EEET2482/EEET2653 Software Engineering Design
COSC2082/COSC2721 Advanced Programming Techniques
Semester 2, 2025

Group Project Report

E-MOTORBIKE RENTAL APPLICATION

Lecturer: Dr Tri Huynh

Group No. 5

Group Members:

Lu Duc Thinh (S3992133)

Jang Soohyuk (S3928379)

Pham Tuan Hai (S3975144)

Vu The Quyen (S4027077)

Date : 18/09/2025

Table of Contents

I. INTRODUCTION

This project is about developing a peer-to-peer electric motorbike rental application that allows users to share their personal motorbikes with other community members. The system works like a sharing platform where people can list their motorbikes for rent and book motorbikes from others using credit points.

The application supports three types of users: Guests who can browse basic motorbike information, Members who can list and rent motorbikes, and Administrators who manage the system. Members can register their motorbikes with rental details, search for available bikes by date and location, make booking requests, and rate each other after completed rentals.

Key features include user authentication and registration, motorbike listing management, search and filtering based on user eligibility, booking approval system, credit point transactions, and mutual rating system. The system also enforces various constraints such as license requirements for larger motorbikes and credit balance management.

The project demonstrates object-oriented programming concepts such as class inheritance and encapsulation through a modular file structure.

II. APPLICATION DESIGN AND DEVELOPMENT

1. Software Design (Class Diagram)

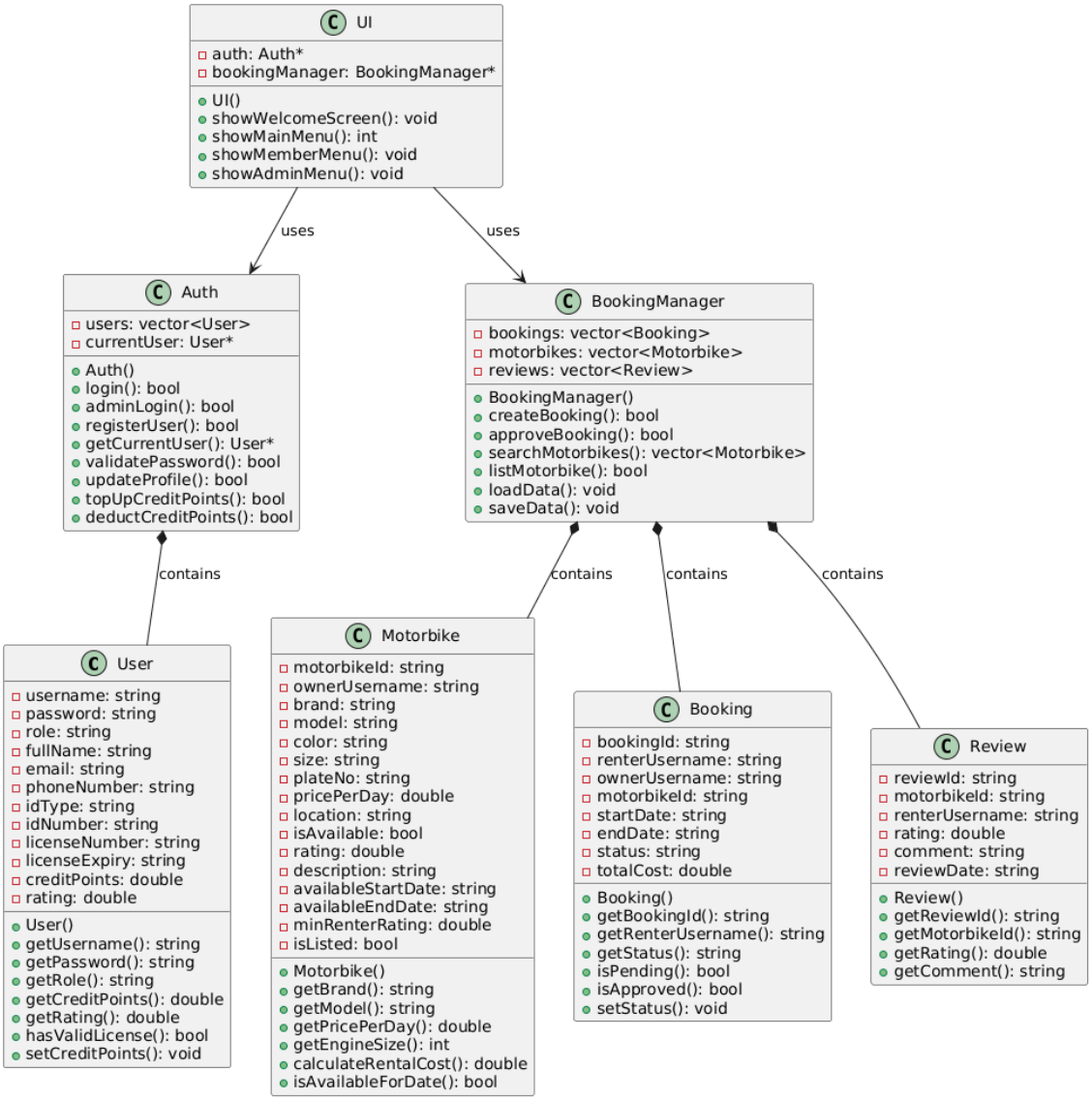


Figure 1: Class diagram for E-motorbike rental application

Description of Each Class:

