Milestone 3 System and software requirements

WinWin

Ride-hailing platform for local-motorcycle service provider and user

Present to

Dr. Pittipol Kantavat and Dr. Ronnakorn Vaiyavuth

By G14 Waterfall

6230123921 Thitaree Setwipattanachai

6230252121 Tarm Kalavantavanich

6231301421 Kanokpich Chaiyawan

6231304321 Kittipong Deevee

6231307221 Jirawat Kusalangkurwat

6231333521 Nopdanai Sayamnet

6231353021 Raviporn Akekunanon

6231372021 Atiwat Deepo

Analysis phase document

2110335 Software Engineering I, Semester 1 of Academic year 2021

Department of Computer Engineering,

Faculty of Engineering, Chulalongkorn University

Table of Contents

List of figures

List of tables

Project name: WinWin

Ride-hailing platform for local-motorcycle service provider and user

Introduction

Nowadays, motorcycle taxis are a major type of transportation in Bangkok (second only to MRT). However, riders have low income because ride-hailing platforms snatch their market share. Moreover, ride-hailing platforms can make users more satisfied than motorcycle taxis e.g., users can call ride-hailing platforms everywhere.

Therefore, WinWin wants to digitalize the motorcycle taxi system and to utilize route familiarity and locality of local motorcycle taxis to be an advantage that other ride-hailing platforms do not have.

WinWin, an online platform that connects customers with the local motorcycle taxis, has two business partners. Firstly, the Department of Land Transport, Ministry of Transport, which provides WinWin with the information about motorcycle taxis in the Bangkok area. Secondly, Winnonie, a startup founded by Bangchak Corporation group that rents out electric motorcycles.

WinWin believes that employing local motorcycle taxis as service providers is the best option since the riders are familiar with the route and can arrive at the customer's location faster than other riders from other ride-hailing platforms.

WinWin also thinks that building this application would satisfy stakeholders such as the Department of Land Transportation, Ministry of Transport, Winnonie, Motorcycle taxis, and consumers. Because Winnonie would be the market leader in the electric-vehicle rental market, the government and the Ministry of Transportation would receive a lot of positive credit for reorganizing local motorbike service. Riders would have more ways to make money, and consumers would benefit from a low-cost service provided by locals.

Objective of the analysis document

- To provide a description of how the system works
- To illustrate an overview of the to-be system
- To define the requirement specification and software specification of the system

Details of requirements

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
REG01	**Create an account (Customer)** - The system shall allow the customer to fill-in their profile including number, first-name, last-name, and phone number.	username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs are valid create an account for the customer in the system	error messages for invalid customer's userID, password, first name, last name, telephone number, and email
		password	string /20	string length 8 - 20 with at least one uppercase, one lowercase, one digit number	string length less than 8 or more than 20, string with no uppercase or lowercase or one digit number		
		first name	string /50	string length 1 - 50	string length 0 or more than 50		
		last name	string /50	string length 1 - 50	string length 0 or more than 50		
		Telephone number	string /10	10-digit number	Has more or less than 10 digits		
		email	string /50	valid form of email	invalid form of email		
REG02	**Create an account (Rider)** - The system shall allow the customer to fill-in their profile including number, first-name, last-name, and phone number. - The system shall allow the rider to fill-in their profile including Reference number, first-name, last-name, citizen ID number, and phone number.	username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs are valid create an account for the rider in the system	error messages for invalid rider's userID, password, first name, last name, telephone number, email, reference number, and citizen ID
		password	string /20	string length 8 - 20 with at least one uppercase, one lowercase, one digit number	string length less than 8 or more than 20, string with no uppercase or lowercase or one digit number		
		first name	string /50	string length 1 - 50	string length 0 or more than 50		
		last name	string /50	string length 1 - 50	string length 0 or more than 50		
		Telephone number	String /10	10-digit number	Has more or less than 10 digits		
		email	string /50	valid form of email	invalid form of email		
		reference number	string /16	16-digit number	Has more or less than 16 digits		
		citizen ID	string /13	13-digit number	Has more or less than 13 digits		

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
LOG01	**Login/logout system** - The system shall allow the customer and rider to login/logout the system.	username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs are valid and there exists an account corresponding to userID, password, and account type, then allow the user to login	error messages for invalid user's userID, password
		password	string /20	string length 8 - 20 with at least one uppercase, one lowercase, one digit number	string length less than 8 or more than 20, string with no uppercase or lowercase or one digit number	if all inputs are valid and there does not exist an account corresponding to userID, password, and account type, then show the error message 'Cannot login to the system'	
		account type	enum /1	r, c			
MAT01	**Rider availability** - The system shall allow riders to set their availability to either available or unavailable.	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	set rider's availability to input availability	error messages for invalid rider's username
		availability	boolean	true, false			
MAT02	**Rider's new ride notification** - The system shall make notifications to riders about ride requests made by users in their acceptable vicinity.	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	send notifications to rider and show ride information	error messages for invalid rider's username
		new ride flag	boolean	true, false			
MAT03	**Accept or decline ride** - The system shall allow riders to accept ride requests that are available. - The system shall	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs are valid and both accept and decline are false, keep showing the ride information	error messages for invalid rider's username

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
	allow riders to decline available ride requests that are notified to them.	accept	boolean	true, false	N/A	if all inputs are valid with input accept is false and input decline is true, then initiate the ride	
		decline	boolean	true, false	N/A	if all inputs are valid with input accept is true and input decline is true, then the system will stop showing the ride information	
MAT04	**Cancel accepted ride** - The system shall allow riders to cancel their acceptance of a ride request.	rideID	char /8	string length 8	string length less than or more than 8	cancel the accepted ride and send alert message to customer	error messages for invalid rideID
		cancel flag	boolean	true, false			
REC01	**Show ride records** - The system shall record every ride every rider has accepted.	rideID	char /8	string length 8	string length less than or more than 8	if input is valid and there exists a ride record corresponding to input rideID, show the record to the user	error messages for invalid rideID
						if input is valid and there does not exist a ride record corresponding to input rideID, show the error message 'Cannot find the record'	

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
BAR01	**Find rider by location** - The system shall allow the customer to look up available riders by location.	latitude	double	double value range from -90 to 90	double value less than -90 or more than 90	if all inputs are valid, show list of all available riders in 1 km radius	error messages for invalid latitude and longitude
		longitude	double	double value range from -180 to 180	double value less than -180 or more than 180		
BAR02	**Set destination** - The system shall allow the customer to set their destination for the ride.	latitude	double	double value range from -90 to 90	double value less than -90 or more than 90	if all inputs are valid, set the desired destination in a ride record	error messages for invalid latitude and longitude
		longitude	double	double value range from -180 to 180	double value less than -180 or more than 180		
BAR03	**Customer sets rider preference** - The system should allow the customer to choose their preference for the ride.	preference	String /100	string length 1 - 100 with no special characters	string length more than 100, string with at least one special character	if all inputs are valid, set the ride preference in a ride record	error messages for invalid preference
BAR04	**Select booking type** - The system shall allow the customer to choose between booking a ride right away or booking a ride in advance.	in advance type flag	boolean	true, false	N/A	if false, set booking type as right away in a ride record. if true, set booking type as in advance in a ride record	N/A
BAR05	**Cancel the ride** - In case of a right-away ride, the system shall allow the customer to cancel the ride before the ride is accepted, without any penalty - In case of an in-advance booked ride, the system shall allow the customer to cancel the ride before the scheduled time, without any penalty.	rideID	char /8	string length 8	string length less than or more than 8	send message "Cancel the requested ride complete"	error messages for invalid rideID
		cancel flag	boolean	true, false	N/A		

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
BAR06	**See price and estm. time** - The system shall allow the customer to see the price rate of the requested ride. - The system should allow customers to see the start time and predicted arrival time for the ride.	start latitude	double	double value range from -90 to 90	double value less than -90 or more than 90	if all inputs are valid, show price and estimate start time and arrival time to the customer	error messages for invalid start latitude, start longitude, stop latitude and stop longitude
		start longitude	double	double value range from -180 to 180	double value less than -180 or more than 180		
		stop latitude	double	double value range from -90 to 90	double value less than -90 or more than 90		
		stop longitude	double	double value range from -180 to 180	double value less than -180 or more than 180		
INI01	**Customer's notification for accepted ride** - The system shall make notification to the customer about the acceptance of their ride request.	customer username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	send notifications to customer about the start of the ride and show ride status	error messages for invalid customer's username
		ride accepted flag	boolean	true, false	N/A		
INI02	**View rider profile** - The system shall show the customer the profile of the rider who accepted their ride request.	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	Show rider profile of the rider who accepted the ride request	error messages for invalid rider's username
INI03	**View rider location** - The system shall allow the customer to be able to see the current location of the rider who accepted their ride request.	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	Show rider location of the rider who accepted the ride request	error messages for invalid rider's username
INI04	**Customer's notification for rider arrival** - The system shall make notification to the customer of the arrival of the rider who accepted their ride request.	customer username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	Send notification to customer for rider arrival	error messages for invalid customer's username
		rider arrival flag	boolean	true, false	N/A		

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
INI05	**Cancel in-progress ride** - The system shall allow customers to cancel their rides that are currently in progress but with a penalty.	rideID	char /8	string length 8	string length less than or more than 8	Send message "Cancel a ride complete"	error messages for invalid rideID
		cancel flag	boolean	true, false	N/A		
PAY01	**Select payment method** - Before the customer books a ride, the service shall allow the customer to select their desired payment method.	transactionID	char /8	string length 8	string length less than or more than 8	if all inputs are valid, send message "Select payment method is complete"	error message for invalid rideId or paymentMethodID
		paymentMethodID	enum /2	bt, cc, ca (bank, card and, cash respectively)	otherwise		
PAY02	**Payment by bank transfer** - In case the customer decides to pay the service by transferring to a bank account, the system shall allow the customer to transfer service fee when the customer reaches the destination.	customer username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs (username, bankAccountID, feeAmount) are valid, send message "Payment via bank transfer"	error message for invalid bankAccountID or feeAmount
		bankAccountID	string /20	string of valid bank account number	string include non-number character		
		feeAmount	double	positive double value	positive double value		
PAY03	**Payment by credit or debit card** - In case the customer decides to make a payment automatically from their credit or debit card, the system shall automatically make a payment from that credit or debit card after the	customer username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs (credit card number, security code, telephone number) are valid, execute external bank system and send message "Payment via credit card is complete"	error message for invalid customer's credit card number, invalid customer' credit card security code, and telephone number

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
	rider marks the service as done.	feeAmount	double	positive double value	positive double value	if all inputs are valid, but the amount of total product/service above credit limit left, send a message "The amount of total product/service above credit limit. The Payment via credit card is incomplete"	
		Credit Card Number	String /16	Exactly 16-digits credit card number	Has more or less than 16 digits		
		Security Code	String /3	Three digits	Has more than 3 digits or less than 3 digits		
		Expiry Date	Date/6 (mmyyyy)	Choose from a list of month and year that the system provides	N/A		
		Telephone number	string/10	10-digit number	Has more or less than 10 digits		
PAY04	**Payment by cash** - In case the customer decides to pay by cash, the system shall deduct the rider's cash credit equivalent to the service fee for that ride after the rider marks the service as done. (Customer pays the rider when they reach the destination.)	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	Send message "Payment by cash complete"	error messages for invalid rider's username and feeAmount
		feeAmount	double	positive double value	positive double value		
PAY05	**Rider's cash credit topup** - In case the rider's cash credit is under 50 baht, the system shall allow rider to top-up credit by bank transfer, credit card, and debit card.	rider username	string /50	string length 1 - 50 with no special characters	string length 0 or more than 50, string with at least one special character	if all inputs (rideID, feeAmount) are valid, send message "Cash credit topup is complete"	error messages for invalid rider's username and feeAmount
		feeAmount	double	positive double value	positive double value		

Requirement		Input				Output	
Require-mentID	**Requirement Name**	**Name**	**Type /length**	**Valid value**	**Invalid value**	**Output for valid input**	**Output for invalid input**
REV01	**Make a review** - In case the service is success, the system shall allow customers that use the service to review their rider via anonymous comment and rate them from 0 to 5 after the ride. - The system should allow the customer to view comments and ratings they have given to past rides.	rideID	char /8	string length 8	string length less than or more than 8	if all inputs (riderID, review rating,review comment) are valid, send make a review complete message	error message for invalid rideID, review rating ,and review comment
		review rating	int /1	integer between 1 to 5	integer length less than or more than 1, integer value less than 1 or more than 5		
		review comment	string /200	string length 0 - 200	string length more than 200		
REV02	**View reviews and comments** - The system should allow the rider to view comments and ratings given to them.	rideID	char /8	string length 8	string length less than or more than 8	if all inputs (riderID) are valid, show reviews and comments	error message for invalid rideID

Overview of the to-be (proposed) system context

This application is designed to bring benefits to both users and motorcycle taxis. Both customers and riders will use the same application, however, their interface will be different based on their user type.

