Sri Lanka Institute of Information Technology

Topic : Online Land Sales System

Group no : MLB_05.02_09

Campus : Malabe

Submission Date : 20/05/2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
it21294198	Karunarathna K. R. M. R. T.	0763780858
it21296246	Hapuarachchi H. A. R. S.	0703768998
it21292668	Herath H.M.N.R.	0704134193
it21294648	Wickramarathne M.D.M.C.L.	0713860096
it21292422	Akmal M.A.M.	0761136787

Contents

1. Description.. 2
2. Identified classes.. 3
3. CRC cards... 4
4. Class diagram 7
5. Code.. 8
6. Individual Contribution 19

1. Description

“Golden lands” is an online land selling system. The system has 2 types of users. Un-registered users can only view posts. They can create an account by giving their name, password, email, address and optionally, a profile picture and a description about themselves.

There are 2 types of posts, request posts and sale posts. Registered user can create, edit and delete both types of both types of posts. They can also save posts to their account to view later. They can edit their personal information in their profile too.

Users’ posts can get featured in the front page by selecting a promotional package and making the relevant payment.

There are 3 types of registered users,

- Normal user
- Moderator
- Admin

Moderator can do everything a normal user can do. Additionally, they can warn, suspend, and ban users from using the system.

Admin can do everything a moderator can. Additionally, they can add and remove moderators from the system.

All posts contain a title, location, city, district, province, a description and client contact information.

Additionally, request posts should state a distance, a price range, a range for the requested land area and photos of the place.

Sale posts contain a price, land area and the address of the land.

Users can report posts for 4 reasons,

- False Advertisement
- Spam and Abuse
- False Information
- Transaction Denial

Users can also add an additional description of the issue before sending in the report.

After sending, the report is received by a moderator. The moderator reviews the report, takes the necessary action and marks the report as “reviewed” at the end.

2. Identified classes

1. Post
2. Sale Post
3. Request Post
4. User
5. Moderator
6. Admin
7. Complaint
8. Post Type
9. Date
10. Photo
11. Payment

3. CRC cards

Class Name: Complaint	
Responsibilities	**collaborators**
Store complaint details	
Get owner	User
Get complained post	Post

Class Name: Post type	
Responsibilities	**collaborators**
Store post type details	
Calculate price	

Class Name: Date	
Responsibilities	**collaborators**
Store a date	
Calculate date (add/subtract days)	

Class Name: Photo	
Responsibilities	**collaborators**
Store photo details	
Display image	

Class Name: User	
Responsibilities	**Collaborators**
Create sale post	Sale post
Create request post	Request post
Manage post	Post
Make payment	Payment
Manage profile	
Make complaint	Complaint
Save posts	Post
View saved posts	Post

Class Name: Admin	
Responsibilities	**Collaborators**
Add moderators	Moderator
Remove moderators	Moderator
Change moderator permissions	

Class Name: Request Post	
Responsibilities:	**Collaborators:**
Enter details	Post
Manage Post	

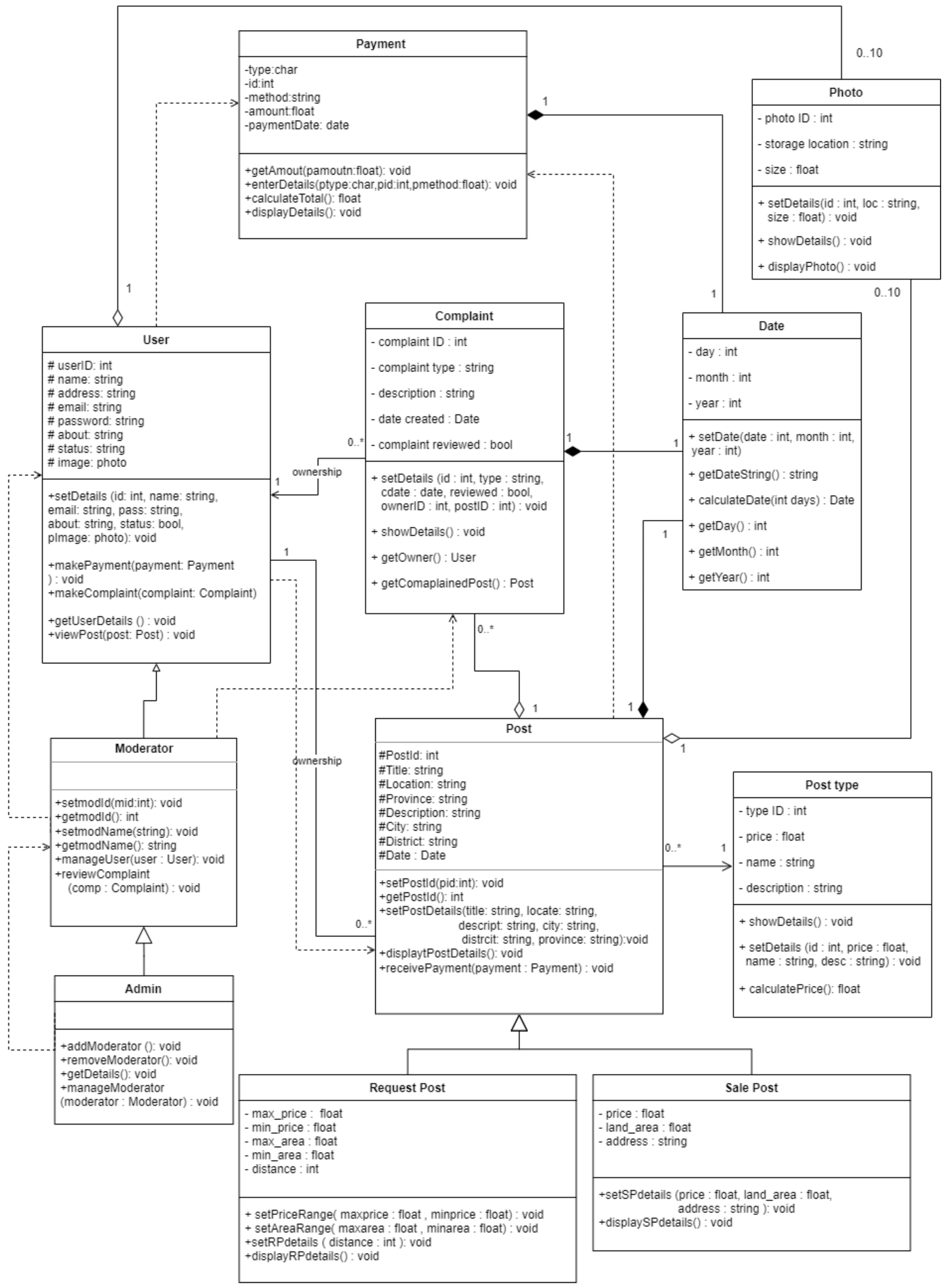
Class Name: Sale post	
Responsibilities:	**Collaborators:**
Enter details	Post
Keep track of the days remaining for the Ad	Date
Validate payment	Payment
Manage post	

Class Name: Payment	
Responsibilities	**Collaborators**
Input the fee of the payment	Post
Calculate the fee for the payment	
Validate the paid method	User
Calculate the end date	Date
Store the details	Post

