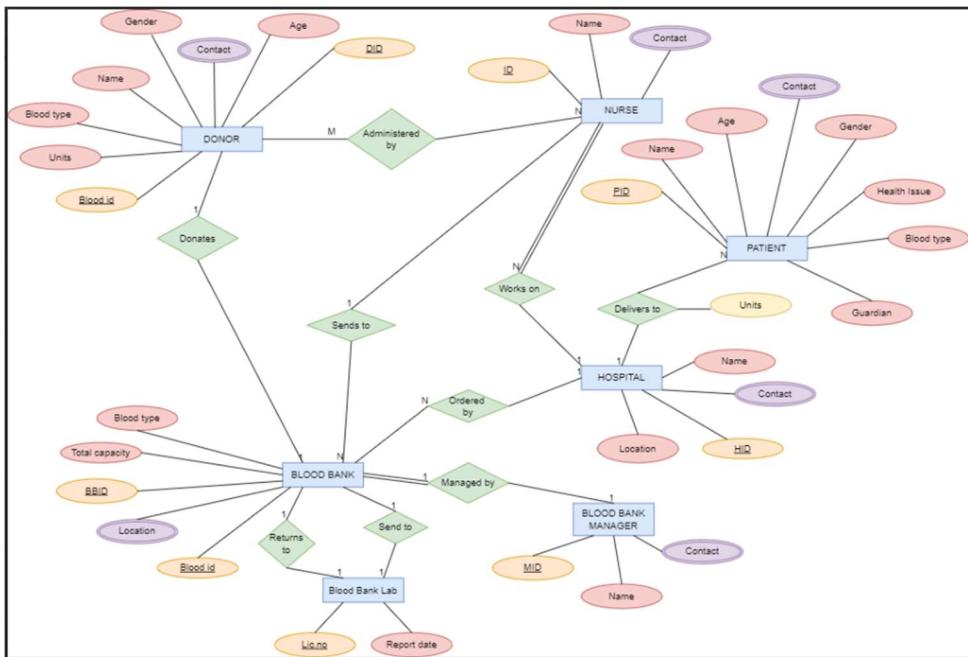
Components: Built using React components to create a dynamic and responsive web application.

Communication with Backend: Utilizes asynchronous HTTP requests (such as fetch or Axios) to interact with the ASP.NET Core Web API, performing CRUD operations.

## **Reason for choosing SQL Server**

- SQL Server is a Relational Database Management software that helps structure our database.
- Ensures integrity and reliability with constraints
- Supports scalability, user-friendly interface
- Comprehensive security features - encryption, access controls, etc.
- Allows for integration with other applications to enable workflow automation (SSIS)
- Integration with communication systems like SMS, and email enables automated notifications
- Integration with EHR (Electronic Health Record)
- Backup and restoring capabilities
- SSRS (SQL Server Reporting Services), SSAS (Analysis services).

## ER Diagram



## Attributes

- Donor
- Nurse
- Blood Bank
- Blood Bank Lab
- Blood Bank Manager
- Patient
- Hospital

## Relationship

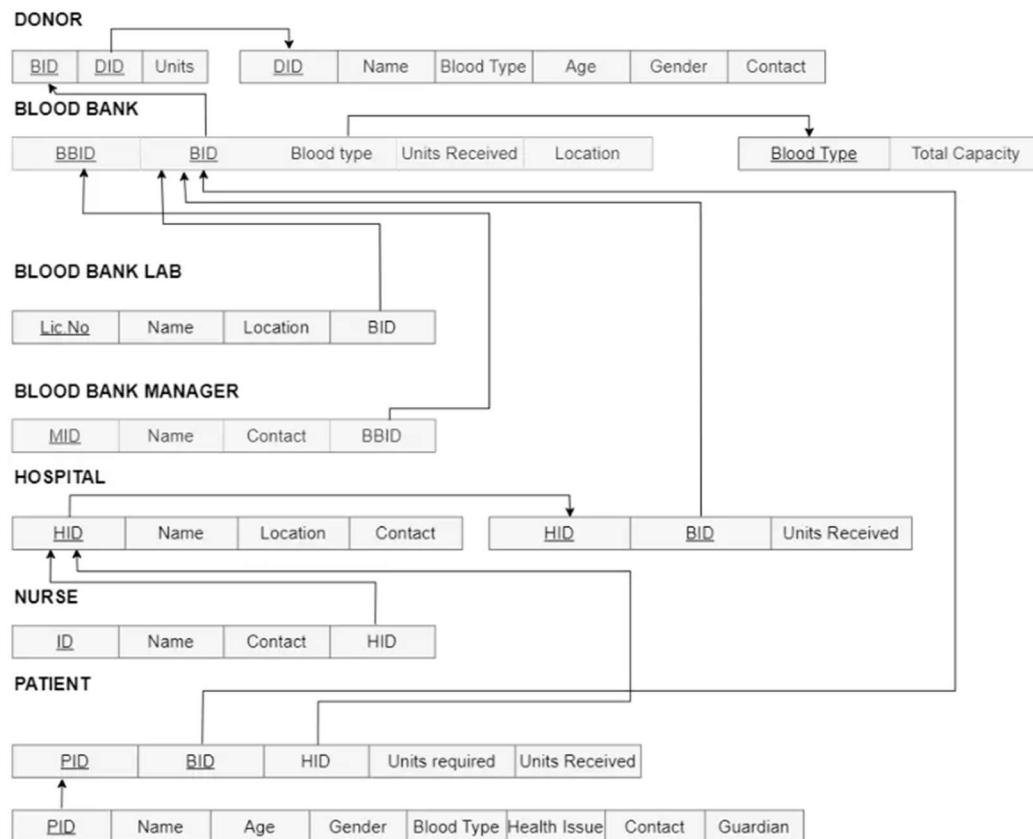
- Donor administered by Nurse
- Donor donates to Blood Bank
- Nurse sends Blood to Blood Bank
- Blood Bank sends the donated blood to Blood Bank Lab
- Blood Bank managed by Blood Bank Manager
- Hospital orders Blood from Blood Bank
- Hospital delivers Blood to the Patient
- Nurse works for the Hospital

## Primary Keys

- BID- Blood ID
- BBID- Blood Bank ID
- HID- Hospital ID
- DID- Donor ID
- MID- Manager ID
- PID- Patient ID

## Relational Schema

The Relational Schema is refined to **3NF**.



## TABLES CREATION

The tables are created using SQL queries with NOT NULL and PRIMARY KEY constraints.

At first, the database named “**BLOODDON1**” is created. Under the database, tables for the donor, blood bank, blood bank lab, blood bank manager, Nurse, Hospital, and patients are created.

### DDL Commands

The screenshot shows four separate SQL queries in the Object Explorer of SQL Server Management Studio:

- SQLQuery1.sql - L...8CGC7R\emaxr (51)\***: Contains the creation of three tables: DONOR, DONOR1, and bb1. The DONOR table has columns BID (INT), DID (int), and Units (int) with a primary key constraint on BID and DID. The DONOR1 table has columns DID (int), Name (varchar(50)), Age (int), Gender (char), Contact (int), and Bloodtype (varchar(5)) with a primary key constraint on DID. The bb1 table has columns BBID (int), BID (int), and Location1 (varchar(40)) with a primary key constraint on BBID and BID.
- SQLQuery2.sql - L...8CGC7R\emaxr (54)**: Contains the creation of three tables: bb2, bbl, and bbm. The bb2 table has columns Bloodtype (varchar(5)) and Totalcap (int) with a primary key constraint on Bloodtype. The bbl table has columns Lno (int), Name (varchar(50)), BID (int), and Location1 (varchar(50)) with a primary key constraint on Lno. The bbm table has columns MId (int), Name (varchar(50)), Contact (int), BBID (int) with a primary key constraint on MId.
- SQLQuery1.sql - L...8CGC7R\emaxr (51)\***: Contains the creation of four tables: hospital, hospital1, nurse, and patient. The hospital table has columns HID (int), Name (varchar(50)), Contact (int), and Locationn (varchar(50)) with a primary key constraint on HID. The hospital1 table has columns HID (int), BId (int), and Unitsrec (int) with a primary key constraint on HID and BId. The nurse table has columns NId (int), HID (int), Name (varchar(50)), and Contact (int) with a primary key constraint on NId. The patient table has columns PId (int), Name (varchar(50)), Age (int), Gender (char), Contact (int), Bloodtype (varchar(5)), Health\_issue (varchar(50)), and Guardian (varchar(50)) with a primary key constraint on PId.
- SQLQuery2.sql - L...8CGC7R\emaxr (54)**: Contains the creation of one table: patient1. It has columns PId (int), BId (int), HID (int), Unitsreq (int), and Unitsrec (int) with a primary key constraint on PId and BId.

