بسم الله الرحمن الرحیم

پروژه آزمایشگاه پایگاه داده

سامانه مدیریت خوابگاه

اعضای گروه:

عالیه سرابندی۹۵۵۲۲۰۹۴

ترمه طباطبایی ۹۵۵۲۱۳۰۶

فهرست

DOMAIN DESCRIPTION(شرح دامنه):

این پروژه سامانه‌ای برای مدیریت خوابگاه می‌باشد. در این سامانه دانشجوها و کارکنان خوابگاه عضو شده و هر یک پروفایلی خواهند داشت. اطلاعات بلوک‌ها، طبقات و اتاق‌های خوابگاه، مکان‌های عمومی، و همچنین لوازم موجود در هر طبقه در این سامانه موجود هستند و برای دانشجویان و کارکنان قابل مشاهده می‌باشند. اطلاعات جانبی نیز راجع به این موارد قابل دسترس است از جمله ظرفیت اتاق‌ها، مسئول اماکن عمومی، ... همچنین امکاناتی نظیر امکان اسکان دهی به دانشجویان، امکان تعیین مسئول اماکن عمومی، امکان شرح وضعیت لوازم،... نیز موجود می‌باشند.

ENTITIES (موجودیت‌ها):

- **کاربر**

کاربران به دو دسته‌ی دانشجو و کارمند تقسیم می‌شوند که یکسری ویژگی‌های مشترک دارند از جمله کد ملی، جنسیت، نام، ایمیل، تاریخ تولد، شماره تلفن و پسورد ورود به سامانه.

- **دانشجو**

دانشجو نوعی کاربر است که اطلاعات وی در پروفایلی که به او اختصاص دارد قابل مشاهده است و اطلاعاتی که اضافه بر اطلاعات کلی کاربر دارد نوع تحصیل (شبانه/روزانه)، سال ورودی، شماره‌ی دانشجویی، شهر، سطح تحصیلی و رشته می باشند. زمان ورودی و خورجی وی نیز سبط می‌شود و اگر خواهد خارج از زمان مقبول وارد خوابگاه شود باید درخواست آن را بدهد. وی می تواند گزارش وضع لوازم و درخواست‌هایی که در بخش بعد به آن‌ها می‌پردازیم را نیز در سامانه انجام دهد.

- **کارمند**

اطلاعاتی که این موجودیت اضافه بر اطلاعات کاربر دارد شغل و شماره‌ی کارمند می‌باشد. کارمند می‌تواند درخواست‌های دانشجویان را مطابق با درسطح

دسترسی شغلش مشاهده کرده و تعیید یا رد کند. زمان ورودی و خروجی کارمند نیز در سامانه سبت می‌شود.

- **مقطع تحصیلی**

هر دانشجو یک مقطع تحصیلی دارد که آن را به عنوان یکی از موجودیت ها در نظر میگیریم که سطح تحصیل دانشجو را مشخص میکند و شامل کارشناسی، دکتری، ... می باشد.

- **رشته**

رشته‌ی تحصیلی نیز یکی از موجودیت‌های ماست و ما را از رشته‌ی دانشجو مطلع می‌سازد. نام و ای دی رشته را از این موجودیت می‌توان یافت که ای در رشته برای کارکردن با باقی سامانه‌های دانشگاه کاربرد دارد.

- **شهر**

شهر یا شهرستانی است که دانشجو ساکن ان می‌باشد و استانی که در آن واقع است را نیز شامل می‌شود.

- **شغل**

شغل اطلاعاتی مانند سطح دسترسی و درامد ماهانه را برای کارمندان برای ما شرح می‌دهد.

- **دانشگاه**

این سامانه برای خوابگاه های دانشگاه ها می باشد، از این رو اطلاعات مربوط به دانشگاه و همچنین خوابگاه‌های مختلف متعلق به دانشگاه را با این موجودیت و روابط آن شرح می‌دهیم.

- **خوابگاه**

خوابگاه ها هرکدام از چندین بلوک و اماکن عمومی ساخته شدند و دانشجویانی در آن‌ها ساکن هستند. جدا از این موارد، شماره‌ی تلفن خوابکاه، این که دخترانه است یا پسرانه، آدرسش و شمارش نیز در این موجودیت ذکر شده اند.

- **بلوک**

بلوک‌ها هر کدام از چندین طبقه ساخته شده‌اند.

- **طبقه**

هر طبقه تعدادی لوازم عمومی دارد و تعدادی اتاق.

- **اتاق**

هر اتاق شماره، میزان ظرفیت، ظرفیت خالی، هزینه، طبقه و بلوک دارد و دانشجویان می‌توانند در صورت وجود ظرفیت خالی برای ان درخواست اسکان دهند.

- **أموال عمومی**

هر طبقه دارای اموال عمومی از جمله سشوار، اتو، یخچال،... می‌باشد.

- **اماکن عمومی**

خوابگاه‌ها هرکدام شامل مکان‌های عمومی مانند کتابخنه، باشگاه، سایت، نمازخانه، آرایشگاه، کافه،... هستند که هر یک از آن‌ها مسئولی از میان دانشجویان و نوع و ساعت کار دارند.

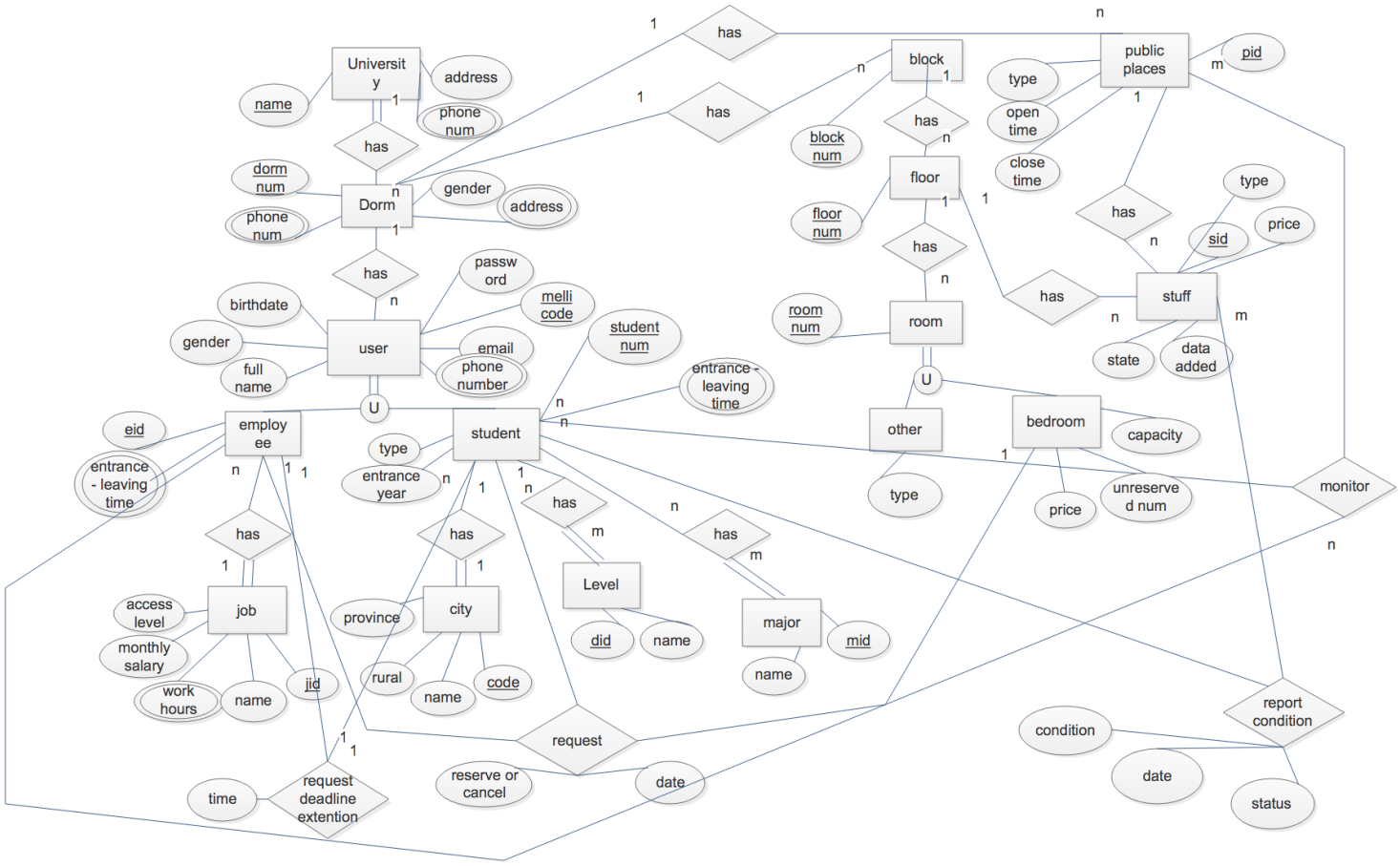
گزارش‌ها:

- کاربر قادر است اطلاعات شخصی خود را در پروفایلش ببیند و به روز رسانی کند.

- کاربر می تواند خوابگاه‌های دانشگاه را به صورت لیست مشاهده کند و اطلاعات کلی ان ها را نیز ببیند
- کاربر قادر است اطلاعات بیشتری راجع به خوابگاه منتخب بیابد از جمله بلوک‌ها و و اماکن عمومی ان.
- کاربر می تواند لیست اتاق های خوابگاه منتخب را همرا ه با مشخصاتشان مانند ظرفیت،... ببیند.
- کاربر می تواند طلاعات اماکن عمومی مانند ساعت کار آن‌ها، نام و شماره‌ی تماس مسئول در صورت وجود و .. را مشاهده کند.
- دانشجو می‌تواند درخواست رزرو برای اتاق‌هایی که ظرفیت خالی دارند بدهد.
- دانشجو می‌تواند درخواست کنسل شدن رزرویشن خود را دهد.
- دانشجو می‌تواند درخواست مسئول شدن برای اماکن عمومی را دهد.
- دانشجو می‌تواند درخواست تمدید زمان مجاز برای بیرون ماندن را دهد.
- دانشجویان می‌توانند شرایت لوازم عمومی را دهند.
- کارمندان متناسب با میزان دسترسی شغلشان می توانند به درخواست‌های دانشجویان جواب دهند (مثلان آبدارچی می تواند وضعیت لوازم را مشاهده کرده و تغییر دهد ولی نمی تواند درخواست رزرو اتاق را مشاهده کرده و قبول یا رد کند).
- کارمندان متناسب با شغل می توانند لیست‌های مختلف (لوازم، وضعیت اتاق‌ها، ...) را تغییر دهند.