For riders to get started, they will need to register through filling in their full name, citizen ID and reference number from the Department of Land Transport, pay an entrance fee including taking a picture of themselves to verify the identity of motorcycle taxis. This is to assure users that all motorcycle taxis in the application will be legal motorcycle taxis and to build confidence to customers that the rider is the approved one.

On the user side to register, it is necessary to verify identity through personal information.

The matchmaking system starts when the user selects the pick-up location and destination they want to go to, selects payment method, and selects rider preferences. The motorcycle taxis in the surrounding location will be notified that there is a new user's ride booking. When a motorcycle taxi accepts a ride from any user, users will see motorcycle plate number and rider name, and then wait for a motorcycle taxi to pick up.

During the service, on the motorcycle taxi side, there shall be an update to let the system know that the motorcycle taxi has arrived, on the way to the destination, or arrived at the destination.

When the service is completed. The customer will be charged for the ride fare via the selected payment method and gives a review to the rider who serves them with rating and description.

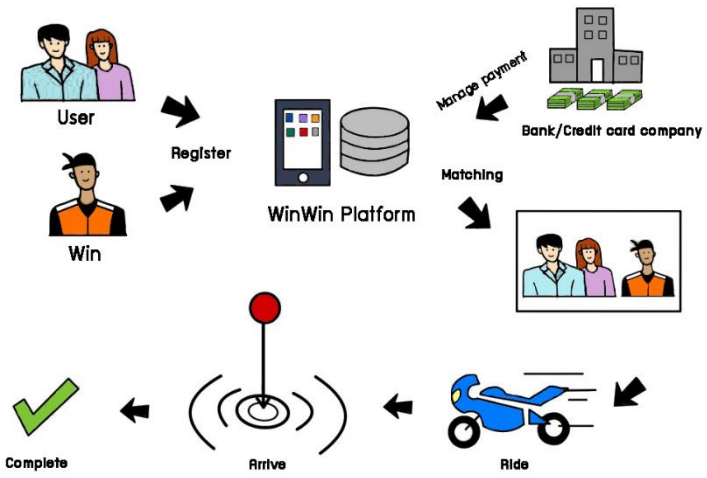


Figure 1: The flow of to-be system

Term definitions

Table 1: Definitions

Terms	Definition
Personal Information	Full name, Phone number, Address, Email
Admin	WinWin platform provider
Job	A service sequence, starting from booking a motorcycle taxi and end with reviewing
Rider	In-service motorcycle taxi
Review	Writing comments and impressions of the service, including star rating and description in various fields such as cleanliness, speed, courtesy of the service provider.

List of stakeholders and their responsibilities

Motorcycle taxis

- Accept or cancel the ride
- Pick up customers and deliver them to the destination
- Confirm the payment after the ride is completed
- Update their status (available/busy)
- Inform manager if they move to the new station
- Top up enough credit for the cash payment method
- Update any changes on the details of the vehicle

Customers

- Request for a ride
- Review the rider after the ride
- Make a payment after the ride is completed

Department of Land Transport, Ministry of Transport

- Provide WinWin the information about motorcycle taxis in Bangkok

Manager

- Accompany changes for motorcycle taxis under their control

Proposed method of systems analysis

Informal benchmarking

- As we are aiming to be a ride hailing service provider, we studied the business process from one of the successful companies in the same field, Grab Bike, to guide us through a better development of the overall design of the system.

Technology analysis

- With today's growing number of smartphone users and the need for convenience of customers, we have identified the opportunity to incorporate the use of mobile ride hailing in the business process to accommodate the existing customer of motorcycle taxis and attract more customers to use the service.
- Nowadays, mobile payment is increasing worldwide. Therefore, we decide to bring mobile payment to be one of the payment methods that customers can pay the ride fare easily

Remarks

- The word "user" in project proposal is changed to "customer" for clarification
- Changing "userID" in system functionalities to "username"

