,

RMIT UNIVERSITY

EEET2482/COSC2082 – Software Engineering Design, Advanced Programming Techniques.

Assignment Report

"TIME BANK" APP FOR DAILY ACTIVITIES

Lecturer: Mr. Linh Tran

Team Members:

Nguyen Vinh Gia Bao (s3986287)

Le Viet Bao (s3979654)

Truong Phung Tan Tai (s3974929)

Tran Phu Van (s3822040)

Date : 20/01/2024

I. INTRODUCTION

This assignment offers students an opportunity to put into practice and implement all knowledge and concepts they have learned in the Software Engineering Design course so far. This also helps students familiarize with the analysis, design and development process involved in creating a software application through a practical project idea, equipping them for real life challenges.

This task requires students to design and implement a C++ programming language to make an app called Time Bank for people to support each other regarding daily activities such as plumbing repairs, tutoring and house cleaning. For simplicity, all interaction associated with the application will be conducted merely through a basic text interface, excluding the demand of GUI. However, the problem must accommodate all fundamental functionalities of a practical application. For instance, member registration, list of skills, member availability, booking request, member block, member and skill rating are some of the essential features that need to be included in the application. All features and operating behaviors will be further discussed and explained in below sections.

II. APPLICATION DESIGN AND DEVELOPMENT

1. Software Design (Class Diagram)

From the application description, provide class diagram and description/ explanation as below:

Class Diagram:

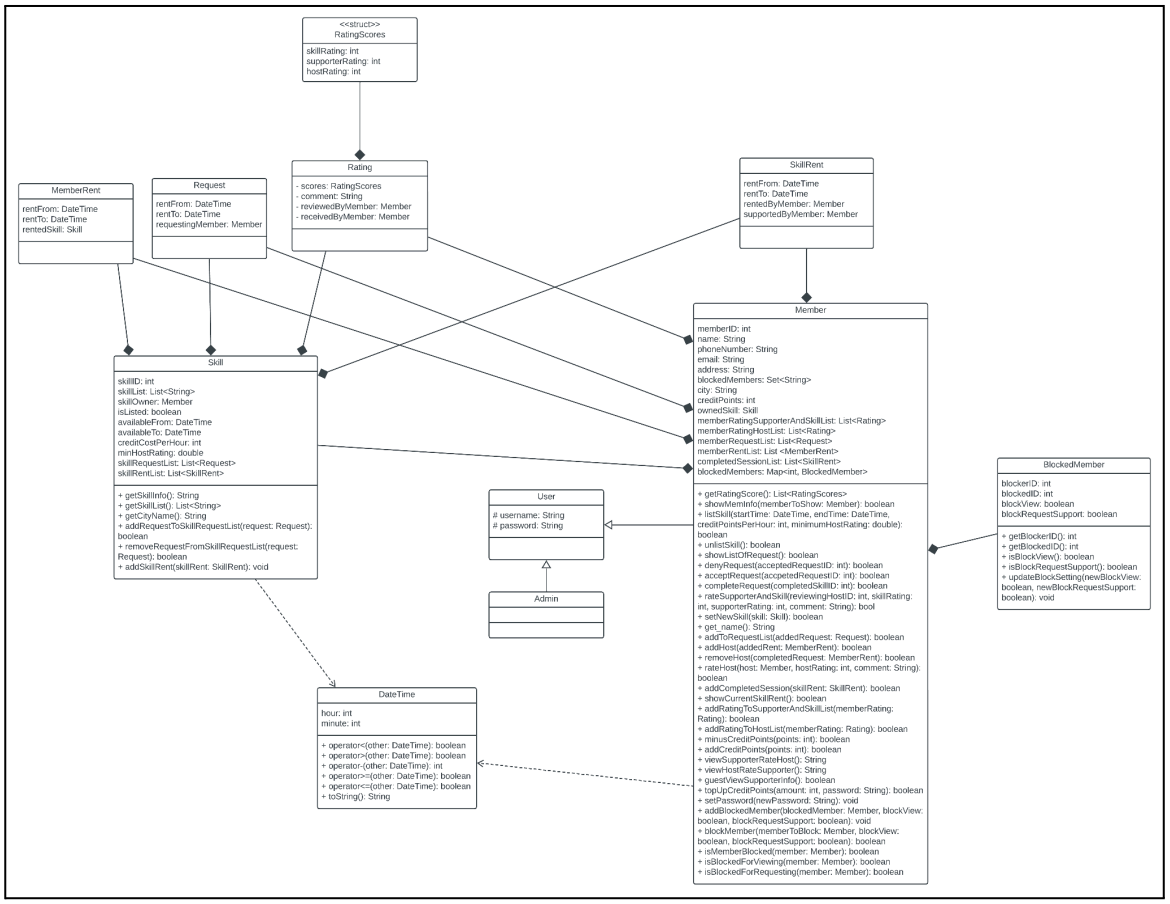

Figure 1: Class Diagram

Description of Each Class:

Provide descriptions for each class using the table below:

Class Name		Name and Data type	Description	Reason/Explanation (*why we need it as an attribute/ method*).
MemberRent: This class is responsible for storing data of the time slot members spend on seeking skills for support.	Attribute(s)	rentFrom: DateTime*	To keep track with the total amount of time the member spent on each rental	This helps the system register the rental period of that member, hence preventing them from having another rent within the designated time frame.
		rentTo: DateTime*		
		rentedSkill: Skill*	Store the information of the rented skill in the database	This indicates the type of activities the host seeks for support.
SkillRent: This class stores data of the renter and the time duration to rent a certain skill.	Attribute(s)	rentFrom: DateTime*	Update the rental period of the skill being served	The system records the timeframe of the rented skill, hence preventing other users from hiring this skill within this specified period
		rentTo: DateTime*		
		rentedByMember: Member*	Store the information of the host renting the skill	This provides the supporter information about the host who hires them.
		supportedByMember: Member*	Store the information of the supporter who performs the skill	This provides the information about the supporter who did the work, useful for CompletedSessionList
Request: This class is used to save data of member's request for skill rental	Attribute(s)	rentFrom: DateTime*	The period of time a member request for skill rental	The supporter will be notified of the number the requests sent to them and the time slots, helping them manage and avoid clashes between time slots. Moreover, the system can compute credit points based on the renting time frame in the request
		rentTo: DateTime*		
		requestingMember: Member*	The information of members who sent rental requests	As supporters receive requests, they will have details of the requestors (hosts), including their rating, minimum required host rating, and renting history; therefore, making sure supporters won't be booked by poor hosts.

Rating:	Attribute(s)	scores: RatingScores	Reserve the review details regardless of whether it was evaluated by the previous supporters or hosts	The system will gather all historical scores corresponding to supporters or hosts and give out the average rating scores. If the supporter is being reviewed, hosts will have comprehensive insights into the level of performance supporters provide. As a result, hosts can make proper decisions in choosing valid supporters. And vice versa, the supporters can rely on the average rating scores of hosts to determine for whom they will provide their services.
		comment: string		
		reviewedByMember: Member*	Store the data of who had commented	The member who gives the review to other members, be it a host or a supporter
		receivedByMember: Member*	Store the data of who received the rating	The member who receives the review by other members, be it a host or a supporter
RatingScore:	Attribute(s)	skillRating : int	Identify the rating given for the skill that was rented	This is used to rate and assess the quality of the skill. Therefore, it is crucial for future users to choose which skill has a high performance.
		supporterRating: int	Represent the rating given to supporter	This allows the host to monitor the performance of the support with their assistance.
		hostRating: int	Represent the rating given to the host that the supporters rate the host who rents them.	The rating of the host based on the averaging ratings of supporters for their treatment and further feedback.
DateTime: This class handles operations related to time data.	Attribute(s)	hour: int	Keep track of when skillrent or member are available when request are made within the rage of time	Tracks the hour and minute component of a time. Necessary for recording precise times of availability, requests, or rentals .Important for fine-grained time tracking and ensuring accuracy in scheduling within the system
		minute: int		

	Method(s)	operator<(other:DateTime) : boolean	Determines if this DateTime is earlier than another.	Maintaining chronological order in processes is crucial for comparing two events in time. (i.e. to find available supporters, reject requests that overlapped)
		operator>(other:DateTime) : boolean	Determines if this DateTime is earlier than another.	
		operator<=(other:DateTime) : boolean	Determines if this DateTime is exactly or earlier than another.	
		operator>=(other:DateTime) : boolean	Determines if this DateTime is exactly or later than another.	
		operator-(other:DateTime) : int	Calculates the difference in time between this and another DateTime.	calculate the difference between the duration of time available. This might help to calculate the skill available for how long a skill has been rented or the time until a rental starts, which is crucial for billing, availability if the member makes a request.
		toString(): string	Convert a DateTime instance into a string representation of date and time values and store it.	It displays the DateTime in format with hour and time within the system
Skill: This class is used to manage and store the skill information including time period, owner, rating points, request,cityID.	Attribute(s)	skillID: int	Unique identifier for each skill and store the skillID	Using the skillId, the system can make it possible for abilities to be uniquely identified and retrieved inside the system. Moreover, the skillID where it is stored as CSV files, serves as a unique for each skill. When loading or updating data, this allows for efficient searching and matching within the CSV structure.When a member rents a skill, having a skillID allows for quick referencing and updating the relevant records.
		skillList: List <String>	A list of skill names and different types of skill available and stored it	Provides a collection of skills that can be offered or requested by users ,and rental options that are appropriate for their skill level

		skillOwner: Member	Member who rent or required this skill	Associate a member's profile with the specific skill they have listed during registration. Moreover, it can be used to keep a record of each member's skills which can verify that members are capable of providing the service they offer.
		isListed: boolean	Regardless of whether the skill is listed, store it.	When isListed is set to true, it means that the person is available for bookings by other members who want assistance in that specific area and that they are actively providing that skill.
		availableFrom: DateTime	Define the timeframe available to perform a specific skill.	The time range for availability which the member can book for that skill. This ensures that the service can be served at that time which is convenient for them, and manage that it cannot overbook.
		availableTo: DateTime		
		creditCostPerHour:int	Keep track of how many points you consume each hour.	Represent the number of credit points that the owner charges per hour to perform a certain skill
		minHostRating :double	Record the rental's minimum rating.	Indicate the lowest host-rating score they are willing to accept from other members who wish to book them for a particular skill.The hosts rate the supporter for the skill performed, as well as their overall performance and kindness.Similarly, supporters rate the hosts on how well they were treated during their interaction
		skillRequestList:List< Request>	Record of request objects that member request for a specific skill	Manages and tracks the demand for a particular skill
		skillRentList:List<Skill Rent>	Record of the owner who rent a skill	A list of past and current rentals or bookings where the skill has been utilized including duration , consuming points, rating. This would record the historical record of all instances where the skill was rented out or used. As a result, it can record-keeping the service