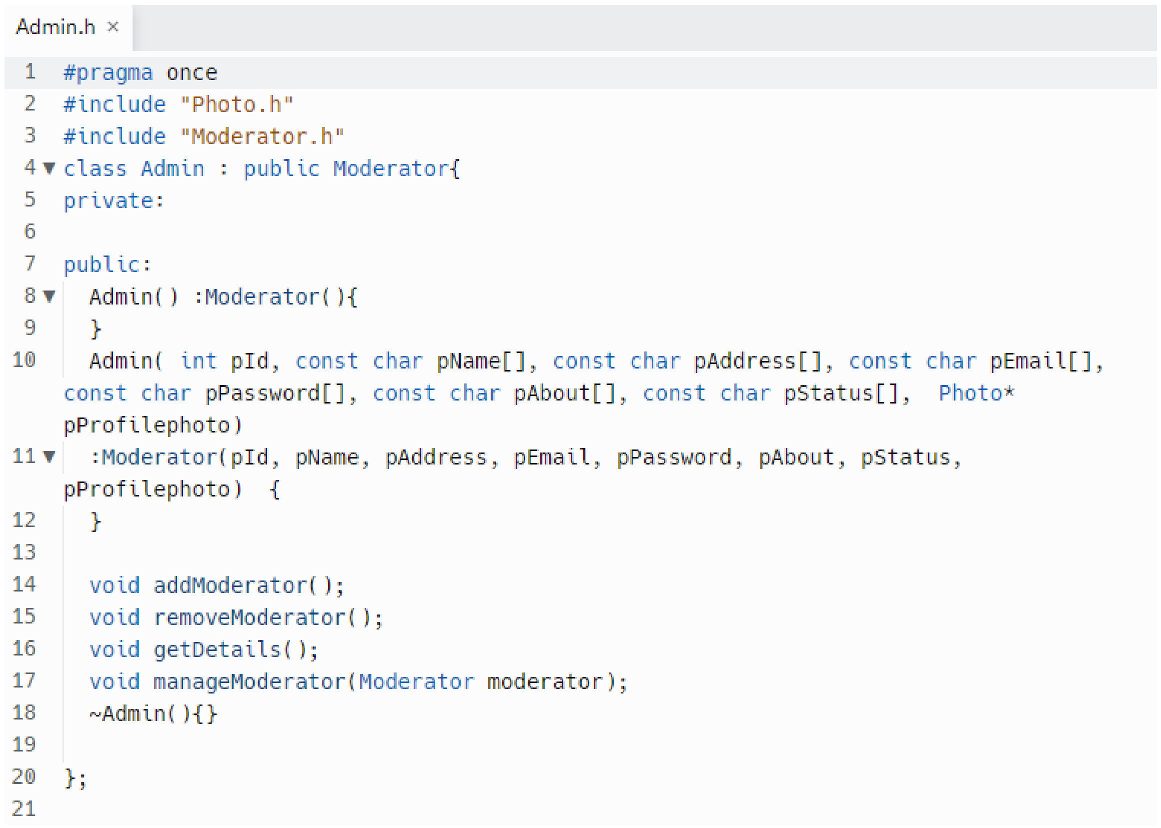
Class Name: Post	
Responsibilities	**Collaborators**
Store Post details	
Modify Details	
View Post	
Validate payment	Payment

Class Name: Moderator	
Responsibilities	**Collaborators**
Store Moderator details	
Validate Moderator Credentials	
Review User Complaints	Complaint, User
Add or delete users	User