Business process modeling

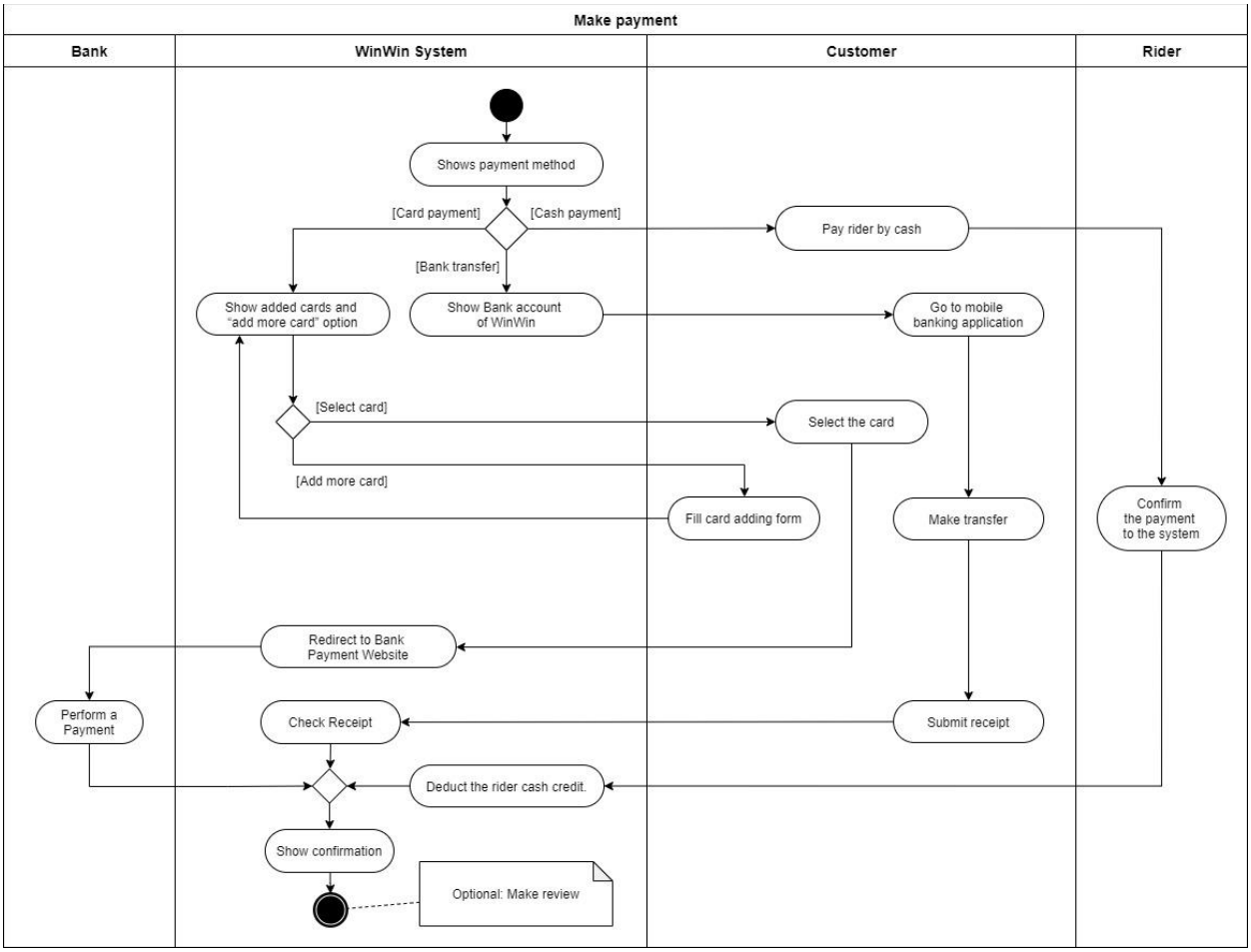


Figure 2: Activity diagram of “Make payment” Use case

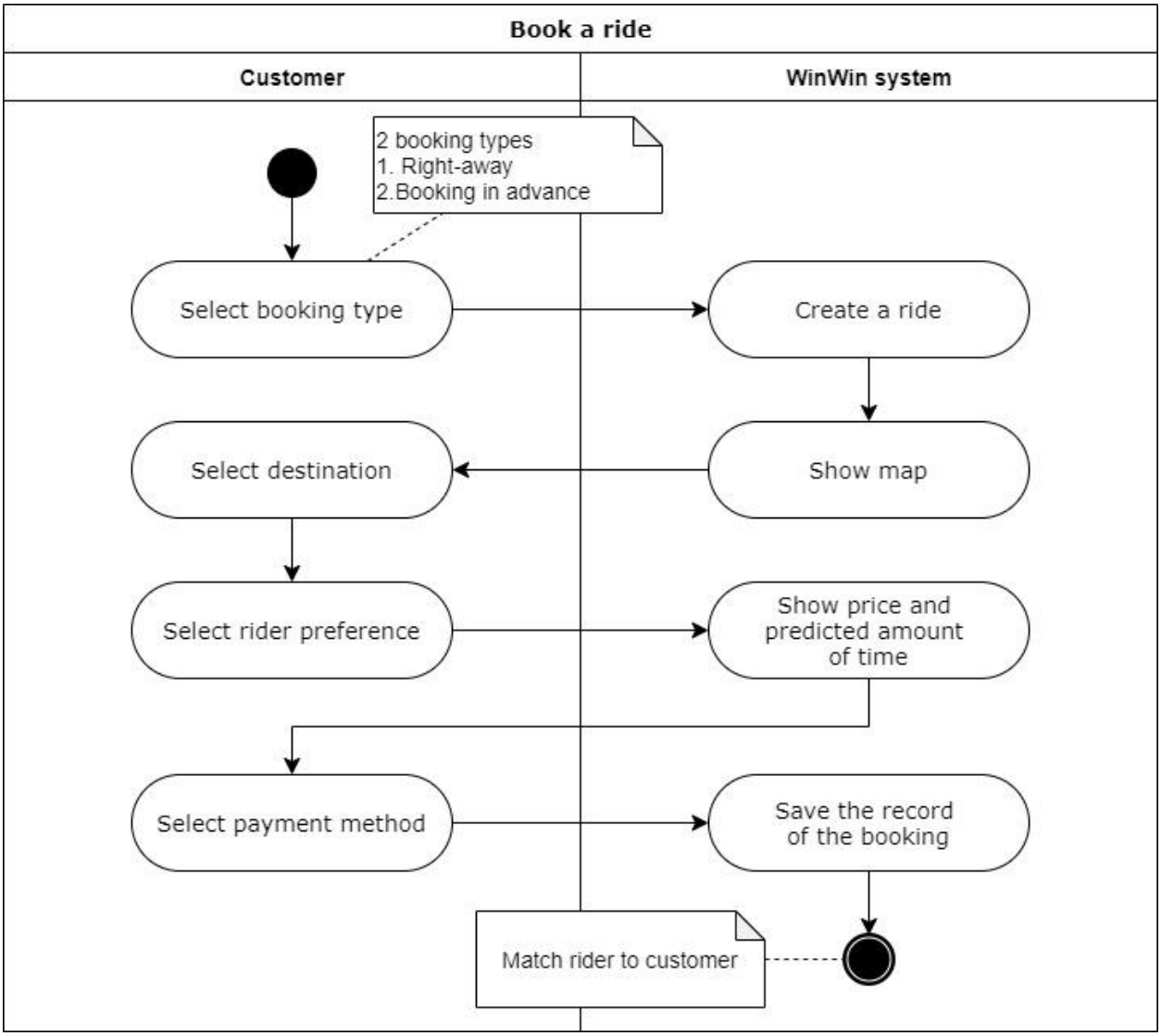


Figure 3: Activity diagram of “Book a Ride” Use case

Detail Essential Use Case diagram and description

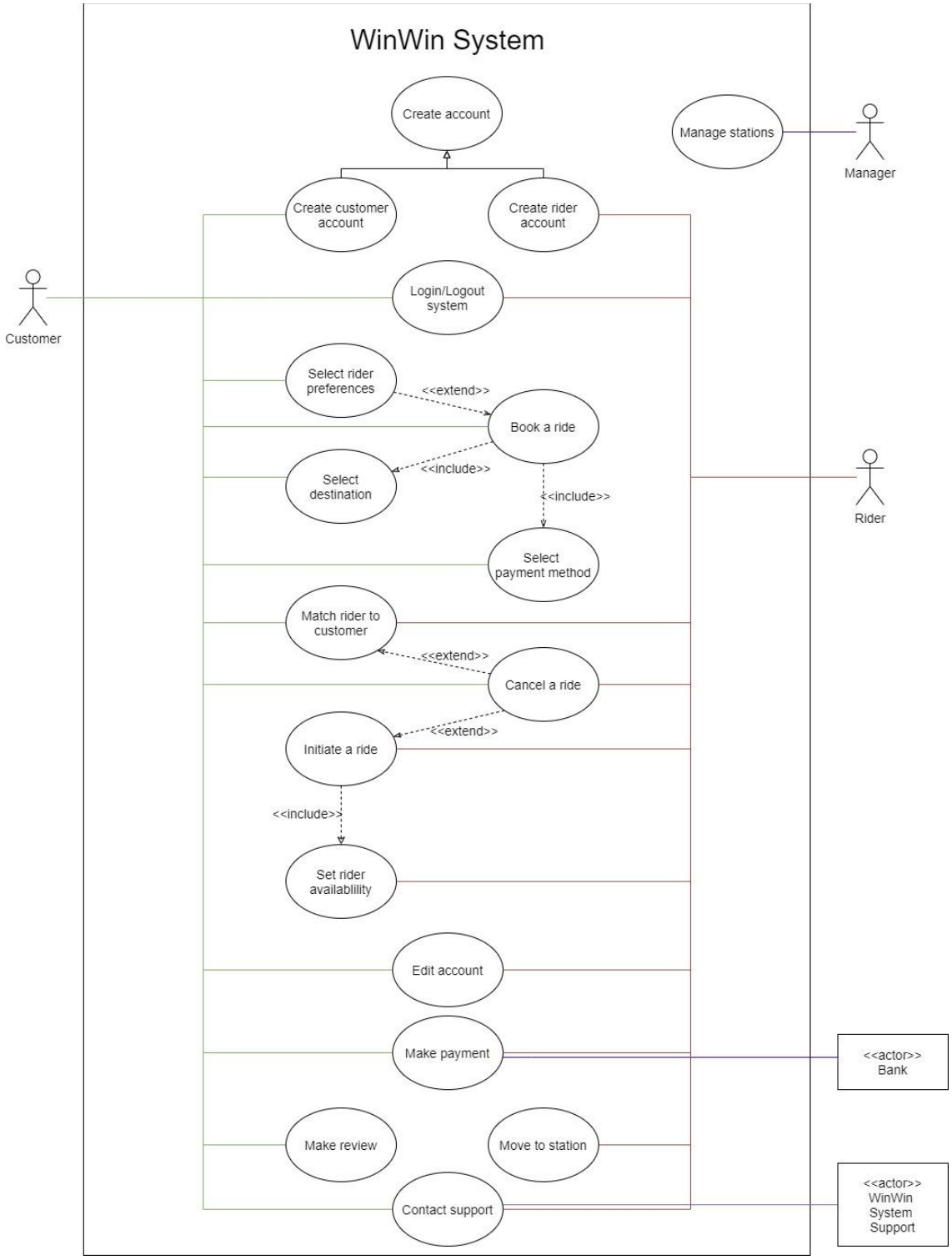


Figure 4: Use case diagram of WinWin System

Use case explanation

Create account	This use case is a generalization of “Create customer account” and “Create rider account” use cases.
Create customer account	This use case describes how to create a customer account.
Create rider account	This use case describes how to create a rider account.
Login/Logout system	This use case describes how customers and riders log in or log out the system.
Book a ride	This use case describes how customers book a ride.
Select rider preferences	This use case describes how customers select rider preferences.
Select destination	This use case describes how customers select a destination.
Select payment method	This use case describes how customers select the payment method.
Match rider to customer	This use case describes how the system matches a rider to the customer.
Cancel a ride	This use case describes how customers and riders cancel rides.
Initiate a ride	This use case describes how riders initiate a ride.
Set rider availability	This use case describes how riders set their availability.
Edit account	This use case describes how customers and riders edit the account.
Make payment	This use case describes how customers make a payment.
Make review	This use case describes how customers make a review.
Contact support	This use case describes how customers and riders contact support.
Move to station	This use case describes how riders move to stations.

Table 2: Use case description of “Book a ride”

Use Case Name: Book a ride	**ID:** 4	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	
Stakeholder and Interests: Customer - wants to book a ride service.		
Brief Description: This use case describes how customers book a ride.		
Trigger: Customer asks to book a new ride service **Type**: External		
Relationships: Association: Customer Include: Select Payment Method, Select Destination Extend: Select Rider Preferences Generalization: -		
Precondition: Customer logs in.		
Postcondition: After a customer has booked a ride, the system initiates the “Match rider to customer” process.		
Normal Flow of Events: 1. The customer selects “Booking” menu 2. The customer chooses a booking type between booking a right-away ride or booking a ride in advance. 3. The customer sets their destination for the ride. Include flow: Select Destination 4. The customer chooses their preference for the ride. Extension points: Select Rider Preferences 5. The system shows the price rate and predicts the amount of time of the requested ride. 6. The customer chooses their method of payment. Include flow: Select Payment Method 7. The system saves the record of the booking.		
Subflows: -		
Alternate/Exceptional Flow: 1-a1: If the customer does not want to book a ride anymore at any time, the customer can select exit button to go back to home menu 2-e1: If the customer books another ride at the same riding time as another ride request, the system notifies the customer to prevent a duplication of ride request.		

Table 3: Use case description of “Make payment”

Use Case Name: Make payment	**ID: 6**	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	
Stakeholder and Interests: Customer - wants to make a payment. WinWin - wants to receive the fee and distribute it to riders. Bank - wants to ensure that the transaction is legit. Rider - wants to receive the income from the services.		
Brief Description: This use case describes how customers make payment.		
Trigger: Rider marks the ride as done **Type**: External		
Relationships: Association: Customer, Bank, Rider Include: - Extend: - Generalization: -		
Precondition: - Customer logs in - Rider marks the ride as done		
Postcondition: -		
Normal Flow of Events: 1. The system shows payment screen according to the chosen payment method 1. If the customer chooses to pay in cash, The S-1: make cash payment subflow is performed 2. If the customer chooses to pay in bank transfer The S-2: make transfer payment subflow is performed 3. If the customer chooses to pay in credit card or debit card The S-3: make card payment subflow is performed 2. The system shows the confirmation of the payment and the option to review the rider		
Subflows: S-1: Make cash payment 1. The customer pays to the rider directly 2. The rider confirms the payment has been done 3. The system deducts the rider’s cash credit		

S-2: Make bank transfer payment

1. The system shows the bank account of WinWin system
2. The customer goes to mobile banking application and make a transfer
3. The customer goes back to WinWin application
4. The customer submits the receipt to the system attachment slot
5. The system checks the submitted receipt

S-3: Make card payment

1. The system shows the added cards of the customer and “add more card” option
    1. If the customer selects “add more card” option, the customer fills the card adding form
2. The customer selects the card to use as this ride payment
3. The system redirects to the bank payment website (extension system from banks)

Alternate/Exceptional Flow:

1-a1: If the customer wants to change payment method, the customer can select “change payment method” option to go back to payment screen (N-1)

2-e1: If the payment is failed, shows the alert, and let customer chooses the new payment method and repeat payment screen (N-1)

Table 4: Use case description of “Initiate a ride”

Use Case Name: Initiate a ride	**ID:** 5	**Importance Level:** High
Primary Actor: Rider	**Use Case Type:** Detail, Essential	
Stakeholder and Interests: Customer - wants to track the rider who accepted their ride request. Rider - wants to initiate a ride to get income.		
Brief Description: This use case describes how riders initiate a ride.		
Trigger: Rider accepts a ride. **Type**: External		
Relationships: Association: Rider Include: Set rider availability Extend: Cancel a ride Generalization: -		
Precondition: - Rider logs in. - Rider accepts a ride.		
Postcondition: - After a rider marks the ride as done, the system initiates the “Make payment” process.		
Normal Flow of Events: 1. The system makes a notification to customer about the acceptance of their ride request Include flow: Set rider availability 2. The system shows the customer the profile of the rider who accepted their ride request 3. The system shows the current location of the rider who accepted their ride request 4. The system makes a notification to the customer about the arrival of the rider who accepted their ride request		
Subflows: -		
Alternate/Exceptional Flow: 2-a1: If the customer or the rider wants to cancel a ride, they can select “cancel a ride” option to cancel a ride		