The other constraints like the FOREIGN key and UNIQUE key constraints are added using the ALTER command.

```
SQLQuery1.sql - L...8CGC7R\emaxr (51)*  SQLQuery2.sql - L...8CGC7R\emaxr (51)
exec sp_help donor;

alter table bb2
add constraint U_BB2_BT unique(Bloodtype);

alter table DONOR
add constraint U_Donor_BID unique(BID);

alter table DONOR1
add constraint U_Donor1_DID unique(DID);

alter table HOSPITAL
add constraint U_Hospital_HID unique(HID);

alter table Patient1
add constraint U_Patient1_PID unique(PID);

alter table bb1
add constraint U_bb1_BID unique(BID);

alter table bb1
add constraint U_bb1_BBID unique(BBID);

alter table HOSPITAL
add constraint U_hhospital_HID unique(HID);

alter table bb1
add constraint bb1_donor_bid_fk foreign key (BID) references DONOR(BID);

alter table bbl
add constraint bbl_bb1_bid_fk foreign key (BID) references bb1(BID);

alter table bbm
add constraint bbm_bb1_bbid_fk foreign key (BBID) references bb1(BBID);

alter table hospital1
add constraint h1_bb1_bid_fk foreign key (BID) references bb1(BID);

alter table nurse
add constraint n_h_hid_fk foreign key (hID) references Hospital(HID);

alter table patient1
add constraint p1_bb1_bid_fk foreign key (BID) references bb1(BID);

alter table patient1
add constraint p1_h_hid_fk foreign key (HID) references hospital(HID);

alter table donor
add constraint d_d1_did_fk foreign key (DID) references donor1(DID);

alter table bb1
add constraint bb1_bb2_bt_fk foreign key (Bloodtype) references bb2(Bloodtype);
```

```
SQLQuery1.sql - L...8CGC7R\emaxr (51)*  SQLQuery2.sql - L...8CGC7R\emaxr (54)
add constraint n1_doi_dio_fk foreign key (DOI) references DOI(DOI);

alter table nurse
add constraint n_h_hid_fk foreign key (hID) references Hospital(HID);

alter table patient1
add constraint p1_bb1_bid_fk foreign key (BID) references bb1(BID);

alter table patient1
add constraint p1_h_hid_fk foreign key (HID) references hospital(HID);

alter table donor
add constraint d_d1_did_fk foreign key (DID) references donor1(DID);

alter table bb1
add constraint bb1_bb2_bt_fk foreign key (Bloodtype) references bb2(Bloodtype);

alter table hospital
add constraint h1_h1_hid_fk foreign key (HID) references hospital1(HID);

alter table patient
add constraint p_p1_pid_fk foreign key (PID) references patient1(PID);
```

## OUTPUT

### DONOR

### DONOR1

```
exec sp_help donor;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>DONOR</b>	dbo	user table	2024-04-26 22:34:52.207

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	ID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	CID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
3	Units	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	donor_pk	clustered, unique, primary key located on PRIMARY	ID, CID
2	U_Donor_BD	nonsclustered, unique, unique key located on PRIMA	ID

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

1	FOREIGN KEY	d, d1, d2, &	No Action	No Action	Enabled	Is For Replication	DO
2	PRIMARY KE...	donor_pk	(n/a)	(n/a)	(n/a)	(n/a)	BLOODTYPE
3	UNIQUE (non-clu...	U_Donor_BD	(n/a)	(n/a)	(n/a)	(n/a)	ID

Table is referenced by foreign key  
1 blooddon1.dbo.bb1.bb1\_donor\_fk

Query executed successfully.

```
exec sp_help donor1;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>DONOR1</b>	dbo	user table	2024-04-26 22:34:54.130

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	ID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Age	int	no	4	10	0	no	(n/a)	(n/a)	NULL
3	Gender	char	no	1	10	0	no	(n/a)	(n/a)	NULL
4	Contact	varchar	no	4	10	0	no	(n/a)	(n/a)	NULL
5	Bloodtype	varchar	no	5	10	0	no	(n/a)	(n/a)	NULL

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	donor_pk	clustered, unique, primary key located on PRIMARY	ID
2	U_Donor1_BD	nonsclustered, unique, unique key located on PRIMA	ID

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

1	PRIMARY KEY (clustered)	donor_pk	(n/a)	(n/a)	(n/a)	(n/a)	DO
2	UNIQUE (non-clustered)	U_Donor1_BD	(n/a)	(n/a)	(n/a)	(n/a)	DO

Table is referenced by foreign key  
1 blooddon1.dbo.DONOR1.bb1\_donor\_fk

Query executed successfully.

### BB1 (BloodBank 1)

### BB2 (Blood Bank 2)

```
exec sp_help bb1;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>BB1</b>	dbo	user table	2024-04-26 22:34:44.70

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	ID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Name	varchar	no	40	10	0	no	(n/a)	(n/a)	NULL
3	Location1	varchar	no	40	10	0	no	(n/a)	(n/a)	SQL_Latin1_General_CI_AS
4	Bloodtype	varchar	no	5	10	0	no	(n/a)	(n/a)	SQL_Latin1_General_CI_AS
5	Unbreak	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	bb1_pk	clustered, unique, primary key located on PRIMARY	ID, BB1
2	U_BB1_BD	nonsclustered, unique, unique key located on PRIMA	BB1
3	U_BB1_BT	nonsclustered, unique, unique key located on PRIMA	BB1

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

1	FOREIGN KEY	bb1_bb2_bb2_fk	No Action	No Action	Enabled	Is For Replication	BB2(Bloodtype)
2	FOREIGN KEY	bb1_donor_bb1_fk	No Action	No Action	Enabled	Is For Replication	BB1(DONOR)
3	PRIMARY KE...	bb1_pk	(n/a)	(n/a)	(n/a)	(n/a)	BB1(BB1)
4	UNIQUE (non-clu...	U_BB1_BD	(n/a)	(n/a)	(n/a)	(n/a)	BB1
5	UNIQUE (non-clu...	U_BB1_BT	(n/a)	(n/a)	(n/a)	(n/a)	BB1

Table is referenced by foreign key  
1 blooddon1.dbo.bb1.bb1\_bb2\_bb2\_fk

Query executed successfully.

```
exec sp_help bb2;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>BB2</b>	dbo	user table	2024-04-26 22:36:05.800

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	Mid	varchar	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Transload	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	bb2_pk	clustered, unique, primary key located on PRIMARY	Mid
2	U_BB2_BT	nonsclustered, unique, unique key located on PRIMA	Bloodtype

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

1	PRIMARY KEY (clustered)	bb2_pk	(n/a)	(n/a)	(n/a)	(n/a)	Bloodtype
2	UNIQUE (non-clustered)	U_BB2_BT	(n/a)	(n/a)	(n/a)	(n/a)	Bloodtype

Table is referenced by foreign key  
1 blooddon1.dbo.bb1.bb1\_bb2\_bt\_fk

Query executed successfully.

### BBL (Blood Bank Lab)

### BBM (Blood Bank manager)

```
exec sp_help bbl;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>bbl</b>	dbo	user table	2024-04-26 22:37:29.253

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	Lno	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Name	varchar	no	50	10	0	no	(n/a)	(n/a)	SQL_Latin1_General_CI_AS
3	BD	int	no	4	10	0	no	(n/a)	(n/a)	NULL
4	Location1	varchar	no	50	10	0	yes	(n/a)	(n/a)	SQL_Latin1_General_CI_AS

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	bbl_pk	clustered, unique, primary key located on PRIMARY	Lno				
2	FOREIGN KEY	bbl_bb1_bb1_fk	No Action	No Action	Enabled	Is For Replication	BB1(BD)
3	PRIMARY KEY (clustered)	bbl_pk	(n/a)	(n/a)	(n/a)	(n/a)	Lno

Query executed successfully.

```
exec sp_help bbm;
```

Results Messages

Name	Owner	Type	Created,LastUpdated
<b>bbm</b>	dbo	user table	2024-04-26 22:38:32.590

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullSource Collation

1	Mid	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Name	varchar	no	50	10	0	no	(n/a)	(n/a)	SQL_Latin1_General_CI_AS
3	Contact	int	no	4	10	0	no	(n/a)	(n/a)	NULL
4	BBID	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity  
1 No identity column defined. Seed Increment Not For Replication  
2 RowGuidCol  
3 No rowguidcol column defined.

Data\_located\_on\_Filegroup PRIMARY

index\_name index\_description index\_keys

1	bbm_pk	clustered, unique, primary key located on PRIMARY	Mid				
2	FOREIGN KEY	bbm_bb1_bb1_fk	No Action	No Action	Enabled	Is For Replication	BB1(Mid)
3	PRIMARY KEY (clustered)	bbm_pk	(n/a)	(n/a)	(n/a)	(n/a)	Mid

Query executed successfully.

# HOSPITAL

# HOSPITAL1

```
exec sp_help hospital;
```

Results (0 rows)

Messages

Name	Owner	Type	Created_datetime
hospital	dbo	user table	2024-04-26 22:41:13.527

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullInSource Collation

HId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Name	varchar	no	50			no			SQL\_Latin1\_General\_CI\_AS
Contact	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Location	varchar	no	50			yes	no		SQL\_Latin1\_General\_CI\_AS

Identity

| No identity column defined. | Seed | Increment | Not For Replication |

RowGuidCol

| No rowguidcol column defined. |

Data\_located\_on\_Negroup

| PRIMARY |

index\_name index\_description index\_keys

| Hospital\_pk | clustered, unique primary key located on PRIMARY | HId |
| U\_Hospital\_HId | nonclustered, unique, unique key located on PRIMA | HId |

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

FOREIGN KEY	n\_h1\_hId\_fk	No Action	No Action	Enabled	Is\_For\_Replication	HId
3	PRIMARY KEY (clustered) hospital\_pk	(n/a)	(n/a)	(n/a)		REFERENCES bloodon1.dbo.hospital (HId)
4	UNIQUE (non-clustered) U\_Hospital\_HId	(n/a)	(n/a)	(n/a)		HId

Table is referenced by foreign key

| bloodon1.dbo.nurse n\_h1\_hId\_fk |
| bloodon1.dbo.patient p1\_h1\_hId\_fk |

Query executed successfully.

# NURSE

# PATIENT