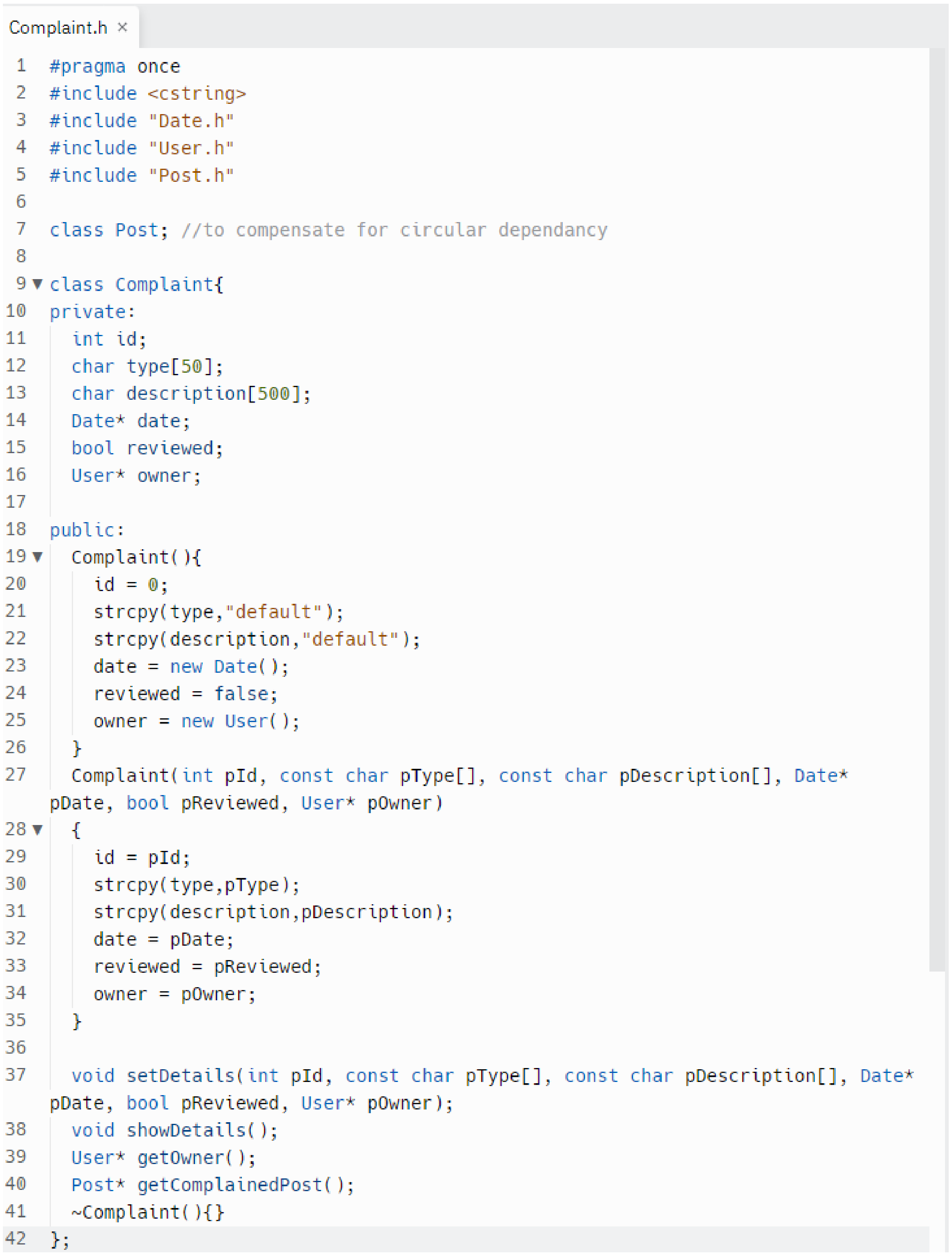
4. Class diagram


Payment
-type:char
-id:int
-method:string
-amount:float
-paymentDate: date
+getAmout(pamoutn:float): void
+enterDetails(ptype:char,pid:int,pmethod:float): void
+calculateTotal(): float
+displayDetails(): void
Photo
- photo ID : int
- storage location : string
- size : float
+ setDetails(id : int, loc : string, size : float) : void
+ showDetails() : void
+ displayPhoto() : void
0..10
1
0..10
User
# userID: int
# name: string
# address: string
# email: string
# password: string
# about: string
# status: string
# image: photo
+setDetails (id: int, name: string, email: string, pass: string, about: string, status: bool, pImage: photo): void
+makePayment(payment: Payment) : void
+makeComplaint(complaint: Complaint)
+getUserDetails () : void
+viewPost(post: Post) : void
Complaint
- complaint ID : int
- complaint type : string
- description : string
- date created : Date
- complaint reviewed : bool
+ setDetails (id : int, type : string, cdate : date, reviewed : bool, ownerID : int, postID : int) : void
+ showDetails() : void
+ getOwner() : User
+ getComaplainedPost() : Post
Date
- day : int
- month : int
- year : int
+ setDate(date : int, month : int, year : int)
+ getDateString() : string
+ calculateDate(int days) : Date
+ getDay() : int
+ getMonth() : int
+ getYear() : int
ownership
0..*
1
1
1
1
1
0..*
1
1
Moderator
+setmodId(mid:int): void
+getmodId(): int
+setmodName(string): void
+getmodName(): string
+manageUser(user : User): void
+reviewComplaint (comp : Complaint) : void
Admin
+addModerator (): void
+removeModerator(): void
+getDetails(): void
+manageModerator (moderator : Moderator) : void
ownership
0..*
Post
#PostId: int
#Title: string
#Location: string
#Province: string
#Description: string
#City: string
#District: string
#Date : Date
+setPostId(pid:int): void
+getPostId(): int
+setPostDetails(title: string, locate: string, descript: string, city: string, distrcit: string, province: string):void
+displaytPostDetails(): void
+receivePayment(payment : Payment) : void
0..*
1
Post type
- type ID : int
- price : float
- name : string
- description : string
+ showDetails() : void
+ setDetails (id : int, price : float, name : string, desc : string) : void
+ calculatePrice(): float
Request Post
- max_price : float
- min_price : float
- max_area : float
- min_area : float
- distance : int
+ setPriceRange(maxprice : float , minprice : float) : void
+ setAreaRange(maxarea : float , minarea : float) : void
+setRPdetails (distance : int): void
+displayRPdetails() : void
Sale Post
- price : float
- land_area : float
- address : string
+setSPdetails (price : float, land_area : float, address : string): void
+displaySPdetails() : void

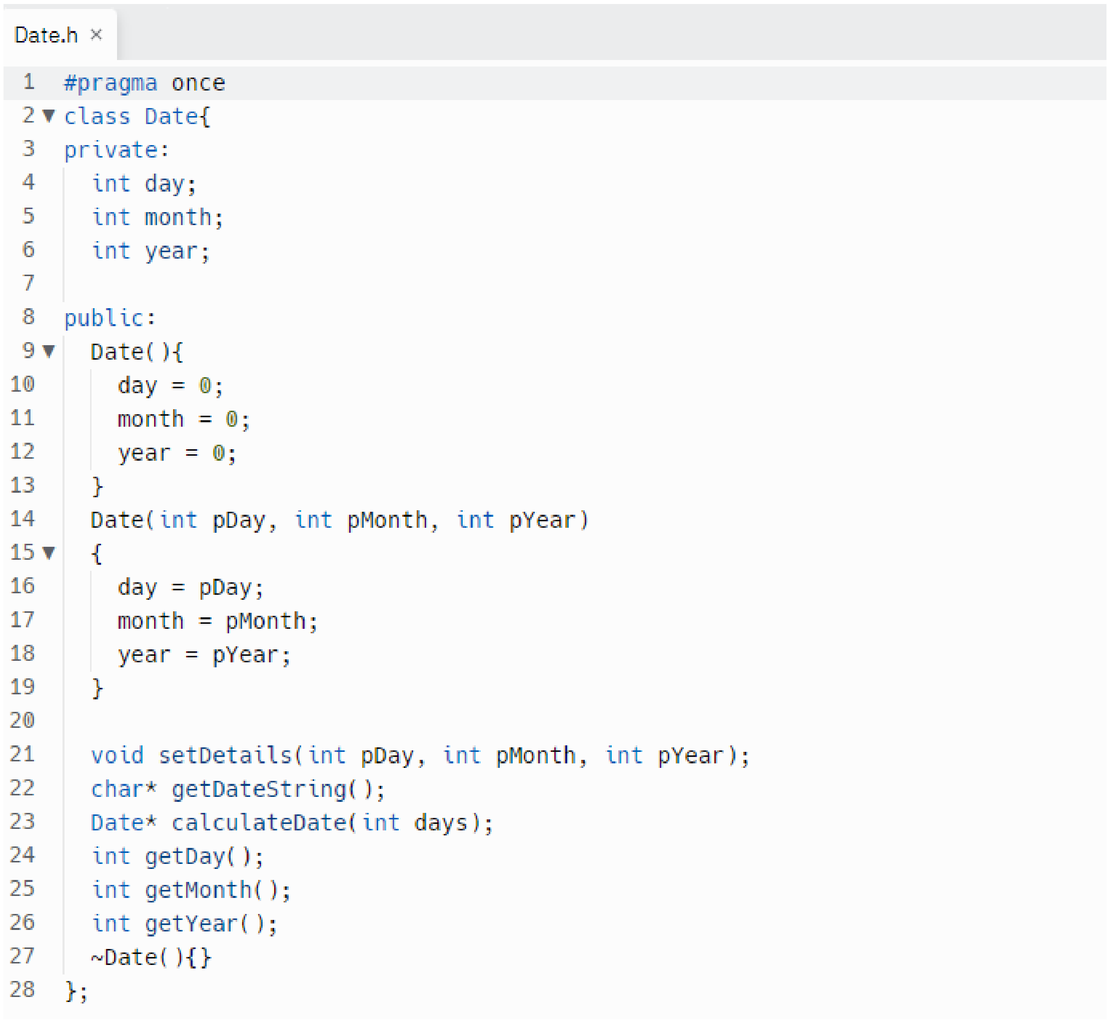
5. Code



```
#pragma once
#include "Photo.h"
#include "Moderator.h"
class Admin : public Moderator{
private:

public:
  Admin() :Moderator(){
  }
  Admin( int pId, const char pName[], const char pAddress[], const char pEmail[], const char pPassword[], const char pAbout[], const char pStatus[],  Photo* pProfilephoto)
  :Moderator(pId, pName, pAddress, pEmail, pPassword, pAbout, pStatus, pProfilephoto)  {
  }

  void addModerator();
  void removeModerator();
  void getDetails();
  void manageModerator(Moderator moderator);
  ~Admin(){}

};