Class: User	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	username: string	Unique identifier for user login	Need unique ID to distinguish users and prevent duplicate accounts
	password: string	User's login password	Essential for authentication and account security
	role: string	User role (member/admin)	Different roles have different access permissions in the system
	fullName: string	User's complete name	Required for registration and profile identification

	email: string	User's email address	Contact information and account verification
	phoneNumber: string	User's phone number	Contact information for communication
	idType: string	ID document type (Citizen ID/Passport)	Required for user identity verification process
	idNumber: string	ID document number	Unique identification for verification
	licenseNumber: string	Driver's license number	Required to validate license for motorbikes >50cc
	licenseExpiry: string	License expiration date	Check if license is still valid for rental eligibility
	creditPoints: double	User's credit balance	Currency for rental transactions in the system
	rating: double	User's renter rating	Trust system - other users rate renter behavior
Method(s)	getUsername(): string	Returns user's username	Need to access username for login and identification
	getCreditPoints(): double	Returns current credit balance	Essential for checking rental affordability
	hasValidLicense(): bool	Checks if license is valid	Enforces 50cc+ motorbike rental restrictions
	setCreditPoints(): void	Updates credit balance	Required for top-ups and rental payments

Table 1: Description table for User class

Class: Auth	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	users: vector<User>	Collection of all registered users	Store all user accounts in the system
	currentUser: User*	Pointer to currently logged-in user	Track active session for personalized features
Method(s)	login(): bool	Authenticate member login	Verify credentials and establish user session
	adminLogin(): bool	Authenticate admin login	Separate admin authentication for security
	registerUser(): bool	Create new user account	Allow new members to join the platform
	validatePassword(): bool	Check password strength	Enforce strong password policy as required
	topUpCreditPoints(): bool	Add credits to user account	Enable users to purchase rental currency
	deductCreditPoints(): bool	Remove credits from account	Process rental payments

Table 2: Description table for Auth class

Class: Motorbike	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)

Attribute(s)	motorbikeId: string	Unique motorbike identifier	Distinguish between different motorbikes
	ownerUsername: string	Username of motorbike owner	Link motorbike to its owner for rental requests
	brand: string	Motorbike brand (VinFast, Honda, etc.)	Important for user selection and preferences
	model: string	Motorbike model name	Specific model information for rental decisions
	size: string	Engine size (50cc, 125cc, etc.)	Determines license requirements and pricing
	pricePerDay: double	Daily rental rate in CP	Owner sets pricing for their motorbike
	location: string	City location (HCMC/Hanoi)	Geographic filtering for search functionality
	availableStartDate: string	Rental period start date	Define when motorbike becomes available
	availableEndDate: string	Rental period end date	Define rental availability window
	minRenterRating: double	Minimum required renter rating	Owner sets quality standards for renters
Method(s)	getBrand(): string	Returns motorbike brand	Display brand information in listings
	getPricePerDay(): double	Returns daily rental rate	Calculate total rental costs
	getEngineSize(): int	Extract engine size from size string	Determine license requirements (50cc rule)
	isAvailableForDate(): bool	Check date availability	Validate booking requests against availability

Table 3: Description table for Motorbike class

Class: Booking	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	bookingId: string	Unique booking identifier	Track individual rental transactions
	renterUsername: string	Username of person renting	Link booking to renter for management
	ownerUsername: string	Username of motorbike owner	Link booking to owner for approval
	motorbikeId: string	ID of motorbike being rented	Connect booking to specific motorbike
	startDate: string	Rental start date	Define rental period beginning
	endDate: string	Rental end date	Define rental period end
	status: string	Booking status (Pending/Approved/Rejected)	Track booking lifecycle
	totalCost: double	Total rental cost in CP	Store calculated payment amount
	ratingGivenByRenter: double	Stars renter gave the motorbike after completion	Need to store feedback per transaction

	commentByRenter: string	Comment renter gave the motorbike after completion	Allows qualitative feedback
	ratingGivenByOwner: double	Stars owner gave the renter	Need to update renter's average rating
	commentByOwner: string	Comment owner gave the renter	Allows qualitative feedback
	completed: bool	Has the booking been marked completed	Ensures ratings only allowed post-completion
Method(s)	getStatus(): string	Returns current booking status	Check booking state for UI display
	isPending(): bool	Check if booking awaits approval	Determine if owner action is needed
	isApproved(): bool	Check if booking is confirmed	Identify active rentals
	setStatus(): void	Update booking status	Change status during approval process
	markCompleted()	Marks the booking as completed	Unlocks the rating stage
	submitRatings() renterStars: double renterComment: string bikeStars: double bikeComment: string	Stores ratings from both sides and updates aggregated ratings in Member and Motorbike objects	Implements the mutual rating system

Table 4: Description table for Booking class

Class: Review	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	reviewId: string	Unique review identifier	Track individual reviews
	motorbikeId: string	ID of reviewed motorbike	Link review to specific motorbike
	renterUsername: string	Username of reviewer	Identify who wrote the review
	rating: double	Numerical rating (1-5)	Quantified feedback for motorbike quality
	comment: string	Written review comment	Detailed feedback for other users
	reviewDate: string	Date when review was written	Track review timeline
	ownerUsername: string	Owner who wrote review of renter	Link each review to reviewer
Method(s)	getComment(): string	Returns review comment	Display customer feedback to potential renters
	getRating(): double	Returns numerical rating	Calculate average motorbike ratings
	LinkToBooking() bookingId: string	Associates review with a specific booking transaction	Ensures traceability between ratings and rental