```
exec sp_help nurse;
```

Results (0 rows)

Messages

Name	Owner	Type	Created_datetime
nurse	dbo	user table	2024-04-26 22:44:23.130

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullInSource Collation

NId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
no	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Name	varchar	no	50			yes	no		SQL\_Latin1\_General\_CI\_AS
Contact	int	no	4	10	0	no	(n/a)	(n/a)	NUL

Identity

| No identity column defined. | Seed | Increment | Not For Replication |

RowGuidCol

| No rowguidcol column defined. |

Data\_located\_on\_Negroup

| PRIMARY |

index\_name index\_description index\_keys

| nurse\_pk | clustered, unique, primary key located on PRIMARY | NId |

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

| FOREIGN KEY | n\_h1\_hId\_fk | No Action | No Action | Enabled | Is\_For\_Replication | HId |
| 3 | PRIMARY KEY (clustered) nurse\_pk | (n/a) | (n/a) | (n/a) |  | REFERENCES bloodon1.dbo.hospital (HId) |

exec sp\_help patient;

Results (0 rows)

Messages

Name	Owner	Type	Created_datetime
patient	dbo	user table	2024-04-26 22:46:31.770

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullInSource Collation

PId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
HId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
HId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Unireq	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Unireq	int	no	4	10	0	no	(n/a)	(n/a)	NUL

Identity

| No identity column defined. | Seed | Increment | Not For Replication |

RowGuidCol

| No rowguidcol column defined. |

Data\_located\_on\_Negroup

| PRIMARY |

index\_name index\_description index\_keys

| patient\_pk | clustered, unique, primary key located on PRIMARY | PId |
| U\_Patient\_PID | nonclustered, unique, unique key located on PRIMA | PId |

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

FOREIGN KEY	p1\_bb1\_hId\_fk	No Action	No Action	Enabled	Is\_For\_Replication	BId
3	FOREIGN KEY p1\_h1\_hId\_fk	No Action	No Action	Enabled	Is\_For\_Replication	HId
4	PRIMARY KE... patient\_pk	(n/a)	(n/a)	(n/a)		REFERENCES bloodon1.dbo.hospital..
5	UNIQUE (non... U\_Patient\_PID	(n/a)	(n/a)	(n/a)		PId

Table is referenced by foreign key

| bloodon1.dbo.hospital h1\_hId\_fk |
| bloodon1.dbo.bb1 bb1\_hId\_fk |

Query executed successfully.

# PATIENT1

```
exec sp_help patient1;
```

Results (0 rows)

Messages

Name	Owner	Type	Created_datetime
patient1	dbo	user table	2024-04-26 22:47:36.703

Column\_name Type Computed Length Prec Scale Nullable TrimTrailingBlanks FixedLenNullInSource Collation

PId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
HId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
HId	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Unireq	int	no	4	10	0	no	(n/a)	(n/a)	NUL
Unireq	int	no	4	10	0	no	(n/a)	(n/a)	NUL

Identity

| No identity column defined. | Seed | Increment | Not For Replication |

RowGuidCol

| No rowguidcol column defined. |

Data\_located\_on\_Negroup

| PRIMARY |

index\_name index\_description index\_keys

| patient1\_pk | clustered, unique, primary key located on PRIMARY | PId |
| U\_Patient1\_PID | nonclustered, unique, unique key located on PRIMA | PId |

constraint\_type constraint\_name delete\_action update\_action status\_enabled status\_for\_replication constraint\_keys

FOREIGN KEY	p1\_bb1\_hId\_fk	No Action	No Action	Enabled	Is\_For\_Replication	BId
3	FOREIGN KEY p1\_h1\_hId\_fk	No Action	No Action	Enabled	Is\_For\_Replication	HId
4	PRIMARY KE... patient1\_pk	(n/a)	(n/a)	(n/a)		REFERENCES bloodon1.dbo.bb1 (BId)
5	UNIQUE (non... U\_Patient1\_PID	(n/a)	(n/a)	(n/a)		PId

Table is referenced by foreign key

| bloodon1.dbo.hospital h1\_hId\_fk |
| bloodon1.dbo.bb1 bb1\_hId\_fk |

## DML Commands

The insertion on tables is done using the multiple row insert command. The primary key column entries are inserted followed by other column entries.

The insertion queries implemented in the SQL server and their Output are listed below.

### **DONOR1**

```
insert into donor1 (DID, Name, Age, Gender, Contact, Bloodtype) values
(101, 'Mark', 40, 'M', 987654321, 'O+'),
(102, 'Rose', 38, 'F', 9876555551, 'A+'),
(103, 'Jenny', 55, 'F', 987654221, 'B+'),
(104, 'Park', 24, 'M', 987654311, 'B+'),
(105, 'Mary', 34, 'F', 777654321, 'A-');
```

Results

DID	Name	Age	Gender	Contact	Bloodtype
101	Mark	40	M	987654321	O+
102	Rose	38	F	9876555551	A+
103	Jenny	55	F	987654221	B+
104	Park	24	M	987654311	B+
105	Mary	34	F	777654321	A-

### **DONOR**

```
insert into donor(BID, DID, Units) values
(901, 101, 5),
(902, 102, 4),
(903, 102, 3),
(904, 105, 2);

select * from donor;
```

Results

BID	DID	Units
901	101	5
902	102	4
903	102	3
904	105	2

## BB2(Blood bank 2)

```
insert into bb2 (Bloodtype, Totalcap) values
('O+', 0),
('A+', 4),
('B+', 0),
('O-', 0),
('A-', 0),
('B-', 0),
('AB+', 0),
('AB-', 0);

select * from bb2;
```

100 %

Results Messages

	Bloodtype	Totalcap
1	A+	4
2	A-	0
3	AB+	0
4	AB-	0
5	B+	0
6	B-	0
7	O+	0
8	O-	0

## BB1(Blood bank 1)

```
insert into bb1 (BBID, BID, Location1, Bloodtype, Unitsrec) values
(701, 901, 'Chennai', 'O+', 2),
(702, 904, 'Bangalore', 'A-', 1),
(703, 902, 'Chennai', 'A+', 2),
(704, 903, 'Pune', 'A+', 2);

select * from bb1;
```

100 %

Results Messages

	BBID	BID	Location1	Bloodtype	Unitsrec
1	701	901	Chennai	O+	2
2	702	904	Bangalore	A-	1
3	703	902	Chennai	A+	2
4	704	903	Pune	A+	2

## BLOOD BANK LAB (BBL)

```
insert into bbl (Lno, BID, Location1) values
(301, 901, 'Chennai'),
(302, 901, 'Salem'),
(303, 902, 'Coimbatore'),
(304, 903, 'Hosur');

select * from bbl;
```

The screenshot shows the SQL Server Management Studio interface. In the top pane, there is a code editor window containing the provided SQL script. Below it is a results pane titled 'Results' which displays a table with four rows of data. The columns are labeled 'Lno', 'BID', and 'Location1'. The data is as follows:

	Lno	BID	Location1
1	301	901	Chennai
2	302	901	Salem
3	303	902	Coimbatore
4	304	903	Hosur

## RETRIEVAL OF DATUM

A few retrieval queries like **select**, **aggregate**, **group by** are performed to check the database is created and implemented without any error.

### Query

List the details of donor whose blood type is B+.

### Command and Output

The screenshot shows the SQL Server Management Studio interface. In the top pane, there is a code editor window containing the provided SQL script. Below it is a results pane titled 'Results' which displays a table with two rows of data. The columns are labeled 'DID', 'Name', 'Age', 'Gender', 'Contact', and 'Bloodtype'. The data is as follows:

	DID	Name	Age	Gender	Contact	Bloodtype
1	103	Jenny	55	F	987654221	B+
2	104	Park	24	M	987654311	B+

### Query

List the total units received by the blood bank for each blood type available.

### Command and output

The screenshot shows the SQL Server Management Studio interface. In the top pane, there is a code editor window containing the provided SQL script. Below it is a results pane titled 'Results' which displays a table with three rows of data. The columns are labeled 'Total\_Units\_Received' and 'bloodtype'. The data is as follows:

	Total_Units_Received	bloodtype
1	5	A+
2	1	A-
3	2	O+

## Query

List all the donor ID, blood ID, and units donated by donors whose blood type is A+.

## Command and output

```
--List DID, BID, Units donated by donor whose bloodtype is A+
select d1.DID, d.BID, d.Units from donor d join donor1 d1 on d.DID=d1.DID where d1.Bloodtype='A+';
```

100 %

Results Messages

	DID	BID	Units
1	102	902	4
2	102	903	3

**PL/SQL- Trigger** function is used on the table **Blood bank 2 (BB2)**. The insertion, deletion, or updation of the Blood Bank 1 table is triggered and the BB2 table is updated.