```

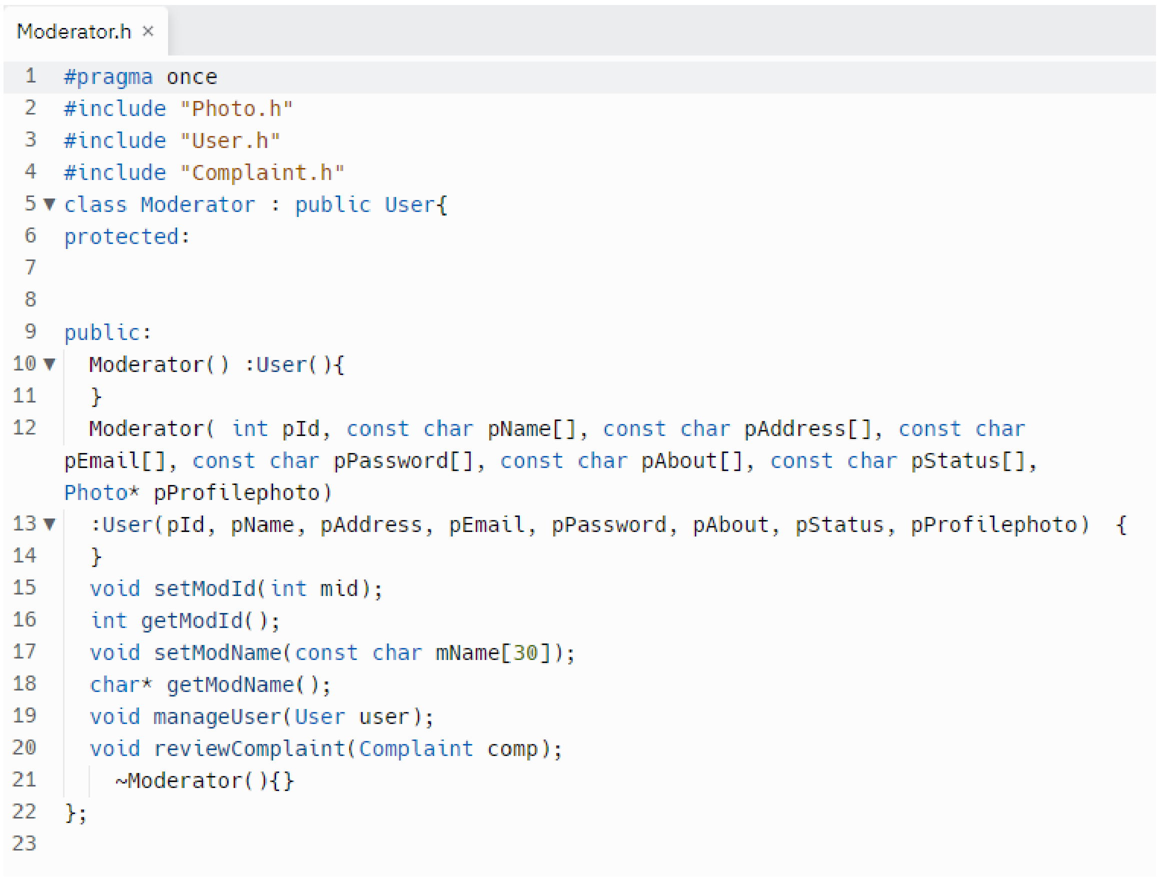


```cpp
#pragma once
#include <cstring>
#include "Date.h"
#include "User.h"
#include "Post.h"

class Post; //to compensate for circular dependancy

class Complaint{
private:
  int id;
  char type[50];
  char description[500];
  Date* date;
  bool reviewed;
  User* owner;

public:
  Complaint(){
    id = 0;
    strcpy(type,"default");
    strcpy(description,"default");
    date = new Date();
    reviewed = false;
    owner = new User();
  }
  Complaint(int pId, const char pType[], const char pDescription[], Date*
pDate, bool pReviewed, User* pOwner)
  {
    id = pId;
    strcpy(type,pType);
    strcpy(description,pDescription);
    date = pDate;
    reviewed = pReviewed;
    owner = pOwner;
  }

  void setDetails(int pId, const char pType[], const char pDescription[], Date*
pDate, bool pReviewed, User* pOwner);
  void showDetails();
  User* getOwner();
  Post* getComplainedPost();
  ~Complaint(){}
};
```

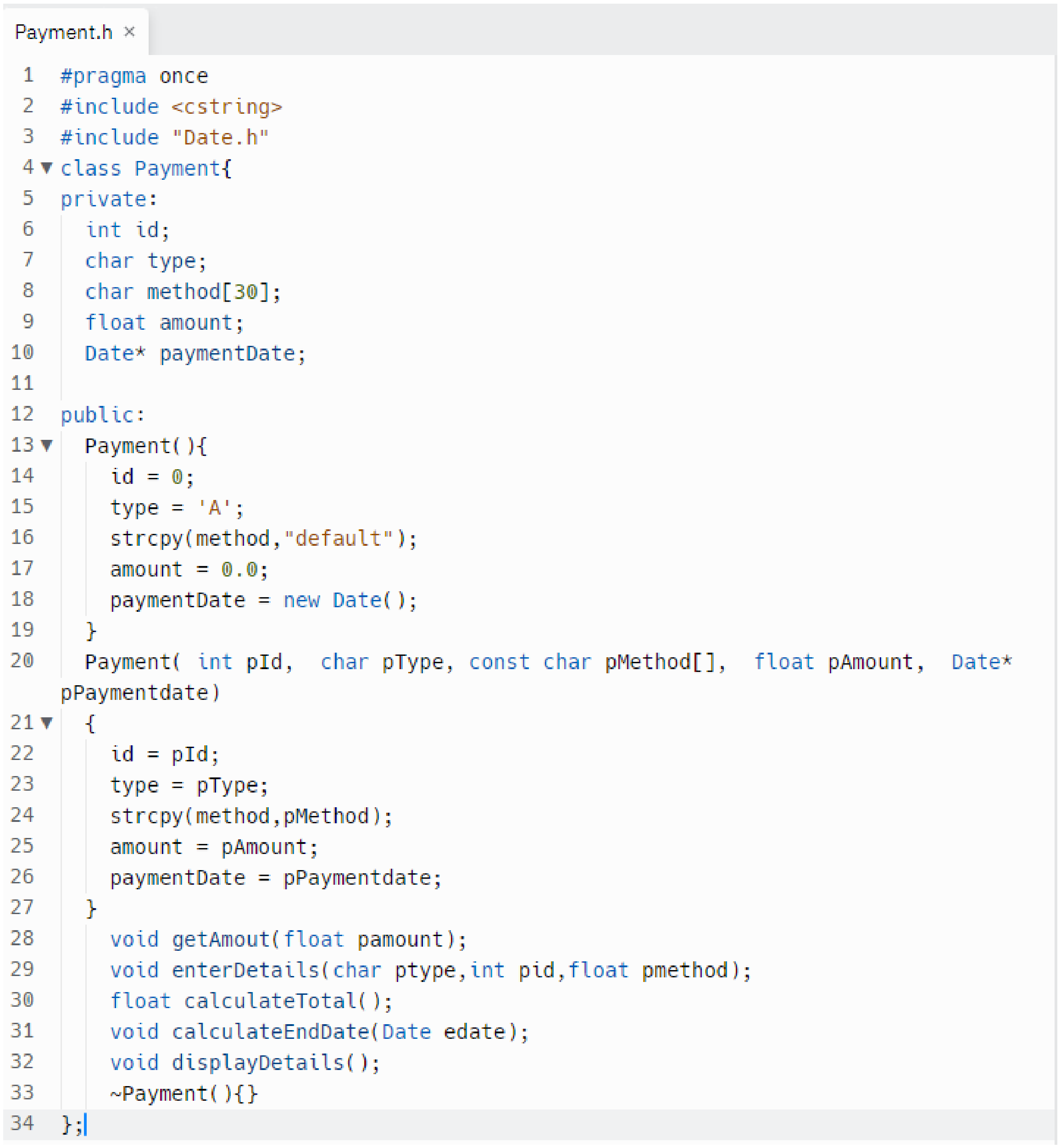


Date.h ×

```
#pragma once
class Date{
private:
  int day;
  int month;
  int year;

public:
  Date(){
    day = 0;
    month = 0;
    year = 0;
  }
  Date(int pDay, int pMonth, int pYear)
  {
    day = pDay;
    month = pMonth;
    year = pYear;
  }

  void setDetails(int pDay, int pMonth, int pYear);
  char* getDateString();
  Date* calculateDate(int days);
  int getDay();
  int getMonth();
  int getYear();
  ~Date(){}
};
```


Moderator.h ×

```
#pragma once
#include "Photo.h"
#include "User.h"
#include "Complaint.h"
class Moderator : public User{
protected:


public:
  Moderator() :User(){
  }
  Moderator( int pId, const char pName[], const char pAddress[], const char
pEmail[], const char pPassword[], const char pAbout[], const char pStatus[],
Photo* pProfilephoto)
  :User(pId, pName, pAddress, pEmail, pPassword, pAbout, pStatus, pProfilephoto)  {
  }
  void setModId(int mid);
  int getModId();
  void setModName(const char mName[30]);
  char* getModName();
  void manageUser(User user);
  void reviewComplaint(Complaint comp);
    ~Moderator(){}
};

