SOFTWARE SOLUTION

We provide software solution for your every problem.

CS2231: BM

Portfolio Maker

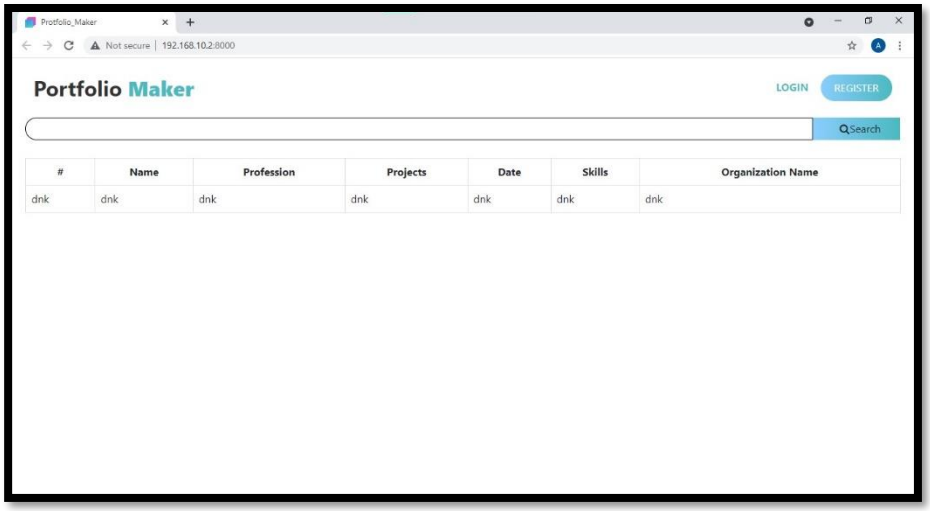
A project of FA Software Solution

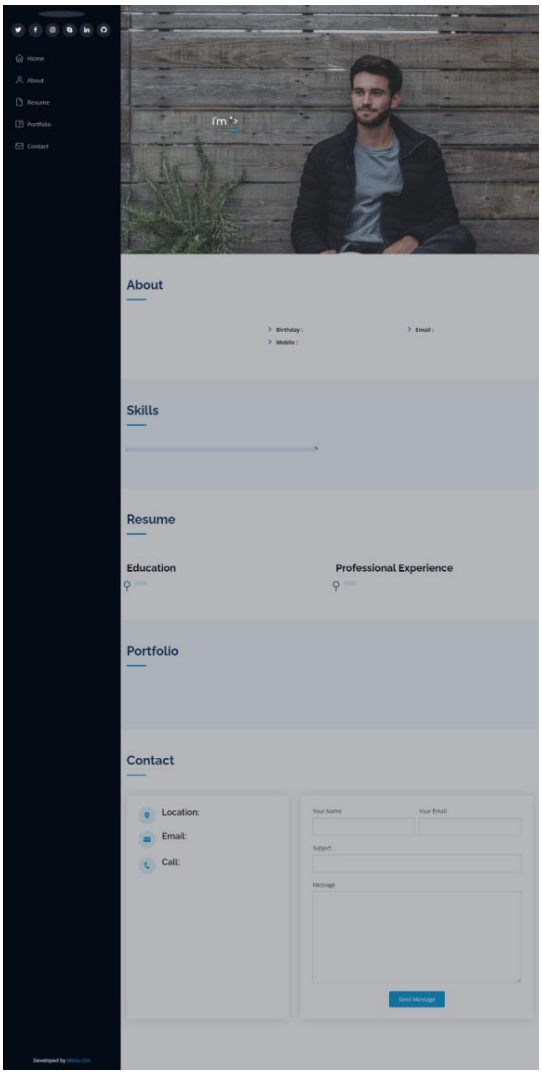
Team Members

Muhammad Fahad (FA19-BSSE-0014)