```
--update trigger
CREATE TRIGGER trgAfterUpdate ON bb1
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    -- Debug message
    PRINT 'update trigger fired';
    UPDATE bb2
    SET Totalcap = Totalcap - d.Unitsrec + i.Unitsrec
    FROM bb2 b
    JOIN deleted d ON b.Bloodtype = d.Bloodtype
    JOIN inserted i ON b.Bloodtype = i.Bloodtype;
    -- Debug message
    PRINT 'Update to bb2 completed';
END;

UPDATE bb1 SET Unitsrec = 3 WHERE BBID = 703;
SELECT * FROM bb2;
```

100 %

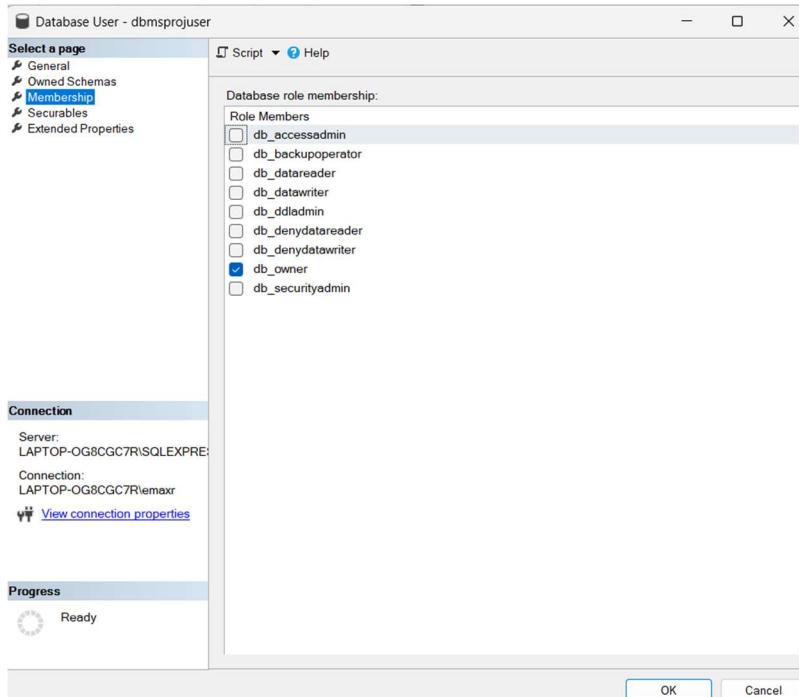
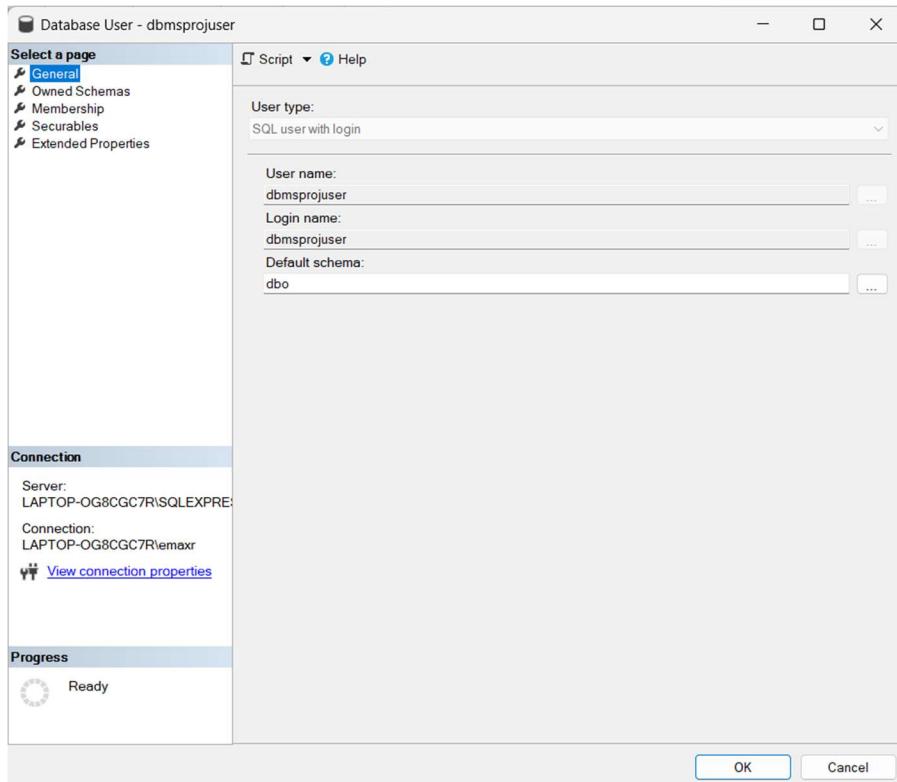
Results Messages

	Bloodtype	Totalcap
1	A+	5
2	A-	0
3	AB+	0
4	AB-	0
5	B+	0
6	B-	0
7	O+	0
8	O-	0

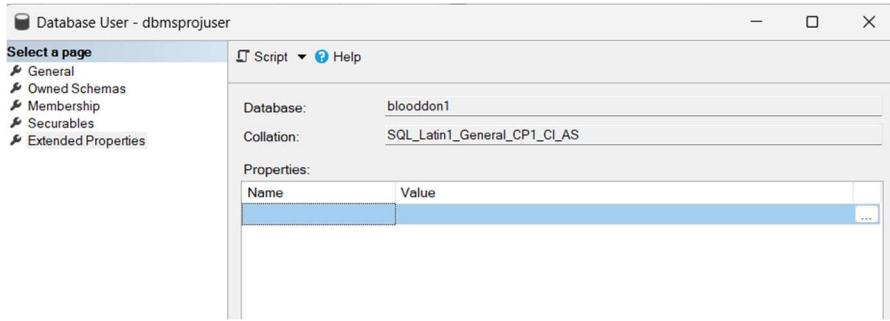
Query executed successfully.

## Creating users with roles on the database blooddon1 in MS SQL server:

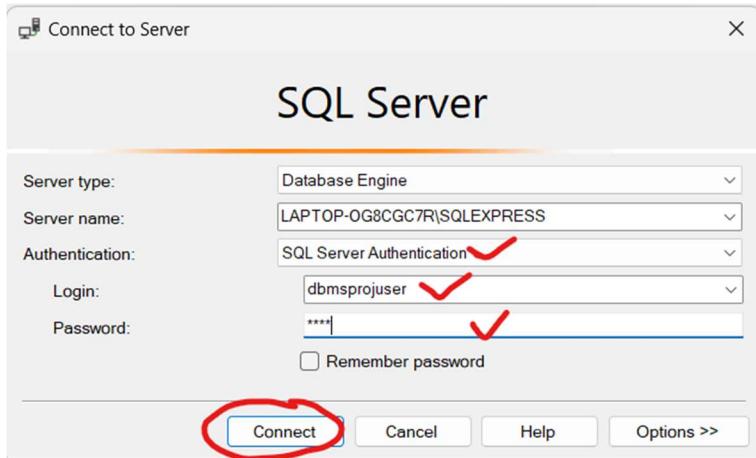
The login name “**dbmsprojuser**” and password are created, and settings in Owned Schemas, and Membership are made under the SQL server EXPRESS that is used to connect the database with the backend.



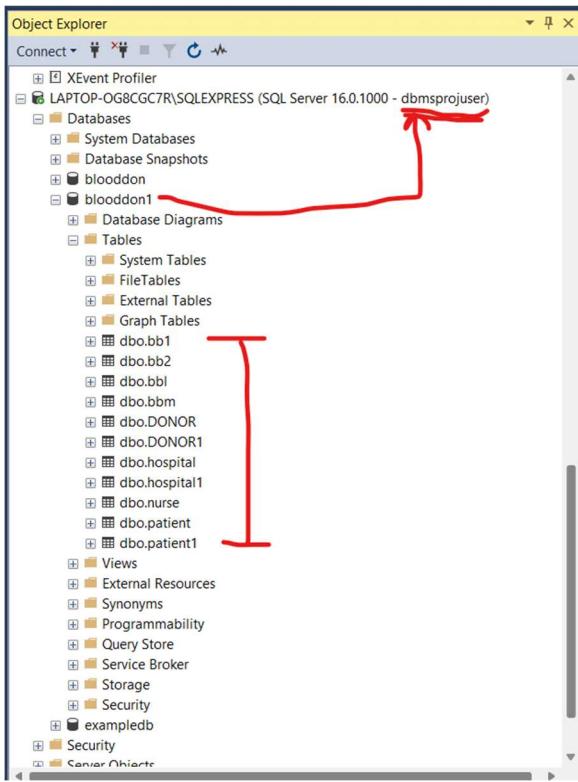
The user is assigned the roles on the database ‘Blooddon1’.



The local host is connected to the server and the authentication is made as “SQL Server Authentication”. The login name and password are given so that the database connects to the backend.

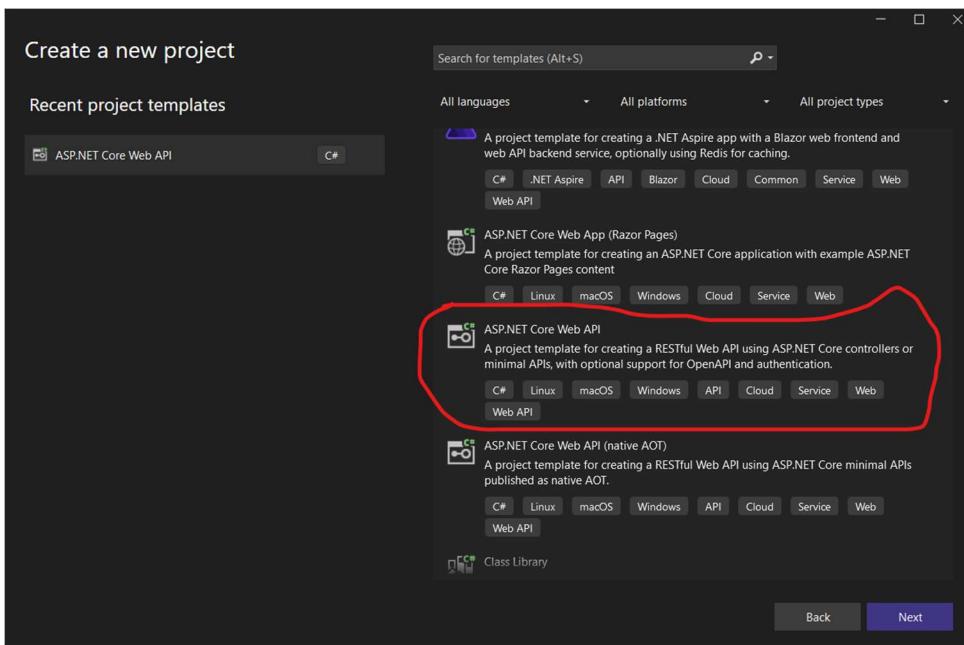


The database “**blooddon1**” is assigned to the “**dbmsprojuser**” and all the tables are accessed such that it can be connected to the backend and frontend processes.



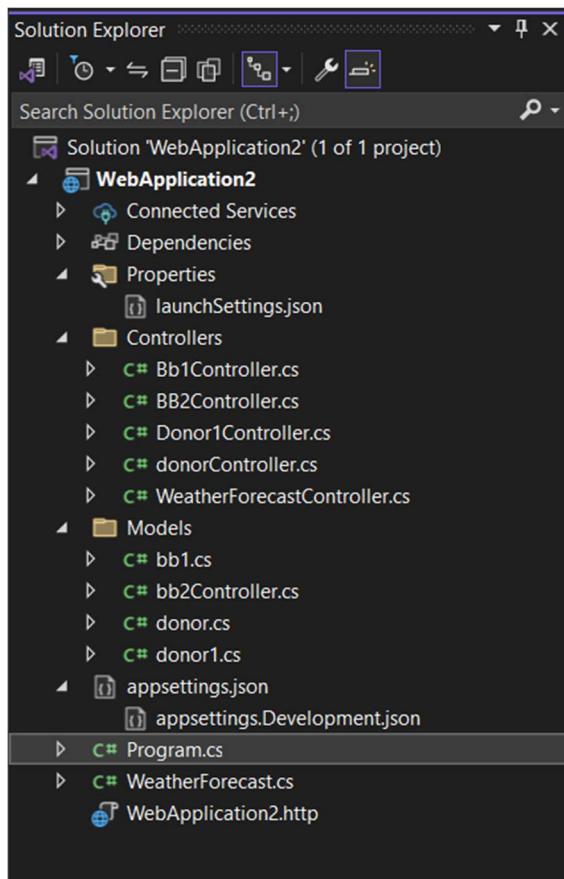
## Creating ASP.NET Core Web API in Visual Studio:

A new project of the template ASP.NET Core Web API is created.



The following is the list of C# files in the project, where the Program.cs file creates the web host which helps the app to listen to HTTP requests.

Appsettings.Development.json file contains the configuration details like the database details, and the connection strings.



## Program.cs