```


Payment.h ×

```cpp
#pragma once
#include <cstring>
#include "Date.h"
class Payment{
private:
  int id;
  char type;
  char method[30];
  float amount;
  Date* paymentDate;

public:
  Payment(){
    id = 0;
    type = 'A';
    strcpy(method,"default");
    amount = 0.0;
    paymentDate = new Date();
  }
  Payment( int pId,  char pType, const char pMethod[],  float pAmount,  Date* pPaymentdate)
  {
    id = pId;
    type = pType;
    strcpy(method,pMethod);
    amount = pAmount;
    paymentDate = pPaymentdate;
  }
    void getAmout(float pamount);
    void enterDetails(char ptype,int pid,float pmethod);
    float calculateTotal();
    void calculateEndDate(Date edate);
    void displayDetails();
    ~Payment(){}
};
```

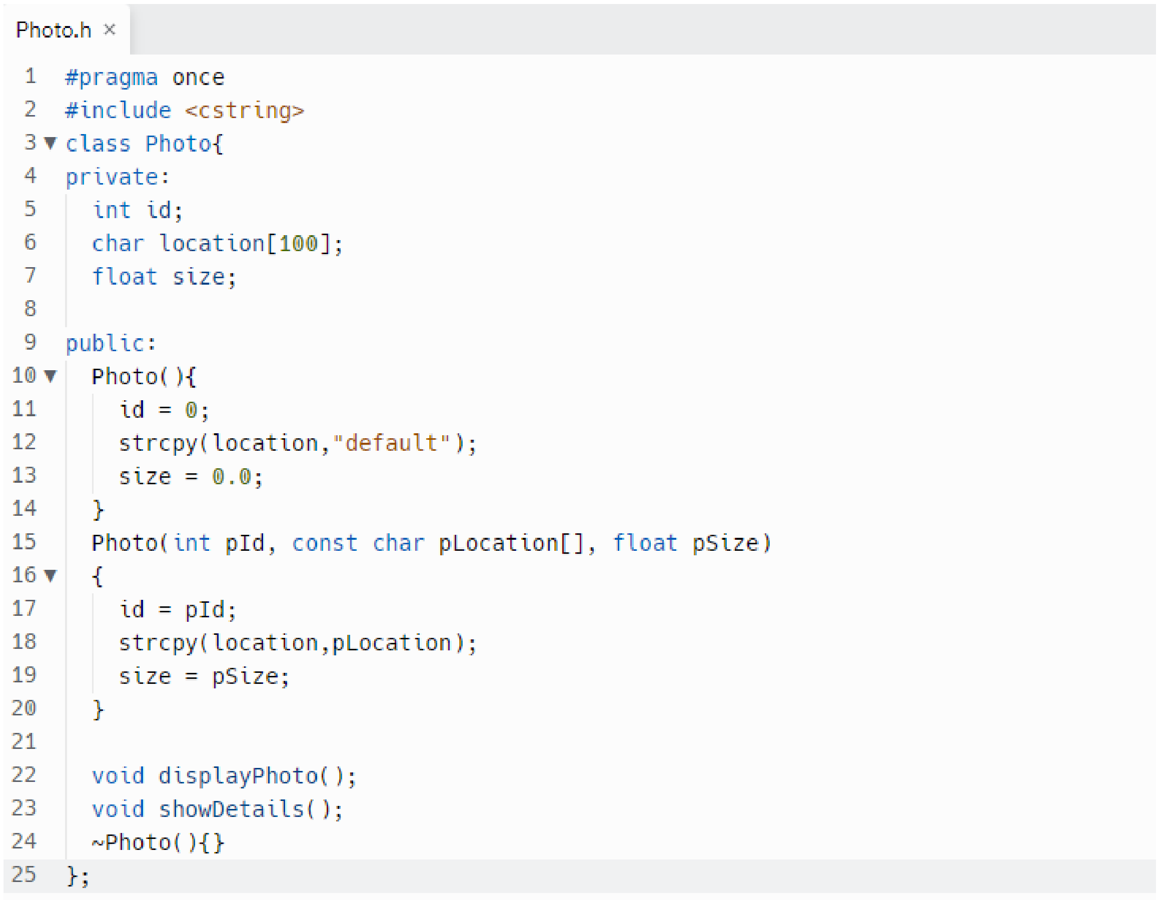
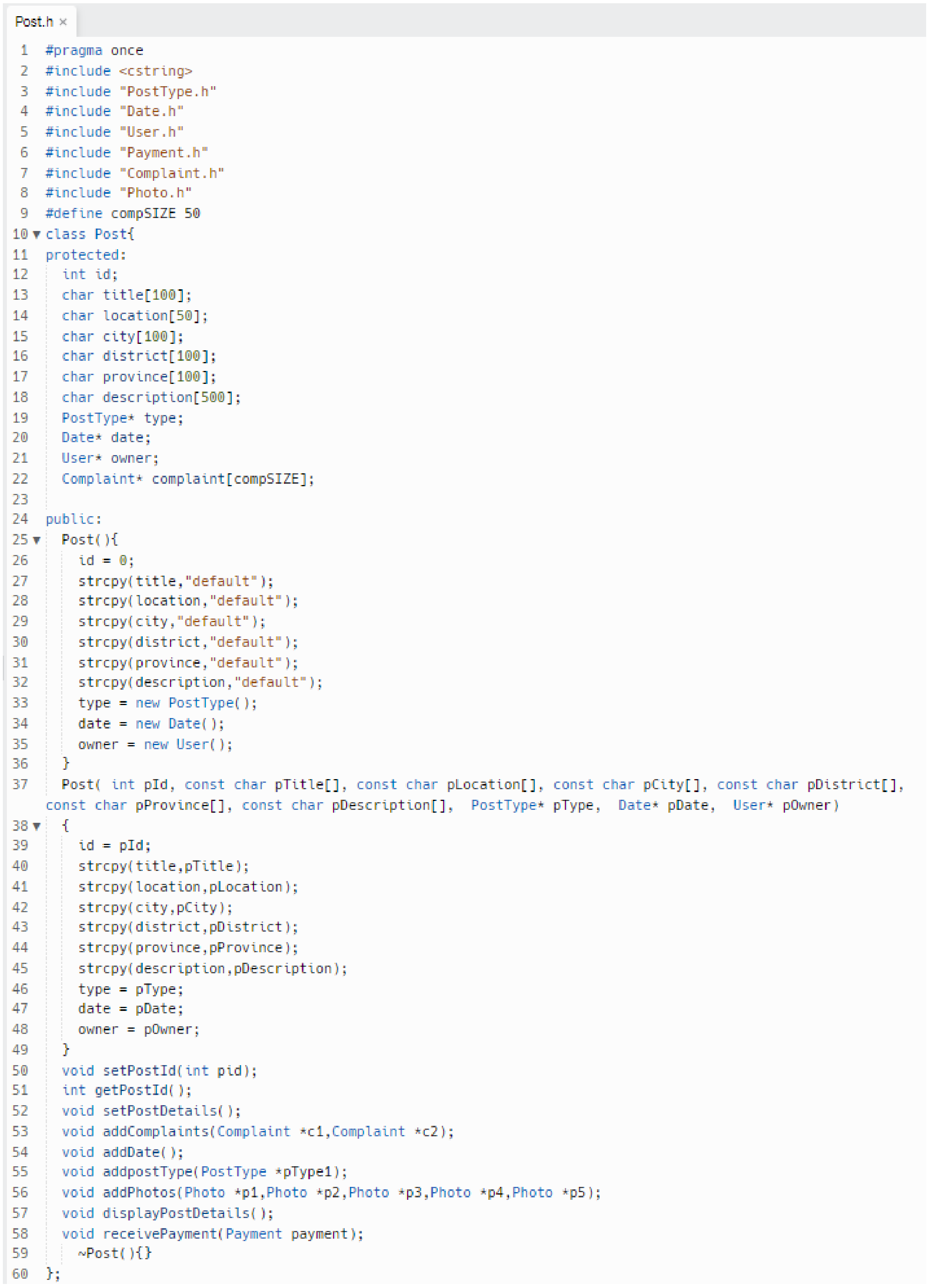
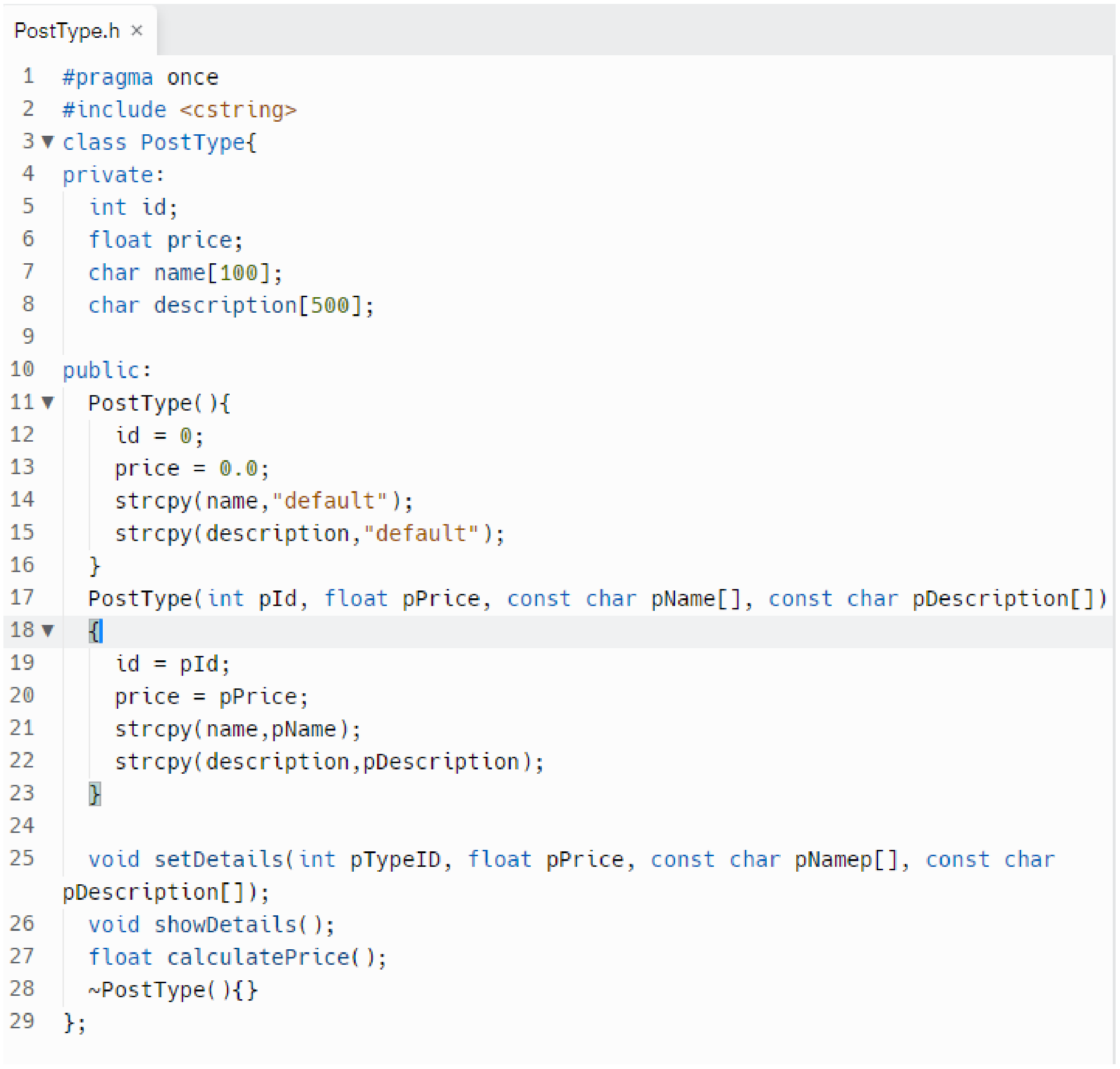


