Milestone 4 System and software design specification

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

Design 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.

To-be system overview

This application is designed to bring benefits to both customers and motorcycle taxis. Both customers and riders will use the same application, however, their interface will be different based on their customer 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 customers 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 customer side to register, it is necessary to verify identity through personal information.

The matchmaking system starts when the customer 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 customer's ride booking. When a motorcycle taxi accepts a ride from any customer, the customer 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.

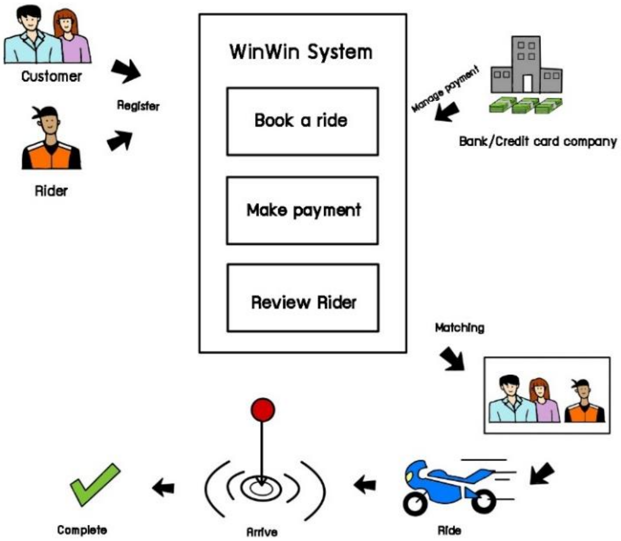


Figure 1: The flow of to-be system

Objective of the project

The objective of this project is to develop a mobile application for motorcycle taxi riders and customers to use. WinWin is to be an intermediate platform between motorcycle taxi riders and customers, doing matchmaking between them to provide ride-hailing services. After registering with the system, WinWin allows customers to request rides without having to come to the motorcycle taxi stations, providing conveniences to customers as well as boosting motorcycle taxi riders' work.

Reference document

- Project proposal document pdf
- Analysis phase document pdf
- Design phase document template pdf
- WinWin Final Pitch Deck pdf
- Class slide
    - Chapter 6 for CRUDE matrix
    - Chapter 7 for Package diagram, Component diagram
    - Chapter 8 for Method specification
    - Chapter 10 for User Interface design
    - Chapter 11 for Physical Architecture design
    - Chapter 13 for Migration plan, Post-implementation activities

Objective of the design document

This document is created as a part of the development process of the system with the objective of providing a description of how the to-be system operates in a mobile-based application, to describe the detail on the procedures of designing, and to ensure that the design corresponds with the purpose of the system and its users. And this document is to be used as reference for correct understandings of how the system operates. In this document, we will represent the to-be system by doing the following.

1. Clarify the overview of the to-be system
2. Elaborate on the system constraints by explaining design criteria, precondition, postconditions, and invariants. And engage in design activities.
3. Various activities such as making diagrams, which includes use-case diagram, class diagram, and component diagram, making CRC class, writing up CRUDE matrix and method specification.
4. Create HCI (Human-Computer Interaction) design as User Interface.
5. Conduct physical architecture design to create architectural models, hardware, and software specifications, also review non-functional requirements of the architectural models as the physical architecture design and present them in the technical term for technicians and developers and guarantee the system performance to the users of the system.
6. Specify the system installation and post-implementation activities by identifying the impact of system design, Conversion Plan, Change Management, and Project Team Review.

Design criteria and design activities

For the long-term viability of a system, simplicity of maintenance, and future adjustments, cohesion, which refers to how single-minded a module is, and coupling, which refers to how independent or interrelated the modules are, are used as a set of metrics to evaluate the designed system. When modules and classes are highly interdependent, the system is more likely to be fixed and difficult to change after implementation, resulting in a poorly designed system.

Coupling

Dependencies between classes were considerably minimized by utilizing the Law of Demeter to reduce interaction coupling and using inheritance to support just generalization/specialization to reduce inheritance coupling. As a result, our system has a relatively low coupling, which is quite acceptable.