Class diagrams and CRC Cards

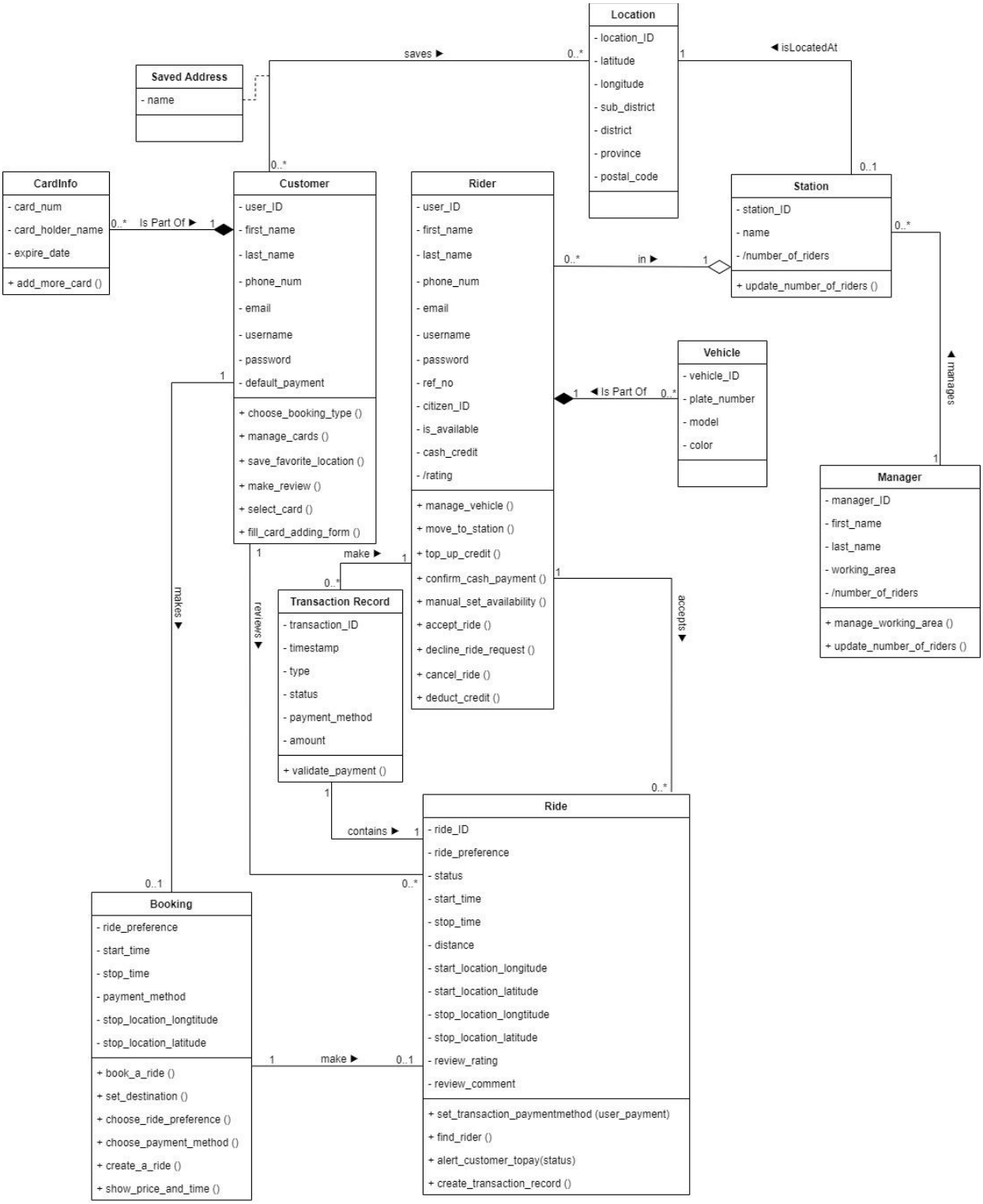


Figure 5: Class diagram of WinWin System

Class explanation

Customer	Users who want to find a ride
Rider	Users who provide a ride
Manager	Person who takes care riders in the area
Booking	Booking from customer who wants a ride, use as a control object to create a ride
Ride	Ride from rider that is provided to customer
Saved Address	Customer saves their favorite locations
Location	Location of customer and station
Station	Station of a rider
Vehicle	Vehicle of a rider
CardInfo	Information of credit/debit card
TransactionRecord	Record of the transaction in system

Front:

Class name: Customer	**ID:** 1	**Type:** Concrete, Domain
Description: An individual who wants to use motorcycle taxi services	**Associated Use Cases:** • Make Payment • Book a Ride	
Responsibilities Choose a booking type Make review Manage cards Fill card adding form Select cards Save favorite location	**Collaborators** Booking Ride CardInfo CardInfo CardInfo Location	

Back:

Attributes:	
• user_ID (char[8]) • first_name (char[50]) • last_name (char[50]) • phone_num (char[10])	• email (char[50]) • username (char[50]) • password (char[20]) • default_payment (char[2])
Relationships:	
Generalization (a-kind-of):	
Aggregation (has-parts):	CardInfo
Other Associations:	Ride, Booking, Location, Saved Address

Figure 6: CRC Card of Class “Customer”

Front:

Class name: Rider	**ID:** 2	**Type:** Concrete, Domain
Description: An individual who provides a ride	**Associated Use Cases:** • Make payment	
Responsibilities	**Collaborators**	
Manage vehicle	Vehicle	
Move to station	Station	
Top up credit	Transaction Record	
Confirms reception of cash payment	Transaction Record	
Manual set availability		
Accept ride	Ride	
Decline ride request	Ride	
Cancel ride	Ride	
Deduct credit		

Back:

Attributes:	
• user_ID (char[8])	• password (char[20])
• first_name (char[50])	• ref_no (char[16])
• last_name (char[50])	• citizen_ID (char[13])
• phone_num (char[10])	• is_available (boolean)
• email (char[50])	• cash_credit (double)
• username (char[50])	• rating (double)
Relationships:	
Generalization (a-kind-of):	
Aggregation (has-parts):	Vehicle, Station
Other Associations:	Ride, Transaction Record

Figure 7: CRC Card of Class “Rider”

Front:

Class name: Manager	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who takes cares of the riders in working area	**Associated Use Cases:** • Manage station	
Responsibilities Manage working area Update number of riders	**Collaborators** Station	

Back:

Attributes:	
• manager_ID (char[8]) • first_name (char[50]) • last_name (char[50])	• working_area (char[50]) • number_of_rider (integer)
Relationships: Generalization (a-kind-of): Aggregation (has-parts): Other Associations: Station	

Figure 8: CRC Card of Class “Manager”