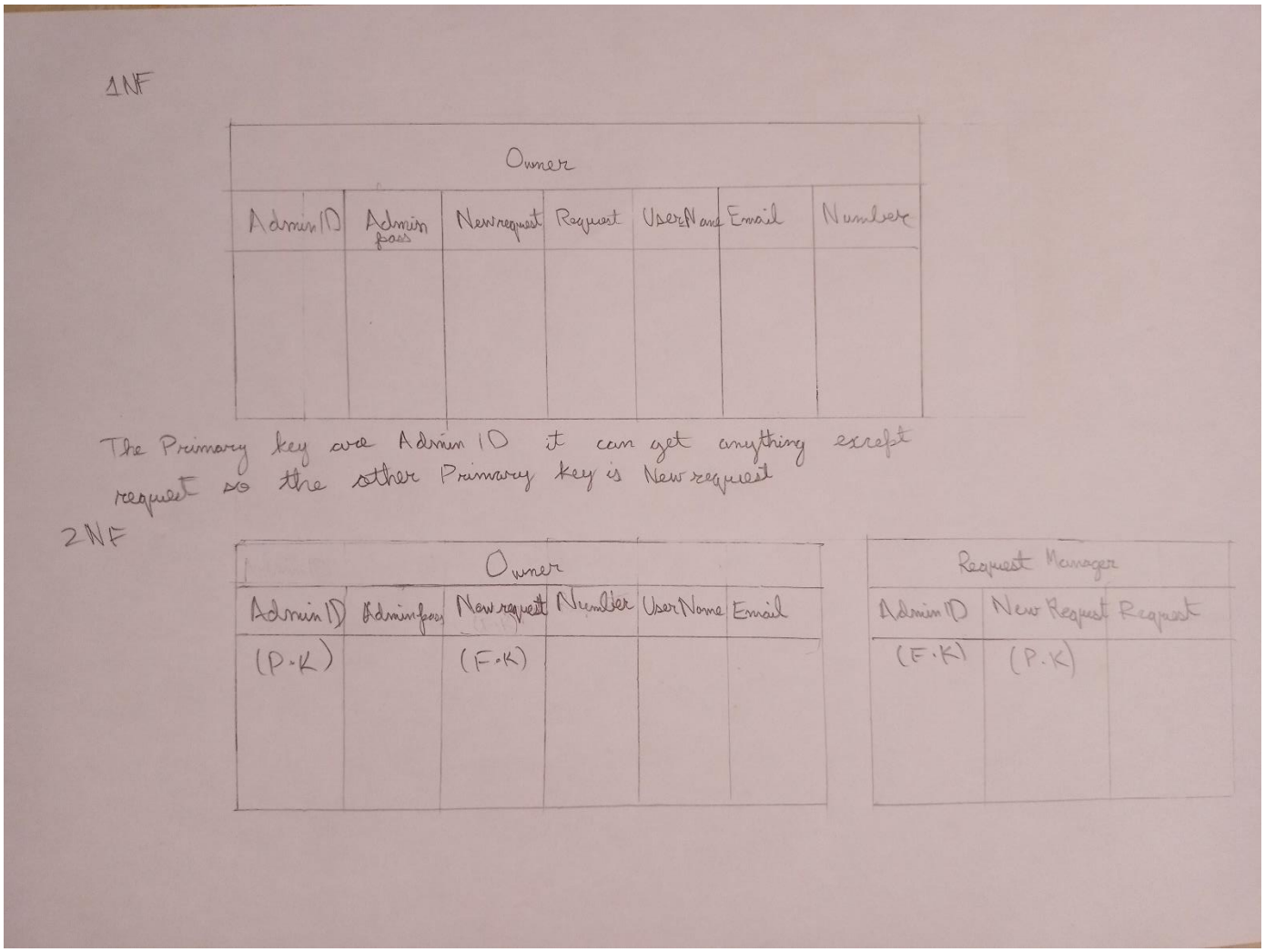
Ahmed Amin (FA19-BSSE-0063)