Interaction Coupling

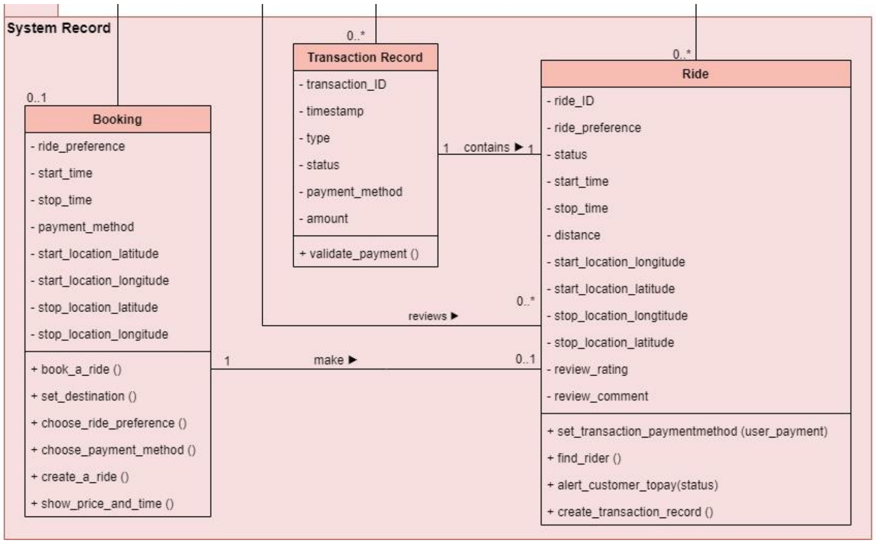


Figure 2: Interaction Coupling

For an example of interaction coupling, the “alert_customer_topay” method has no direct coupling as the “status” parameter was passed from the “Ride” class itself or the “set_transaction_paymentmethod” method which has only a data-type interaction coupling between “Transaction Record” and “Ride” class.

Inheritance Coupling

Inheritance was not employed in the class diagrams since it was not essentially required. As a result, there will be no issues concerning inheritance coupling.

Cohesion

Method Cohesion

Each methods present are made to perform only one purpose

Class Cohesion

Each class present is made to represent only one thing, with all the attributes and methods necessary to fully define the thing.

Generalization Cohesion

Classes in a hierarchy should show a-kind-of relationship, not associations or aggregations. As our design does not have classes in a hierarchy, this issue is redundant.

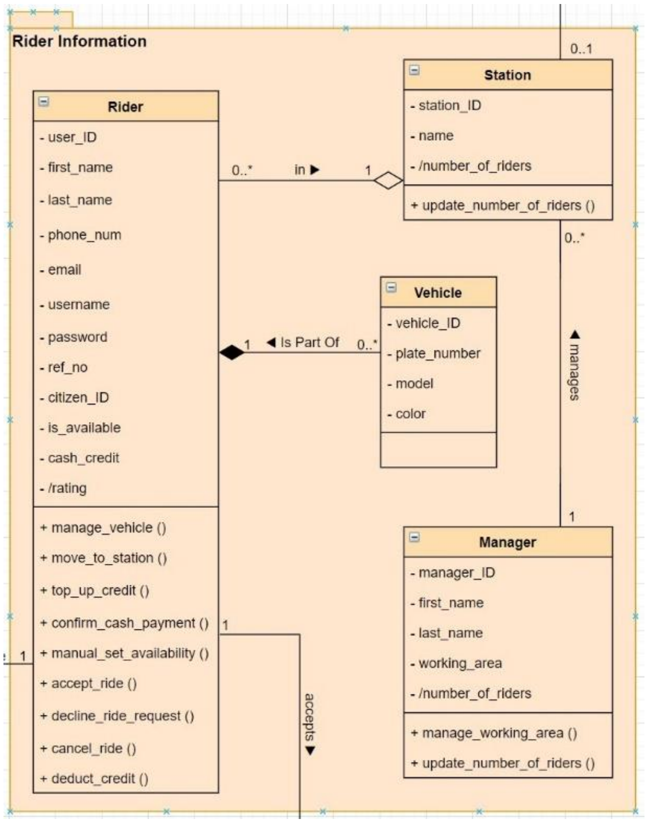


Figure 3: Rider information Class package

Even though the "Rider Information package" contains the most complex functionality of all the packages in the system, we can see that each method operates independently, and so the classes are created to represent just one thing, with all the attributes and methods necessary to fully define the thing.

Design activities

Adding specifications

As we have already reviewed the analysis model, all classes are both essential and sufficient to solve the problem and there are no extra or unused attribute methods. Furthermore, the visibility of classes is thoroughly studied. So does the method signature which consists of the name of the method, parameters and type of return are added. Finally, constraints and the handling of constraint violations are properly described.

Restructuring the Design

1. Factoring

We use factoring techniques to separate aspects from a class to simplify the design, the example of this statement is separating "Booking" class from "Ride" class. By doing so, the role of "Ride" class can be understood better that an instance of "Ride" can be instantiated only when all the necessary attributes are finally set while the attributes of a "Booking" object can keep on getting changed.