```
using Microsoft.AspNetCore.Mvc.Formatters;
using Microsoft.Extensions.Options;
using Newtonsoft.Json.Serialization;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers(options =>
{
    // Configure Newtonsoft.Json options here
    options.OutputFormatters.RemoveType<SystemTextJsonOutputFormatter>();
}).AddNewtonsoftJson(options =>
{
    // Configure Newtonsoft.Json serializer settings here
    options.SerializerSettings.ReferenceLoopHandling =
    Newtonsoft.Json.ReferenceLoopHandling.Ignore;
    options.SerializerSettings.ContractResolver = new DefaultContractResolver();
});
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
```

```

builder.Services.AddSwaggerGen();

// Enable CORS
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowOrigin",
        builder => builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

// Enable CORS
app.UseCors("AllowOrigin");

app.UseAuthorization();

app.MapControllers();

app.Run();

```

## appsettings.Development.json

```

{
  "ConnectionStrings": {
    "DefaultConnection": "Server=LAPTOP-0G8CGC7R\\SQLEXPRESS;Database=blooddon1; UserID=dbmsprojuser; Password=1234; Trusted_Connection=False; "
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}

```

## **Models: Class files:**

Contains the models of the project defined as classes.

### **Bb1.cs**

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication2.Models
{
    public class Bb1
    {
        public required int BBID { get; set; }
        public required int BID { get; set; }
        public string Location1 { get; set; }
        public string Bloodtype { get; set; }

        public int Unitsrec { get; set; }
    }
}
```

### **Bb2:**

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication2.Models
{
    public class Bb2
    {
        public required string BloodType { get; set; }
        public int TotalCap { get; set; }
    }
}
```

### **Donor:**

```
namespace WebApplication2.Models
{
    public class Donor
```

```
{
    public int BID { get; set; }
    public int DID { get; set; }
    public int Units { get; set; }
}
}
```

### **Donor1:**

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication2.Models
{
    public class Donor1
    {
        public required int DID { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public long Contact { get; set; }
        public string Bloodtype { get; set; }
    }
}
```

### **Controllers:**

The empty controller API is added for each table details. The contents of each table are copied to the *controller API*. *Get, Post* method is used in the API for selection and insertion. *Put, delete http* requests are included such that updation and deletion are done respectively.

### **Bb1Controller.cs:**

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using System;
using System.Data;
using System.Data.SqlClient;
using WebApplication2.Models;

namespace WebApplication2.Controllers
```

```

{
    [Route("api/[controller]")]
    [ApiController]
    public class Bb1Controller : ControllerBase
    {
        private readonly IConfiguration _configuration;

        public Bb1Controller(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        [HttpGet]
        public IActionResult Get()
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {
                    conn.Open();
                    using (SqlCommand cmd = new SqlCommand("SELECT * FROM bb1", conn))
                    {
                        SqlDataReader reader = cmd.ExecuteReader();
                        DataTable dataTable = new DataTable();
                        dataTable.Load(reader);

                        return Ok(dataTable);
                    }
                }
            }
            catch (SqlException ex)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
            }
        }

        [HttpPost]
        public IActionResult Post(Bb1 bb1)
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {
                    conn.Open();
                    string query = "INSERT INTO bb1 (BBID, BID, Location1, Bloodtype, Unitsrec) VALUES (@BBID, @BID, @Location1, @Bloodtype, @Unitsrec)";
                    using (SqlCommand cmd = new SqlCommand(query, conn))
                    {

```

```

        cmd.Parameters.AddWithValue("@BBID", bb1.BBID);
        cmd.Parameters.AddWithValue("@BID", bb1.BID);
        cmd.Parameters.AddWithValue("@Location1", bb1.Location1);
        cmd.Parameters.AddWithValue("@Bloodtype", bb1.Bloodtype);
        cmd.Parameters.AddWithValue("@Unitsrec", bb1.Unitsrec);
        cmd.ExecuteNonQuery();
    }
}
return Ok("Record inserted successfully.");
}
catch (SqlException ex)
{
    return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
}
}

[HttpPut("{id}")]
public IActionResult Put(int id, Bb1 bb1)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "UPDATE bb1 SET Location1=@Location1, Unitsrec=@Unitsrec WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {

                cmd.Parameters.AddWithValue("@Location1", bb1.Location1);
                cmd.Parameters.AddWithValue("@Unitsrec", bb1.Unitsrec);
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} updated successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try

```

```

    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "DELETE FROM bb1 WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} deleted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

// Add other actions (POST, PUT, DELETE) here as needed
}
}

```

## Bb2Controller.cs:

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using System;
using System.Data;
using System.Data.SqlClient;
using WebApplication2.Models;

namespace WebApplication2.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class Bb2Controller : ControllerBase
    {
        private readonly IConfiguration _configuration;

        public Bb2Controller(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        [HttpGet]
        public IActionResult Get()
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {
                    conn.Open();

```

```

        using (SqlCommand cmd = new SqlCommand("SELECT * FROM bb2", conn))
    {
        SqlDataReader reader = cmd.ExecuteReader();
        DataTable dataTable = new DataTable();
        dataTable.Load(reader);

        return Ok(dataTable);
    }
}
catch (SqlException ex)
{
    return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
}

[HttpPost]
public IActionResult Post(Bb2 bb2)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "INSERT INTO bb2 (bloodtype, totalcap) VALUES (@BloodType, @TotalCap)";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@BloodType", bb2.BloodType);
                cmd.Parameters.AddWithValue("@TotalCap", bb2.TotalCap);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok("Record inserted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

[HttpPut("{id}")]
public IActionResult Put(int id, Bb2 bb2)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {

```

```

        conn.Open();
        string query = "UPDATE bb2 SET totalcap = @TotalCap WHERE id = @Id";
        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@TotalCap", bb2.TotalCap);
            cmd.Parameters.AddWithValue("@Id", id);
            cmd.ExecuteNonQuery();
        }
    }
    return Ok($"Record with ID {id} updated successfully.");
}
catch (SqlException ex)
{
    return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
}
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "DELETE FROM bb2 WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} deleted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

// Add other actions (POST, PUT, DELETE) here as needed
}
}

```

## DonorController.cs

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using System;
using System.Data;
using System.Data.SqlClient;
using WebApplication2.Models;

```

```

namespace WebApplication2.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class DonorController : ControllerBase
    {
        private readonly IConfiguration _configuration;

        public DonorController(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        [HttpGet]
        public IActionResult Get()
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is
not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {
                    conn.Open();
                    using (SqlCommand cmd = new SqlCommand("SELECT * FROM donor", conn))
                    {
                        SqlDataReader reader = cmd.ExecuteReader();
                        DataTable dataTable = new DataTable();
                        dataTable.Load(reader);