Table 5: Description table for Review class

Class: BookingManager	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	bookings: vector<Booking>	Collection of all bookings	Manage all rental transactions
	motorbikes: vector<Motorbike>	Collection of all motorbikes	Store all available motorbikes
	reviews: vector<Review>	Collection of all reviews	Store customer feedback data
	completedBookings: vector<Booking>	Storage of completed bookings ready for ratings	Need to drive “rate after completion” workflow
Method(s)	createBooking(): bool	Create new rental request	Allow members to request motorbike rentals
	approveBooking(): bool	Approve pending booking	Let owners confirm rental requests
	searchMotorbikes(): vector<Motorbike>	Find available motorbikes	Help users discover suitable rentals
	loadData(): void	Load data from files	Restore system state on startup
	saveData(): void	Save data to files	Persist data when application closes
	markBookingCompleted() bookingId: string	Marks a booking as completed after rental period ends	Require to enable rating
	handleRatings() bookingId: string renterStars: double renterComment: string bikeStars: double bikeComment: string	Calls Member and Motorbike update methods to update aggregated ratings	Implements mutual rating and aggregation
	loadData()	Loads all data from files	Restores system state on startup
	saveData()	Saves all data to files	Persists data when application closes

Table 6: Description table for BookingManager class

Class: UI	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method)
Attribute(s)	auth: Auth*	Pointer to authentication system	Access user login and authentication features
	bookingManager: BookingManager*	Pointer to booking system	Access motorbike and booking functionality
Method(s)	showWelcomeScreen(): void	Display application intro	Provide user-friendly application startup
	showMainMenu(): int	Display main menu options	Allow users to choose their role and actions
	showMemberMenu(): void	Display member-specific options	Provide member functionality access

	showAdminMenu(): void	Display admin-specific options	Provide administrative tools access
	showCompletedBookings()	Displays list of rentals eligible for rating	Allows user to select which to rate
	showRatingPrompt()	Displays interface for entering star ratings and comments for both sides	Implements the interactive UI for ratings
	saveAndExit()	Call BookingManager::saveData() before program exit	Implements data persistence at shutdown

Table 7: Description table for UI class

Class Relationships:

- Auth contains User objects (Composition):

The Auth class contains and manages a collection of User objects through a vector<User>. Auth owns the lifecycle of User objects - when Auth creates a new user through registration, it adds the User object to its collection. When Auth is destroyed, all User objects are also destroyed. This composition relationship ensures centralized user management and authentication control.

- BookingManager contains Booking objects (Composition):

BookingManager contains and owns all Booking objects in the system. It creates new Booking objects when users make rental requests, manages their status changes (pending, approved, rejected), and handles their persistence to files. The BookingManager controls the complete lifecycle of booking transactions and ensures business rules are enforced.

- BookingManager contains Motorbike objects (Composition):

BookingManager owns and manages all Motorbike objects in the system. It handles motorbike registration when users list their vehicles, manages search and filtering operations, and controls motorbike availability status. This composition allows centralized motorbike management and ensures data consistency across the rental system.

- BookingManager contains Review objects (Composition):

BookingManager contains all Review objects and manages the review system. It loads reviews from files, provides review data for motorbike listings, and handles review-related operations. This composition ensures that reviews are properly managed as part of the overall booking and rental system.

- UI depends on Auth (Dependency):

The UI class has a dependency relationship with Auth, using it for user authentication and session management. UI calls Auth methods to handle login processes, user registration, and access to current user information. This dependency allows UI to provide authentication features without implementing the authentication logic itself.

- UI depends on BookingManager (Dependency):

The UI class depends on BookingManager to access all motorbike and booking functionality. UI delegates business operations like motorbike search, booking creation, and rental management to BookingManager. This dependency maintains separation of concerns where UI handles user interaction while BookingManager handles business logic.

- Review associates with Booking (Association):

Every Booking can generate two Review entries — one from the renter and another from the owner. Reviews are tied to a specific booking and can only be made once the booking is marked as completed. This association reflects the relationship between ratings and a particular rental transaction.

- Booking associates with User (Association):

Each Booking involves two User objects: one serving as the renter and another as the owner. The Booking does not create or eliminate these users; rather, it references them for handling transactions, deducting credit points, and updating ratings. This association allows a user to participate in multiple bookings throughout time, either as a renter or an owner.

- Booking associates with Motorbike (Association):

Each Booking references a specific Motorbike object that is being rented. The Booking does not own the Motorbike; rather, it uses the Motorbike's information for rental transactions, availability updates, and cost calculations. This association allows multiple bookings to reference the same motorbike over different time periods.

- BookingManager depends on User and Motorbike interfaces (Dependency):

BookingManager depends on the public interfaces of User and Motorbike to adjust credits, update ratings, and check eligibility when processing bookings. Any changes in User or Motorbike attributes directly affect how BookingManager operates. This dependency ensures that BookingManager can interact with these objects without owning them.

- UI depends on BookingManager and Auth (Dependency):

The UI uses BookingManager to create, complete, and rate bookings, and uses Auth to handle login/logout. It does not own these objects but depends on them to provide the functionality. This keeps the business logic separate from presentation and maintains clean architecture principles.

2. Implementation Result

a. Welcome Screen

Result Summary:

The welcome screen successfully displays project information including group members, instructors, and main menu options. Users can choose between five options: Guest access for limited browsing, Member login for full functionality, Admin access for system management, new member registration, or exit to terminate the application.