Sequence diagram

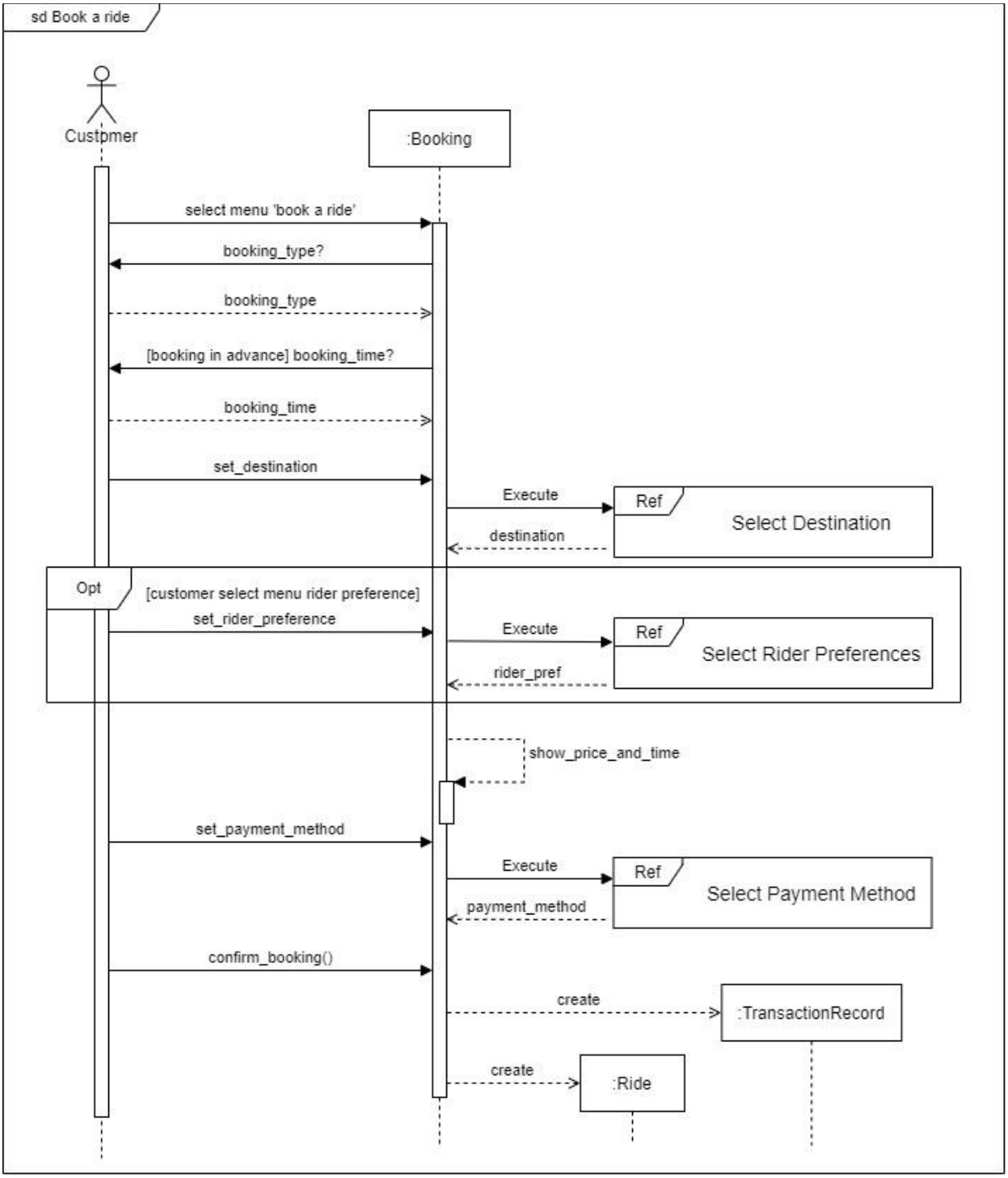


Figure 9: Sequence diagram of “Book a Ride” Use case

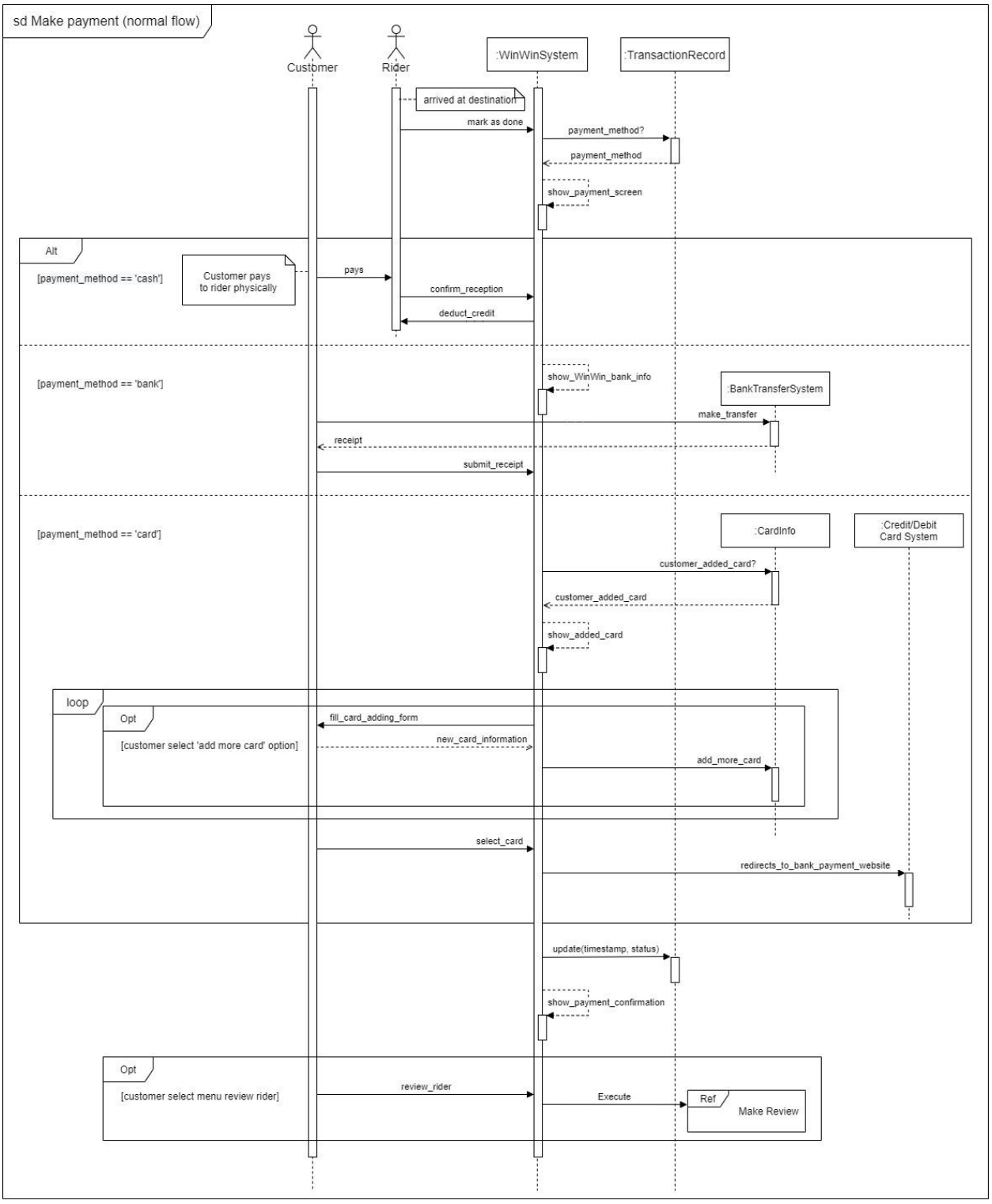


Figure 10: Sequence diagram of “Make payment” Use case

Behavioral State Machine

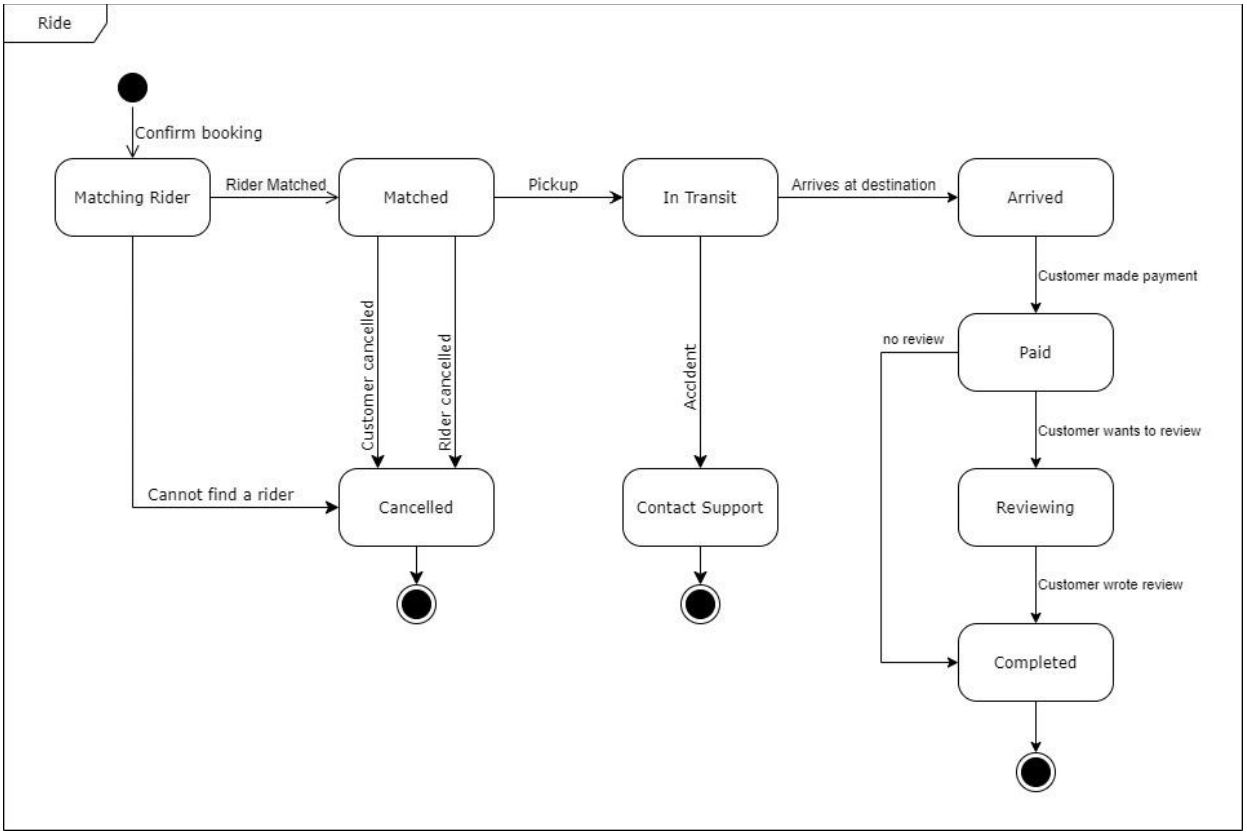
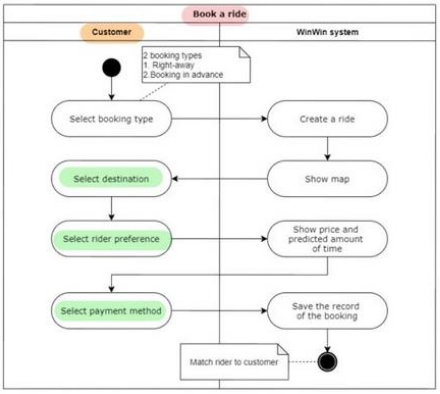


Figure 11: Behavioral state machine of Class “Ride”

Verifying and validating the analysis model

Verifying and validating functional model



Use Case Name: Book a ride	**ID:** 4	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	

Stakeholder and Interests:

Customer - wants to book a ride service.

Brief Description: This use case describes how customers book a ride.

Trigger: Customer asks to book a new ride service

Type: External

Relationships:

Association: Customer

Include: Select Payment Method, Select Destination

Extend: Select Rider Preferences

Generalization: -

Precondition: Customer logs in.

Postcondition: After a customer has booked a ride, the system initiates the "Match rider to customer" process.

Normal Flow of Events:

1. The customer selects "Booking" menu
2. The customer chooses a booking type between booking a right-away ride or booking a ride in advance.
3. The customer sets their destination for the ride.
   Include flow: Select Destination
4. The customer chooses their preference for the ride.
   Extension points: Select Rider Preferences
5. The system shows the price rate and predicts the amount of time of the requested ride.
6. The customer chooses their method of payment.
   Include flow: Select Payment Method
7. The system saves the record of the booking.

Subflows: -

Alternate/Exceptional Flow:

1-a1: If the customer does not want to book a ride anymore at any time, the customer can select exit button to go back to home menu

2-e1: If the customer books another ride at the same riding time as another ride request, the system notifies the customer to prevent a duplication of ride request.

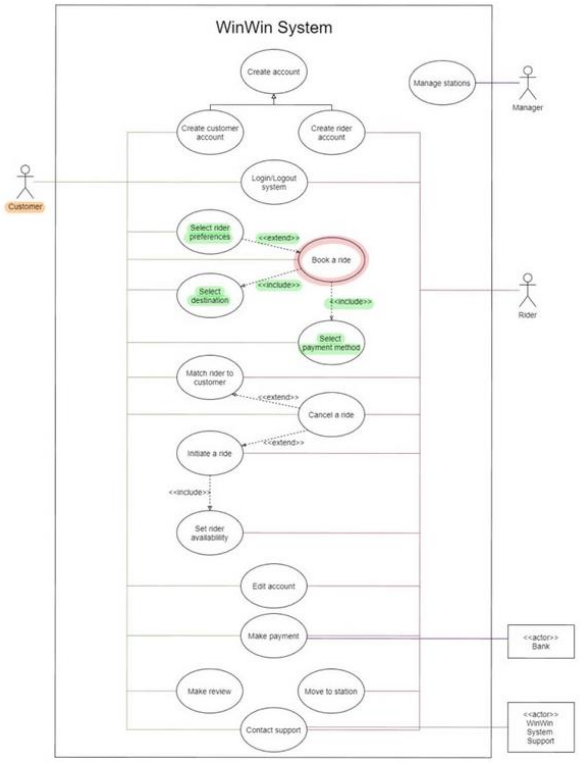
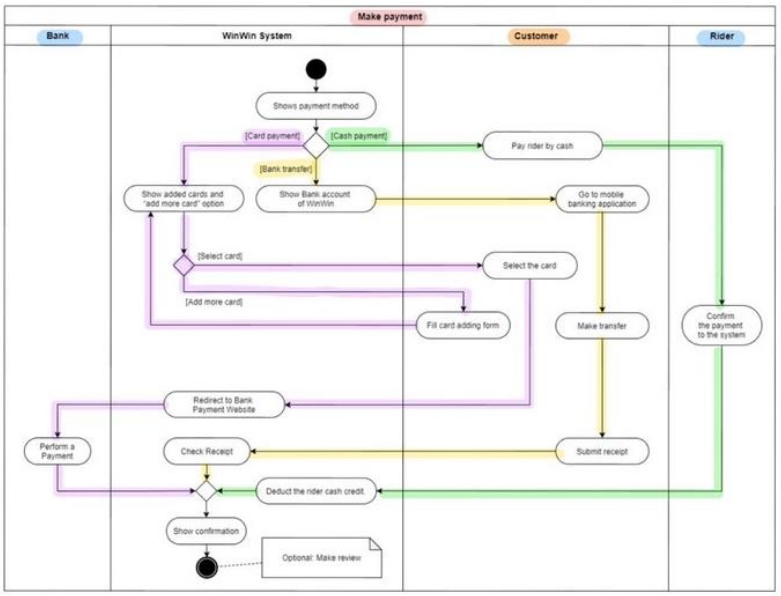


Figure 12: Validating of functional model of use case “Book a ride”



Use Case Name: Make payment	**ID:** 6	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	

Stakeholder and Interests:

Customer	- wants to make a payment.
WinWin	- wants to receive the fee and distribute it to riders.
Bank	- wants to ensure that the transaction is legit.
Rider	- wants to receive the income from the services.

Brief Description: This use case describes how customers make payment.