				history to see the skill's utilization rate.
	Method(s)	getSkillInfo: string	Contain the information include name,rating,description of the particular skill	This provides users with detail about the skill when choosing services.
		getCityName: string	Contain the location of the services in 2 places Ha Noi and Sai Gon	Indicate where the skill is available or where it is located for the owner at that particular place.
		addRequestToSkillRequestList(request:Request): boolean	Contain Information about a service request such as skill ,member request, suitable time for the service.	This would indicate that the request has been added or failed to request. This will happen when the users submit the request and keep track of all active and pending requests.
		removeRequestFromSkillRequestList(request:Request): boolean	Delete the service requests with specific skill from the list	This returns 'true' when the request has been removed from the list and false when failed to delete. It helped to manage and update the status of requests.
		addSkillRent(skillRent:SkillRent): void	Add the new skill that has rent to the SkillRentList	This will allow the system to associate rentals with the skill and update for a skill to keep an exact record for rentals.
		addCompletedSession(rentSession:SkillRent): boolean	Track the completed rentals	This allows the system to keep all data of the skill rental that have been completed by checking for a null and display an error message if it is found,
BlockedMember: Manage interactions between member (can block other member avoid conflict)	Attribute(s)	blockerID: int	Identify the member who initiated the block.	This indicates a member who has been blocked by another member for some reason. This can manage conflict if they have any issues. As a result, it is crucial to maintain a safe ~~application and control~~ privacy for users.
		blockView: boolean	Restrict other members from viewing	
		blockRequestSupport	Restrict other members from requesting for support or communications to the blocker	

		blockedID : int	Identify the member who initiated the block.	
	Method(s)	getBlockerID : int	Return the ID of the member who has blocked the other.	These methods are to identify the member who initiated the block and which member has been blocked.
		getBlockedID : int	Return the ID of the member who has been blocked	
		isBlockView(): boolean	Check if the view is restricted for blocked member	This allows the system and member who has blocked to verify the state of viewing is blocked or not
		isBlockRequestSupport() : boolean	Check if the blocked member is restricted from requesting support	This method ensures that the blocker can manage the interactions.
		updateBlockSetting(newBlockView: boolean, newBlockRequestSupport: boolean): void	Update the attributes of blockView and blockRequestSupport to new value and store it	This allows flexibility to update and control their privacy in users settings.
Member: This class serves the purpose of storing all the essential data of a certain user.	Attribute(s)	# username: string	The required name for account login	These attributes are inherited from class User and it is compulsory for the system authentication
		# password: string	The required password for account login	
		memberID: int	Member ID	These attributes are indicating the personal information of a member and these information will be stored in the system.
		name: string	Member's name	

		phoneNumber: string	Member's phone number	
		email: string	Member's email	
		address: string	Member's address	
		blockedMembers: set<string>	The list of specific sorted order of members being blocked	
		city: string	Member's location	
		creditPoints: int	Member's credit point	This is also a piece of personal information but it can vary depending on the member's current status. Members can earn creditPoints if they do favors to others or they can lose credit points if asking other members for help.
		ownedSkill: Skill*	Member's skills	This attribute tells information of how many and what kind of skills a member has
		memberRatingSupporterAndSkillList: List<Rating*>	The list of information that one member gave to another supporter, including comments and rating scores	This feature helps gather all the comments and rating scores of a member being reviewed by others. This information will be systematically stored in a list and used for measuring member's average rating score
		memberRatingHostList: List<Rating*>	The list of information that one member gave to another host, including comments and rating scores	
		memberRequestList: list<Request*>	The list of requests sent by members to another ones (for example, hosts send requests to supporters)	The system puts all the requests sent to a member into a list. List is the class of the List container, providing sequential access; hence avoiding duplicating requests.
		completedSessionList :List<SkillRent>	Contain the completed session together with the hostID, and the supporterID	The vector is used to represent the list of completed sessions after the supporter finished the request, from this list, the system can keep track and knows which host and

				supporter is related to the service
		blockedMemberList: list<MemberBlock*>	This is a list of the members being blocked by other ones	The system will register these blocked members, preventing them from viewing information or requesting support of the blocking ones.
	Method(s)	getRatingScore(): list <RatingScores>	Obtain a member's average rating scores	The system will measure a member's average rating score based on multiple rating scores stored inside the list. This result can be further viewed by other members, helping have a valid assessment.
		shownMemInfo(memberToShow: Member): boolean	Show all detail of a member	The member can view all of his/she information after the login process.
		listSkill(startTime: *DateTime, endTime: *DateTime, creditPointsPerHour: int, minimumHostRating: double): boolean	List all the information related to that skill	A member can list himself/herself to be available on particular periods in which he/she can offer skills for rent. The cost of credit points per hour when someone books the provided service. Moreover, a minimum host rating score is required for having desired services.
		unlistSkill(): boolean	Remove the skill from the list	This feature provides members (supporters) an option to remove their unwanted skills to perform.
		showListOfRequest(): boolean	Display all the requests being sent by members (hosts) who wanted that skills	The supporter can view how many requests have been sent for each skill they have.
		denyRequest(acceptedRequestID: int): boolean	Remove the undesired request from the list	Once a request approval is made, other requests that clash with that one will be automatically rejected.
		acceptRequest(acceptedRequestID: int): boolean	Accept the request among the list	The supporter accepts the host's request of performing a certain service and since then he/she will not be allowed to cancel the request.
		completeRequest(completeSkillID: int): boolean	Restore the service once it is done	When the service is done, the specific skill and its time slot will be restored available for further bookings.
		rateSupporterAndSkill(reviewingHostID: int, skillRating: int, supporterRating: int, comment: string): boolean	The host will assess the service quality regarding various criteria.	Hosts assess the service provider based on criterion of skill rating, supporter rating and comment.