                        return Ok(dataTable);
                    }
                }
            }
            catch (SqlException ex)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
            }
        }

        [HttpPost]
        public IActionResult Post(Donor donor)
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is
not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {
                    conn.Open();
                    string query = "INSERT INTO donor (BID, DID, Units) VALUES (@BID, @DID,
@Units)";

```

```

        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@BID", donor.BID);
            cmd.Parameters.AddWithValue("@DID", donor.DID);
            cmd.Parameters.AddWithValue("@Units", donor.Units);
            cmd.ExecuteNonQuery();
        }
    }
    return Ok("Record inserted successfully.");
}
catch (SqlException ex)
{
    return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
}

[HttpPut("{id}")]
public IActionResult Put(int id, Donor donor)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is
not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "UPDATE donor SET Units=@Units WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {

                cmd.Parameters.AddWithValue("@Units", donor.Units);
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} updated successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is
not defined in the configuration.");
    }

    try
    {

```

```

        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "DELETE FROM donor WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} deleted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

// Add other actions (POST, PUT, DELETE) here as needed
}
}

```

## Donor1Controller.cs:

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using System;
using System.Data;
using System.Data.SqlClient;
using WebApplication2.Models;
using System.Text.Json;
using System.Collections.Generic;

namespace WebApplication2.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class Donor1Controller : ControllerBase
    {
        private readonly IConfiguration _configuration;

        public Donor1Controller(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        [HttpGet]
        public IActionResult Get()
        {
            string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(sqlDataSource))
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
            }

            try
            {
                using (SqlConnection conn = new SqlConnection(sqlDataSource))
                {

```

```

        conn.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM donor1", conn))
        {
            SqlDataReader reader = cmd.ExecuteReader();
            DataTable dataTable = new DataTable();
            dataTable.Load(reader);

            return Ok(dataTable);
        }
    }
}
catch (SqlException ex)
{
    return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
}

[HttpPost]
public IActionResult Post(Donor1 donor1)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is not defined in the configuration.");
    }

    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "INSERT INTO donor1 (DID, Name, Age, Gender, Contact, Bloodtype) VALUES (@DID, @Name, @Age, @Gender, @Contact, @Bloodtype)";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@DID", donor1.DID);
                cmd.Parameters.AddWithValue("@Name", donor1.Name);
                cmd.Parameters.AddWithValue("@Age", donor1.Age);
                cmd.Parameters.AddWithValue("@Gender", donor1.Gender);
                cmd.Parameters.AddWithValue("@Contact", donor1.Contact);
                cmd.Parameters.AddWithValue("@Bloodtype", donor1.Bloodtype);

                cmd.ExecuteNonQuery();
            }
        }
        return Ok("Record inserted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    string sqlDataSource = _configuration.GetConnectionString("DefaultConnection");
    if (string.IsNullOrEmpty(sqlDataSource))
    {
}

```

```

        return StatusCode(StatusCodes.Status500InternalServerError, "DefaultConnection is
not defined in the configuration.");
    }

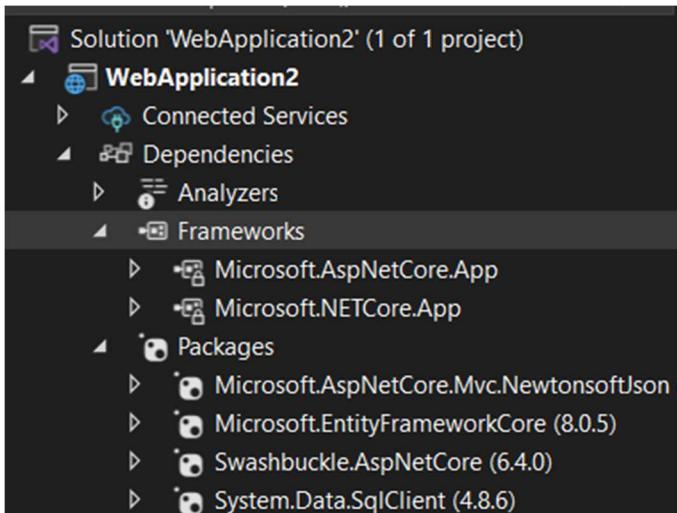
    try
    {
        using (SqlConnection conn = new SqlConnection(sqlDataSource))
        {
            conn.Open();
            string query = "DELETE FROM donor1 WHERE id = @Id";
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@Id", id);
                cmd.ExecuteNonQuery();
            }
        }
        return Ok($"Record with ID {id} deleted successfully.");
    }
    catch (SqlException ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

// Add other actions (POST, PUT, DELETE) here as needed
}
}

```

## Frameworks and Packages used:

- [Microsoft.AspNetCore.App](#)
- [Microsoft.NETCore.App](#)



For the visual representation of the API and the documentation in the interactive format **Swagger** is used. (instead of postman)

The output of the created API is shown below.

The screenshot displays four separate Swagger UI windows, each representing a different API definition:

- WebApplication2**: Shows endpoints for Bb1 and Bb2. Bb1 includes GET, POST, PUT, and DELETE methods for /api/Bb1. Bb2 includes GET, POST, and PUT methods for /api/Bb2.
- Bb1**: Shows endpoints for Bb1. It includes GET, POST, PUT, and DELETE methods for /api/Bb1.
- Bb2**: Shows endpoints for Bb2. It includes GET, POST, and PUT methods for /api/Bb2.
- Donor**: Shows endpoints for Donor. It includes GET, POST, PUT, and DELETE methods for /api/Donor.

## CRUD operation on the BB1 table

### GET

The entries in the table in the backend as well as status of the table are displayed.

```

curl -X 'GET' \
  'http://localhost:13242/api/BB1' \
  -H 'accept: */*'

```

```

http://localhost:13242/api/BB1

```

Code	Details
200	Response body

```

[{"BBID": 701, "BID": 901, "location": "Chennai", "bloodtype": "O+", "unitsrec": 2, "id": 1}, {"BBID": 702, "BID": 902, "location": "Bangalore", "bloodtype": "A-", "unitsrec": 1, "id": 2}, {"BBID": 703,

```

Curl

```
curl -X 'GET' \
'http://localhost:13242/api/Bb1' \
-H 'accept: */*'
```

Request URL

<http://localhost:13242/api/Bb1>

Server response

Code	Details	Links
200	Response body	
	<pre>{   "BBID": 707,   "BID": 907,   "location": "Pune",   "bloodtype": "A+",   "Unitsrec": 2,   "Id": 3 }, {   "BBID": 704,   "BID": 903,   "location": "Pune",   "bloodtype": "A+",   "Unitsrec": 2,   "Id": 4 }, {   "BBID": 705,   "BID": 905,   "location": "Chennai",   "bloodtype": "O+",   "Unitsrec": 2,   "Id": 5 }, {   "BBID": 706,   "BID": 906,   "location": "Pune",   "bloodtype": "A+",   "Unitsrec": 2,   "Id": 6 }</pre>	No links
	Response headers	
	<pre>content-length: 651 content-type: application/json; charset=utf-8 date: Tue, 28 May 2024 12:43:42 GMT server: Microsoft-IIS/10.0 x-powered-by: ASP.NET</pre>	

Responses

Code	Description	Links
200	Success	No links

Curl

```
curl -X 'GET' \
'http://localhost:13242/api/Bb1' \
-H 'accept: */*'
```

Request URL

<http://localhost:13242/api/Bb1>

Server response

Code	Details	Links
200	Response body	
	<pre>{   "Unitsrec": 2,   "Id": 5 }, {   "BBID": 706,   "BID": 906,   "location": "Pune",   "bloodtype": "A+",   "Unitsrec": 2,   "Id": 6 }, {   "BBID": 708,   "BID": 907,   "location": "shimla",   "bloodtype": "A-",   "Unitsrec": 3,   "Id": 13 }, {   "BBID": 709,   "BID": 908,   "location": "Bombay",   "bloodtype": "AB-",   "Unitsrec": 3,   "Id": 16 } ]</pre>	No links
	Response headers	
	<pre>content-length: 651 content-type: application/json; charset=utf-8 date: Tue, 28 May 2024 12:43:42 GMT server: Microsoft-IIS/10.0 x-powered-by: ASP.NET</pre>	

Responses

Code	Description	Links
200	Success	No links

## POST

The insertion is made with the help of POST http request such that the action reflects in the database table too.