Trigger: Rider marks the ride as done

Type: External

Relationships:

Association:	Customer, Bank, Rider
Include:	-
Extend:	-
Generalization:	-

Precondition:

- Customer logs in
- Rider marks the ride as done

Postcondition: -

Normal Flow of Events:

1. The system shows payment screen according to the chosen payment method
    1. If the customer chooses to pay in cash,
       The S-1: make cash payment subflow is performed
    2. If the customer chooses to pay in bank transfer
       The S-2: make transfer payment subflow is performed
    3. If the customer chooses to pay in credit card or debit card
       The S-3: make card payment subflow is performed
2. The system shows the confirmation of the payment and the option to review the rider

Subflows:

S-1: Make cash payment

1. The customer pays to the rider directly
2. The rider confirms the payment has been done
3. The system deducts the rider's cash credit

S-2: Make bank transfer payment

1. The system shows the bank account of WinWin system
2. The customer goes to mobile banking application and make a transfer
3. The customer goes back to WinWin application
4. The customer submits the receipt to the system attachment slot
5. The system checks the submitted receipt

S-3: Make card payment

1. The system shows the added cards of the customer and "add more card" option
    1. If the customer selects "add more card" option, the customer fills the card adding form
2. The customer selects the card to use as this ride payment
3. The system redirects to the bank payment website (extension system from banks)

Alternate/Exceptional Flow:

1-a1: If the customer wants to change payment method, the customer can select "change payment method" option to go back to payment screen (N-3)

2-e1: If the payment is failed, shows the alert, and let customer chooses the new payment method and repeat payment screen (N-1)

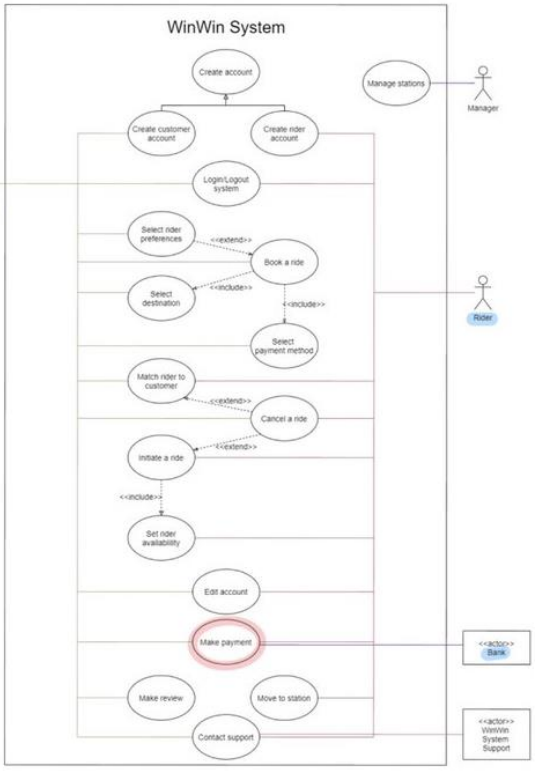


Figure 13: Validating of functional model of use case "Make payment"

Verifying and validating functional model

1. For each action/activity on the activity diagram, we validated the recorded event in the flows of the use-case description. for example, the regular flow of events in “Book a ride” description corresponds to the sequence in “Book a ride” activity diagram.
2. The sequence of "Book a ride" and "Payment" corresponded to the sequence in their activity diagram.
3. There is one and only use-case description for each use-case.
4. There are actors who exist in a use-case description such as customer in “Book a ride” description, Rider and Bank in “Make payment” description which all of them are shown on the use-case diagram with association link.
5. All other relationships in the use-case description such as “select payment method” in the “Book a ride” description is depicted on the use-case diagram.

Verifying and validating functional model

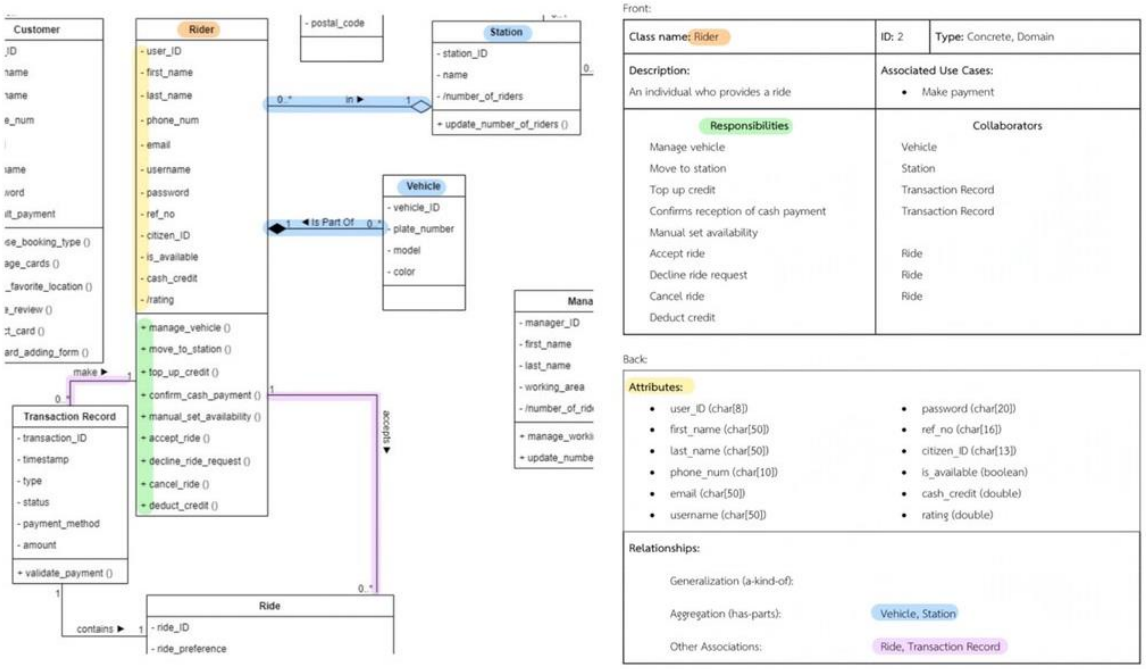
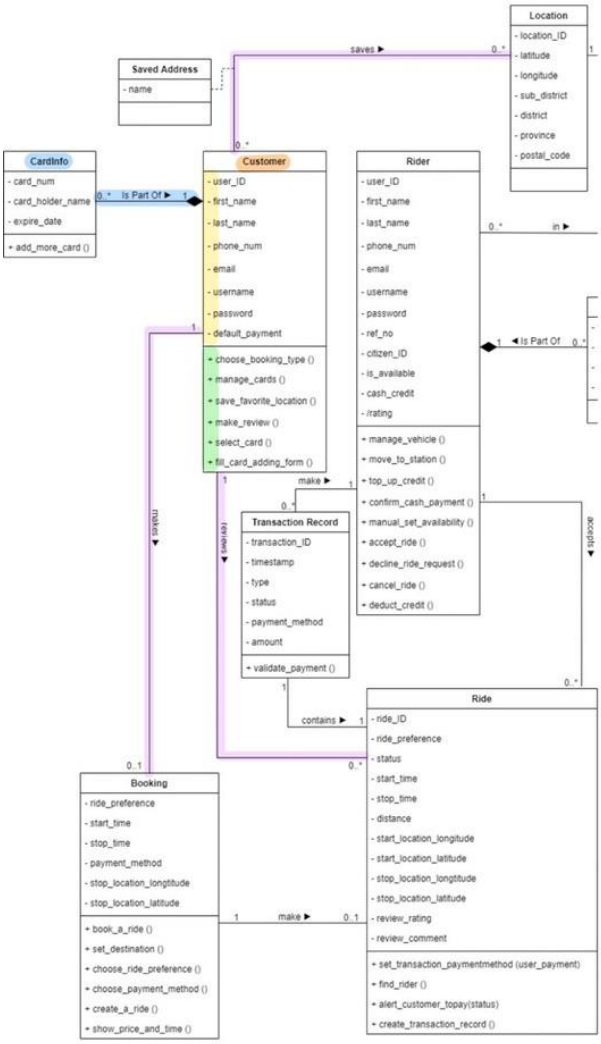


Figure 14: Validating of structural model of class “Rider”



Front:

Class name: Customer	ID: 1	Type: Concrete, Domain
Description: An individual who wants to use motorcycle taxi services	**Associated Use Cases:** • Make Payment • Book a Ride	
Responsibilities	**Collaborators**	
Choose a booking type	Booking	
Make review	Ride	
Manage cards	CardInfo	
Fill card adding form	CardInfo	
Select cards	CardInfo	
Save favorite location	Location	

Back:

Attributes:

- user_ID (char[8])
- first_name (char[50])
- last_name (char[50])
- phone_num (char[10])
- email (char[50])
- username (char[50])
- password (char[20])
- default_payment (char[2])

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts): CardInfo

Other Associations: Ride, Booking, Location, Saved Address

Use Case Name: Book a ride	ID: 4	Importance Level: High
Primary Actor: Customer	Use Case Type: Detail, Essential	

Stakeholder and Interests:

Customer - wants to book a ride service.

Brief Description: This use case describes how customers book a ride.

Trigger: Customer asks to book a new ride service

Type: External

Relationships:

Association:	Customer
Include:	Select Payment Method, Select Destination
Extend:	Select Rider Preferences
Generalization:	-

Precondition: Customer logs in.

Postcondition: After a customer has booked a ride, the system initiates the "Match rider to customer" process.

Normal Flow of Events:

1. The customer selects "Booking" menu
2. The customer chooses a booking type between booking a right-away ride or booking a ride in advance.
3. The customer sets their destination for the ride.
   Include flow: Select Destination
4. The customer chooses their preference for the ride.
   Extension points: Select Rider Preferences
5. The system shows the price rate and predicts the amount of time of the requested ride.
6. The customer chooses their method of payment.
   Include flow: Select Payment Method
7. The system saves the record of the booking.

Subflows: -

Alternate/Exceptional Flow:

1-a1: If the customer does not want to book a ride anymore at any time, the customer can select exit button to go back to home menu

2-e1: If the customer books another ride at the same riding time as another ride request, the system notifies the customer to prevent a duplication of ride request.