		setNewSkill(skill: Skill*): boolean	Add a new skill into the skill list	The system creates a skill list to which members can add new skills.
		get_name(): string	Obtain member's name	Collect member's name for better clarification in terms of who is using whom service and vice versa
		addToRequestList(addedRequest: Request*): boolean	Add a request into the request list.	The request will be added into the list along with existing ones, and it will be sequentially arranged.
		addHost(addedRent: MemberRent*): boolean	Add host to the particular period of the service they requested	When the host's request is approved, they will be allocated to their desired session.
		removeHost(completedRequest: MemberRent*): boolean	When the service is done, remove the host from the service time frame	This function will remove the time frame of the host's appointment after finishing the service
		rateHost(host: Member*, hostRating: int, commnet: string): boolean	The supporter will rate the host	After performing a service, supporters can rate hosts and leave comments for them.
		showCurrentSkillRent(): boolean	Show the current rented skill	The system will display which skill is being rented in a specific period and the information of the host
		addRatingToMemberRentList(memberRating: Rating*(): boolean	Include rating score and commnet of a member to the Rating database	As supporters gave reviews towards hosts, this information will be added and stored in the specified database. This allows supporters to view all the existing reviews they have made
		minusCreditPoints(points: int): boolean	Deduct credit points from member's balance whenever members have their services done by others	When hosts ask supporters to performance their desired service, hosts' credit points will be deducted corresponding to the service point consumption
		addCreditPoints(points: int): boolean	Add credit points the supporter's balance whenever they have done a requested service	The cost the hosts pays for services will be transferred to supporters' balance account.
		viewSupporterRateHost(): string	Display the review that supporters gave to hosts	The system will check the member rating list to see whether the specified hosts have been rated before or not. If found, display rating scores and comments to that hosts
		viewHostRateSupporter(): string	Display the review that hosts gave to supporters	From the member rating list, the system displays the reviews the supporters have received from hosts, including supporter rating

				scores, skill scores and comment
		guestViewSupporterInfo(): boolean	Non-members view supporter's details	This function allows non-members to review all supporter's details without the ratings (comments and scores), hence encourages non-members to register to become a member.
		setPassword(newPassword: string): boolean	Create member's newpassword	The system will replace the password the new one and record it in the database
		topUpCreditPoints(amount: int, password: String): boolean	Add more credit points to use the service	The member can top up their credit points and confirm the transaction by entering their password
		addBlockedMember(blockedMember: Member*, blockView: boolean, blockRequestSupport: boolean): void	Add the requests for blocking certain members	The function will check the member's ID that would be blocked and then create a map and store these will-be-blocked IDs in it.
		blockMember(memberToBlock: Member*, blockView: boolean, blockRequestSupport: boolean): boolean	Block other members that had conflicts	When the blocking request is approved, based on the ID, the system will block those specified members, preventing them from reviewing or making requests. Then it will store these blocked IDs in a specified map.
		isMemberBlocked(member: Member*): boolean	Check if certain members are blocked	This system grants members information regarding whether he/she was blocked by others or to confirm the one that needs to be blocked, has already been blocked.
		isBlockedForViewing(member: Member*): boolean	Check if members could be able to view or make requests or not.	This function is designed to validate whether the blocking function is working well or not.
		isBlockedForRequesting(member: Member*): boolean		
Admin	As a part of the system, by inheriting from 'User', the Admin object can access and modify the member's password so it doesn't have any special functions.			

Class Relationships:

- Class User is the base class: the common attributes like 'username' and 'password' which are required to use the member account .

- Class Admin is a subclass of the Class User: therefore also has all the characteristics of a parent class User including ‘username’ and ‘password’. An ‘Admin’ being a specialized ‘User’ will have permissions to change and manage ‘Member’ data, ‘Skill’ data.
- The class Skill is composed of class Members that can indicate the skill management, processing request, contain or manage lists of objects and also rating with each skill. If a ‘Member’ is deleted, ‘Skill’ is also deleted.
- The class Request is a composition of the class Skill. A member will make a request to rent a skill if the skill is available to the member. The request would be created when a skill is needed and if the skill is no longer offered or available, it wouldn’t exist.
- The Request is composed of class Member. A member will submit the Request for rental and manage request, so if the member is moved or deleted, the Request will be cleared.
- The class Rating is composed of the ‘Member’ class which indicates that a rating cannot exist without a member. In case the ‘Member’ is deleted, then the ‘Rating’ would also be deleted.
- The Rating is a composition of Skill class which is used to comment or evaluate through feedback and including the data given to the skill from the member who provided rating. If the ‘Skill’ objects are destroyed, the ‘Rating’ objects would also be deleted.
- The MemberRent is composed of the class ‘Member’ which represents for recording a rental transaction including the time and the skill that the members are currently renting and storing the data.
- The MemberRent is a composition of Skill class, which indicates the information about the rentals including the time, and the skill has rented. When the Member is cleared, it will affect the MememberRent would also be deleted.
- The SkillRent is a composition of the Member class. The SkillRent class indicates a transaction where a skill is temporarily transferred from one member to another. The ‘SkillRent’ would track the details like the period for which skill is rented and the ‘Member’ who is renting out their skill. The SkillRent is also used in the Skill class to keep track of the completed session list, as this is vital for the host who would rate the supporter and skill later on after the supporter completed the request.
- The DateTime class has a dashed line and is connected from the Skill and Member class which typically are used to represent a dependency. The Skill and Member class used DateTime for time-related attributes and operations.
- The RatingScore is composed of the ‘Rating’, which is used to hold the score for different aspects of rating, typically cannot exist without the ‘Rating' object. If the ‘Rating’ object is destroyed, the ‘RatingScore’ object would also be removed.
- The BlockedMember is a composition relationship of the ‘Member’, it cannot exist without a member. It is created when a member is blocked, when a member is

unblocked, it would cease to exist or be deleted. If a 'Member' is removed from the system, the MemberBlock would be removed.