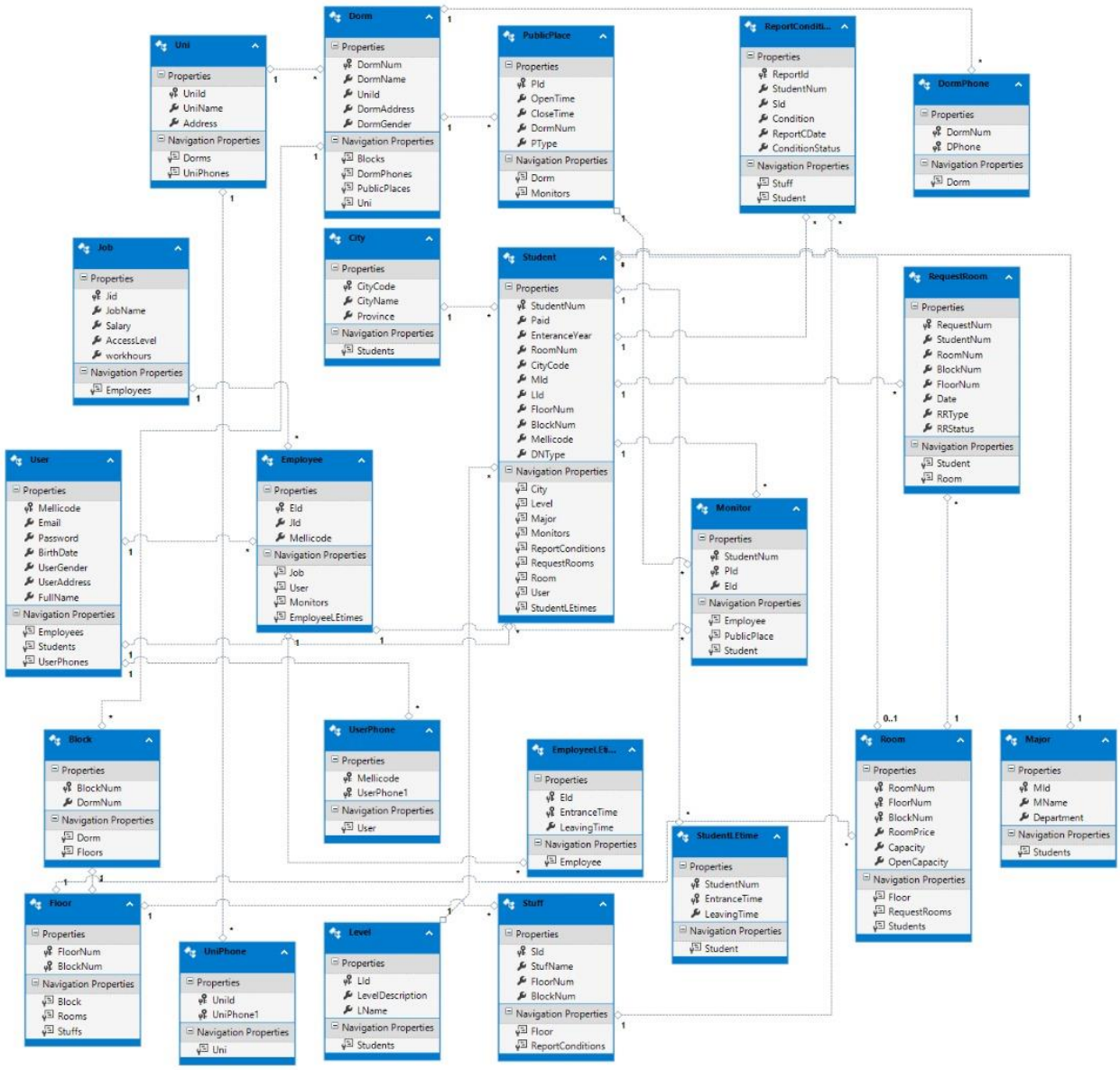
نمودار ER:

شکل۱ – نمودار ER



در کار نهایی ما برخی از موارد غیر الزامی از جمله فیلد rural برای شهر و بخش اتاق‌های دیگر که هیچ کاربردی نداشت را حذف کردیم و اتاق و اتاق خواب را ادغام کردیم.

نمودار ارجاع:



شکل۲ – نمودار ارجاع

نرمال سازی:

ما از اول کار با در نظر داشتن شرایت نرمال بودن، جداول نمودار ER را ساختیم و در ادامه نمونه‌های برای نشان دادن این موارد می‌آوریم.

سطح ۱:

موجودیت یا جدولی در فرم اول نرمال است که تمامی المان‌های اطلاعاتی ان یکتا باشند. بنابر این ما جدول جداگانه‌ای برای ذخیر شماره تلفن‌ها (برای هر موجودی جدا) ساختیم زیرا امکان وجود چندین شماره برای هر مورد از ان موجودیت وجود داشت.

University

	Domain	**Primary key**	**Foreign key**	**Nullable**	**Constraint**
Unid	Integer()	✓			
Phone num	Integer()	✓			

سطح ۲:

برای نرمال بودن در سطح ۲ باید موجودیت هم در سطح ۱ نرمال باشد و هم تمامی اتریبیوت‌های غیر کلیدی ان وابستگی تابعی به تمام کلید اصلی موجودیت داشته باشند نه به بخشی از ان. این مورد برای موجودیت‌هایی است که کلید مرکب دارند. مثالی از این مورد جدول زمان ورود و خروج کارمندان می‌باشد که کلید مرکب دارد زیرا یک کارمند می تواند چندین ورودی در جدول داشته باشد و چندین کارمند در یک تاریخ و زمان امکان است وارد شوند و یک کارمند در یک تاریخ و زمان فقط یک بار ظاهر می شود و نیازی به مورد دیگری هم نیست که اضافه شود به کلید.

Employee entrance – leaving time

	Domain	Primary key	Foreign key	Nullable	Constraint
Eid	Integer()	✓	✓		
Entrance time	DATETIME	✓			
Leaving time	DATETIME				

سطح ۳:

موجودیت یا جدولی در فرم سوم نرمال است که در دو فرم قبا نرمال باشد و تمامی ایتم‌های غیر کلید ان وابستگی تابعی به کلید اصلی داشته باشند نه به یک ایتم غیر کلیدی. برای مثال در جدول زیر هیچ ایتمی به ایتم‌های دیگر غیر کلیدی وابسته نیست.

Public places

	Domain	Primary key	Foreign key	Nullable	Constraint
PId	Integer()	✓			
Dorm num	Integer()		✓		
Type	Nvchar(100)				
Open time	DATETIME				
Close time	DATETIME				

نرمال سازی تا سطح ۳ به طور کامل انجام شده است.

پیاده سازی:

ما با c#, sql server و windows form برنامه را پیاده سازی کردیم.