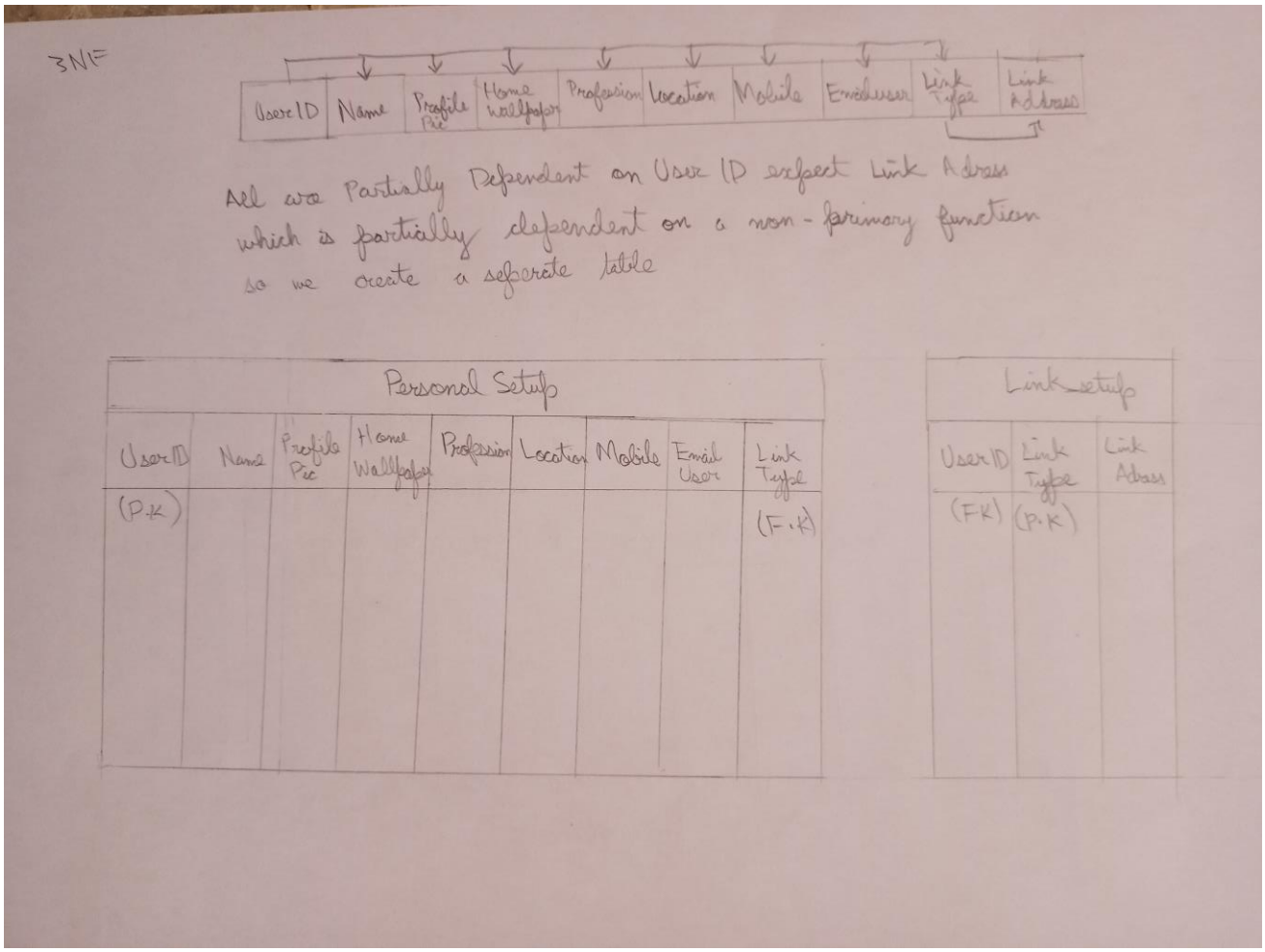
Introduction





Entity Relationship Model Diagram, Normalize Relationships and functional dependencies




3NF
UserID
Name
Profile Pic
Home wallpaper
Profession
Location
Mobile
Email user
Link Type
Link Address
All are Partially Dependent on User ID expect Link Adress which is partially dependent on a non-primary function so we create a seperate table
Personal Setup
UserID
Name
Profile Pic
Home Wallpaper
Profession
Location
Mobile
Email User
Link Type
(P.K)
(F.K)
Link_setup
UserID
Link Type
Link Adress
(FK)
(P.K)

1 NF

Personal setup											
UserID	Profilepic	Name	home wallpaper	Proffesions	Location	mobile	emailuser	Facebook	Github	Instagram	Tweet
(P-K)											

The Primary key can be UserID because it can get all information of table
The Data is already in atomic form

2NF

To reduce Data Dependency we create two colum to reduce other

Personal setup									
UserID	Profile Pic	Home Wallpaper	Proffesion	Location	Mobile	email user	Name	Link Type	Link Adress

Create Tables

```
CREATE TABLE admin_users (
  username varchar(250) NOT NULL,
  user_id varchar(10) NOT NULL,
  user_pass varchar(250) NOT NULL,
  Lkey int NOT NULL
)

CREATE TABLE aboutus_setup (
  user_id varchar(10) NOT NULL, -- FK
  dob varchar(250) NOT NULL, -- FK+PK
  heading text NOT NULL,
  subheading text NOT NULL,
  shortdesc text NOT NULL,
  longdesc text NOT NULL,
  website varchar(250) NOT NULL

)



CREATE TABLE basic_setup (
  user_id varchar(10) NOT NULL,-- FK
  LKey int NOT NULL,--PK = FK + PK
  title varchar(250) NOT NULL,
  description text NOT NULL,
  keyword text NOT NULL,
  icon varchar(250) NOT NULL,
  Theme varchar(10) NOT NULL -- fk
)


CREATE TABLE Theme (
  Theme varchar(10) NOT NULL, -- pk
  Location varchar(10) NOT NULL,
  Customize int  NOT NULL DEFAULT 0,
)

CREATE TABLE Theme_Customize (
  Theme varchar(10) NOT NULL, -- fk.pk
  user_id varchar(10) NOT NULL,-- FK,pk
  fcolor nvarchar(6) NOT NULL,
  bcolor varchar(6) NOT NULL,
  fontStyle varchar(250) NOT NULL,
  fontSize int NOT NULL,
)


CREATE TABLE contact (
  user_id varchar(10) NOT NULL,-- FK
  Cid int NOT NULL, --PK = FK + PK
  cname varchar(250) NOT NULL,
  cemail varchar(250) NOT NULL,
  csubject text NOT NULL,
  cmessage text NOT NULL
```

```
)

CREATE TABLE personal_setup (
  user_id varchar(10) NOT NULL, -- fk
  profilepic varchar(250) NOT NULL,
  name varchar(250) NOT NULL,
  homewallpaper varchar(200) NOT NULL,
  professions varchar(200) NOT NULL,
  location text NOT NULL,
  mobile varchar(200) NOT NULL,
  emailuser varchar(200) NOT NULL -- pk
)

CREATE TABLE Link_setup(
  user_id varchar(10) NOT NULL, -- FK
  Name varchar(200) NOT NULL, -- PK = FK + PK
  Link varchar(200) NOT NULL,
)


CREATE TABLE ProjectDetail (
  user_id varchar(10) NOT NULL, -- fk,pk
  projectname varchar(250) NOT NULL, -- pk
  projectpic varchar(250) NOT NULL,
  projectlink text NOT NULL,
  projectdesc varchar(250) NOT NULL,
)


CREATE TABLE Experience (
  user_id varchar(10) NOT NULL, -- fk , pk
  category varchar(250) NOT NULL,
  title varchar(250) NOT NULL,
  year varchar(250) NOT NULL, -- pk
  ogname varchar(250) NOT NULL, -- pk
  workdesc text NOT NULL
)

CREATE TABLE skills (
  user_id varchar(10) NOT NULL, -- fk , pk
  skill varchar(250) NOT NULL, -- pk
  score varchar(250) NOT NULL
)

CREATE TABLE Licence (
  user_id varchar(10) NOT NULL, -- fk , pk
  LKey int NOT NULL, -- pk
  ExpireDate varchar(12) NOT NULL,
  IssueDate varchar(12) NOT NULL,
  LStatus int NOT NULL DEFAULT 1
)


CREATE TABLE Owner_Setup(
AdminID varchar(10) Not NUll,
AdminPass varchar(20) Not Null,
NewRequest varchar(10) Not Null,
Number varchar(12) Not NUll,
```

```
UserName varchar (20) Not Null,
Email varchar(30) Not Null,
);


CREATE TABLE Request_Manager(
AdminID varchar(10) Not NUll,
NewRequest varchar(10) Not Null,
Request varchar(10) Not Null,
);
```