Figure 15: Validating of structural model of class "Customer"

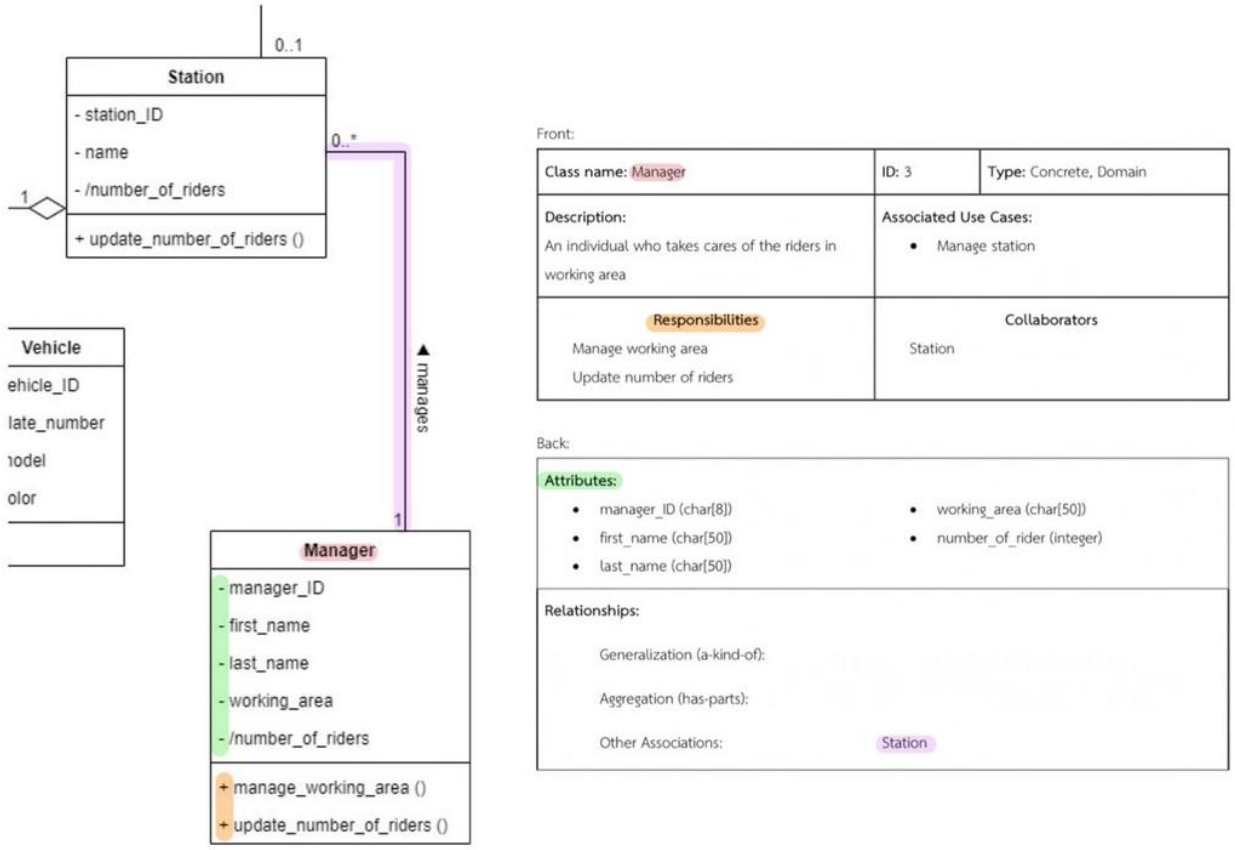


Figure 16: Validating of structural model of class “Manager”

Verifying and validating structural model

1. For each CRC card is associated with a class. For example, “Customer” CRC card is associated with “Customer” class on the class diagram.
2. The operations on the class diagram correspond to the responsibilities on the front of the card. For example, in the “Customer” CRC card, the “Choose a booking type” responsibility is included as an operation in the “Customer” class.
3. Collaborators on the front of the card imply some type of relationship on the back of the card as we can see from the “Customer” CRC card as “Booking” “Ride” “CardInfo” and “Location” all of which showed on the back of the card as some type of relationship.
4. On the class diagram, the attributes on the back of the card are displayed as attributes. For example, “user ID,” “email,” “first name,” and other attributes from the “Customer” CRC card are shown as attributes on the “Customer” class.
5. Relationships on the back of the card were depicted on the class diagram.

Verifying and validating behavioral model

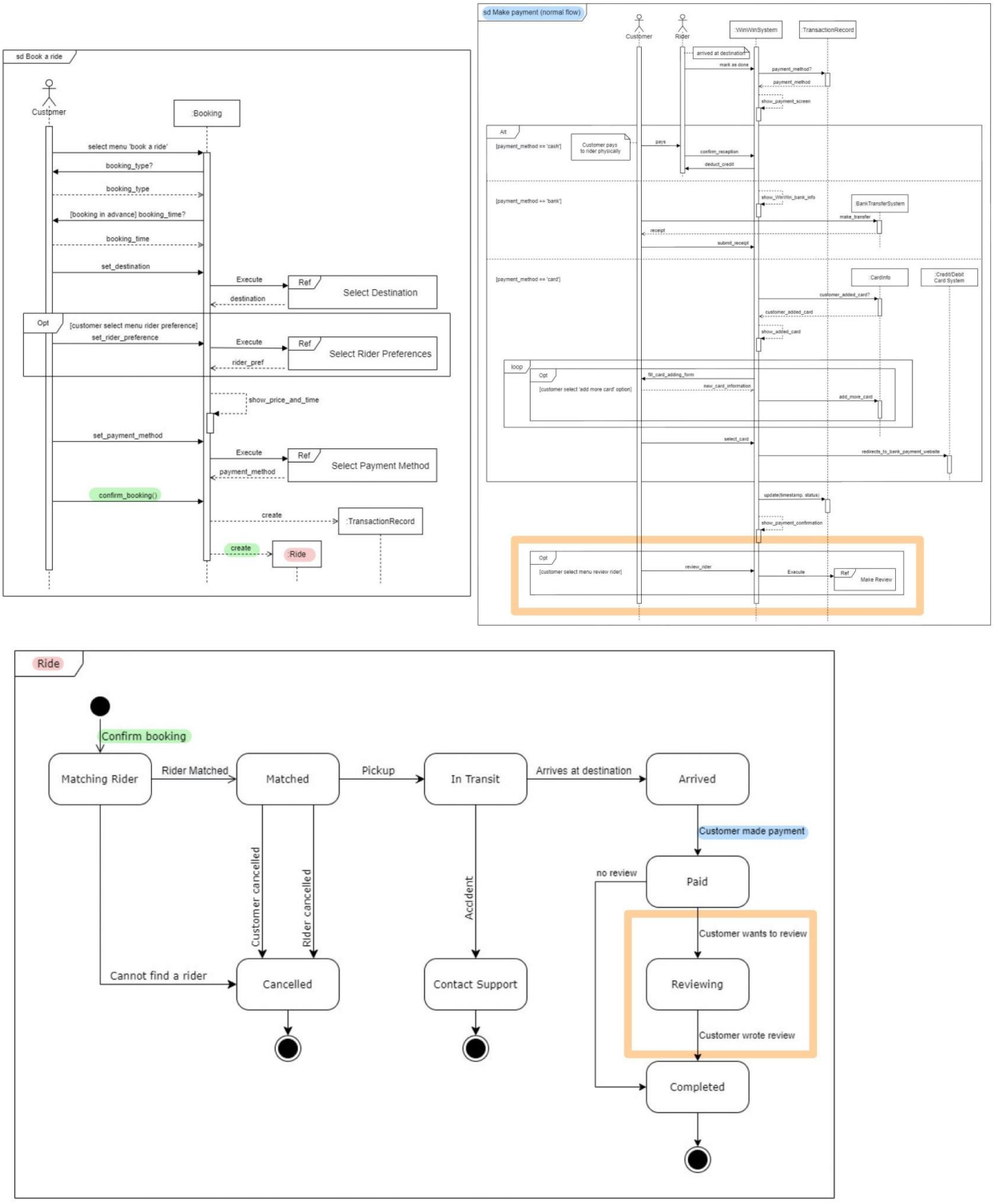


Figure 17: Validating of behavioral model of use case “Book a ride” and use case “Make payment”

Verifying and validating behavioral model

1. The message “confirm booking” in sequence diagram “Book a ride” is consistent with starting point of behavioral state machine of class “Ride”
2. The entire of use case “Make payment” is match with transition “Customer made payment” in behavioral state machine of class “Ride”

Verifying and validating between models

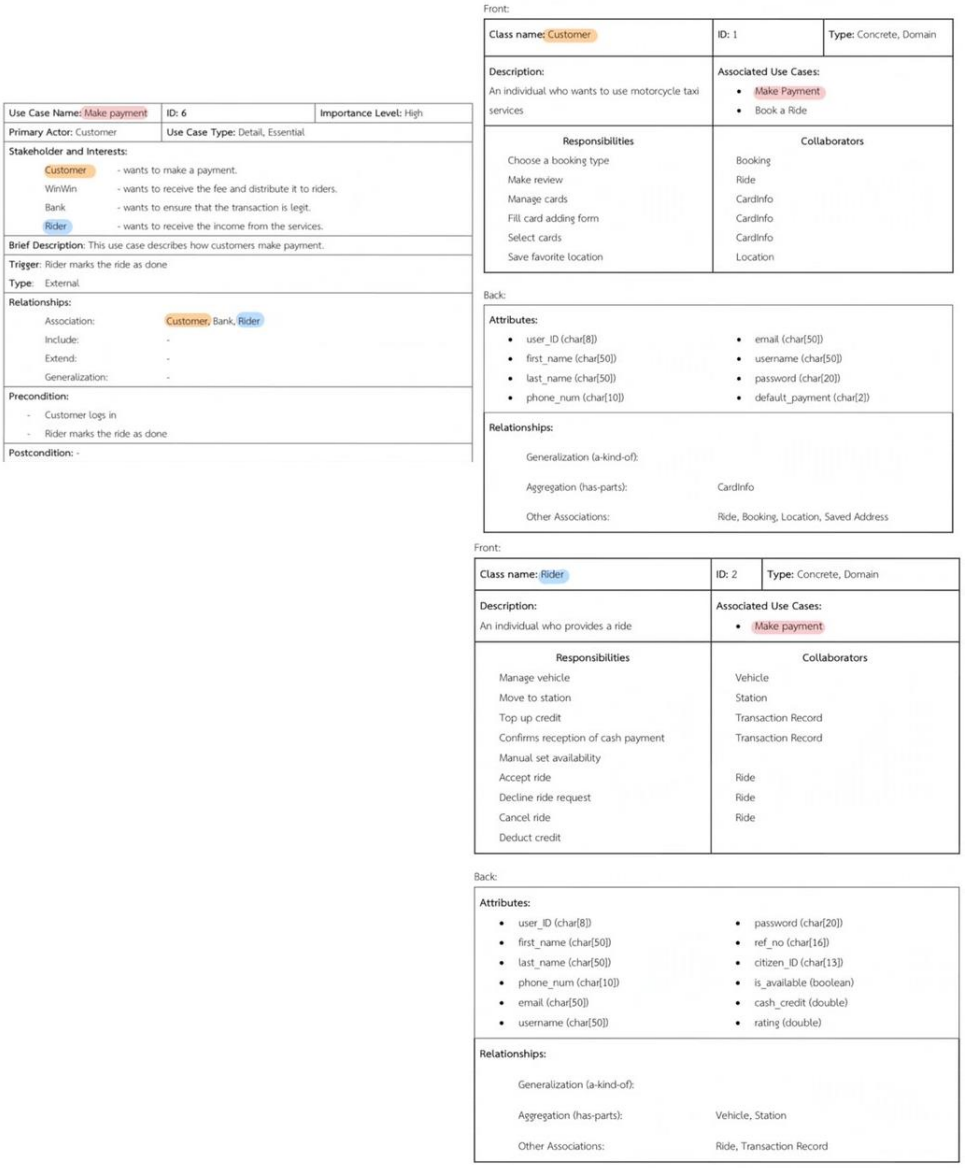
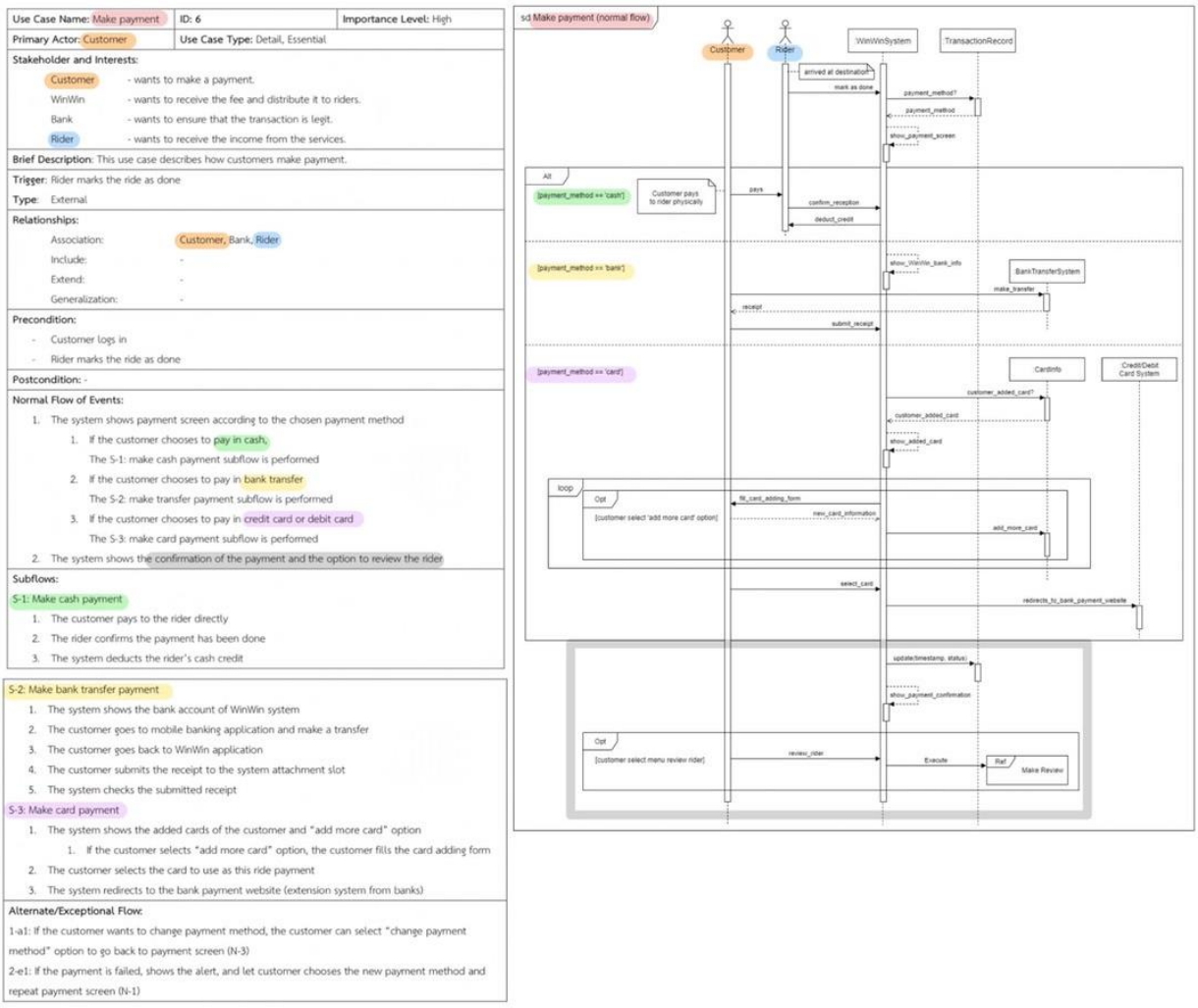