کد ساخت جدول‌ها:

```
CREATE TABLE [dbo].[UniPhone] (
    [UniId] INT           NOT NULL,
    [UnPhone]          NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([UniId] ASC, [Phone] ASC),
    UNIQUE NONCLUSTERED ([UnPhone] ASC),
    FOREIGN KEY ([UniId]) REFERENCES [dbo].[Uni] ([UniId])
);
```

```
CREATE TABLE [dbo].[Users] (
    [Mellicode] INT            NOT NULL,
    [Email]  NVARCHAR (100) NOT NULL,
    [Password]  NVARCHAR (100) NOT NULL,
    [BirthDate] DATE           NOT NULL,
    [UserGender]    NVARCHAR (20)  NOT NULL,
    [UserAddress]   NVARCHAR (300) NOT NULL,
    [FullName]      NVARCHAR (200) NOT NULL,

    PRIMARY KEY CLUSTERED ([Mellicode] ASC),

    CHECK ([UserGender]='F' OR [UserGender]='M')
);
```

```
CREATE TABLE [dbo].[UserPhone] (
    [Mellicode] INT           NOT NULL,
    [UserPhone]       NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([Mellicode] ASC, [Phone] ASC),
    UNIQUE NONCLUSTERED ([UserPhone] ASC),
    FOREIGN KEY ([Mellicode]) REFERENCES [dbo].[Users] ([Mellicode])
);
```

```
CREATE TABLE [dbo].[Student] (
    [StudentNum]     INT  NOT NULL,
    [Paid]          BIT  NOT NULL,
```

```
    [EnteranceYear] DATE NOT NULL,
    [RoomNum]        INT  NULL,
    [CityCode]      INT  NOT NULL,
    [MId]     INT  NOT NULL,
    [LId]     INT  NOT NULL,
    [FloorNum]       INT  NULL,
    [BlockNum]       INT  NULL,
    [Mellicode]     INT  NOT NULL,
    [DNType] NVARCHAR(50) NOT NULL,
    PRIMARY KEY CLUSTERED ([StudentNum] ASC),
    FOREIGN KEY ([RoomNum], [BlockNum], [FloorNum]) REFERENCES [dbo].[Room] ([RoomNum],
[BlockNum], [FloorNum]),
    FOREIGN KEY ([Mellicode]) REFERENCES [dbo].[Users] ([CodeMelli]),
    FOREIGN KEY ([CityCode]) REFERENCES [dbo].[City] ([CityCode]),
    FOREIGN KEY ([MId]) REFERENCES [dbo].[Major] ([MId]),
    FOREIGN KEY ([LId]) REFERENCES [dbo].[Level] ([LId]),

       CHECK ([DNType]='day' OR [DNType]='night')
);



CREATE TABLE [dbo].[Floor] (
    [FloorNum] INT NOT NULL,
    [BlockNum] INT NOT NULL,
    [DormNum] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([BlockNum] ASC, [FloorNum] ASC, [DormNum] ASC),
    UNIQUE NONCLUSTERED ([FloorNum] ASC),
    CONSTRAINT [fk_block_floor] FOREIGN KEY ([BlockNum]) REFERENCES [dbo].[Block]
([BlockNum]) ON DELETE CASCADE
);



CREATE TABLE [dbo].[Room] (
    [RoomNum]  INT NOT NULL,
    [FloorNum] INT NOT NULL,
    [BlockNum] INT NOT NULL,
    [RoomPrice]  INT            NOT NULL,
    [Capacity] INT            NOT NULL,
    [OpenCapacity] INT            NOT NULL,
CONSTRAINT [PK_Room] PRIMARY KEY CLUSTERED ([RoomNum] ASC, [BlockNum] ASC,[FloorNum]
ASC),
    CONSTRAINT [fk_room_floor] FOREIGN KEY ([BlockNum], [FloorNum]) REFERENCES
[dbo].[Floor] ([BlockNum], [FloorNum]) ON DELETE CASCADE,

);



CREATE TABLE [dbo].[Level] (
    [LId]   INT            NOT NULL,
    [LevelDescription] NVARCHAR (100) NOT NULL,
    [LName] NVARCHAR(100) NOT NULL,
    PRIMARY KEY CLUSTERED ([LId] ASC),
    UNIQUE NONCLUSTERED ([Description] ASC)
);
```

```
CREATE TABLE [dbo].[PublicPlace] (
    [PId]   INT      NOT NULL,
    [OpenTime] TIME (7) NOT NULL,
       [PType] NVARCHAR(100)  NOT NULL,

    [CloseTime] TIME (7) NOT NULL,
    [DormNum] INT      NOT NULL,
    PRIMARY KEY CLUSTERED ([PId] ASC),
    FOREIGN KEY ([DormNum]) REFERENCES [dbo].[Dorm] ([DormNum])
);

CREATE TABLE [dbo].[Major] (
    [MId]   INT            NOT NULL,
    [MName] NVARCHAR (100) NOT NULL,
    [Department] NVARCHAR(100) NOT NULL,
    PRIMARY KEY CLUSTERED ([MId] ASC),
    UNIQUE NONCLUSTERED ([MName] ASC)
);



CREATE TABLE [dbo].[Stuff] (
    [SId]   INT            NOT NULL,
    [StufName] NVARCHAR (150) NOT NULL,
    [FloorNum]      INT            NOT NULL,
    [BlockNum]      INT            NOT NULL,
 PRIMARY KEY CLUSTERED ([SId] ASC),
    CONSTRAINT [fk_stuff_block_floor] FOREIGN KEY ([BlockNum], [FloorNum]) REFERENCES
[dbo].[Floor] ([BlockNum], [FloorNum]) ON DELETE CASCADE
);



CREATE TABLE [dbo].[DormPhone] (
    [DormNum] INT           NOT NULL,
    [DPhone]         NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([DormNum] ASC, [DPhone] ASC),
    UNIQUE NONCLUSTERED ([DPhone] ASC),
    FOREIGN KEY ([DormNum]) REFERENCES [dbo].[Dorm] ([DormNum])
);




CREATE TABLE [dbo].[Employee] (
    [EId] INT NOT NULL,
    [JId]      INT NOT NULL,
    [Mellicode]  INT NOT NULL,
    PRIMARY KEY CLUSTERED ([EId] ASC),
    FOREIGN KEY ([Mellicode]) REFERENCES [dbo].[Users] ([Mellicode]),
```

```
    FOREIGN KEY ([JId]) REFERENCES [dbo].[Job] ([JId])
);
```

```
CREATE TABLE [dbo].[City] (
    [CityCode]    INT           NOT NULL,
    [CityName]    NVARCHAR (100) NOT NULL,
    [Province] NVARCHAR(100)           NOT NULL,
    PRIMARY KEY CLUSTERED ([CityCode] ASC),
    UNIQUE NONCLUSTERED ([CityName] ASC)
);

CREATE TABLE [dbo].[Uni] (
    [UniId]   INT           NOT NULL,
    [UniName] NVARCHAR (150) NOT NULL,
    [UniAddress]        NVARCHAR (500) NOT NULL,
    PRIMARY KEY CLUSTERED ([UniId] ASC),
    UNIQUE NONCLUSTERED ([UniName] ASC)
);
```

```
CREATE TABLE [dbo].[Dorm] (
    [DormNum]   INT           NOT NULL,
    [DormName] NVARCHAR (150) NOT NULL,
    [UniId]  INT           NOT NULL,
    [Address]       NVARCHAR (500) NOT NULL,
    [DormGender] NCHAR(10) NOT NULL,
    PRIMARY KEY CLUSTERED ([DormNum] ASC),
    UNIQUE NONCLUSTERED ([DormName] ASC),
    UNIQUE NONCLUSTERED ([Address] ASC),
    CONSTRAINT [fk_dorm_uni] FOREIGN KEY ([UniId]) REFERENCES [dbo].[Uni] ([UniId]) ON
DELETE CASCADE,
       CHECK ([DormGender]='M' or [DormGender]='F')
);
```

```
CREATE TABLE [dbo].[Block] (
    [BlockNum]     INT           NOT NULL,
    [DormNum] INT           NOT NULL,
    PRIMARY KEY CLUSTERED ([BlockNum] ASC,[DormNum] ASC),

    FOREIGN KEY ([DormNum]) REFERENCES [dbo].[Dorm] ([DormNum])
);

CREATE TABLE [dbo].[Job] (
    [Jid]     INT           NOT NULL,
```

```
    [JobName] NVARCHAR(100)          NOT NULL,
    [Salary] INT NULL,
    [AccessLevel] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([Jid] ASC),


    CONSTRAINT [CK_accesslevel_Column] CHECK ([AccessLevel] = 1 or [AccessLevel] = 2 or
[AccessLevel] = 3)
);




CREATE TABLE [dbo].[EmployeeLEtime] (
    [EId] INT           NOT NULL,

    [EntranceTime]         DATETIME NOT NULL,
       [LeavingTime]         DATETIME NOT NULL,

    PRIMARY KEY CLUSTERED ( [EId] ASC, [EntranceTime] ASC),
    UNIQUE NONCLUSTERED ([Phone] ASC),
    CONSTRAINT [fk_et] FOREIGN KEY ([EId]) REFERENCES [dbo].[Dorm] ([EId]) ON DELETE
CASCADE

);


CREATE TABLE [dbo].[StudentLEtime] (
    [StudentNum] INT           NOT NULL,

    [EntranceTime]         DATETIME NOT NULL,
       [LeavingTime]         DATETIME NOT NULL,

    PRIMARY KEY CLUSTERED ( [EId] ASC, [EntranceTime] ASC),
    UNIQUE NONCLUSTERED ([Phone] ASC),
    CONSTRAINT [fk_st] FOREIGN KEY ([StudentNum]) REFERENCES [dbo].[Student]
([StudentNum]) ON DELETE CASCADE

);


CREATE TABLE [dbo].[ReportCondition] (
    [ReportId] INT           NOT NULL,

    [StudentNum]         int NOT NULL,
       [SId]         int NOT NULL,
       [Condition]         NVARCHAR(100)  NOT NULL,

       [ReportCDate]         DATETIME NOT NULL,
       [ConditionStatus]         NVARCHAR(100)  (50) NOT NULL,

    PRIMARY KEY CLUSTERED ([ReportId] ASC),

 CONSTRAINT [fk_rcsn]   FOREIGN KEY ([StudentNum]) REFERENCES [dbo].[Student]
([StudentNum]) ON DELETE CASCADE
,
```

```
CONSTRAINT [fk_rcsi] FOREIGN KEY ([SId]) REFERENCES [dbo].[Stuff] ([SId]) ON DELETE
CASCADE


);




CREATE TABLE [dbo].[RequestDeadlineExtension] (
    [StudentNum] INT           NOT NULL,
    [DEDate]         DATE NOT NULL,
    [EId]         int NOT NULL,
    [DesiredTime]         DATETIME NOT NULL,
    [DeadlineStatus]         int NULL,

    PRIMARY KEY CLUSTERED ([EId] ASC, [StudentNum] ASC),

   CONSTRAINT [fk_rdesn] FOREIGN KEY ([StudentNum]) REFERENCES [dbo].[Student]
([StudentNum]) ON DELETE CASCADE,
CONSTRAINT [fk_rdep] FOREIGN KEY ([PId]) REFERENCES [dbo].[PublicPlace] ([PId]) ON DELETE
CASCADE,
CHECK ([DeadlineStatus]='Accept' or [DeadlineStatus]='Reject' or [DeadlineStatus]=NULL)


);


CREATE TABLE [dbo].[Monitor] (
    [StudentNum] INT           NOT NULL,
    [PId]         int NOT NULL,
    [EId]         int NOT NULL,

    PRIMARY KEY CLUSTERED ([PId] ASC, [StudentNum] ASC),

    FOREIGN KEY ([StudentNum]) REFERENCES [dbo].[Student] ([StudentNum]),
FOREIGN KEY ([PId]) REFERENCES [dbo].[PublicPlace] ([PId]),
FOREIGN KEY ([EId]) REFERENCES [dbo].[Employee] ([EId])
);




CREATE TABLE [dbo].[RequestRoom] (
    [RequestNum] INT           NOT NULL,
    [StudentNum]         int NOT NULL,
    [RoomNum]         int NOT NULL,
    [BlockNum]         int NOT NULL,
[FloorNum]         int NOT NULL,

    [Date]         DATE NOT NULL,
    [RRType]         NVARCHAR (150)  NOT NULL,
    [RRStatus]         int NULL,

    PRIMARY KEY CLUSTERED ( [RequestNum] ASC),
```

```
 FOREIGN KEY ([StudentNum) REFERENCES [dbo].[Student] ([StudentNum],
CONSTRAINT [fk_rrr] FOREIGN KEY ([RoomNum] ],[BlockNum],[FloorNum]) REFERENCES
[dbo].[Room] ([RoomNum] ,[BlockNum],[FloorNum])) ON DELETE CASCADE,

CHECK ([RRStatus]='Accept' or [RRStatus]='Reject' or [RRStatus]=NULL),

CHECK ([RRType]='Reserve' or [RRType]='Cancel')
);
```