Screenshots of Sample Result:

```
EEET2482/EEET2653/COSC2082/COSC2721 GROUP PROJECT
Semester 2 2025
E-MOTORBIKE RENTAL APPLICATION
Instructor: Dr Ling Huo Chong, Dr Ushik Shrestha, Dr Tri Huynh
Group: 5
S3992133, Lu Duc Thinh
S3928379, Jang Soohyuk
S3975144, Pham Tuan Hai
S4027077, Vu The Quyen

Use the app as 1. Guest  2. Member  3. Admin  4. Register  5. Exit
Enter your choice:
```

Figure 2: Welcome screen displaying project information and main menu options

b. Basic Features

Result Summary:

Feature Name	Feature Description	Status (Implemented/Not Implemented)	Bugs/Limitations if it has
1. Guest access	Guests can browse general motorbike listings with limited details: brand, model, engine size and location.	Implemented	No limitations. Works as designed with appropriate access restrictions.
2. Admin access	Admin can view all member profiles and motorbike listings without restriction.	Implemented	No limitations. Admin has full system access and can view comprehensive statistics.
3. Member registration	Users can register as members and register one electric motorbike. Registration must include a strong password policy. Upon successful registration and payment, the member receives 20 credit points and a default renter rating of 3. All members and motorbike information must be recorded and stored.	Implemented	Registration fee payment is simulated (not real payment). Motorbike registration during signup is separate menu option rather than integrated.
4. Profile and account management	Members can log in with their username and password. They can view and update their profile information as specified in the Profile management section, change their password, and top up their credit points.	Implemented	Credit point top-up is simulated (not real payment). Username cannot be changed for the system integrity.

5. Motorbike listing	Members can list an electric motorbike as specified in the Motorbike listing section.	Implemented	No limitations. System enforces one motorbike per member rule effectively.
6. Motorbike unlisting	Members can un-list their motorbike at any time, unless it is already booked.	Implemented	No limitations. System properly checks for active bookings before allowing unlisting.
7. Motorbike search and filtering	Members can search for available motorbikes by date and city. Search results should follow the filtering criteria defined in the Motorbike search and filtering section.	Implemented	Date validation uses simple string comparison rather than proper date parsing.
8. Viewing listings and reviews	Members can view complete listings, including the average rating score and user comments for each motorbike.	Implemented	Reviews are loaded from file. Review writing is implemented as well.
9. Rental request submission	Members can submit rental requests. A license restriction must be enforced: members without a valid motorbike license cannot rent electric motorbikes over 50cc.	Implemented	License validation uses simple date comparison. There might be room for improvement if we have spare time.
10. Rental request management	Motorbike owners can view all rental requests for their listed motorbike and choose to accept one. All overlapping rental requests are automatically rejected upon acceptance.	Implemented	No limitations. Automatic rejection of overlapping requests works correctly.
11. Rental confirmation and credit deduction	When a request is accepted, CPs are deducted from the renter's balance, and the motorbike is marked as rented for the selected period. Rental cancellations are not allowed once confirmed.	Implemented	No limitations. Credit deduction is working properly.
12. Ride completion and ratings	After the rental period ends, both the renter and the owner must rate each other (1-5 stars and a comment). Ratings and comments are tied to the specific transaction.	Implemented	No limitations. Mutual rating system is implemented.
13. Rating aggregation	Member and motorbike ratings are automatically averaged over time and updated accordingly.	Implemented	No limitations. A simple averaging function is used.
14. Data persistence	All data must be saved to data file(s) before the application closes. Upon starting the application, the data	Implemented	No limitations. File-based storage works reliably.

	must be loaded and made available for continued use.		

Table 8: Result summary table for the program

Screenshots of Sample Result (for each feature):

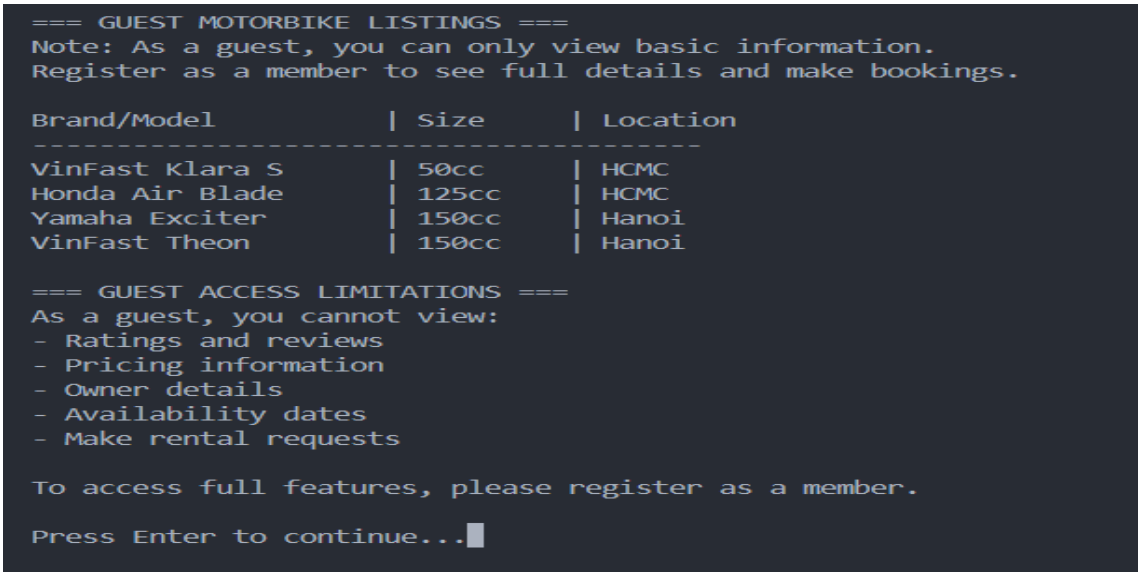


Figure 3: Guest motorbike listings showing only limited details (Feature 1)