Figure 18: Validating between functional model and structural model in use case “Make payment”



Use Case Name: Make payment	**ID:** 6	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	

Stakeholder and Interests:

Customer	- wants to make a payment.
WinWin	- wants to receive the fee and distribute it to riders.
Bank	- wants to ensure that the transaction is legit.
Rider	- wants to receive the income from the services.

Brief Description: This use case describes how customers make payment.

Trigger: Rider marks the ride as done

Type: External

Relationships:

Association:	Customer, Bank, Rider
Include:	-
Extend:	-
Generalization:	-

Precondition:

- Customer logs in
- Rider marks the ride as done

Postcondition: -

Normal Flow of Events:

1. The system shows payment screen according to the chosen payment method
    1. If the customer chooses to pay in cash,
       The S-1: make cash payment subflow is performed
    2. If the customer chooses to pay in bank transfer
       The S-2: make transfer payment subflow is performed
    3. If the customer chooses to pay in credit card or debit card
       The S-3: make card payment subflow is performed
2. The system shows the confirmation of the payment and the option to review the rider

Subflows:

S-1: Make cash payment

1. The customer pays to the rider directly
2. The rider confirms the payment has been done
3. The system deducts the rider's cash credit

S-2: Make bank transfer payment

1. The system shows the bank account of WinWin system
2. The customer goes to mobile banking application and make a transfer
3. The customer goes back to WinWin application
4. The customer submits the receipt to the system attachment slot
5. The system checks the submitted receipt

S-3: Make card payment

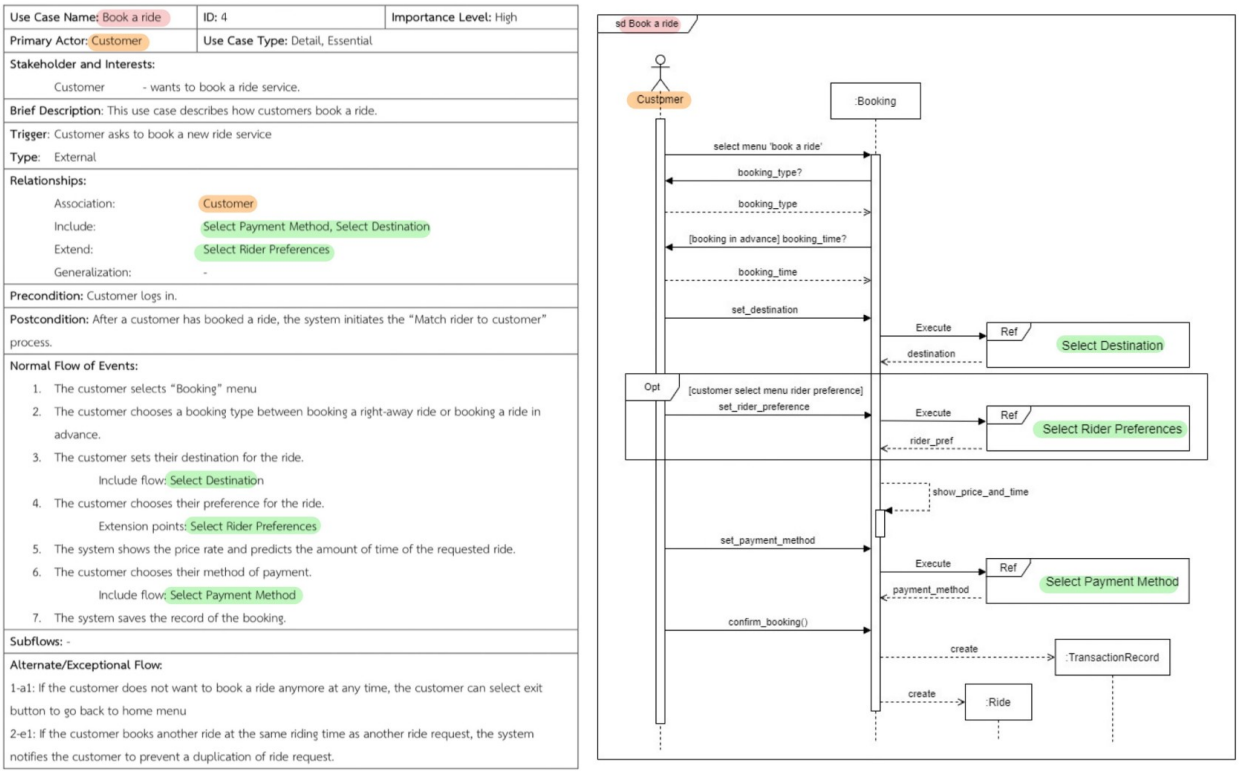
1. The system shows the added cards of the customer and "add more card" option
    1. If the customer selects "add more card" option, the customer fills the card adding form
2. The customer selects the card to use as this ride payment
3. The system redirects to the bank payment website (extension system from banks)

Alternate/Exceptional Flow:

1-a1: If the customer wants to change payment method, the customer can select "change payment method" option to go back to payment screen (N-3)

2-e1: If the payment is failed, shows the alert, and let customer chooses the new payment method and repeat payment screen (N-1)

Figure 19: Validating between functional model and behavioral model of use case "Make payment"



Use Case Name: Book a ride	**ID:** 4	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	

Stakeholder and Interests:
Customer - wants to book a ride service.

Brief Description: This use case describes how customers book a ride.

Trigger: Customer asks to book a new ride service

Type: External

Relationships:
Association: Customer
Include: Select Payment Method, Select Destination
Extend: Select Rider Preferences
Generalization: -

Precondition: Customer logs in.

Postcondition: After a customer has booked a ride, the system initiates the "Match rider to customer" process.

Normal Flow of Events:
1. The customer selects "Booking" menu
2. The customer chooses a booking type between booking a right-away ride or booking a ride in advance.
3. The customer sets their destination for the ride.
   Include flow: Select Destination
4. The customer chooses their preference for the ride.
   Extension points: Select Rider Preferences
5. The system shows the price rate and predicts the amount of time of the requested ride.
6. The customer chooses their method of payment.
   Include flow: Select Payment Method
7. The system saves the record of the booking.

Subflows: -

Alternate/Exceptional Flow:
1-a1: If the customer does not want to book a ride anymore at any time, the customer can select exit button to go back to home menu
2-e1: If the customer books another ride at the same riding time as another ride request, the system notifies the customer to prevent a duplication of ride request.

Figure 20: Validating between functional model and behavioral model of use case "Book a ride"

Verifying and validating between models

1. Between structural model and functional model such as CRC card and Use-case description, all the actors and objects are consistent with each other, for example the "Customer" CRC card is showed as the stakeholder on the "Make payment" use-case description
2. Between functional model and behavioral model such as "Book a ride" sequence diagram and "Book a ride" use-case description, all the behaviors are referred to one scenario described in description.
3. All the actors are consistent between models. Customer and rider in the "Make payment" description, for example, are compatible with the "Make payment" sequence diagram.
4. The flows of the sequence diagrams are related to the normal flow of events in their use-case description.

Contributions

Table 5: Contributions

Name	Contributed part	Level of Achievement
6230123921 Thitaree Setwipattanachai	Proposed method of systems analysis, Activity Diagram, CRC Card	5
6230252121 Tarm Kalavantavanich	Use Case Diagram, Sequential Diagram, Behavioral State Machine, Verifying and validating the analysis model	5
6231301421 Kanokpich Chaiyawan	Proposed method of systems analysis, List of stakeholders, Use Case Description, CRC Card	5
6231304321 Kittipong Deevee	List of stakeholders, Activity Diagram, Class Diagram	5
6231307221 Jirawat Kusalangkurwat	Proposed method of systems analysis, Use Case Description, CRC Card	5
6231333521 Nopdanai Sayamnet	Objective of the analysis document, Proposed method of systems analysis, List of stakeholders, Class Diagram, CRC Card, Verifying and validating the analysis model	5
6231353021 Raviporn Akekunanon	Definitions, Details of requirements, Use Case Description, Class Diagram, Sequential Diagram, Document correction and organization	5
6231372021 Atiwat Deepo	Details of requirements, Activity Diagram, CRC Card, Sequential Diagram	5