چند نمونه QUERY:

- students profile info

```
SELECT *

FROM Student INNER JOIN User ON
Student.Mellicode=User.Mellicode
```

- list of dormitories of a university with unid=1

```
SELECT *

FROM Dorm

WHERE UniId = 1
```

- check all deadline extension requests of a student from student number

```
SELECT *

FROM RequestDeadlineExtention

WHERE studentNum=95512167
```

- list of PublicPlaces of a dormitory with dormNum=1

```
SELECT *
FROM PublicPlace
WHERE DormNum = 1
```

- all telephone numbers of a university

```
SELECT *
FROM Uni INNER JOIN UniPhone
ON Uni.UniId = UniPhone.UniId
WHERE UniId = 1
```

- list of students with majorName = 'Computer Engineering'

```
SELECT *
FROM Student INNER JOIN Major
ON Student.MId = Major.MId
WHERE MName = 'Computer'
```

- count of female students

```
SELECT count (*) FROM Student INNER JOIN User ON
Student.Mellicode=User.melliCode
WHERE UserGender='F'
```

- the salary of an admin

```
SELECT Salary
```

```
FROM Job
WHERE JName='Admin'
```

:TRIGGERS

در این بخش از trigger برای این استفاده می‌کنیم به این منظور که وقتی دانشجو درخواست رزرو اتاقی را داد و کارمند آن را تعیید کرد، ستون‌های RoomNum، FloorNum و BlockNum در جدول دانشجو تغییر به روز رسانی شوند.

- Trigger for updating student table after request was checked by employee

```
CREATE TRIGGER updateroomnum
ON [dbo].[RequestRoom]
AFTER UPDATE
AS
BEGIN
    IF UPDATE(checked)
        UPDATE [dbo].[Student]
        SET RoomNum = i.RoomNum
        SET FloorNum = i.FloorNum
        SET BlockNum = i.BlockNum
        FROM
```

```
                Updated i
END
```

CURSORS:

در این بخش از cursor برای چاپ اطلاعات تمامی دانشجویان یک خوابگاه استفاده می‌کنیم. اول سطر به سطر اطلاعاتی که می‌خواهیم را استخراج می‌کنیم (ایا paid هستند یا خیر) و بعد چاپ می‌کنیم.

- Cursor for printing all student names and whether they have paid or not

```
DECLARE c_studentpaylist CURSOR
FOR SELECT
     StudentNum,
     Paid

     FROM
          [dbo].[Student] c
          INNER JOIN [dbo].[User] t
          ON c.Mellicode = t.Mellicode;
OPEN c_studentpaylist;
FETCH NEXT FROM c_studentpaylist INTO
```

```
    @StudentNum VARCHAR(MAX),
    @Student_Name VARCHAR(MAX),
    @paid BIT;
WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT @StudentNum + @Student_Name + CAST(@paid AS
VARCHAR);
        FETCH NEXT FROM c_studentpaylist INTO
            @StudentNum,
            @Student_Name,
            @paid;
    END;
CLOSE c_studentpaylist;
DEALLOCATE c_studentpaylist;
```

FUNCTIONS:

در این بخش برای query یک از function میانگین گیری ()avg استفاده شده است و در query دوم از function شموردن()count استفاده شده است.

- number of employees with more than average salaries

```
SELECT count (*)
FROM Employee INNER JOIN Job on Employee.JId=Job.JId
```

```
WHERE Salary > (SELECT avg (Salary) FROM Job)
```

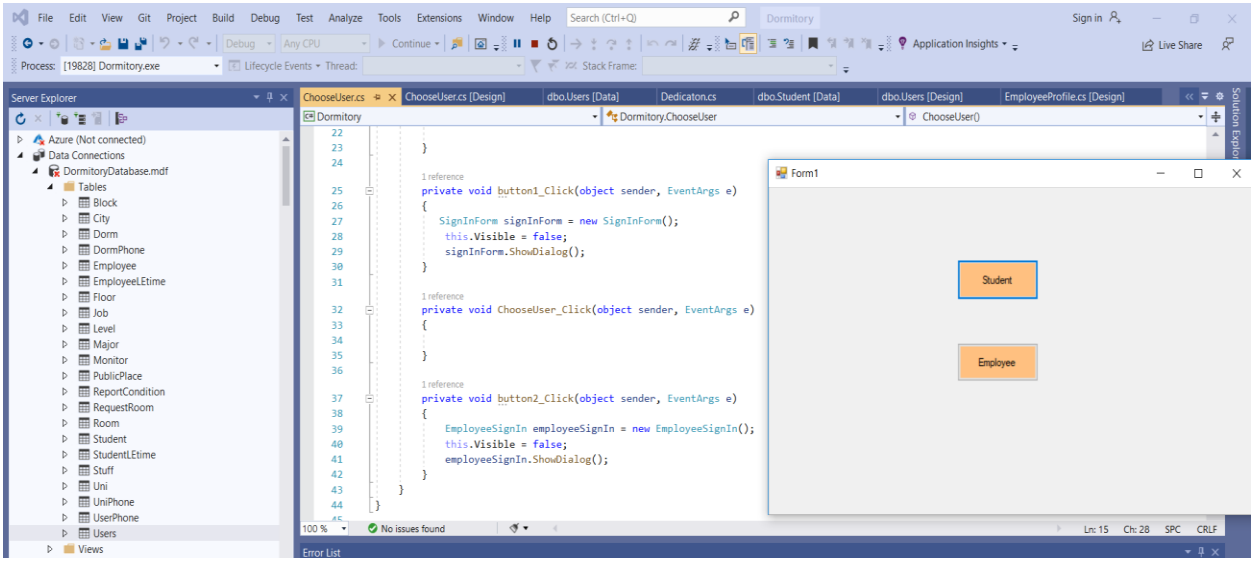
- name of dormitories with more than 2 blocks

```
SELECT DormitoryName,count(*)
FROM Dorm INNER JOIN Block ON Dorm.dormNum=Block.dormNum
GROUP BY DormNum having count(*)>2
```

تصاویر اجرای پروژه و کد :

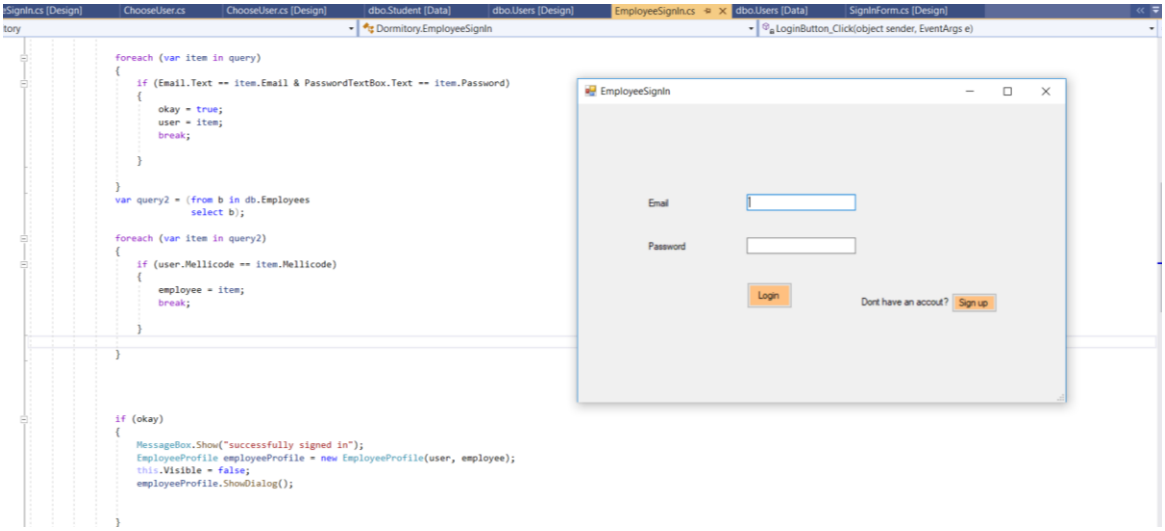
با اجرا برنامه در ابتدا صفحه ای باز می شود که از کاربر می خواه که نوع کاربری خود که کارمند یا دانشجو هست را مشخص کند.

در تصویر زیر این صفحه به همراه کد مربوطه را مشاهده می کنیم.