Photo.h ×

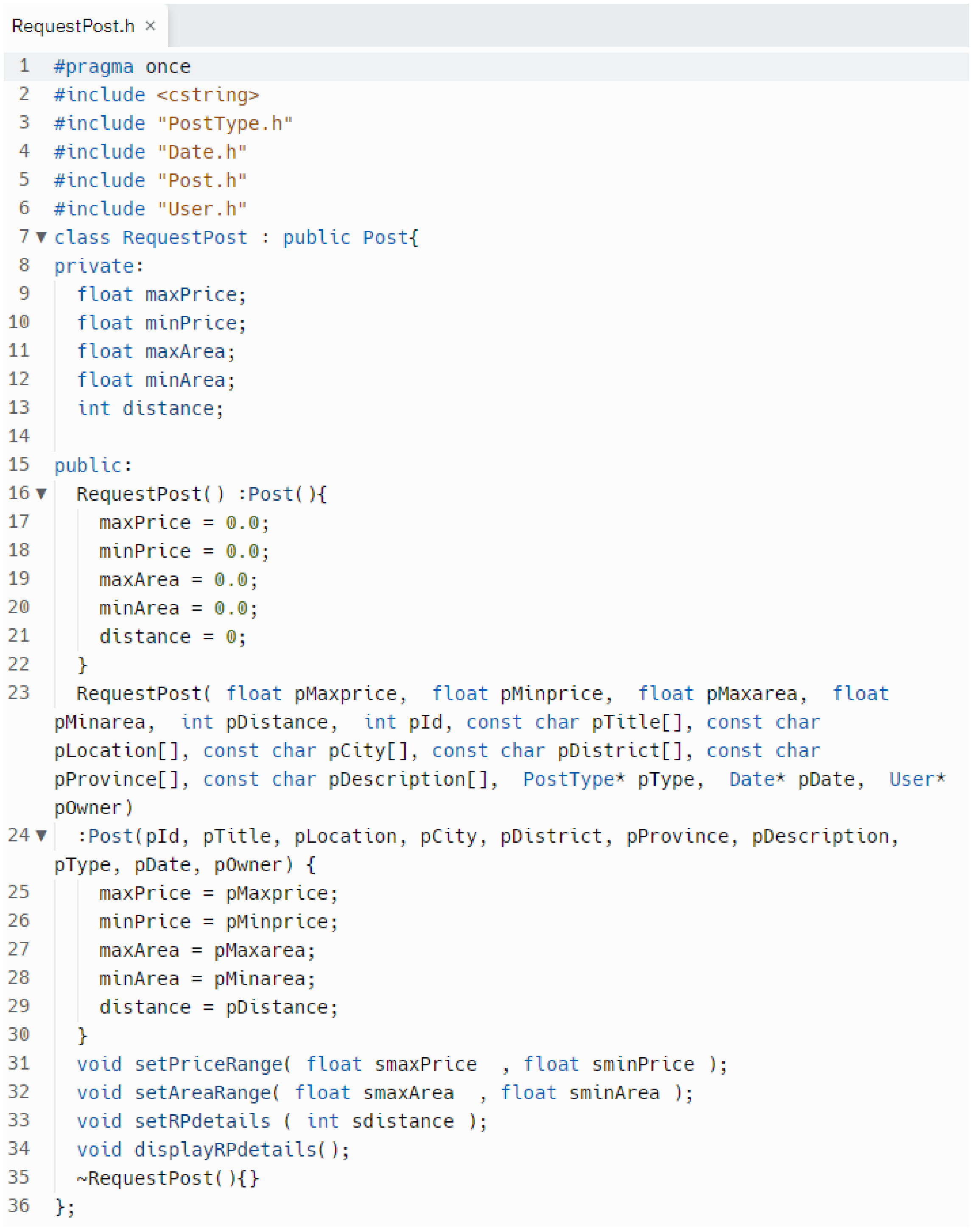
```cpp
#pragma once
#include <cstring>
class Photo{
private:
  int id;
  char location[100];
  float size;

public:
  Photo(){
    id = 0;
    strcpy(location,"default");
    size = 0.0;
  }
  Photo(int pId, const char pLocation[], float pSize)
  {
    id = pId;
    strcpy(location,pLocation);
    size = pSize;
  }

  void displayPhoto();
  void showDetails();
  ~Photo(){}
};
```