Primary Keys

```
ALTER TABLE admin_users
  ADD PRIMARY KEY (user_id);
ALTER TABLE aboutus_setup
  ADD PRIMARY KEY (user_id,dob);
ALTER TABLE basic_setup
  ADD PRIMARY KEY (user_id,Theme);
ALTER TABLE Theme
  ADD PRIMARY KEY (Theme);
ALTER TABLE Theme_Customize
  ADD PRIMARY KEY (user_id,Theme);
ALTER TABLE contact
  ADD PRIMARY KEY (user_id,Cid);
ALTER TABLE personal_setup
  ADD PRIMARY KEY (user_id,emailuser);
ALTER TABLE Link_setup
  ADD PRIMARY KEY (user_id,Name);
ALTER TABLE ProjectDetail
  ADD PRIMARY KEY (user_id,projectname);
ALTER TABLE Experience
  ADD PRIMARY KEY (user_id,year,ogname);
ALTER TABLE skills
  ADD PRIMARY KEY (user_id,skill);
ALTER TABLE Licence
  ADD PRIMARY KEY (LKey);
ALTER TABLE Owner_Setup
  ADD PRIMARY KEY (AdminID);
ALTER TABLE Request_Manager
ADD PRIMARY KEY (NewRequest);
```

Foreign Keys

```
ALTER TABLE Licence
   ADD CONSTRAINT aboutus_setupfk
   FOREIGN KEY (user_id)
   REFERENCES admin_users (user_id);
ALTER TABLE admin_users
   ADD CONSTRAINT Licencefk
   FOREIGN KEY (LKey)
   REFERENCES Licence (LKey);
ALTER TABLE aboutus_setup
   ADD CONSTRAINT admin_usersfk
   FOREIGN KEY (user_id)
   REFERENCES admin_users (user_id);
ALTER TABLE basic_setup
       ADD CONSTRAINT basic_setupfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE basic_setup
       ADD CONSTRAINT basic_setup2Themefk
       FOREIGN KEY (Theme)
       REFERENCES Theme (Theme);
ALTER TABLE Theme_Customize
       ADD CONSTRAINT Theme_Customizefk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE Theme_Customize
       ADD CONSTRAINT Theme_Customize2Themefk
       FOREIGN KEY (Theme)
       REFERENCES Theme (Theme);
ALTER TABLE contact
       ADD CONSTRAINT contactfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE personal_setup
       ADD CONSTRAINT personal_setupfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE Link_setup
       ADD CONSTRAINT Link_setupfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE ProjectDetail
       ADD CONSTRAINT projectdetailfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE Experience
       ADD CONSTRAINT Experiencefk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE skills
       ADD CONSTRAINT skillsfk
       FOREIGN KEY (user_id)
       REFERENCES admin_users (user_id);
ALTER TABLE Owner_Setup
       ADD CONSTRAINT OwnerSetupFK
```

```
	FOREIGN KEY (NewRequest)
	REFERENCES Owner_Setup (AdminID);
```

Function

```
DROP FUNCTION LoginCheck;

CREATE FUNCTION FN_LoginCheck_user(@UserName VARCHAR(10),@Password VARCHAR(8))
RETURNS INT
AS
BEGIN
	DECLARE @PASSWORD_ VARCHAR(8);
	DECLARE @MATCH INT;

	SELECT @PASSWORD_ = Pass FROM Admininfo
	WHERE Name = @UserName;
	IF (@PASSWORD_ = @Password)
	BEGIN
		SET @MATCH = 1;
	END;
	ELSE
	BEGIN
		SET @MATCH = 0;
	END;
	RETURN @MATCH
END;
GO;

CREATE FUNCTION FN_LoginCheck_ADMIN(@UserName VARCHAR(10),@Password VARCHAR(8))
RETURNS INT
AS
BEGIN
	DECLARE @PASSWORD_ VARCHAR(8);
	DECLARE @MATCH INT;

	SELECT @PASSWORD_ = AdminPass FROM Owner_Setup
	WHERE AdminID = @UserName;
	IF (@PASSWORD_ = @Password)
	BEGIN
		SET @MATCH = 1;
	END;
	ELSE
	BEGIN
		SET @MATCH = 0;
	END;
	RETURN @MATCH
END;
GO;

CREATE FUNCTION FN_SEARCH_userNAME(@NAME VARCHAR(100))
RETURNS TABLE
AS
	RETURN (SELECT * FROM Admininfo
	WHERE Name LIKE @NAME)

GO;
```

```
CREATE FUNCTION FN_SEARCH_PROJECT(@INPUT VARCHAR(100))
RETURNS TABLE
AS
	RETURN (SELECT * FROM BasicInfo WHERE Experties LIKE '%' );

GO;

CREATE FUNCTION FN_SEARCH_Profession(@INPUT VARCHAR(100))
RETURNS TABLE
AS

	RETURN (SELECT * FROM UserInfo
	WHERE Profession LIKE @INPUT );

GO;


CREATE FUNCTION FN_ADDING_EXTRA_QOUTE(@VAL VARCHAR(MAX))
RETURNS VARCHAR(MAX)
AS
BEGIN
DECLARE @RET VARCHAR(MAX);
	SET @RET = CHAR(39)+@VAL+CHAR(39);
	RETURN (@RET);
END;
GO;
```