**POST** /api/Bb1

**Parameters**

No parameters

**Request body**

application/json-patch+json

```
{
    "bid": 710,
    "bbid": 999,
    "location1": "Kerala",
    "bloodtype": "AB+",
    "unitsrec": 5
}
```

**Responses**

**Curl**

```
curl -X 'POST' \
'http://localhost:13242/api/Bb1' \
-H 'accept: */*' \
-H 'Content-Type: application/json-patch+json' \
-d '{
    "bid": 710,
    "bbid": 999,
    "location1": "Kerala",
    "bloodtype": "AB+",
    "unitsrec": 5
}'
```

**Responses**

**Curl**

```
curl -X 'POST' \
'http://localhost:13242/api/Bb1' \
-H 'accept: */*' \
-H 'Content-Type: application/json-patch+json' \
-d '{
    "bid": 710,
    "bbid": 999,
    "location1": "Kerala",
    "bloodtype": "AB+",
    "unitsrec": 5
}'
```

**Request URL**

<http://localhost:13242/api/Bb1>

**Server response**

Code	Details
200	<b>Response body</b> Record inserted successfully.  <b>Response headers</b> access-control-allow-origin: * content-encoding: gzip content-type: text/plain; charset=utf-8 date: Tue, 28 May 2024 12:45:29 GMT server: Microsoft-IIS/10.0 transfer-encoding: chunked vary: Accept-Encoding x-powered-by: ASP.NET

**Responses**

Code	Description	Links
200	Success	No links

The updated table in the database is

	BBID	BID	Location1	Bloodtype	Unitsrec	Id
1	701	901	Chennai	O+	2	1
2	702	904	Bangalore	A-	1	2
3	707	902	Pune	A+	2	3
4	704	903	Pune	A+	2	4
5	705	905	Chennai	O+	2	5
6	706	906	Pune	A+	2	6
7	708	907	shimla	A-	3	13
8	709	908	Bombay	AB-	3	16
9	710	909	Kerala	AB+	5	19

## PUT

The updation is made with the help of PUT http request such that the physical table is also updated.

PUT /api/Bb1/{id}

Parameters

Name	Description
Id * required	integer(\$int32) 19 (parts)

Request body

application/json-patch+json

```
{
  "location1": "East Kerala",
  "bloodtype": "AB+",
  "unitsrec": 6
}
```

Execute Clear

Responses

```
curl -X 'PUT' \
'http://localhost:13242/api/Bb1/19' \
-H 'accept: */*' \
-H 'Content-Type: application/json-patch+json' \
-d '{
  "location1": "East Kerala"
}'
```

Responses

```
Curl
curl -X 'PUT' \
http://localhost:13242/api/Bb1/19' \
-H 'Accept: */*' \
-H 'Content-Type: application/json-patch+json' \
-d '{
  "Location": "East Kerala",
  "Bloodtype": "AB+",
  "Unitsrec": 6
}'
```

Request URL  
<http://localhost:13242/api/Bb1/19>

Server response

Code	Details	Links
200	<p>Response body</p> <p>Record with ID 19 updated successfully.</p>	<a href="#">Copy</a> <a href="#">Download</a>
	Response headers	
	<pre>access-control-allow-origin: * accept-encoding: gzip content-type: text/plain; charset=utf-8 date: Mon, 17 Jul 2017 10:17:07 GMT server: Microsoft-IIS/10.0 transfer-encoding: chunked vary: Accept-Encoding x-powered-by: ASP.NET</pre>	

Responses

Code	Description	Links
200	Success	No links

The updated table in the database is

	BBID	BID	Location1	Bloodtype	Unitsrec	Id
1	701	901	Chennai	O+	2	1
2	702	904	Bangalore	A-	1	2
3	707	902	Pune	A+	2	3
4	704	903	Pune	A+	2	4
5	705	905	Chennai	O+	2	5
6	706	906	Pune	A+	2	6
7	708	907	shimla	A-	3	13
8	709	908	Bombay	AB-	3	16
9	710	909	East Kerala	AB+	6	19

## DELETE

The deletion is made with the help of DELETE http request such that the physical table is also reflected.

**DELETE** /api/Bb1/{id}

**Parameters**

Name	Description
<b>id</b> <small>* required</small>	integer(\$int32) (path)

**Responses**

Curl

```
curl -X 'DELETE' \
'http://localhost:13242/api/Bb1/19' \
-H 'accept: */*'
```

Request URL

<http://localhost:13242/api/Bb1/19>

Server response

**Code** **Details**

200 Response body

```
Record with ID 19 deleted successfully.
```

Response headers

```
access-control-allow-origin: *
content-encoding: gzip
content-type: text/plain; charset=utf-8
date: Tue, 28 May 2024 12:48:54 GMT
server: Microsoft-IIS/10.0
transfer-encoding: chunked
vary: Accept-Encoding
x-powered-by: ASP.NET
```

The updated table in the database is

	BBID	BID	Location1	Bloodtype	Unitsrec	Id
1	701	901	Chennai	O+	2	1
2	702	904	Bangalore	A-	1	2
3	707	902	Pune	A+	2	3
4	704	903	Pune	A+	2	4
5	705	905	Chennai	O+	2	5
6	706	906	Pune	A+	2	6
7	708	907	shimla	A-	3	13
8	709	908	Bombay	AB-	3	16

## FRONTEND

For the frontend, react.js is used. *Index.js* is used to see the contents of the app. Components are injected into the div with id. Bootstrap is installed for styling and the header is added.

### Index.html

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
        manifest.json provides metadata used when your web app is installed on a
        user's mobile device or desktop. See
        https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
        Notice the use of %PUBLIC_URL% in the tags above.
        It will be replaced with the URL of the `public` folder during the build.
        Only files inside the `public` folder can be referenced from the HTML.

        Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
        work correctly both with client-side routing and a non-root public URL.
        Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
        This HTML file is a template.
        If you open it directly in the browser, you will see an empty page.

        You can add webfonts, meta tags, or analytics to this file.
        The build step will place the bundled scripts into the <body> tag.

        To begin the development, run `npm start` or `yarn start` .
    -->
```

```

    To create a production bundle, use `npm run build` or `yarn build`.

-->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## App.js

*App.js* is the root component which contains the contents that would be displayed on the screen. It includes pages that connects to each table separately to display data and to perform CRUD operations.

```

import logo from './logo.svg';
import './App.css';
import {Home} from './Home';
import {BB1} from './BB1';
import {BB2} from './BB2';
import {Donor} from './Donor';
import {Donor1} from './Donor1';
import { BrowserRouter, Route, Switch, NavLink } from 'react-router-dom';

function App() {
  return (
    <BrowserRouter>
      <div className="App-container">
        <h3 className="d-flex justify-content-center m-3">
          Blood Donation Management System
        </h3>

        <nav className="navbar navbar-expand-sm bg-light navbar-dark">
          <ul className="navbar-nav">
            <li className="nav-item- m-1">
              <NavLink className="btn btn-light btn-outline-primary" to="/home">
                Home
              </NavLink>
            </li>
            <li className="nav-item- m-1">
              <NavLink className="btn btn-light btn-outline-primary" to="/bb1">
                BB1
              </NavLink>
            </li>
            <li className="nav-item- m-1">
              <NavLink className="btn btn-light btn-outline-primary" to="/bb2">
                BB2
              </NavLink>
            </li>
          </ul>
        </nav>
      </div>
    </BrowserRouter>
  );
}

export default App;

```

```

        </NavLink>
    </li>
    <li className="nav-item- m-1">
        <NavLink className="btn btn-light btn-outline-primary" to="/donor">
            Donor
        </NavLink>
    </li>
    <li className="nav-item- m-1">
        <NavLink className="btn btn-light btn-outline-primary" to="/donor1">
            Donor1
        </NavLink>
    </li>
</ul>

</nav>

<Switch>
    <Route path='/home' component={Home}/>
    <Route path='/bb1' component={BB1}/>
    <Route path='/bb2' component={BB2}/>
    <Route path='/donor' component={Donor}/>
    <Route path='/donor1' component={Donor1}/>
</Switch>

</div>
</BrowserRouter>
);
}

export default App;

```

## Variables.js

*variable.js* is used such that it is connected to the API using the port number *13242* specific to the localhost here.

```

export const variables={
    API_URL:"http://localhost:13242/api/"
}

```

## Bb1.js

```
import React, {Component} from 'react';
import {variables} from './Variables';

export class BB1 extends Component{
    constructor(props){
        super(props);
        this.state={
            bb1:[],
            modalTitle:"",
            BBID:"",
            BID:"",
            Location1:"",
            Bloodtype:"",
            Unitsrec:"",
            Id:0
        };
    }

    refreshList() {
        fetch(variables.API_URL + 'bb1')
            .then(response => {
                if (!response.ok) {
                    throw new Error('Network response was not ok ' + response.statusText);
                }
                return response.json();
            })
            .then(data => {
                this.setState({ bb1: data });
            })
            .catch(error => {
                console.error('There was a problem with the fetch operation:', error);
            });
    }

    componentDidMount(){
        this.refreshList();
    }
    changeBBID = (e) => {
        this.setState({ BBID: e.target.value });
    }
}
```

```

changeBID = (e) => {
    this.setState({ BID: e.target.value });
}
changeLocation = (e) => {
    this.setState({ Location1: e.target.value });
}

changeBloodtype = (e) => {
    this.setState({ Bloodtype: e.target.value });
}

changeUnitsrec=(e)=>{
    this.setState({Unitsrec:e.target.value})
}

addClick = () => {
    this.setState({
        modalTitle: "Add new BBID",
        BBID: "", // Set BBID to an empty string to allow user input
        BID: "",
        Location1: "",
        Bloodtype: "",
        Unitsrec: "",
        Id:0
    });
}

editClick = (bba1) => {
    this.setState({
        modalTitle: "Edit BBID",
        Id: bba1.Id,
        BBID: bba1.BBID,
        BID: bba1.BID,
        Bloodtype: bba1.Bloodtype,
        Unitsrec: bba1.Unitsrec,
        Location1: bba1.Location1
    });
}

createClick(){
    fetch(variables.API_URL+'bb1',{
        method:'POST',
        headers:{
            'Accept':'application/json',
            'Content-type': 'application/json'
        },
        body:JSON.stringify({