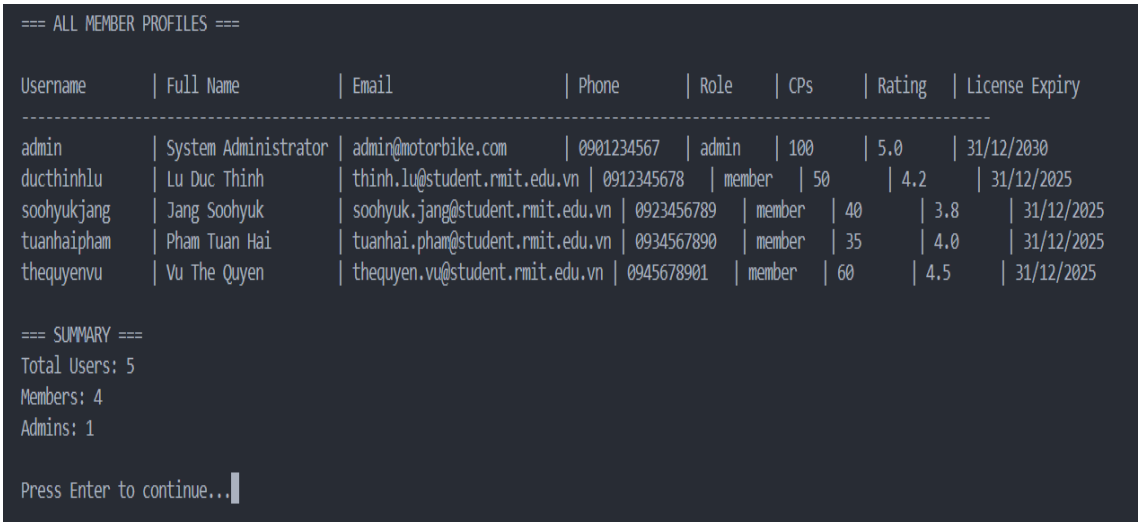
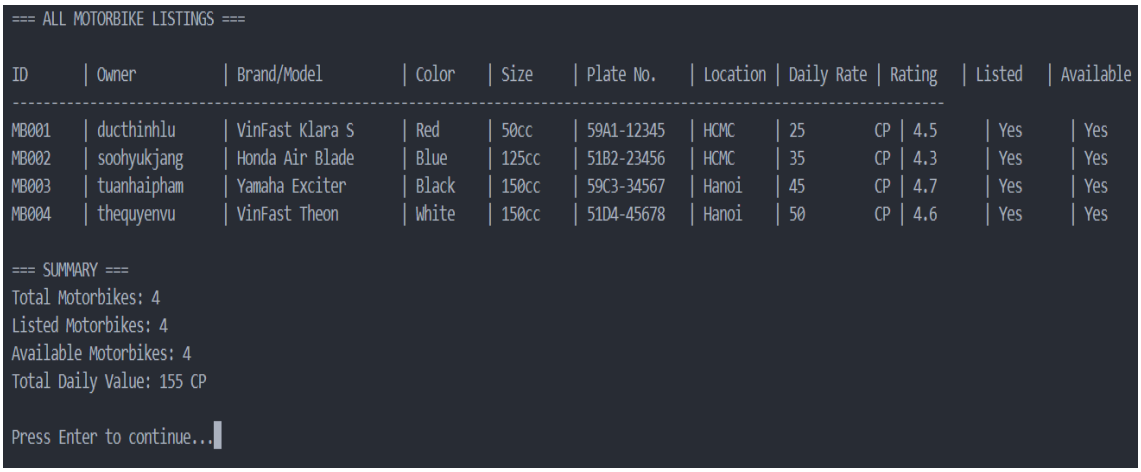
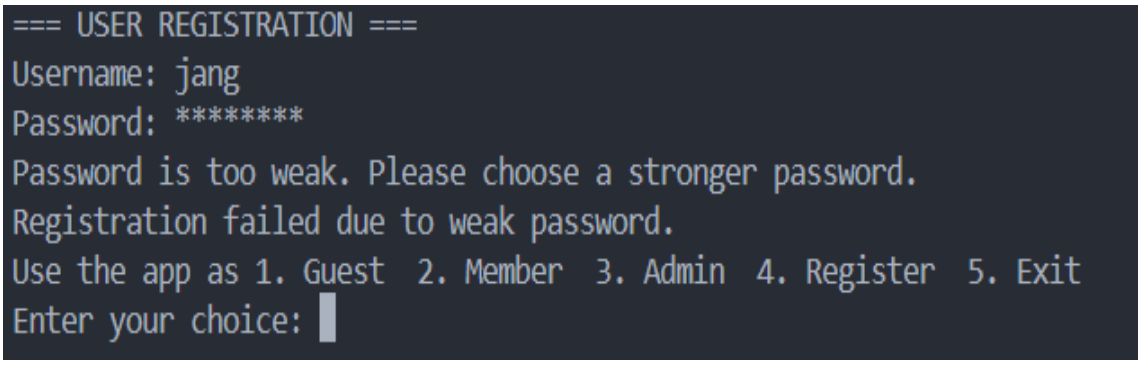


Figure 4: Admin view showing all member profiles without restrictions (Feature 2)