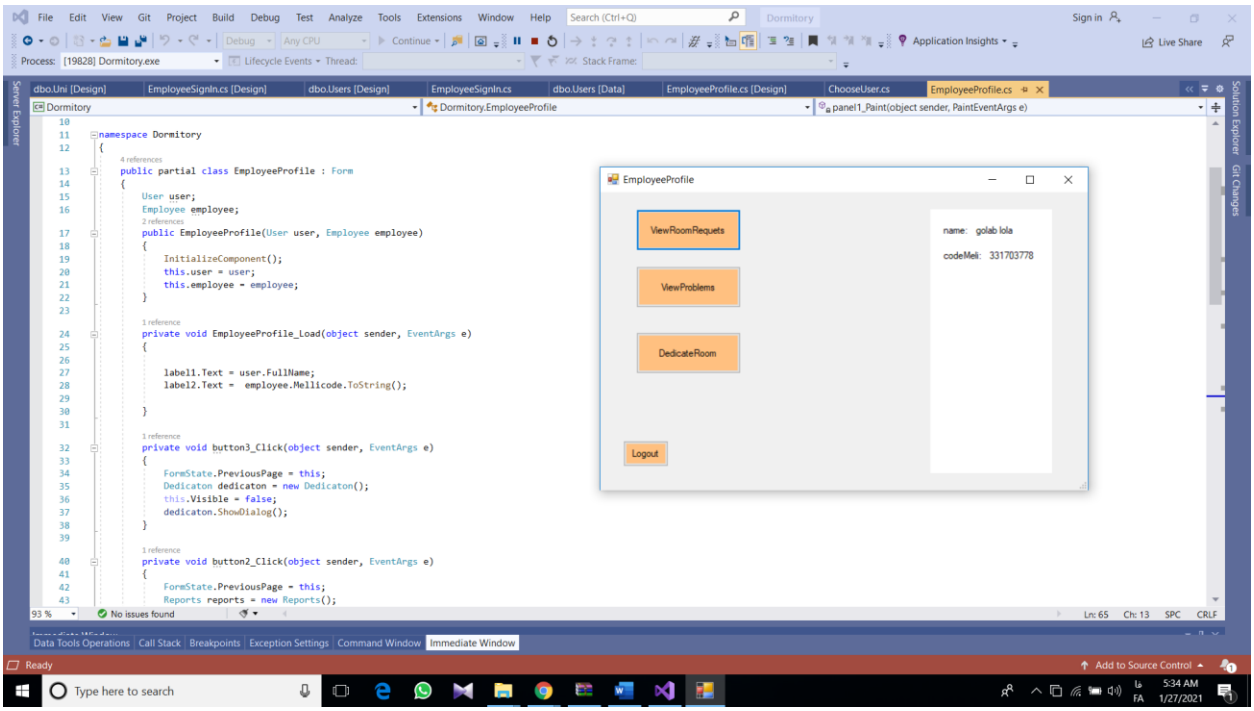


پس از انتخاب کاربر (در انجا ما برای مثال کارمند را انتخاب کرده ایم) صفحه SignIn باز می شود. در این صفحه اگر کاربر دارای حساب کاربری باشد با وارد کردن ایمیل و رمز خود وارد می شود. این صفحه و کد نحوه

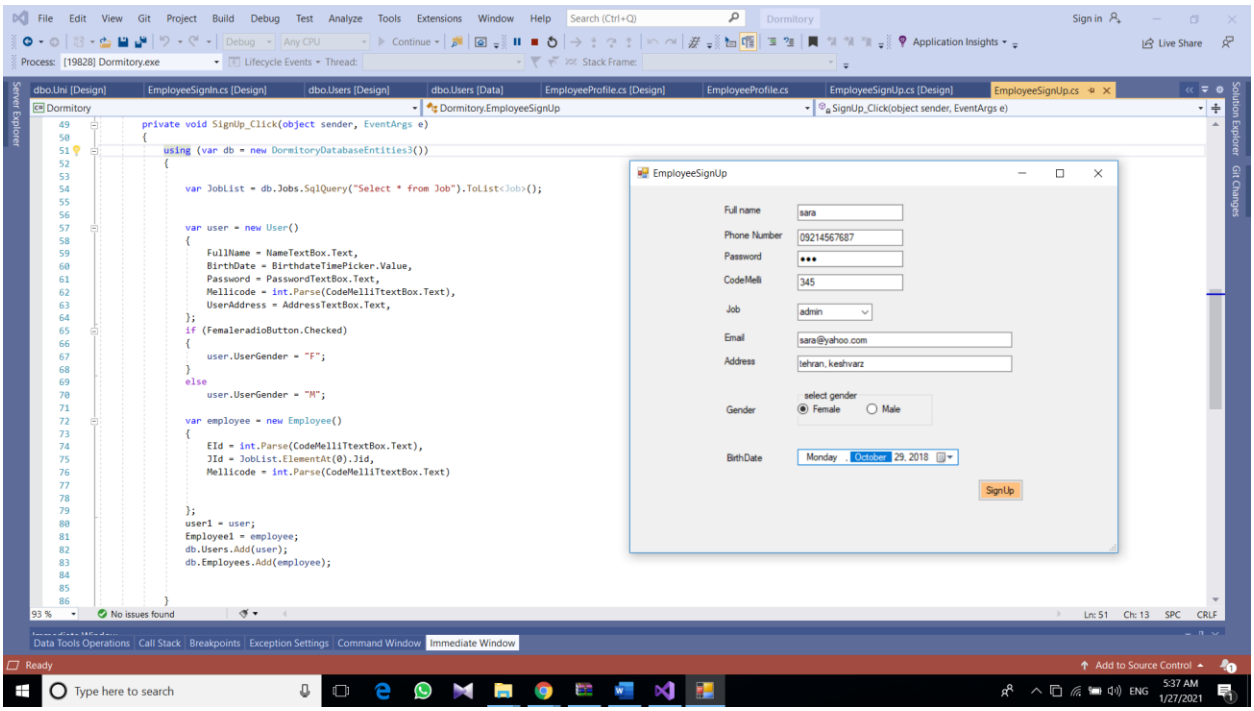
چک کردن ایمیل و رمز را در تصویر زیر مشاهده می کنیم.

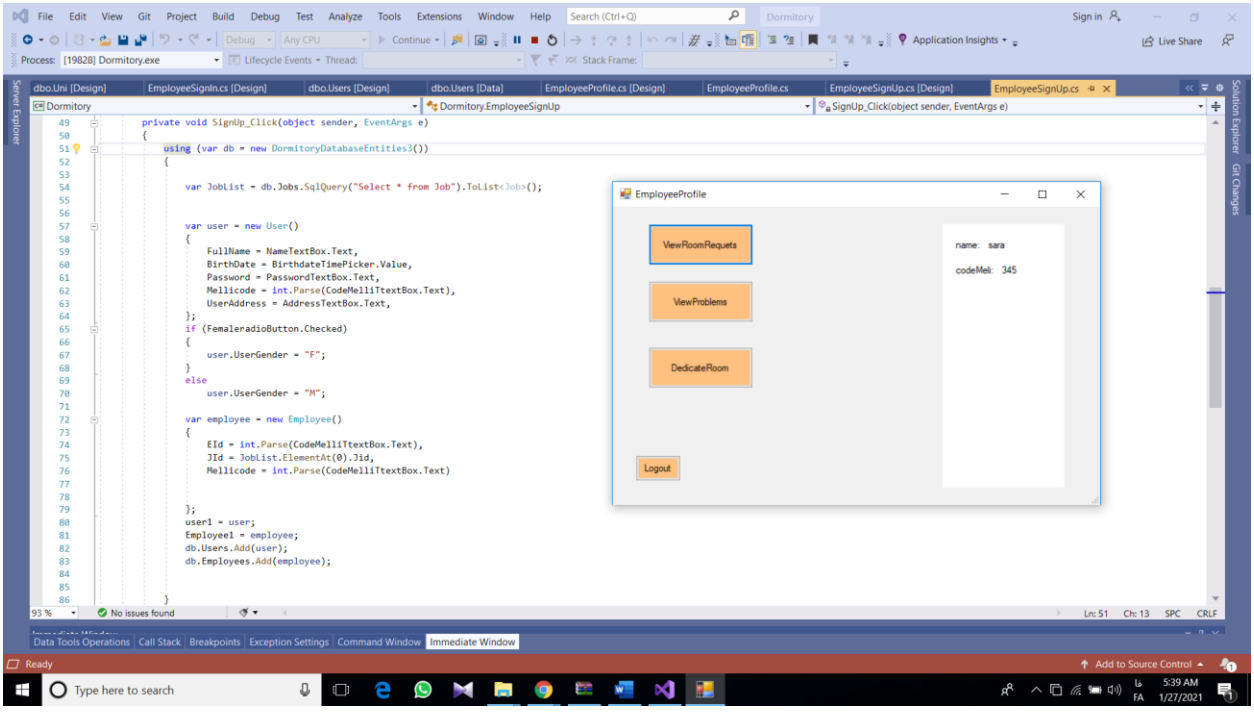


با وارد کردن ایمیل و رمز وارد صفحه کاربر می شویم.