```

```

        BBID:this.state.BBID,
        BID:this.state.BID,
        Location1:this.state.Location1,
        Bloodtype:this.state.Bloodtype,
        Unitsrec:this.state.Unitsrec

    })
})
.then(res => {
    if (!res.ok) {
        throw new Error('Failed to create BBID');
    }
    return res.json();
})
.then(result => {
    alert('New BBID added successfully');
    this.refreshList();
})
.catch(error => {
    alert('Failed to create BBID: ' + error.message);
});
}

updateClick(){
    fetch(variables.API_URL + 'bb1/' + this.state.Id, {
        method: 'PUT',
        headers: {
            'Accept': 'application/json',
            'Content-type': 'application/json'
        },
        body: JSON.stringify({
            BBID: this.state.BBID,
            BID: this.state.BID,
            Location1: this.state.Location1,
            Bloodtype: this.state.Bloodtype,
            Unitsrec: this.state.Unitsrec
        })
    })
    .then(res => {
        if (!res.ok) {
            throw new Error('Failed to update BBID');
        }
        return res.json();
    })
    .then(result => {
        alert('BBID updated successfully');
        this.refreshList();
    })
}

```

```

        })
        .catch(error => {
            alert('Failed to update BBID: ' + error.message);
        });
    }

    render(){
        const {
            bb1,
            modalTitle,
            BBID,
            BID,
            Location1,
            Bloodtype,
            Unitsrec,
            Id
        } =this.state;

        return(
<div>
    <button type="button"
        className="btn btn-primary m-2 float-end"
        data-bs-toggle="modal"
        data-bs-target="#exampleModal"
        onClick={()=>this.addClick}>
        Add new BBID
    </button>
    <table className="table table-striped">
        <thead>
            <tr>
                <th>
                    BBID
                </th>
                <th>
                    BID
                </th>
                <th>
                    Location
                </th>
                <th>
                    Bloodtype
                </th>
                <th>
                    Units recieved
                </th>
                <th>

```

```

        Options
    </th>
</tr>
</thead>
<tbody>
{bba1.map(bba1=>
<tr key={bba1.BBID}>
<td>{bba1.BBID}</td>
<td>{bba1.BID}</td>
<td>{bba1.Location1}</td>
<td>{bba1.Bloodtype}</td>
<td>{bba1.Unitsrec}</td>

<td>
<button type="button"
className="btn btn-light mr-1"
data-bs-toggle="modal"
data-bs-target="#exampleModal"
onClick={()=>this.editClick(bba1)}>
<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" className="bi bi-pencil-square" viewBox="0 0 16 16">
<path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-
2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293z"/>
<path fillRule="evenodd" d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5
1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-
.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 1 2.5z"/>
</svg>
</button>
</td>
</tr>
)})

</tbody>
</table>

<div className="modal fade" id="exampleModal" tabIndex="-1" aria-hidden="true">
<div className="modal-dialog modal-lg modal-dialog-centered">
<div className="modal-content">
<div className="modal-header">
<h5 className="modal-title">{modalTitle}</h5>
<button type="button" className="btn-close" data-bs-dismiss="modal" aria-
label="Close">
</button>
</div>

```

```

<div className="modal-body">

    <div className="input-group mb-3">
        <span className="input-group-text">BBID</span>
        <input type="text" className="form-control" value={BBID} onChange={(e) =>
this.setState({ BBID: e.target.value })} />
    </div>
    <div className="input-group mb-3">
        <span className="input-group-text">BID</span>
        <input type="text" className="form-control" value={BID} onChange={(e) =>
this.setState({ BID: e.target.value })} />
    </div>
    <div className="input-group mb-3">
        <span className="input-group-text">Bloodtype</span>
        <input type="text" className="form-control" value={Bloodtype}
onChange={(e) => this.setState({ Bloodtype: e.target.value })} />
    </div>
    <div className="input-group mb-3">
        <span className="input-group-text">Location</span>
        <input type="text" className="form-control" value={Location1}
onChange={(e) => this.setState({ Location1: e.target.value })} />
    </div>
    <div className="input-group mb-3">
        <span className="input-group-text">Unitsrec</span>
        <input type="text" className="form-control" value={Unitsrec}
onChange={(e) => this.setState({ Unitsrec: e.target.value })} />
    </div>

    {Id==0?
    <button type="button"
    className="btn btn-primary float-start"
    onClick={()=>this.createClick()}
    >Create</button>
    :null}

    {Id!=0?
    <button type="button"
    className="btn btn-primary float-start"
    onClick={()=>this.updateClick()}
    >Update</button>
    :null}

    </div>
</div>
</div>

```

```
        )  
    }  
}
```

## FRONTEND OUTPUT

### Displaying data

The screenshot shows a web browser window titled "Blood Donation Management System". The URL is "localhost:3000/bb1". The page has a header with navigation links: Home, BB1 (which is active), BB2, Donor, and Donor1. A blue button "Add new BBID" is located in the top right. Below the header is a table with columns: BBID, BID, Location, Bloodytype, Units received, and Options. The data rows are:

BBID	BID	Location	Bloodytype	Units received	Options
701	901	Chennai	O+	2	<input type="checkbox"/>
702	904	Bangalore	A-	1	<input type="checkbox"/>
707	902	Pune	A+	2	<input type="checkbox"/>
704	903	Pune	A+	2	<input type="checkbox"/>
705	905	Chennai	O+	2	<input type="checkbox"/>
706	906	Pune	A+	2	<input type="checkbox"/>
708	907	shimla	A-	3	<input type="checkbox"/>
709	908	Bombay	AB-	3	<input type="checkbox"/>

### Insertion of new entry

The screenshot shows a modal dialog box in the center of the screen. The title bar says "localhost:3000 says" and the message inside the box is "New BBID added successfully". There is an "ok" button at the bottom right of the modal. In the background, the main table of blood donations is visible. A new row is being added with the following values:

BBID	710
BID	909
Bloodytype	O+
Location	Pune
Unitsrec	4

Below the modal, there is a "Create" button.

Swagger UI   React App   localhost:3000/bb1

### Blood Donation Management System

Home BB1 BB2 Donor Donor1 Add new BBID

BBID	BID	Location	Bloodtype	Units received	Options
701	901	Chennai	O+	2	<input type="checkbox"/>
702	904	Bangalore	A-	1	<input type="checkbox"/>
707	902	Pune	A+	2	<input type="checkbox"/>
704	903	Pune	A+	2	<input type="checkbox"/>
705	905	Chennai	O+	2	<input type="checkbox"/>
706	906	Pune	A+	2	<input type="checkbox"/>
708	907	shimla	A-	3	<input type="checkbox"/>
709	908	Bombay	AB-	3	<input type="checkbox"/>
710	909	Pune	O+	4	<input type="checkbox"/>

The updated table in database

Results Messages

	BBID	BID	Location1	Bloodtype	Unitsrec	Id
1	701	901	Chennai	O+	2	1
2	702	904	Bangalore	A-	1	2
3	707	902	Pune	A+	2	3
4	704	903	Pune	A+	2	4
5	705	905	Chennai	O+	2	5
6	706	906	Pune	A+	2	6
7	708	907	shimla	A-	3	13
8	709	908	Bombay	AB-	3	16
9	710	909	Pune	O+	4	20

## Updation of the records

localhost:3000 says  
BBID updated successfully

OK

Add new BBID

BBID	BID	Location	Options
701	901	Chennai	<input type="checkbox"/>
702	904	Bangalore	<input type="checkbox"/>
707	902	Pune	<input type="checkbox"/>
704	903	Pune	<input type="checkbox"/>
705	905	Chennai	<input type="checkbox"/>
706	906	Pune	<input type="checkbox"/>
708	907	shimla	<input type="checkbox"/>
709	908	Bombay	<input type="checkbox"/>
710	909	Pune	<input type="checkbox"/>

Edit BBID

BBID: 710  
BID: 909  
Bloodtype: O+  
Location: Mumabi  
Unitsrec: 5

Update

BBID	BID	Location	Bloodtype	Units received	Options
701	901	Chennai	O+	2	<input type="checkbox"/>
702	904	Bangalore	A-	1	<input type="checkbox"/>
707	902	Pune	A+	2	<input type="checkbox"/>
704	903	Pune	A+	2	<input type="checkbox"/>
705	905	Chennai	O+	2	<input type="checkbox"/>
706	906	Pune	A+	2	<input type="checkbox"/>
708	907	Shimla	A-	3	<input type="checkbox"/>
709	908	Bombay	AB-	3	<input type="checkbox"/>
710	909	Mumabi	O+	5	<input type="checkbox"/>

The updated table in the database

	BBID	BID	Location1	Bloodtype	Unitsrec	Id
1	701	901	Chennai	O+	2	1
2	702	904	Bangalore	A-	1	2
3	707	902	Pune	A+	2	3
4	704	903	Pune	A+	2	4
5	705	905	Chennai	O+	2	5
6	706	906	Pune	A+	2	6
7	708	907	Shimla	A-	3	13
8	709	908	Bombay	AB-	3	16
9	710	909	Mumabi	O+	5	20

## CONCLUSION

- Microsoft SQL Server Management Studio is used to create the database and its respective tables needed for the Blood donation management system. Specific DML and DDL commands were performed. Retrieval queries and triggers were implemented.
- A full-stack web application is developed using ASP.NET Core Web API, Microsoft SQL Server and React.js. The backend driven by ASP.NET Core Web API is integrated with the database, and the frontend is developed using react.js for dynamic, responsive user interface and is made to communicate with backend to perform CRUD operations on the data.