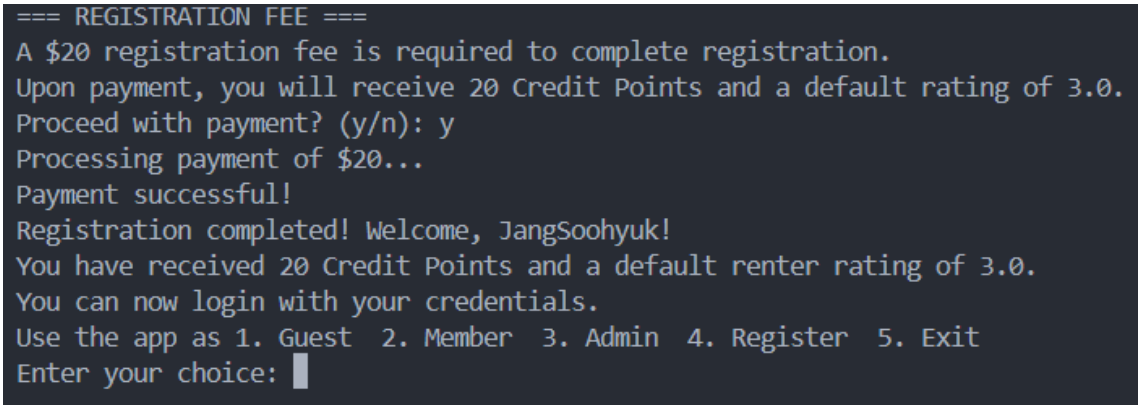
```
=== ALL MOTORBIKE LISTINGS ===

ID      | Owner          | Brand/Model        | Color  | Size   | Plate No.   | Location | Daily Rate | Rating | Listed | Available
-------------------------------------------------------------------------------------------------------------------------------
MB001   | ducthinhlu     | VinFast Klara S    | Red    | 50cc   | 59A1-12345  | HCMC     | 25       CP | 4.5    | Yes    | Yes
MB002   | soohyukjang    | Honda Air Blade    | Blue   | 125cc  | 51B2-23456  | HCMC     | 35       CP | 4.3    | Yes    | Yes
MB003   | tuanhaipham    | Yamaha Exciter     | Black  | 150cc  | 59C3-34567  | Hanoi    | 45       CP | 4.7    | Yes    | Yes
MB004   | thequyenvu     | VinFast Theon      | White  | 150cc  | 51D4-45678  | Hanoi    | 50       CP | 4.6    | Yes    | Yes

=== SUMMARY ===
Total Motorbikes: 4
Listed Motorbikes: 4
Available Motorbikes: 4
Total Daily Value: 155 CP

Press Enter to continue...
```

Figure 5: Admin view showing all motorbikes without restrictions (Feature 2)



```
=== USER REGISTRATION ===
Username: jang
Password: ********
Password is too weak. Please choose a stronger password.
Registration failed due to weak password.
Use the app as 1. Guest  2. Member  3. Admin  4. Register  5. Exit
Enter your choice:
```

Figure 6: Strong password policy during registration (Feature 3)



```
=== REGISTRATION FEE ===
A $20 registration fee is required to complete registration.
Upon payment, you will receive 20 Credit Points and a default rating of 3.0.
Proceed with payment? (y/n): y
Processing payment of $20...
Payment successful!
Registration completed! Welcome, JangSoohyuk!
You have received 20 Credit Points and a default renter rating of 3.0.
You can now login with your credentials.
Use the app as 1. Guest  2. Member  3. Admin  4. Register  5. Exit
Enter your choice:
```

Figure 7: Registration completion with 20 credit points and default rating 3.0 (Feature 3)

```
=== LIST MY MOTORBIKE FOR RENT ===
Please provide the following information:

Brand: Honda
Model: H1234
Color: Black
Size (e.g., 50cc, 110cc): 110cc
Plate Number: T1234
Location (HCMC or Hanoi): HCMC
Available Start Date (DD/MM/YYYY): 17/09/2025
Available End Date (DD/MM/YYYY): 20/09/2025
Daily Rental Rate (CP): 20
Minimum Required Renter Rating (1.0-5.0): 4.0
Motorbike listed successfully!
Motorbike ID: MB21

Motorbike listed successfully!

Press Enter to continue...
```

Figure 8: E-motorbike registration completion (Feature 3, 5)

```
=== USER PROFILE ===
Username: soohyukjang
Full Name: Jang Soohyuk
Email: soohyuk.jang@student.rmit.edu.vn
Phone: 0923456789
Role: member
Credit Points: 40
Rating: 4

Press Enter to continue...
```

Figure 9: Member profile information display (Feature 4)

```
=== TOP UP CREDIT POINTS ===
Current balance: 40 CPs
Rate: $1 = 1 CP

Enter amount to top up ($): 30
Enter your password to confirm: **********
Successfully topped up 30 CPs!
New balance: 100 CPs

Press Enter to continue...
```

Figure 10: Credit point top-up with password authentication (Feature 4)

```
=== UNLIST MY MOTORBIKE ===
Are you sure you want to unlist your motorbike? (y/n): y
Motorbike unlisted successfully.
Motorbike unlisted successfully!

Press Enter to continue...
```

Figure 11: Motorbike unlisting confirmation with booking status check (Feature 6)

```
=== SEARCH AVAILABLE MOTORBIKES ===
Search by:
1. Single date
2. Date range
Enter your choice: 2
Enter start date (DD/MM/YYYY): 01/09/2025
Enter end date (DD/MM/YYYY): 08/09/2025
Enter city (HCMC or Hanoi): HCMC
```