Procedures

```
-- Admin Info
CREATE PROCEDURE InsertRecord(
    @Name VARCHAR(250),
    @ID VARCHAR(10),
    @Pass VARCHAR(250),
    @Liscence_Key INT
) AS
BEGIN
    INSERT INTO Admininfo(NAME, ID, Pass, [Liscence KEY])
VALUES(@Name, @ID, @Pass, @Liscence_Key);
END;


-- AboutInfo
CREATE PROCEDURE InsertAboutInfo(
    @ID VARCHAR(10),
    @dob VARCHAR(250),
    @Heading TEXT,
    @Sub_Heading TEXT,
    @Short_Description TEXT,
    @Long_Description TEXT,
    @Website VARCHAR(250)
) AS
BEGIN
    INSERT INTO AboutInfo(
        ID,
        dob,
        Heading,
        [Sub Heading],
        [Short Description],
        [LONG Description],
        Website
    )
VALUES(
    @ID,
    @dob,
    @Heading,
    @Sub_Heading,
    @Short_Description,
    @Long_Description,
    @Website
);
END;

-- basic Info
CREATE PROCEDURE InsertbasicInfo(
    @ID VARCHAR(10),
    @Liscence_Key INT,
    @Job_title VARCHAR(250),
    @Job_Description TEXT,
    @Experties TEXT,
    @Avatar_Icon VARCHAR(250),
    @Theme VARCHAR(10)
) AS
```

```
BEGIN
    INSERT INTO BasicInfo(
        ID,
        [Liscence KEY],
        [Job title],
        [Job Description],
        Experties,
        [Avatar Icon],
        Theme
    )
VALUES(
    @ID,
    @Liscence_Key,
    @Job_title,
    @Job_Description,
    @Experties,
    @Avatar_Icon,
    @Theme
);
END;

-- Theme Info
CREATE PROCEDURE Themedata(
    @Theme VARCHAR(10),
    @Theme_Location VARCHAR(10),
    @Customize INT
) AS
BEGIN
    INSERT INTO ThemeSelection(
        Theme,
        [Theme Location],
        Customize
    )
VALUES(@Theme, @Theme_Location, @Customize);
END;

-- Project Detail
CREATE PROCEDURE ProjectDetailInsert(
    @ID VARCHAR(10),
    @Project_Name VARCHAR(250),
    @Pics_of_Project VARCHAR(250),
    @Link_of_Projet TEXT,
    @Description VARCHAR(250)
) AS
BEGIN
    INSERT INTO ProjectInfo(
        ID,
        [Project NAME],
        [Pics of Project],
        [Link of Project],
        Description
    )
VALUES(
    @ID,
    @Project_Name,
    @Pics_of_Project,
    @Link_of_Projet,
    @Description
```

```
);
END;

-- Insert History
CREATE PROCEDURE InsetHistroy(
    @UserID VARCHAR(10),
    @ActionPerform VARCHAR(100),
    @Log_ VARCHAR(100),
    @Before_log VARCHAR(100),
    @After_log VARCHAR(100),
    @Date_time VARCHAR(100)
) AS
BEGIN
    INSERT INTO Histroy
VALUES(
    @UserID,
    @ActionPerform,
    @Log_,
    @Before_log,
    @After_log,
    @Date_time
);
END;

-- USER Contact
CREATE PROCEDURE ContactDetail(
    @ID VARCHAR(10),
    @Contact_ID INT,
    @Name VARCHAR(250),
    @Email VARCHAR(250),
    @Subject TEXT,
    @Message TEXT
) AS
BEGIN
    INSERT INTO UserContact(
        ID,
        [Contact_ID],
        NAME,
        Email,
        SUBJECT,
        Message
    )
VALUES(
    @ID,
    @Contact_ID,
    @Name,
    @Email,
    @Subject,
    @Message
);
END;

 -- LinkDetails
CREATE PROCEDURE LinkDetailInsert(
    @ID VARCHAR(10),
    @Social_Networks VARCHAR(200),
    @URL VARCHAR(200)
) AS
```

```
BEGIN
    INSERT INTO LinkDetails(ID, [Social Network], URL)
VALUES(@ID, @Social_Networks, @URL);
END;

 -- Working Experience
CREATE PROCEDURE WorkingExperiencingInsert(
    @ID VARCHAR(10),
    @Category VARCHAR(250),
    @Job_Title VARCHAR(250),
    @Working_Since VARCHAR(250),
    @Organization_Name VARCHAR(250),
    @Working_Description TEXT
) AS
BEGIN
    INSERT INTO WorkingExperience(
        ID,
        Category,
        [Job Title],
        [Working Since],
        [Organization NAME],
        [Working Description]
    )
VALUES(
    @ID,
    @Category,
    @Job_Title,
    @Working_Since,
    @Organization_Name,
    @Working_Description
);
END;



 -- JobSkills
CREATE PROCEDURE JobSkillInsert(
    @ID VARCHAR(10),
    @Skills VARCHAR(250),
    @Points VARCHAR(250)
) AS
BEGIN
    INSERT INTO JobSkills(ID, Skills, Points)
VALUES(@ID, @Skills, @Points);
END;

 -- LiscenceDetail
CREATE PROCEDURE LisceneDetailInsert(