2. Implementation Result

Describe implementation results with proofs (screenshot pictures), and discuss any limitation if it has.

a. Welcome Screen

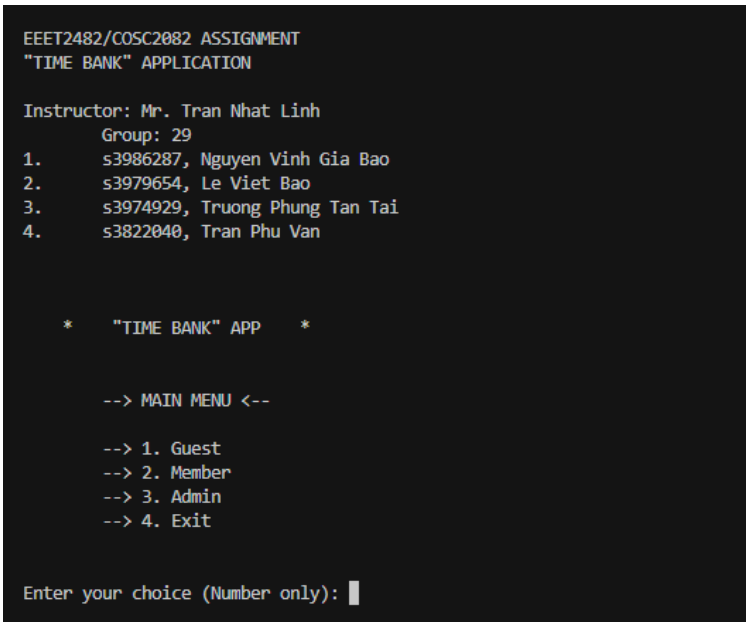

Figure 2: Welcome Screen

b. Basic Features

Result Summary: *Fill in the table below*

Feature Name	Feature Description	Status (*Implemented/ Not Implemented*)	Bugs/Limitations if it has
1. Non-member does registration	A non-member can register to become a member (information is recorded).	Implemented	N/A
2. Non-member view all supporter details	A non-member can view all supporter details (but not their reviews).	Implemented	N/A
3. Login with Admin account	An admin can login with a predefined username and password, and can reset password for any member.	Implemented	N/A
4. Login with registered member	A member can login with the registered username and password, and can view all of his/her information	Implemented	N/A
5. Member availability list	A member can list himself/herself available to be booked (including skills	Implemented	N/A

	can be performed, consuming points per hour, with/without minimum required host-rating score), and unlist if wanted.		
6. Available suitable supporters in specific cities	A member can search for all available suitable supporters for a specified city (suitable with his current credit points and rating score).	Implemented	N/A
7. Booking request	A member can request to book a supporter	Implemented	N/A
8. Member request list	A member can view all requests to his listed skills	Implemented	N/A
9. Request consideration	A member can reject or accept a request (reject other requests)	Implemented	N/A
10. Block member	A member can block another member from viewing his/her information or requesting support.	Implemented	N/A
11. Supporter rating points	A member as a host can rate his/her supporter (score and comments)	Implemented	N/A
12. Host rating points	A member as a supporter can rate his/her host (score and comments)	Implemented	Needs to exit the program and run it again to view the currently supported skills list, and then complete the request
13. System database	Data must be saved into data file(s) before the program is ended, and loaded into the program when it is started.	Implemented	N/A

Screenshots of Sample Result (for each feature):

Feature 1

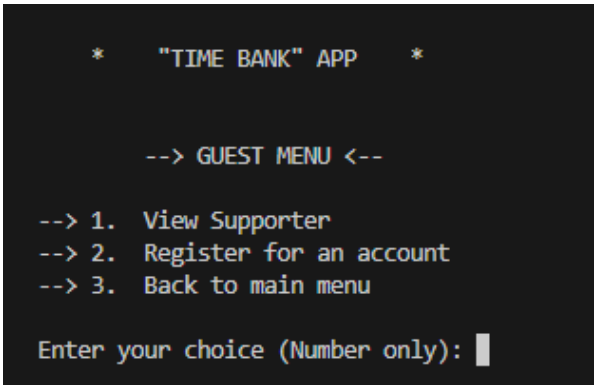

Figure 3: Guest Menu

```
*    "TIME BANK" APP    *


        --> MEMBER REGISTRATION <--

Enter a username: ssetrmit1

Enter a password: cosc2082

Enter your first name: Brian

Enter your last name: Tran

Enter your phone number: 0779822135

Enter your email: brian14tran@gmail.com

Enter your address: 95 Nguyen Van Troi
Registration successful
Loading...
```

Figure 4: Member Registration Interface

Feature 2

```
                --> SUPPORTER LIST <--
ID    Information

1. Username: darkomarwin
2. Username: bronguyen
3. Username: ssetrmit1

4. Back to Guest Menu

Enter your choice (Number only): 1

You have chosen supporter number 1
+ Username: darkomarwin
+ Full name: Darko Marwin
+ Phone number: 0901877037
+ Email: darkomarwin@gmail.com
+ Address: 103 Nguyen Van Linh

+ Skill: Skill List: Gardening, Lawnmowing, City: Saigon

1. Back to supporter list
```