Post.h ×

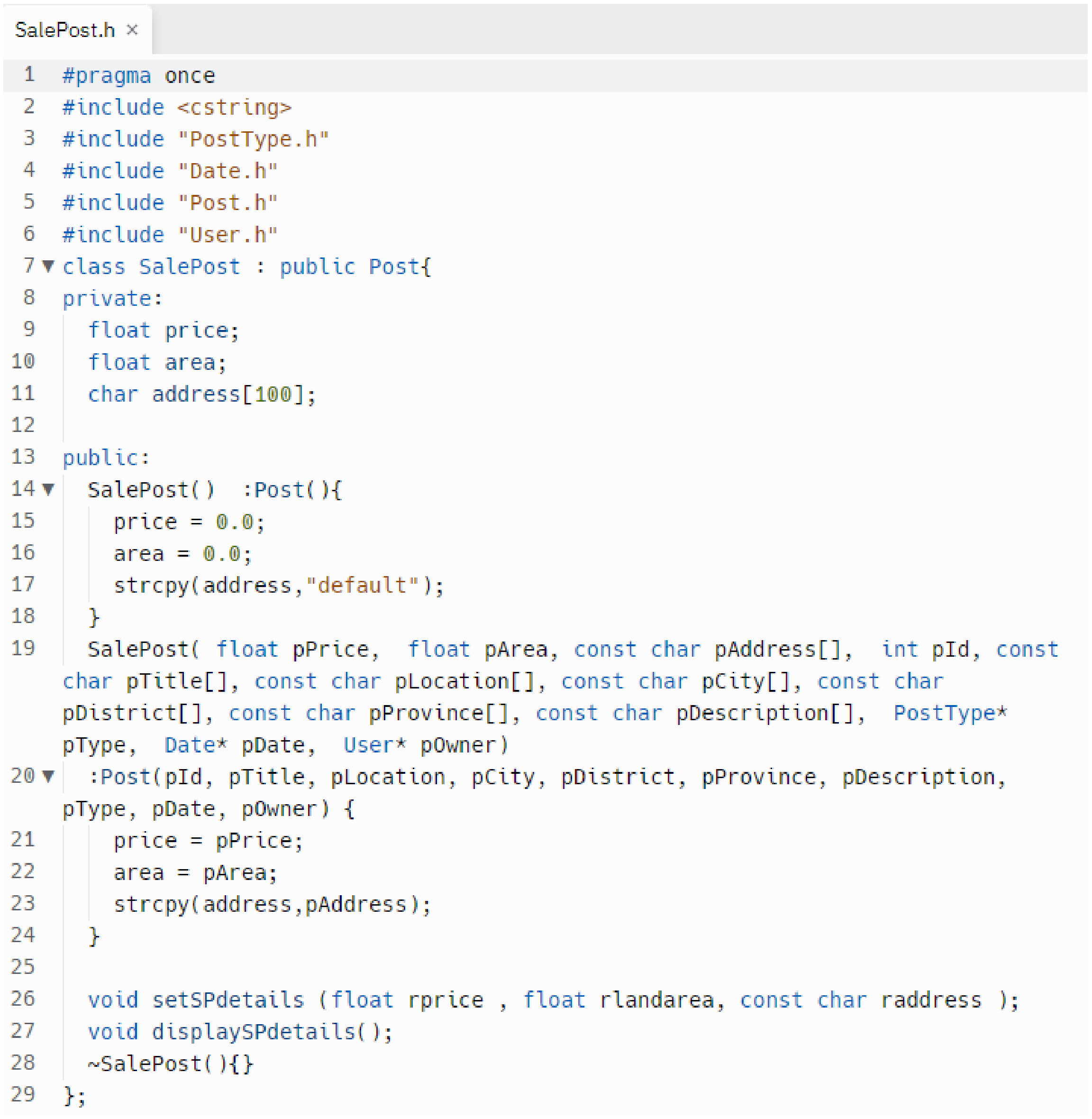
```cpp
#pragma once
#include <cstring>
#include "PostType.h"
#include "Date.h"
#include "User.h"
#include "Payment.h"
#include "Complaint.h"
#include "Photo.h"
#define compSIZE 50
class Post{
protected:
  int id;
  char title[100];
  char location[50];
  char city[100];
  char district[100];
  char province[100];
  char description[500];
  PostType* type;
  Date* date;
  User* owner;
  Complaint* complaint[compSIZE];

public:
  Post(){
    id = 0;
    strcpy(title,"default");
    strcpy(location,"default");
    strcpy(city,"default");
    strcpy(district,"default");
    strcpy(province,"default");
    strcpy(description,"default");
    type = new PostType();
    date = new Date();
    owner = new User();
  }
  Post( int pId, const char pTitle[], const char pLocation[], const char pCity[], const char pDistrict[],
const char pProvince[], const char pDescription[],  PostType* pType,  Date* pDate,  User* pOwner)
  {
    id = pId;
    strcpy(title,pTitle);
    strcpy(location,pLocation);
    strcpy(city,pCity);
    strcpy(district,pDistrict);
    strcpy(province,pProvince);
    strcpy(description,pDescription);
    type = pType;
    date = pDate;
    owner = pOwner;
  }
  void setPostId(int pid);
  int getPostId();
  void setPostDetails();
  void addComplaints(Complaint *c1,Complaint *c2);
  void addDate();
  void addpostType(PostType *pType1);
  void addPhotos(Photo *p1,Photo *p2,Photo *p3,Photo *p4,Photo *p5);
  void displayPostDetails();
  void receivePayment(Payment payment);
    ~Post(){}
};
```


PostType.h ×

```
#pragma once
#include <cstring>
class PostType{
private:
  int id;
  float price;
  char name[100];
  char description[500];

public:
  PostType(){
    id = 0;
    price = 0.0;
    strcpy(name,"default");
    strcpy(description,"default");
  }
  PostType(int pId, float pPrice, const char pName[], const char pDescription[])
  {
    id = pId;
    price = pPrice;
    strcpy(name,pName);
    strcpy(description,pDescription);
  }

  void setDetails(int pTypeID, float pPrice, const char pNamep[], const char
pDescription[]);
  void showDetails();
  float calculatePrice();
  ~PostType(){}
};
```