    @ID VARCHAR(10),
    @Liscence_Key INT,
    @Validity VARCHAR(12),
    @Issue_Date VARCHAR(12),
    @Liscenece_Status INT
) AS
BEGIN
    INSERT INTO LisceneceDetail(
        ID,
```

```
        [Liscence KEY],
        Validity,
        [Issue DATE],
        [Liscenece STATUS]
    )
VALUES(
    @ID,
    @Liscence_Key,
    @Validity,
    @Issue_Date,
    @Liscenece_Status
);
END;

 -- OwnerInfo
CREATE PROCEDURE OwnerInfoInsert(
    @AdminID VARCHAR(10),
    @AdminPass VARCHAR(20),
    @Request_Recieved VARCHAR(10),
    @Contact_No VARCHAR(12),
    @UserName VARCHAR(20),
    @Email VARCHAR(30)
) AS
BEGIN
    INSERT INTO OwnerInfo(
        AdminID,
        AdminPass,
        [Request Recieved],
        [Contact NO],
        UserName,
        Email
    )
VALUES(
    @AdminID,
    @AdminPass,
    @Request_Recieved,
    @Contact_No,
    @UserName,
    @Email
);
END;
-- Request
CREATE PROCEDURE RequestInfoInsert(
    @AdminID VARCHAR(10),
    @Request_Recieved VARCHAR(10),
    @Send_Request VARCHAR(10)
) AS
BEGIN
    INSERT INTO Request(
        AdminID,
        [Request Recieved],
        [Send Request]
    )
VALUES(
    @AdminID,
    @Request_Recieved,
    @Send_Request
);
```

```
END;

 -- CustomiztionTheme
CREATE PROCEDURE CustomizeInsert(
    @Theme VARCHAR(10),
    @ID VARCHAR(10),
    @Font_Color VARCHAR(6),
    @Background_Color VARCHAR(6),
    @fontStyle VARCHAR(250),
    @fontSize INT
) AS
BEGIN
    INSERT INTO CustomizeTheme(
        Theme,
        ID,
        [Font Color],
        [Background Color],
        fontStyle,
        fontSize
    )
VALUES(
    @Theme,
    @ID,
    @Font_Color,
    @Background_Color,
    @fontStyle,
    @fontSize
);
END;


-- USER Info
CREATE PROCEDURE UserInfoInsert(
    @ID VARCHAR(10),
    @Profile_Pic VARCHAR(250),
    @Name VARCHAR(250),
    @Wallpaper VARCHAR(200),
    @Pofession VARCHAR(200),
    @Adress TEXT,
    @Contact_No VARCHAR(200),
    @Email VARCHAR(200)
) AS
BEGIN
    INSERT INTO UserInfo(
        ID,
        Profile_Pic,
        NAME,
        Wallpaper,
        Profession,
        Adress,
        [Contact NO],
        Email
    )
VALUES(
    @ID,
    @Profile_Pic,
    @Name,
    @Wallpaper,
```

```
    @Pofession,
    @Adress,
    @Contact_No,
    @Email
);
END;
-- Insert Complete
---------------------------------Gobal-------------------------------------------------
-------------


-- Search
CREATE PROCEDURE _Search_WebVeiw(@Input VARCHAR(MAX), @SortBy VARCHAR(MAX), @Table
VARCHAR(MAX)) AS
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'SELECT * FROM '+@Table+' where '+@SortBy+' like
'+char(39)+'%'+@Input+char(39)+'%';
              Exec (@SQL);
END



-- Main Insert
CREATE PROCEDURE PN_INSERT_GOBAL @Tab VARCHAR(MAX), @val VARCHAR(MAX), @COL VARCHAR(MAX)
AS
BEGIN
     DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'Insert Into ' + @Tab + '(' + @COL + ') Values (' + @val + ')';
              Exec (@SQL);
END

-- SELECT
CREATE PROCEDURE PN_SELECT_GOBAL @Tab VARCHAR(MAX) AS
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'SELECT * FROM '+@Tab;
              Exec (@SQL);
END

-- UDDATE
CREATE PROCEDURE PN_UPDATE_GOBAL @Tab VARCHAR(MAX),@INPUT VARCHAR(MAX) AS
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'UDDATE '+@Tab+'SET '+@INPUT;
              Exec (@SQL);
END

-- DELETE
CREATE PROCEDURE PN_DELETE_GOBAL @Tab VARCHAR(MAX),@INPUT VARCHAR(MAX) AS
```

```
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'DELETE FROM'+@Tab+'WHERE '+@INPUT;
              Exec (@SQL);
END

-- TRUCAT
CREATE PROCEDURE PN_TRUCAT_GOBAL @Tab VARCHAR(MAX),@INPUT VARCHAR(MAX) AS
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'TRUNCATE  FROM'+@Tab;
              Exec (@SQL);
END

-- DROP
CREATE PROCEDURE PN_DROP_GOBAL @TYPE VARCHAR(MAX),@INPUT VARCHAR(MAX) AS
BEGIN
    DECLARE
        @SQL VARCHAR(MAX)
    SELECT
        @SQL = 'DROP '+@TYPE+' '+@INPUT;
              Exec (@SQL);
END
```