Figure 5: View Supporters' Detail (excluding reviews)

Feature 3

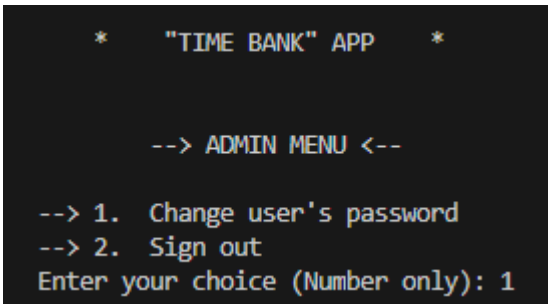

Figure 6: Admin Menu

Note: The admin's **username** is admin123 and password is **12345678**

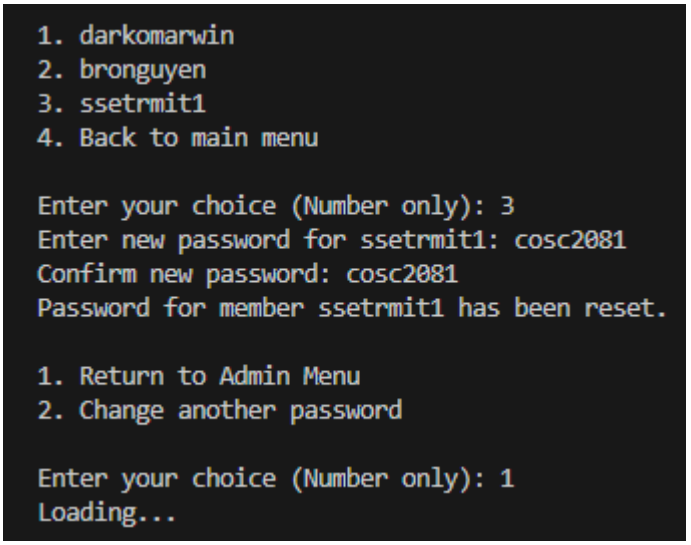

Figure 7: Admin changes user's password

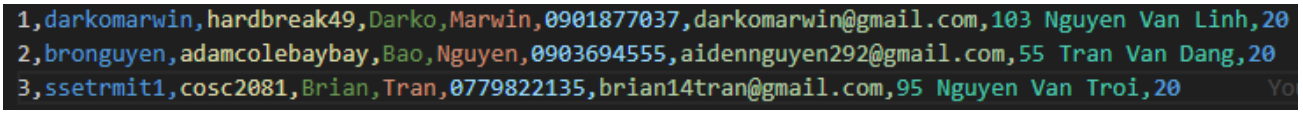

Figure 8: User number 3's password has been changed

Feature 4

```
        --> MEMBER MENU <--

--> 1.  View your information
--> 2.  View other members' information
--> 3.  Skill supporting menu
--> 4.  Search for available supporters
--> 5.  View skill request list
--> 6.  View currently supported skill
--> 7.  View completed session list
--> 8.  Block member
--> 9.  Top up credit points
--> 10. Sign out

Enter your choice (Number only): 1

+ Username: bronguyen
+ Full name: Bao Nguyen
+ Phone number: 0903694555
+ Email: aidennguyen292@gmail.com
+ Address: 55 Tran Van Dang
+ Credit points: 20
+ Owned skill: Skill List: Fishing, Seafaring, City: Hanoi
+ Average Skill Rating: 0
+ Average Supporter Rating: 0
+ Average Host Rating: 0

1. Back to member menu
Enter your choice (Number only):
```

Figure 9: Member views their own information

Feature 5

```
        --> LIST MENU <--

You haven't had any skills on our system yet!!
1. Enter your skill info
2. Back to member menu

Enter your choice (Number only): 1
Please enter your SKILL information

Enter the skill information:
 - Enter a skill (type 'done' to finish): Tutor
 - Enter a skill (type 'done' to finish): Math
 - Enter a skill (type 'done' to finish): English
 - Enter a skill (type 'done' to finish): done

Choose the city your skill is in:

--> 1.  Hanoi
--> 2.  Saigon

Enter your choice (Number only): 2



    *    "TIME BANK" APP    *


        --> LIST MENU <--

Your skill isn't listed

--> 1. List your skill
--> 2. Back to member menu

Enter your choice (Number only): 1

Enter listing information:
 - Start time(hh:mm): 09:00
- End date(hh:mm): 10:00
- Consuming points per hour: 10
Do you want to set a minimum required host-rating score? (yes/no): yes
- Minimum Host Rating (1.0-5.0): 2.5

Successfully listed skill




    *    "TIME BANK" APP    *


        --> LIST MENU <--

Currently listed skill:
Skill List: Tutor, Math, English, City: Saigon

--> 1.  Unlist skill
--> 2.  Back to member menu
```

Figure 10: Member lists their own skill

Feature 6 & 7

```
Search for a suitable skill:


Choose the city you want to search for:

--> 1. Hanoi
--> 2. Saigon

Enter your choice (Number only): 2
Enter start time (hh:mm): 09:32


Enter end time (hh:mm): 11:00
Skill availability is not defined for Skill List: Fishing, Seafaring, City: Hanoi
Checking time: Requested - 09:32 to 11:00, Available - 09:30 to 11:30
Checking time: Requested - 09:32 to 11:00, Available - 15:00 to 18:00
Requested time is outside the skill's available time.

The list of suitable supporters:
1. Supporter's Name: Bao Nguyen, Skill(s): Gardening, Pet Caring
2. Back to member menu

Enter your choice (Number only): 1


--> 1.  Request supporter

--> 2.  View supporter reviews

--> 3.  Back to skill list

Enter your choice (Number only): 1

        Request sent
```