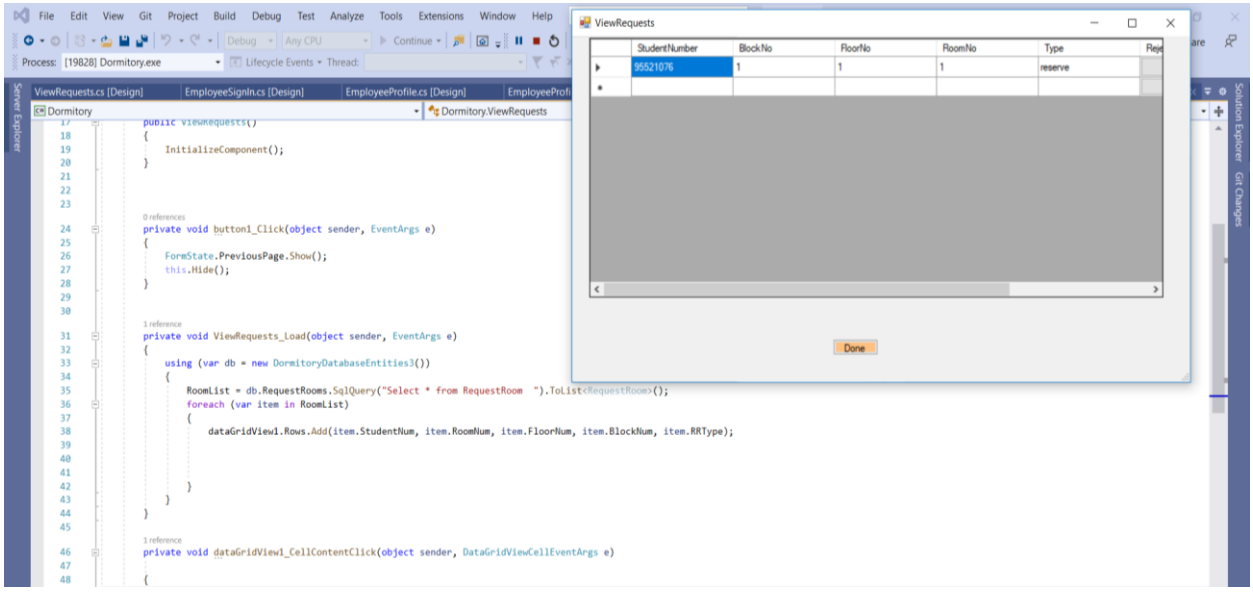
همچنین در صورت نداشتن حساب کاربری با انتخاب گزینه signUp می توانیم یک حساب کاربری جدید بسازیم و وارد صفحه خود بشویم. در شکل اجرا و کد مربوط به ساخت حساب جدید و رفتن به صفحه شخصی را میبینیم.



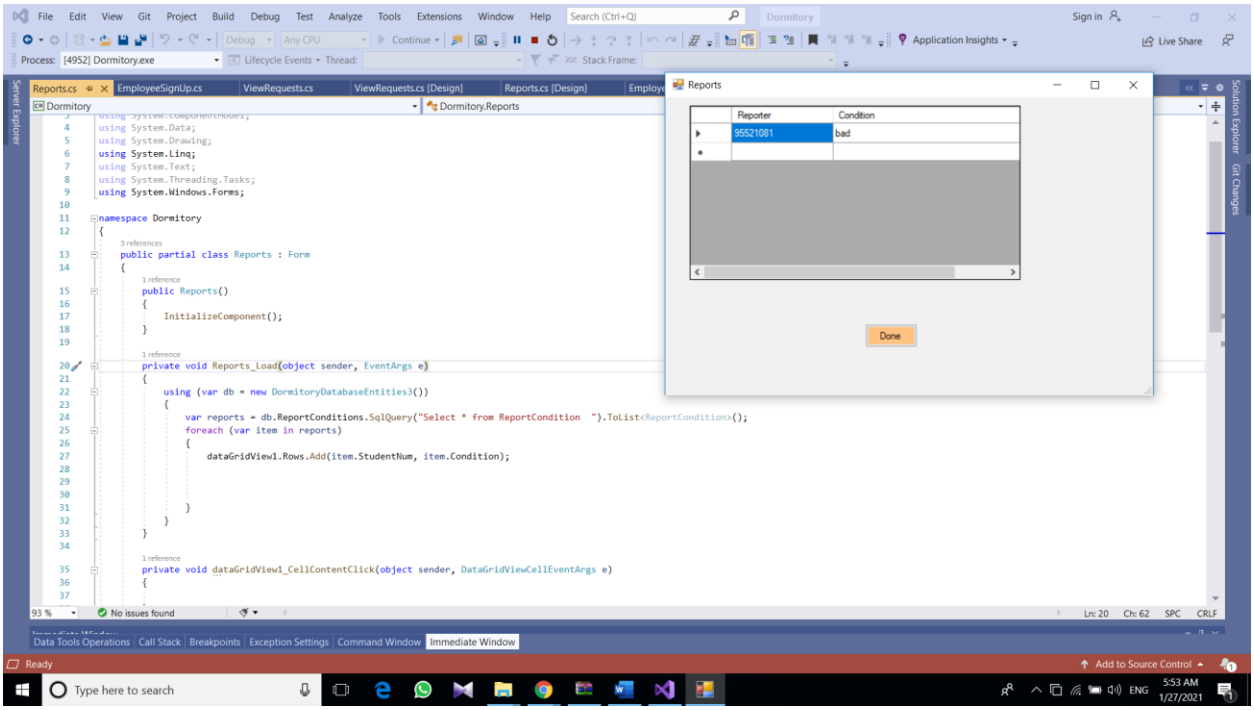


در این صفحه برای هر کارمند امکاناتی از قبیل دیدن درخواست ها، مشاهده مشکلات گزارش شده، و تخصیص اتاق به هر دانشجو را مشاهده می کنیم. هم چنین با استفاده از دکمه logout می توان از حساب کاربری خارج شد.

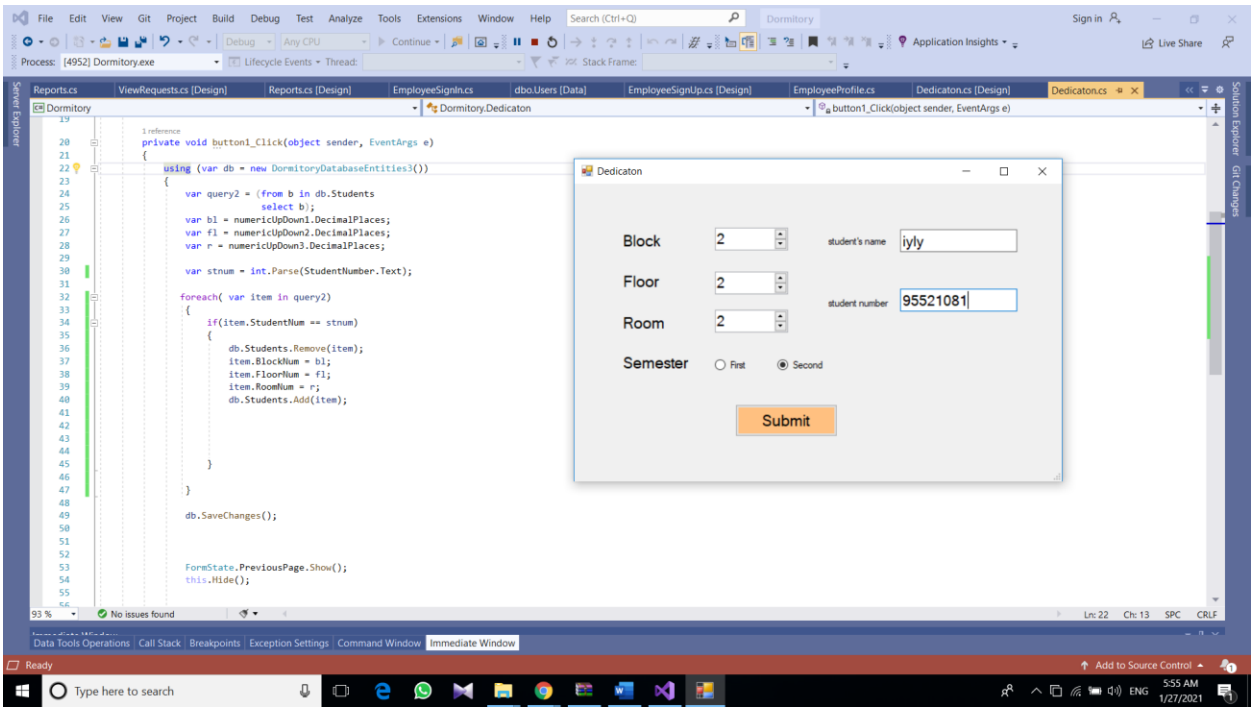
در تصویر زیر صفحه نمایش درخواست ها و همچنین کد مربوط به اجرای آن را مشاهده می کنیم. با زدن دکمه Done به صفحه اصلی کاربر بر می گردیم.



در تصویر زیر صفحه نمایش مشکلات و ک اجرای آن را مشاهده میکنیم.



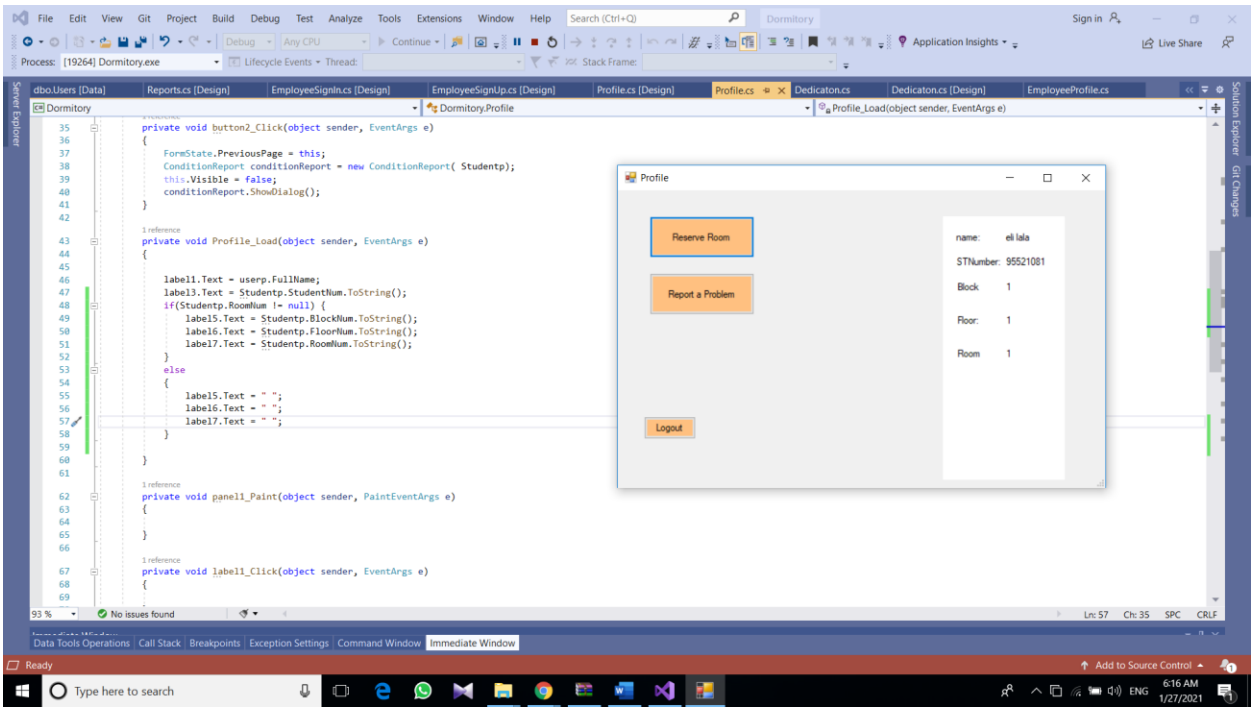
در تصویر زیر فرم اختصاص اتاق و کد اجرای آن را مشاهده میکنیم.