Triggers

```
--Aboutus Setup
CREATE TRIGGER History_log ON
aboutus_setup INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(AB_.dob, AB_.heading, AB_.longdesc, AB_.shortdesc, AB_.subheading,
AB_.website),
            CONCAT(INS.dob, INS.heading, INS.longdesc, INS.shortdesc, INS.subheading,
INS.website),
            GETDATE()
                     FROM INSERTED INS, aboutus_setup AB_ WHERE INS.user_id = AB_.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(AB_.dob, AB_.heading, AB_.longdesc, AB_.shortdesc, AB_.subheading,
AB_.website),
            CONCAT(INS.dob, INS.heading, INS.longdesc, INS.shortdesc, INS.subheading,
INS.website),
            GETDATE()
                     FROM DELETED INS, aboutus_setup AB_ WHERE INS.user_id = AB_.user_id
    END;
END

-- BAsic Setup
CREATE TRIGGER HistoryBasicSet ON
basic_setup INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(BS_.LKey, BS_.title, BS_.description, BS_.keyword, BS_.icon,
BS_.Theme),
            CONCAT(INS.LKey, INS.title, INS.description, INS.keyword, INS.icon,
INS.Theme),
            GETDATE()
                     FROM INSERTED INS, basic_setup BS_ WHERE INS.user_id = BS_.user_id

    END;
```

```
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(BS_.LKey, BS_.title, BS_.description, BS_.keyword, BS_.icon,
BS_.Theme),
            CONCAT(INS.LKey, INS.title, INS.description, INS.keyword, INS.icon,
INS.Theme),
            GETDATE()
                    FROM DELETED INS, basic_setup BS_ WHERE INS.user_id = BS_.user_id
    END;
END

--Admin USer

CREATE TRIGGER HistoryAdminUser ON
admin_users INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
             INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(AU.username, AU.user_pass, AU.Lkey),
                    CONCAT(INS.username, INS.user_pass, INS.Lkey),
            GETDATE()
                    FROM INSERTED INS, admin_users AU WHERE INS.user_id = AU.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(AU.username, AU.user_pass, AU.Lkey),
                    CONCAT(INS.username, INS.user_pass, INS.Lkey),
            GETDATE()
                    FROM DELETED INS, admin_users AU WHERE INS.user_id = AU.user_id
    END;
END

--Contact

CREATE TRIGGER HistoryContact ON
contact INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
             INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(CO.Cid, CO.cname, CO.cemail,CO.csubject, CO.cmessage),
                    CONCAT(INS.Cid, INS.cname, INS.cemail,INS.csubject, INS.cmessage),
```

```
            GETDATE()
                    FROM INSERTED INS, contact CO WHERE INS.user_id = CO.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(CO.Cid, CO.cname, CO.cemail,CO.csubject, CO.cmessage),
                    CONCAT(INS.Cid, INS.cname, INS.cemail,INS.csubject, INS.cmessage),
            GETDATE()
                    FROM DELETED INS, contact CO WHERE INS.user_id = CO.user_id
    END;
END

--Personal Setup

CREATE TRIGGER HistoryPersonalSet ON
personal_setup INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
             INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(PS.profilepic, PS.name, PS.homewallpaper,PS.professions,
PS.location,PS.mobile, PS.emailuser),
                    CONCAT(INS.profilepic, INS.name, INS.homewallpaper,INS.professions,
INS.location,INS.mobile, INS.emailuser),
            GETDATE()
                    FROM INSERTED INS, personal_setup PS WHERE INS.user_id = PS.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(PS.profilepic, PS.name, PS.homewallpaper,PS.professions,
PS.location,PS.mobile, PS.emailuser),
                    CONCAT(INS.profilepic, INS.name, INS.homewallpaper,INS.professions,
INS.location,INS.mobile, INS.emailuser),
            GETDATE()
                    FROM DELETED INS, personal_setup PS WHERE INS.user_id = PS.user_id
    END;
END

--Link Setup

CREATE TRIGGER HistoryLinkSet ON
Link_setup INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
```

```
        BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                        CONCAT(GETDATE(),''),
              CONCAT(LS.Name, LS.Link),
                        CONCAT(INS.Name, INS.Link),
              GETDATE()
                        FROM INSERTED INS, Link_setup LS WHERE INS.user_id = LS.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                        CONCAT(GETDATE(),''),
              CONCAT(LS.Name, LS.Link),
                        CONCAT(INS.Name, INS.Link),
              GETDATE()
                        FROM DELETED INS, Link_setup LS WHERE INS.user_id = LS.user_id
    END;
END

--Theme Customize

CREATE TRIGGER HistoryThemeCustom ON
Theme_Customize INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                        CONCAT(GETDATE(),''),
              CONCAT(TC.Theme, TC.fcolor, TC.bcolor,TC.fontStyle, TC.fontSize),