Figure 12: Motorbike search interface with date and city input (Feature 7)

```
=== SEARCH RESULTS ===
Date Range: 01/09/2025 to 08/09/2025
City: HCMC
Found 1 available motorbike(s)

Available Motorbikes:
ID  | Brand/Model       | Color  | Size  | Plate No.    | Daily Rate | Rating | Min Rating
----|-------------------|--------|-------|--------------|------------|--------|-----------
1   | VinFast Klara S   | Red    | 50cc  | 59A1-12345   | 25.0      CP | 4.5    | 3.0

Enter motorbike number to view details (0 to go back):
```

Figure 13: Filtered search results based on user eligibility criteria (Feature 7)

```
=== MOTORBIKE DETAILS ===
Motorbike ID: MB001
Brand: VinFast
Model: Klara S
Color: Red
Engine Size: 50cc
Plate Number: 59A1-12345
Location: HCMC
Daily Rental Rate: 25.0 CP
Available Period: 01/09/2025 to 31/12/2025
Minimum Required Renter Rating: 3.0
Motorbike Rating: 4.5/5.0
Description: VinFast Klara S - Red 50cc Electric Scooter

=== CUSTOMER REVIEWS ===
Average Rating: 4.5/5.0

Customer Comments:
1. VinFast Klara S runs very smoothly and quietly. Perfect for city rides!

Press Enter to continue...
```

Figure 14: Complete motorbike listing with average ratings and customer reviews (Feature 8)

```
=== MAKE RENTAL REQUEST ===
Motorbike: VinFast Klara S
Daily Rate: 25.0 CP
Available Period: 01/09/2025 to 31/12/2025

Enter rental start date (DD/MM/YYYY): 01/09/2025
Enter rental end date (DD/MM/YYYY): 02/09/2025
Rental request submitted successfully!
Booking ID: BK6
Total Cost: 25.0 CP
Status: Pending approval from owner

Rental request submitted successfully!

Press Enter to continue...
```

Figure 15: Rental request submission form with date selection (Feature 9)

```
=== MAKE RENTAL REQUEST ===
Motorbike: Honda Air Blade
Daily Rate: 35.0 CP
Available Period: 01/09/2025 to 30/11/2025

Enter rental start date (DD/MM/YYYY): 01/09/2025
Enter rental end date (DD/MM/YYYY): 09/09/2025
Valid license required for motorbikes over 50cc.

Failed to submit rental request.

Press Enter to continue...
```

Figure 16: License restriction enforcement for motorbikes over 50cc (Feature 9)

```
=== RENTAL REQUESTS FOR MY MOTORBIKES ===
Pending Requests:
ID  | Renter        | Motorbike        | Period           | Cost  | Status
----|---------------|------------------|------------------|-------|--------
  1 |    thequyenvu |  VinFast Klara S | 01/09/2025-02/09/2025 |  25.0 CP | Pending

Enter request number to approve/reject (0 to go back):
```

Figure 17: Rental requests list for motorbike owners (Feature 10)

```
=== MANAGE RENTAL REQUEST ===
Booking ID: BK6
Renter: thequyenvu
Motorbike: VinFast Klara S
Period: 01/09/2025 to 02/09/2025
Total Cost: 25.0 CP

1. Approve Request
2. Reject Request
3. Back
Enter your choice:
```

Figure 18: Rental request approval/rejection interface (Feature 10)

```
=== MANAGE RENTAL REQUEST ===
Booking ID: BK6
Renter: thequyenvu
Motorbike: VinFast Klara S
Period: 01/09/2025 to 02/09/2025
Total Cost: 25.0 CP

1. Approve Request
2. Reject Request
3. Back
Enter your choice: 1
Booking approved successfully!
Credit points deducted: 25.0 CP
Request approved successfully!

Press Enter to continue...
```

Figure 19: Credit point deduction and rental confirmation message (Feature 11)

```
=== COMPLETE RENTAL ===
Booking ID: BK004
Motorbike: VinFast Theon
Period: 20/09/2025 to 22/09/2025
Total Cost: 100.0 CP

Are you sure you want to complete this rental? (y/n): y
Rental completed successfully!
Rental completed successfully!
You can now rate the motorbike in the 'Rate completed rentals' menu.

Press Enter to continue...
```

Figure 20: Completed rentals available for rating (Feature 12)

```
=== RATE MOTORBIKE ===
Booking ID: BK004
Motorbike: VinFast Theon
Period: 20/09/2025 to 22/09/2025

Rate the motorbike (1.0 - 5.0): 4
Enter your comment: Good, thank you. Nice vehicle
Review added successfully!
Motorbike rated successfully!
Rating: 4.0/5.0
Comment: Good, thank you. Nice vehicle
New average rating: 4.3/5.0
Thank you for rating the motorbike!

Press Enter to continue...
```

Figure 21: Rating input interface with 1-5 stars and comment field (Feature 12)

```
=== PROFILE MANAGEMENT ===
1. View Profile Information
2. Update Profile Information
3. Change Password
4. Top Up Credit Points
5. View Booking History
6. Back to Member Menu
Enter your choice: 1
=== USER PROFILE ===
Username: jang
Full Name: Soohyuk Jang
Email: soohyuk@gmail.com
Phone: 0901111111
Role: member
Credit Points: 50
Rating: 3.5

Press Enter to continue...
```