بار دیگر به عنوان دانشجو وارد می شویم و و با وارد کردن ایمیل و رمز وارد صفحه دانشجو می شویم.

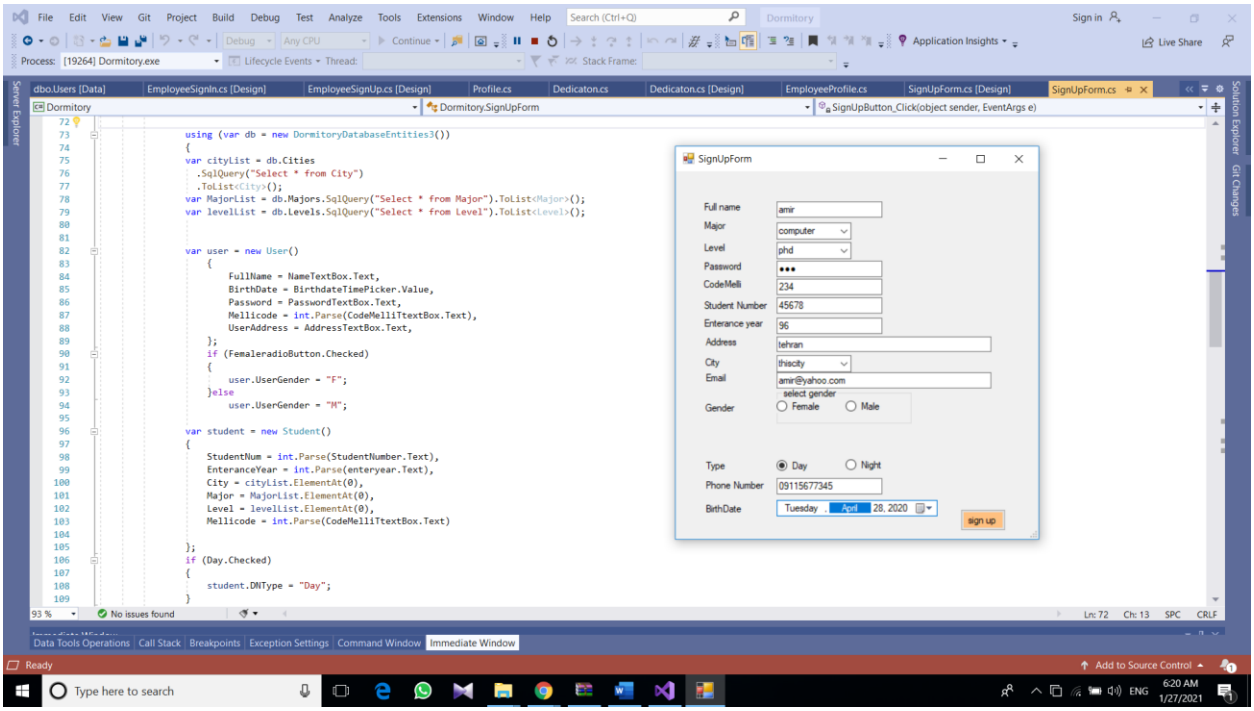
کد لاگین کردن در صفحه دانشجو مشابه کارمند که در بالا آورده شده است ، می باشد.

در تصویر زیر صفحه شخصی دانشجویی به همراه اطلاعات اتاقی که در آن اسکان داده شده است و کد مربوط به آن را می بینیم.

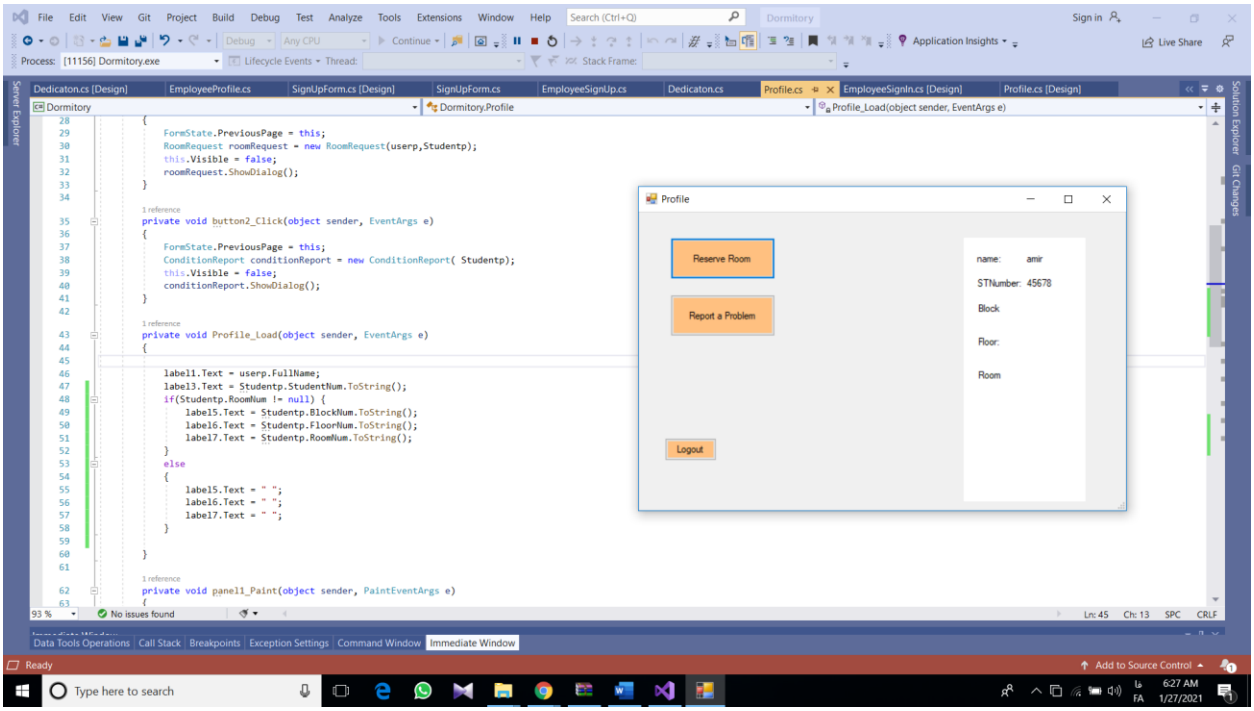


هم چنین در صورتی که دانشجو حساب کاربری نداشته باشد می تواند یک حساب کاربری جدید بسازد.

در شکل فرم ساختن حساب کاربری جدید و هم چنین کد مربوط به آن را مشاهده می کنیم.

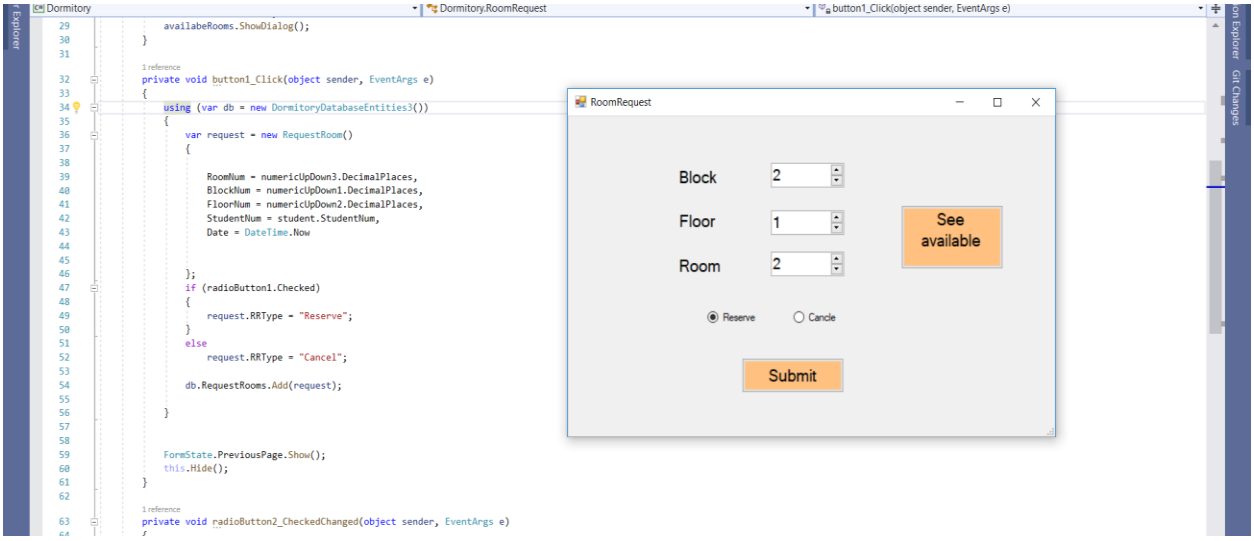


درشکل زیر صفحه حساب جدید ساخته شده را می بینیم که اتاق با آن اختصاص داده نشده.