                        CONCAT(INS.Theme, INS.fcolor, INS.bcolor,INS.fontStyle,
INS.fontSize),
              GETDATE()
                        FROM INSERTED INS, Theme_Customize TC WHERE INS.user_id = TC.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                        CONCAT(GETDATE(),''),
              CONCAT(TC.Theme, TC.fcolor, TC.bcolor,TC.fontStyle, TC.fontSize),
                        CONCAT(INS.Theme, INS.fcolor, INS.bcolor,INS.fontStyle,
INS.fontSize),
              GETDATE()
                        FROM DELETED INS, Theme_Customize TC WHERE INS.user_id = TC.user_id
    END;
END

--ProjectDetail

CREATE TRIGGER HistoryProjectDetail ON
ProjectDetail INSTEAD OF
```

```
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(PD.projectname, PD.projectpic, PD.projectlink,PD.projectdesc),
                     CONCAT(INS.projectname, INS.projectpic,
INS.projectlink,INS.projectdesc),
            GETDATE()
                     FROM INSERTED INS, ProjectDetail PD WHERE INS.user_id = PD.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(PD.projectname, PD.projectpic, PD.projectlink,PD.projectdesc),
                     CONCAT(INS.projectname, INS.projectpic,
INS.projectlink,INS.projectdesc),
            GETDATE()
                     FROM DELETED INS, ProjectDetail PD WHERE INS.user_id = PD.user_id
    END;
END

--Experience

CREATE TRIGGER HistoryExperience ON
Experience INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
              INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(EX.category, EX.title, EX.year,EX.ogname, EX.workdesc),
                     CONCAT(INS.category, INS.title, INS.year,INS.ogname, EX.workdesc),
            GETDATE()
                     FROM INSERTED INS, Experience EX WHERE INS.user_id = EX.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                     CONCAT(GETDATE(),''),
            CONCAT(EX.category, EX.title, EX.year,EX.ogname, EX.workdesc),
                     CONCAT(INS.category, INS.title, INS.year,INS.ogname, EX.workdesc),
            GETDATE()
                     FROM DELETED INS, Experience EX WHERE INS.user_id = EX.user_id
    END;
END
```

```
--Skills

CREATE TRIGGER HistorySkills ON
skills INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
             INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(SK.skill, SK.score),
                    CONCAT(INS.skill, INS.score),
            GETDATE()
                    FROM INSERTED INS, skills SK WHERE INS.user_id = SK.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(SK.skill, SK.score),
                    CONCAT(INS.skill, INS.score),
            GETDATE()
                    FROM DELETED INS, skills SK WHERE INS.user_id = SK.user_id
    END;
END

--Liscence
CREATE TRIGGER HistoryLiscene ON
Licence INSTEAD OF
DELETE,
UPDATE AS
BEGIN
    IF EXISTS (SELECT * FROM INSERTED)
       BEGIN
             INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log,
Date_time) SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(LS.LKey, LS.ExpireDate, LS.IssueDate,LS.LStatus),
                    CONCAT(INS.LKey, INS.ExpireDate, INS.IssueDate,LS.LStatus),
            GETDATE()
                    FROM INSERTED INS, Licence LS WHERE INS.user_id = LS.user_id

    END;
       ELSE IF EXISTS(SELECT * FROM DELETED)
       BEGIN
         INSERT INTO History (UserID,ActionPerform,Log_,Before_log,After_log, Date_time)
SELECT INS.user_id,'DEL',
                    CONCAT(GETDATE(),''),
            CONCAT(LS.LKey, LS.ExpireDate, LS.IssueDate,LS.LStatus),
                    CONCAT(INS.LKey, INS.ExpireDate, INS.IssueDate,LS.LStatus),
            GETDATE()
                    FROM DELETED INS, Licence LS WHERE INS.user_id = LS.user_id
    END;
END
```

```
-- ADD ID TO ALL TABLE
CREATE TRIGGER TR_ID_ADDING2ALL_COLUMN ON
Licence
AFTER INSERT
AS
BEGIN
	DECLARE @ID VARCHAR(10);
	DECLARE @LKEY VARCHAR(10);

	SELECT @ID = I.user_id, @LKEY = I.LKey FROM INSERTED I;

	EXEC InsertRecord '',@ID,'',@LKEY;
	EXEC InsertAboutInfo @ID,'','','','','','';
	EXEC ProjectDetailInsert @ID,'','','','';
	EXEC LinkDetailInsert @ID,'','';
	EXEC WorkingExperiencingInsert @ID,'','','','','';
	EXEC JobSkillInsert @ID,'','';

END

update aboutus_setup set dob = '25/5/2021' where user_id = 'User07'
select * from [dbo].[aboutus_setup]
```

Experience of Project

It was really good because we are working on a new platform MVC which make us do something new and that makes the project very interesting to work, it was a quite difficult but still worth it

Proposed benefit in system

The project is giving all the benefit that is proposed in the project proposal and its in the system

Final Comment

The system can provide a user to create its very own portfolio with a good theme and everything.