Figure 22: Aggregated ratings displayed in user profile or motorbike details (Feature 13)

```
=== LIST MY MOTORBIKE FOR RENT ===
Please provide the following information:

Brand: VinFast
Model: V123
Color: White
Size (e.g., 50cc, 110cc): 50cc
Plate Number: VT123
Location (HCMC or Hanoi): HCMC
Available Start Date (DD/MM/YYYY): 01/01/2025
Available End Date (DD/MM/YYYY): 10/10/2025
Daily Rental Rate (CP): 20
Minimum Required Renter Rating (1.0-5.0): 2
Motorbike listed successfully!
Motorbike ID: MB6

Motorbike listed successfully!

Press Enter to continue...
```

Figure 23: New data added to the system (Feature 14)

```
=== MY MOTORBIKE LISTING ===
Motorbike ID: MB6
Brand: VinFast V123
Color: White
Size: 50cc
Plate Number: VT123
Location: HCMC
Daily Rate: 20 CP
Available Period: 01/01/2025 to 10/10/2025
Minimum Renter Rating: 2
Current Rating: 0/5.0
Status: Available

Press Enter to continue...
```

Figure 24: Data persistence verification after application restart (Feature 14)

c. Advanced Features

Result Summary:

Advanced features are implemented and functional as well. The identity verification system provides a simple verification mechanism with status display. The activity dashboard offers a comprehensive overview of user account status, active rentals, and rental requests as well.

Screenshots of Sample Result:

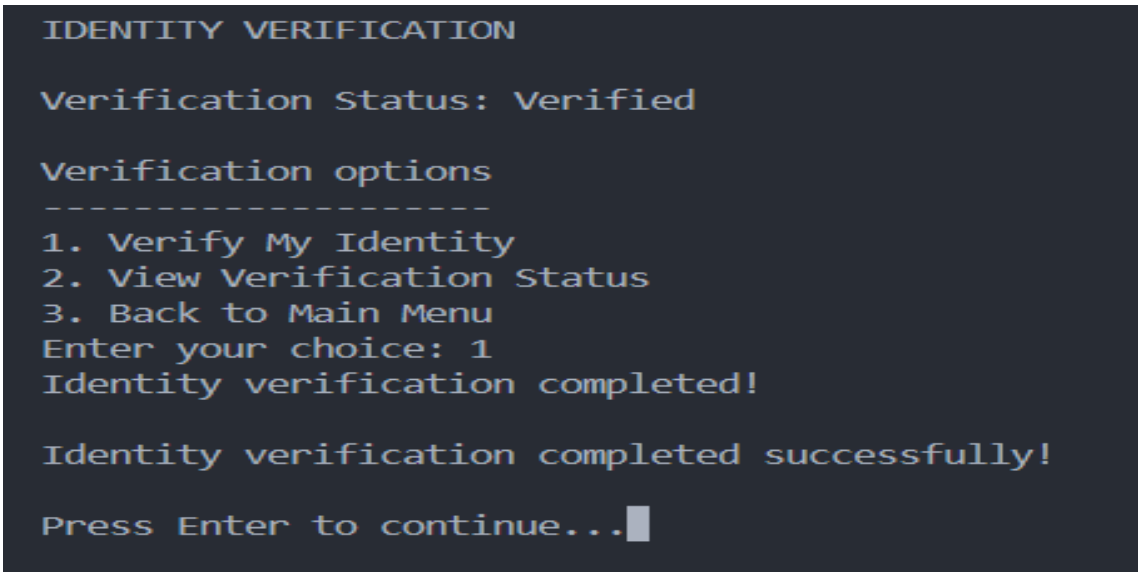


Figure 25: Identity verification menu and verification status display (Advanced feature 1)

Figure 26: Activity dashboard showing account overview, active rentals, and rental requests (Advanced feature 2)

IV. DISCUSSIONS & CONCLUSIONS

The final implementation does not completely match our initial class diagram. During implementation, we faced several logical issues that required design modifications. For example, our initial date validation logic used simple string comparison, which caused errors when comparing dates such as "09/09/2025" and "01/10/2025". This led us to add new validation methods to the BookingManager class. Additionally, the original single UI class became too complex during development, so we modularized it into seven specialized classes (UICore, UIProfile, UIMotorbike, etc.). We also discovered the need for data persistence methods and file management attributes that weren't in our initial design. While the class diagram provided a solid foundation for planning and team communication, the practical implementation revealed requirements and logical issues that necessitated these changes. This iterative process of design refinement during development shows our commitment to continuous improvement and thorough analysis.

V. REFERENCES

[1] W3Schools, “C++ OOP,” *W3Schools.com*. [Online]. Available: https://www.w3schools.com/cpp/cpp_oop.asp. [Accessed: Sep. 17, 2025].

[2] GeeksforGeeks, “C++ Object Oriented Programming,” *GeeksforGeeks.org*. [Online]. Available: https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/. [Accessed: Sep. 17, 2025].

[3] W3Schools, “C++ Files,” *W3Schools.com*. [Online]. Available: https://www.w3schools.com/cpp/cpp_files.asp. [Accessed: Sep. 17, 2025].

[4] GeeksforGeeks, “C++ STL (Standard Template Library),” *GeeksforGeeks.org*. [Online]. Available: https://www.geeksforgeeks.org/the-c-standard-template-library-stl/. [Accessed: Sep. 17, 2025].

[5] GeeksforGeeks, “Header files in C/C++,” *GeeksforGeeks.org*. [Online]. Available: https://www.geeksforgeeks.org/header-files-in-c-cpp-and-its-uses/. [Accessed: Sep. 17, 2025].