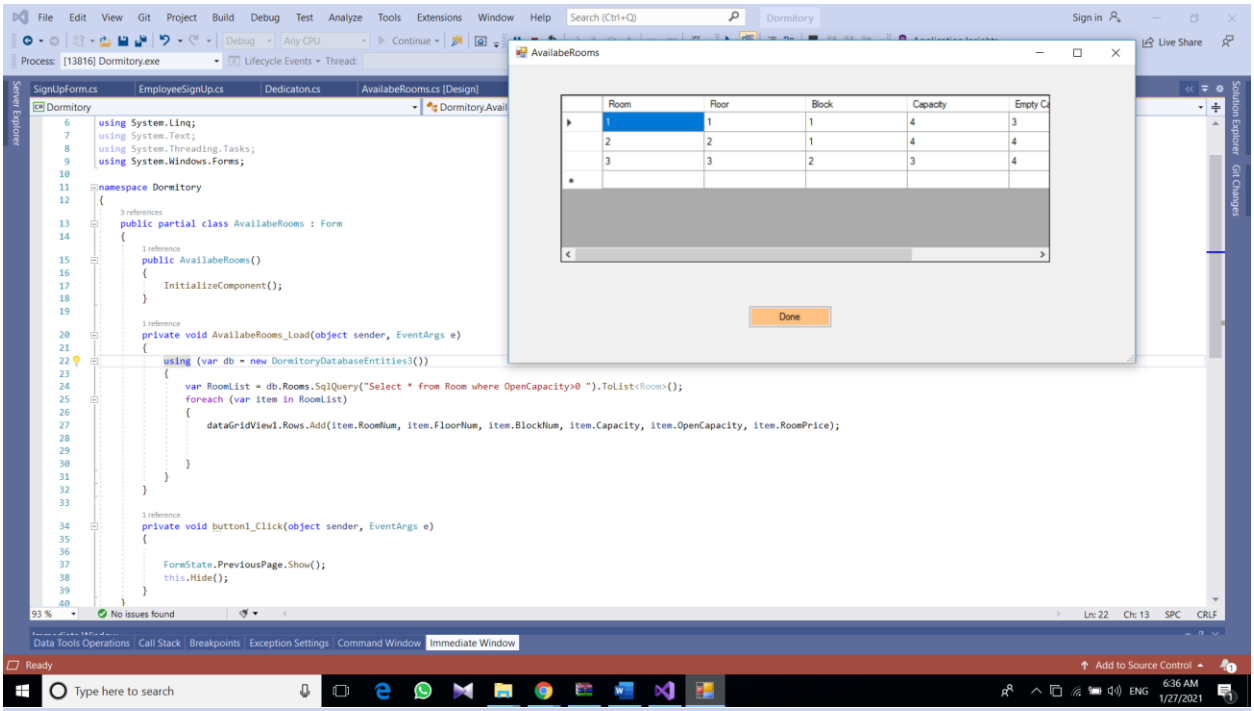
با انتخاب گزینه reserve room دانشجو قادر خواهد بود تا درخواست رزرو یک اتاق و یا کنسل کردن یک اتاق را بدهد.

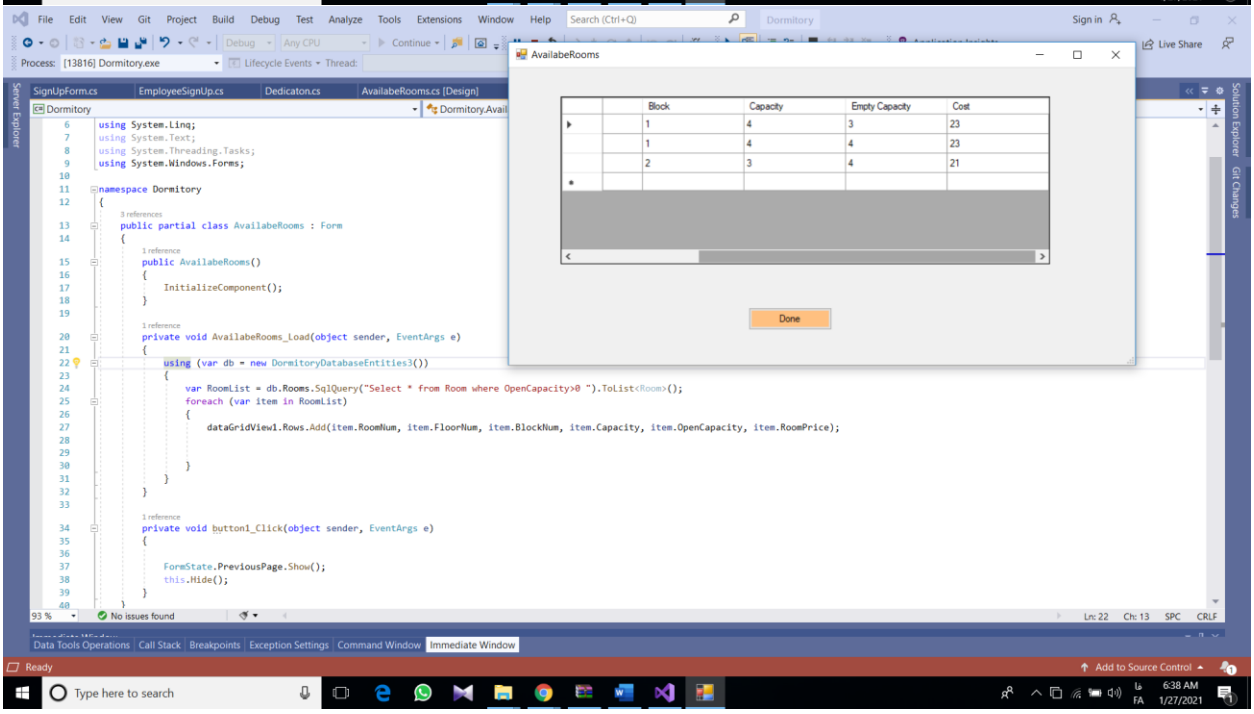
در شکل زیر فرم درخواست رزرو و یا کنسل اتاق با انتخاب شماره بلوک و طبقه و اتاق و کد مربوط به آن را مشاهده می کنیم.



همچنین در این صفحه دکمه مشاهده اتاق های قابل رزرو وجود دارد.

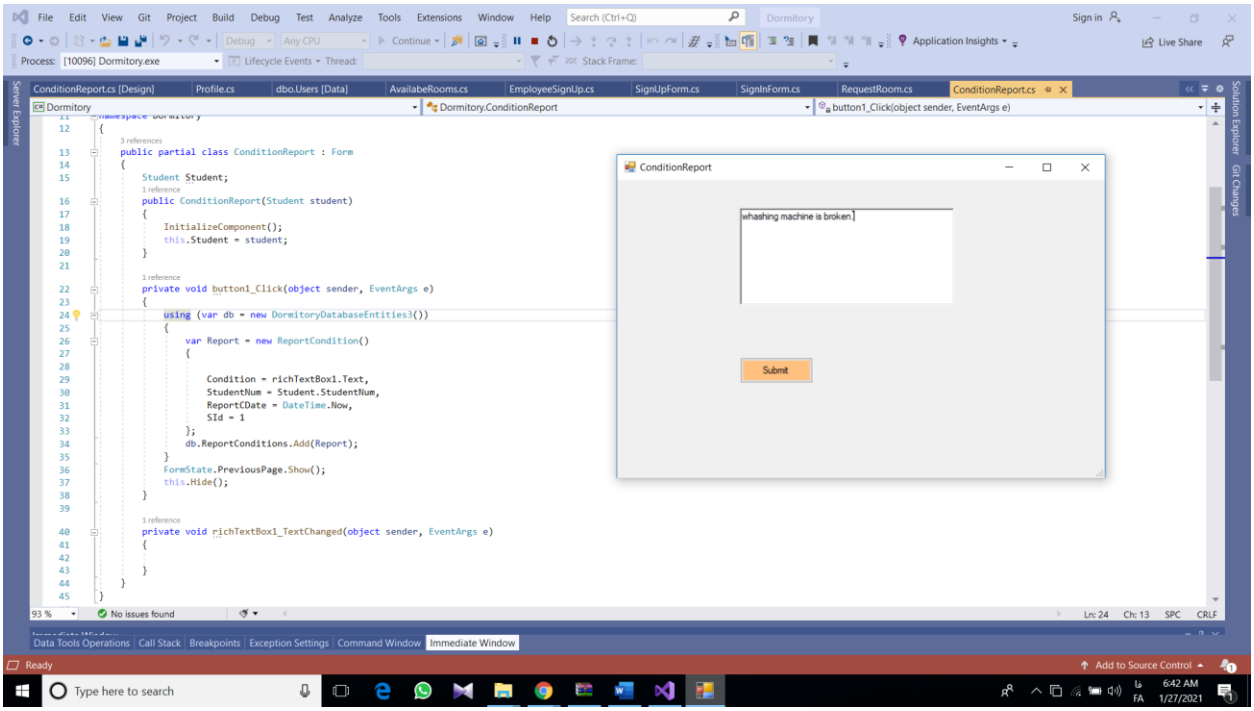
در دو شکل زیر این جدول و کد مربوط به آن را مشاهده می کنیم. در این جدول شماره بلوک ، طبقه ، اتاق و ظرفیت کل و ظرفیت خالی هر اتاق ، و همچنین شهریه هر اتاق مشخص شده است.





همچنین در صفحه هر دانشجو دکمه گزارش مشکلات برای گزارش مشکلات خوابگاه برای کارمندان و مسئولان نیز وجود دارد.

در شکل زیر فرم فرستادن مشکل ایجاد شده و کد مربوط به آن را میبینیم.



در این صفحه شخصی دانشجو هم ، همانند صفحه کارمندان می توان با انتخاب گزینه logout از صفحه خارج شد و دوباره به صفحه لاگین برگشت.