Figure 10: Member can search for supporters with specified checks

Feature 8

```
    *    "TIME BANK" APP    *


         --> MEMBER MENU <--

--> 1.  View your information
--> 2.  View other members' information
--> 3.  Skill supporting menu
--> 4.  Search for available supporters
--> 5.  View skill request list (supporter only)
--> 6.  View currently supported skill (supporter only)
--> 7.  View completed session list
--> 8.  Block member
--> 9.  Top up credit points
--> 10. Sign out

Enter your choice (Number only): 5

Choose a request to view in detail:

The request list for your support:

1. 09:35 - 10:00, Member: bao le, Host Rating: 0

2. Back to member menu
```

Figure 11: Member can see their request list

Feature 9

```
The request list for your support:

1. 09:35 - 10:00, Member: bao le, Host Rating: 0

2. Back to member menu


Enter your choice (Number only): 1


-->     1. Accept the request
-->     2. View the reviews of this member
-->     3. Back to menu

Enter your choice (Number only): 2
This host has no reviews.



--> 1.  Accept the request

--> 2.  Back to menu list

Enter your choice (Number only): 1
Request accepted successfully
Loading...
```

Figure 12: Member can see their request list

Feature 10

```
    *    "TIME BANK" APP    *


        --> MEMBER MENU <--

--> 1.  View your information
--> 2.  View other members' information
--> 3.  Skill supporting menu
--> 4.  Search for available supporters
--> 5.  View skill request list (supporter only)
--> 6.  View currently supported skill (supporter only)
--> 7.  View completed session list
--> 8.  Block member
--> 9.  Top up credit points
--> 10. Sign out

Enter your choice (Number only): 8
Select a member to block:
1. darkomarwin
2. bronguyen
3. ssetrmit1
4. lebao123
5. Back to main menu

Enter your choice (Number only): 1
Do you want to block this member from viewing your information? (yes/no): Invalid i
nput. Please enter 'yes' or 'no': yes

Do you want to block this member from requesting your support? (yes/no): yesyes
Invalid input. Please enter 'yes' or 'no': yes
New block settings added successfully.
```

Figure 13: Member can block other members

Feature 11

```
    *    "TIME BANK" APP    *


        --> MEMBER MENU <--

--> 1.  View your information
--> 2.  View other members' information
--> 3.  Skill supporting menu
--> 4.  Search for available supporters
--> 5.  View skill request list (supporter only)
--> 6.  View currently supported skill (supporter only)
--> 7.  View completed session list
--> 8.  Block member
--> 9.  Top up credit points
--> 10. Sign out

Enter your choice (Number only): 7
Choose 1 for supporter, choose 2 for host
1. Supporter
2. Host
3. Back to member menu

Enter your choice (Number only): 2

Completed Session List:

--> 1. From: 09:35, To: 10:00, Supported By: Bao Nguyen, Hosted By: bao le
2. Back to member menu

Enter your choice (Number only): 1

+ Username: aidentran
+ Full name: Bao Nguyen
+ Phone number: 0764111255
+ Email: vinhbao123@gmail.com
+ Address: 145 Dien Bien Phu
+ Credit points: 30
+ Owned skill: Skill List: Gardening, Pet Caring, City: Saigon
+ Average Skill Rating: 5
+ Average Supporter Rating: 5
+ Average Host Rating: 0

-->     1. Rate this member
-->     2. Back to completed session list

Enter your choice (Number only): 1

--- Rate the Skill from 1 to 5 ---

Enter your choice (Number only): 5

--- Rate the Supporter from 1 to 5 ---

Enter your choice (Number only): 5

Enter a comment about this skill/supporter: nice
You have successfully rated the supporter and their skill(s)!Loading...
```