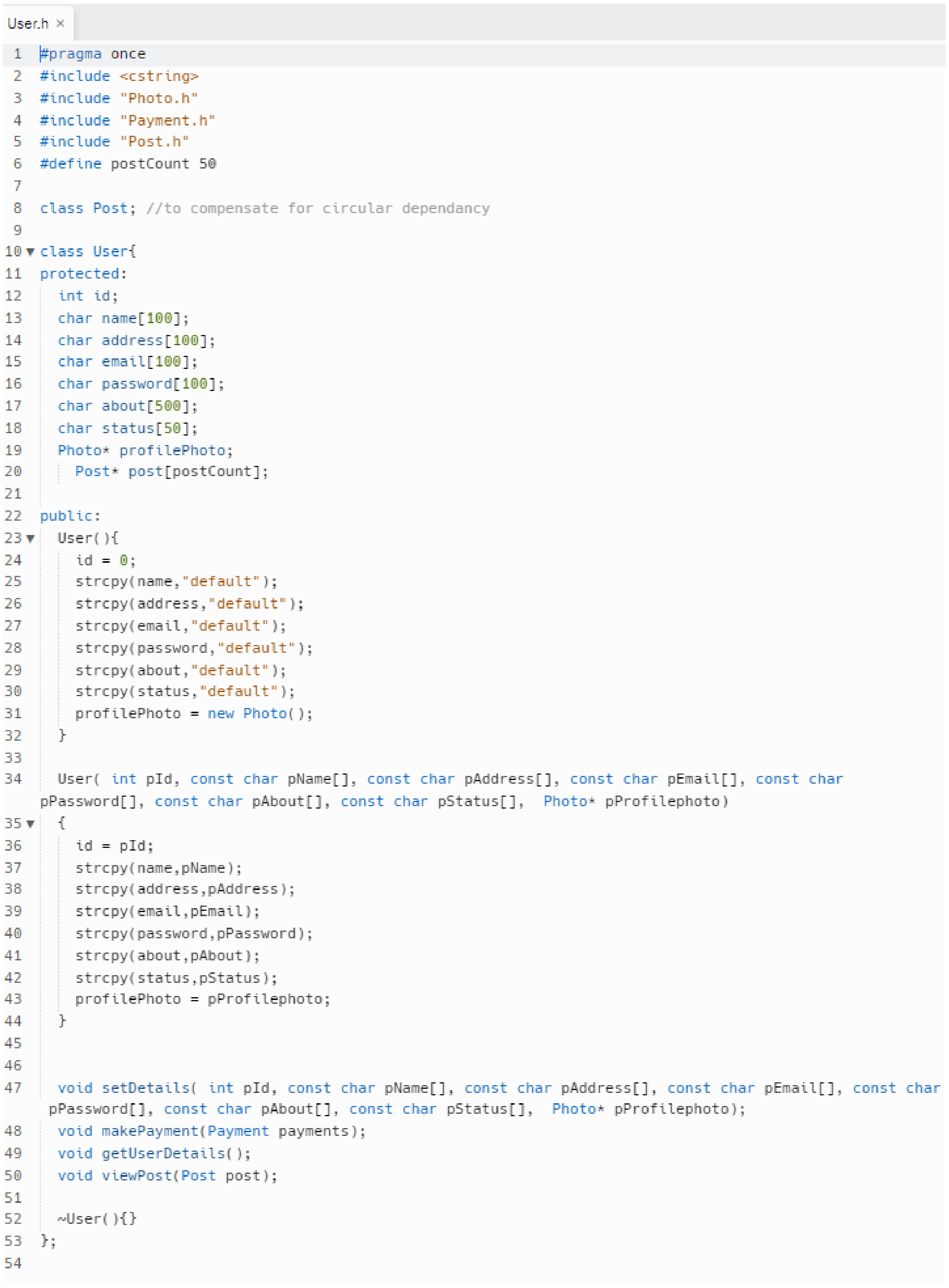

RequestPost.h ×

```cpp
#pragma once
#include <cstring>
#include "PostType.h"
#include "Date.h"
#include "Post.h"
#include "User.h"
class RequestPost : public Post{
private:
  float maxPrice;
  float minPrice;
  float maxArea;
  float minArea;
  int distance;

public:
  RequestPost( ) :Post( ){
    maxPrice = 0.0;
    minPrice = 0.0;
    maxArea = 0.0;
    minArea = 0.0;
    distance = 0;
  }
  RequestPost( float pMaxprice,  float pMinprice,  float pMaxarea,  float 
pMinarea,  int pDistance,  int pId, const char pTitle[], const char 
pLocation[], const char pCity[], const char pDistrict[], const char 
pProvince[], const char pDescription[],  PostType* pType,  Date* pDate,  User* 
pOwner)
  :Post(pId, pTitle, pLocation, pCity, pDistrict, pProvince, pDescription, 
pType, pDate, pOwner) {
    maxPrice = pMaxprice;
    minPrice = pMinprice;
    maxArea = pMaxarea;
    minArea = pMinarea;
    distance = pDistance;
  }
  void setPriceRange( float smaxPrice  , float sminPrice );
  void setAreaRange( float smaxArea  , float sminArea );
  void setRPdetails ( int sdistance );
  void displayRPdetails( );
  ~RequestPost( ){}
};
```



SalePost.h ×

```cpp
#pragma once
#include <cstring>
#include "PostType.h"
#include "Date.h"
#include "Post.h"
#include "User.h"
class SalePost : public Post{
private:
  float price;
  float area;
  char address[100];

public:
  SalePost()  :Post(){
    price = 0.0;
    area = 0.0;
    strcpy(address,"default");
  }
  SalePost( float pPrice,  float pArea, const char pAddress[],  int pId, const char pTitle[], const char pLocation[], const char pCity[], const char pDistrict[], const char pProvince[], const char pDescription[],  PostType* pType,  Date* pDate,  User* pOwner)
  :Post(pId, pTitle, pLocation, pCity, pDistrict, pProvince, pDescription, pType, pDate, pOwner) {
    price = pPrice;
    area = pArea;
    strcpy(address,pAddress);
  }

  void setSPdetails (float rprice , float rlandarea, const char raddress );
  void displaySPdetails();
  ~SalePost(){}
};
```


User.h ×

```
#pragma once
#include <cstring>
#include "Photo.h"
#include "Payment.h"
#include "Post.h"
#define postCount 50

class Post; //to compensate for circular dependancy

class User{
protected:
  int id;
  char name[100];
  char address[100];
  char email[100];
  char password[100];
  char about[500];
  char status[50];
  Photo* profilePhoto;
    Post* post[postCount];

public:
  User(){
    id = 0;
    strcpy(name,"default");
    strcpy(address,"default");
    strcpy(email,"default");
    strcpy(password,"default");
    strcpy(about,"default");
    strcpy(status,"default");
    profilePhoto = new Photo();
  }

  User( int pId, const char pName[], const char pAddress[], const char pEmail[], const char
pPassword[], const char pAbout[], const char pStatus[],  Photo* pProfilephoto)
  {
    id = pId;
    strcpy(name,pName);
    strcpy(address,pAddress);
    strcpy(email,pEmail);
    strcpy(password,pPassword);
    strcpy(about,pAbout);
    strcpy(status,pStatus);
    profilePhoto = pProfilephoto;
  }


  void setDetails( int pId, const char pName[], const char pAddress[], const char pEmail[], const char
 pPassword[], const char pAbout[], const char pStatus[],  Photo* pProfilephoto);
  void makePayment(Payment payments);
  void getUserDetails();
  void viewPost(Post post);

  ~User(){}
};
```

6. Individual Contribution

Registration No	Name	Contribution	
		Class	Tasks
it21294198	Karunarathna K. R. M. R. T.	Payment	CRC card, UML and header file
it21296246	Hapuarachchi H. A. R. S.	Date, PostType, Photo, Complaint	CRC cards, UML, and C++ codes
it21292668	Herath H.M.N.R.	Sale Post, Request Post	CRC cards, UML and C++ codes.
it21294648	Wickramarathne M.D.M.C.L.	User, Admin	CRC cards, UML, C++ codes, Description
it21292422	Akmal M.A.M.	Post, Moderator	CRC cards and class diagram for post and moderator