Figure 14: Host can rate supporters that they have supported

Feature 12

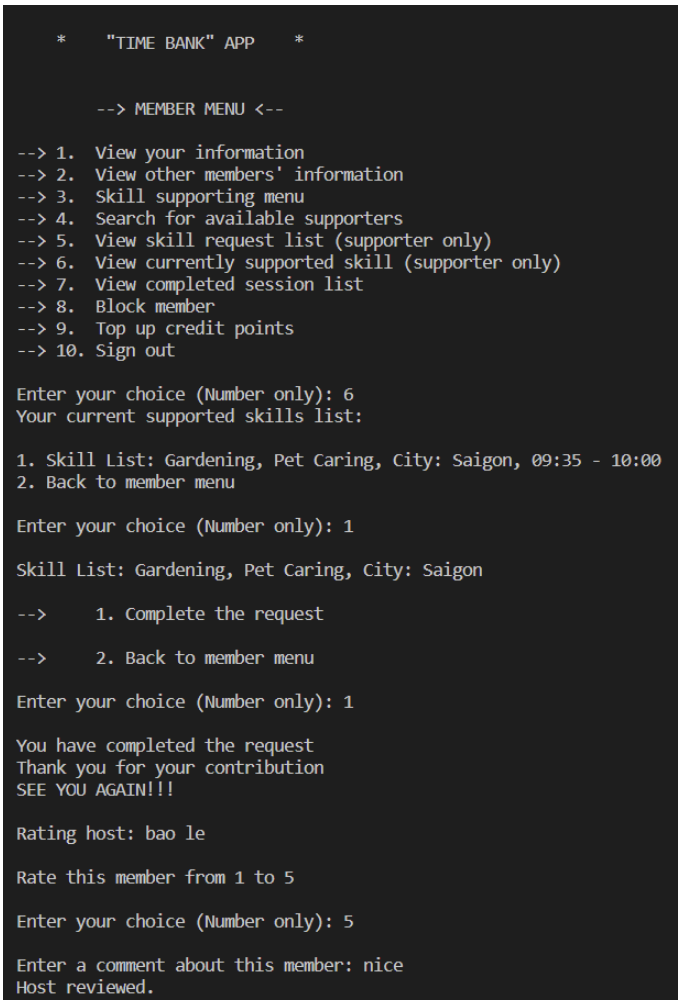

Figure 15: Supporter can rate hosts that they supported

c. Time Period Feature

Result Summary:

We were able to implement a DateTime class with custom formats and data validation, overloaded operators, and other methods to implement checks for overlapping sessions as well as to keep the users' experience consistent, as shown in Figure 16.

Screenshots of Sample Result:

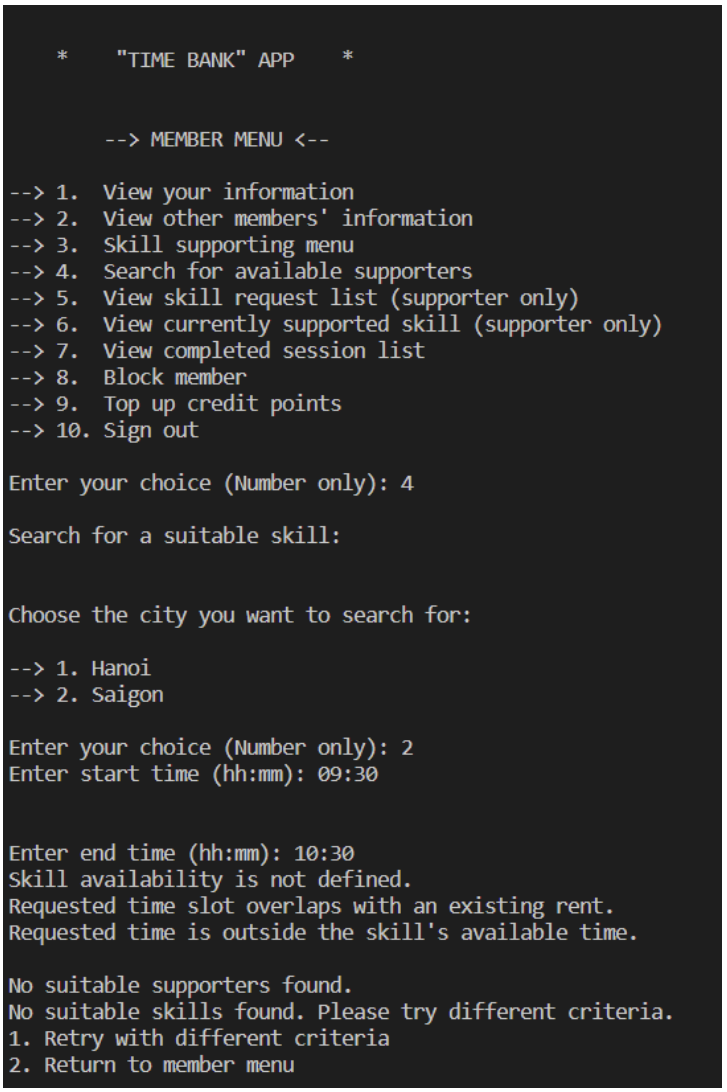

Figure 16: Implementation of time period feature

IV. DISCUSSION & CONCLUSION

In the final result of the “Time Bank” application, the design represented in the class diagram was successfully matched by the implementation.Throughout the implementation process, there were some bugs that had been fixed to meet all requirements of the system. These updates were necessary to accommodate specific requirements, optimize the application's performance, and enter input such as personal information, full name, email to confirm the user and store information date in csv file to avoid confusion during use. DateTime is used to supplement the time confirmed in requests, calculations,etc. All of which meet the proposed basic requirements.

Together, we first create a class diagram to have better visualization and understanding of the system structure, and the dependencies between classes (parent and child class) so that we can develop software based on the existing instructions. Having the design before implementation proves to be immensely helpful in guiding the development process. The class diagram gives an overall visual representation of the system's structure.

It helped ensure that the code was well-organized. During the process of coding for the application, several attributes and methods needed to be updated. These modifications were important to optimize the operation, or unanticipated difficulties that arose during the coding phase. By updating the design along the way, the implementation could align more closely with the intended functionality, resulting in a more robust and coherent application.

Throughout this assignment, we have gained valuable insights and skills in software engineering and application development. Through this project, we are able to apply concepts learned in the course, such as analysis, design, and development, to a real-world scenario. We accumulated knowledge on how to create a comprehensive class diagram, implement various features, handle user input, and manage data storage. The assignment also provided an opportunity to confront the complexities of incorporating the time period functionality into the application, improving the understanding of designing applications with specific temporal considerations. The knowledge and experience gained from this assignment are highly applicable to future studies and careers. It has provided us with useful abilities in software development, project management, and problem-solving. These abilities aid in our realization that in this era of technological development, we need to constantly update information and knowledge to keep up with constantly evolving technology trends. The ability to create and implement complex applications, manage project requirements, and adapt designs during implementation will be invaluable as we pursue a career in software engineering.

V. REFERENCES (USE IEEE STYLES)