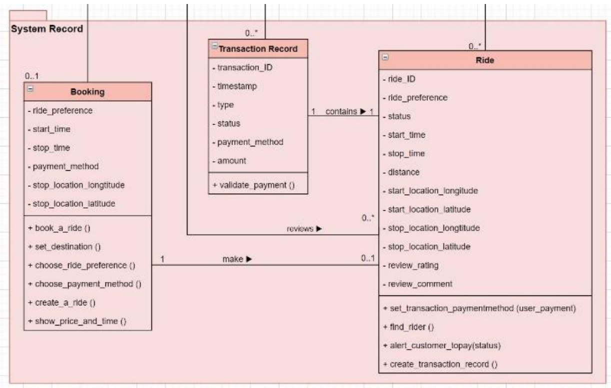


Figure 4: System record Class package

2. Normalization

As we use normalization techniques, we convert the association class ("Saved Address" class) to a normal class.

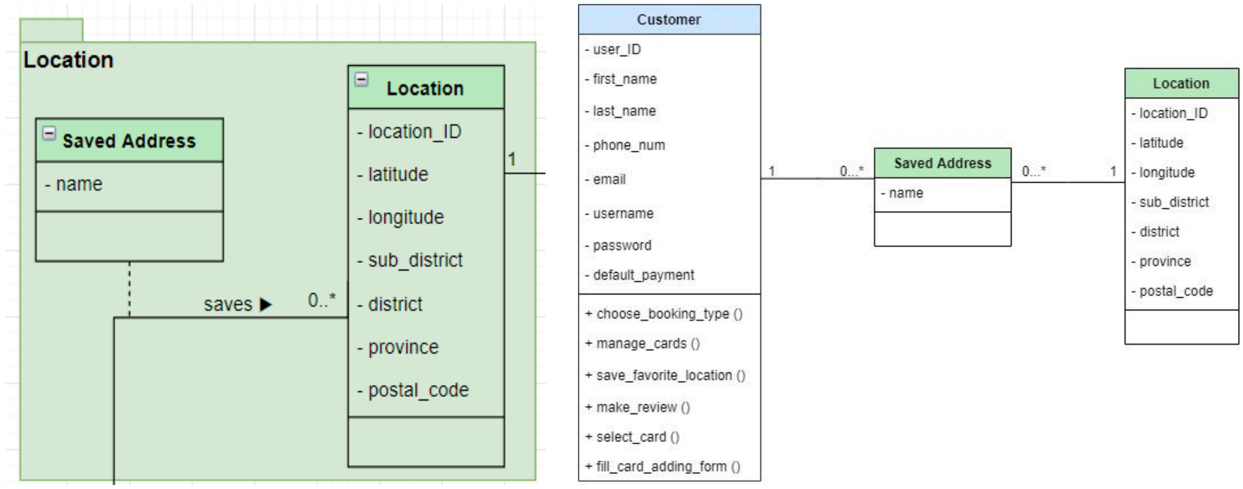


Figure 5: Normalization of Location Class package

In the implementation process, we will use normalization techniques to convert the association class ("Saved Address" class) to a normal class. Though the association class in the class diagram will be maintained as is for better understanding and representation.

Optimizing the Design

1. Review access paths between objects

It may not be essential to add an attribute for the object at the endpoint of the path since there is no path that meets the conditions that

1. The path is long since it involves more than three steps.
2. The message is frequently sent.

2. Review each attribute of each class

After reviewing the class diagram, any attributes that are only read and/or updated by only a single class are already in that calling class.

3. Review direct and indirect fan-out of each method

Because we have already used the factorization approach, the fan-out message of each method is drastically decreased.

4. Consider execution order of statements in often-used methods to arrange them for efficiency

Each method already has the optimal execution order given to it.

5. Avoid re-computation by creating derived attributes and triggers

“Rider” class has a derived attribute that is rating which is an average of all the ratings given for each ride the rider has provided. The derived rating attribute is updated by a trigger when a new rating is given to a not-yet-rated ride, calling for re-computation of the derived attribute.

6. Consider combining classes that form a one-to-one association (as both must exist for either to exist)

Although we have a one-to-one association (Ride and Transaction Record class), collapsing is not preferable as the cohesion of the classes will be lost. There is only one one-to-one association (Ride and Transaction Record class) which cannot combine into a single class.

Introduction of an overview of system design modeling

In this paper, we show how we employed the OOSAD method to build the to-be system in such a way that it maximizes cohesion while reducing coupling in object-oriented aspects.

OOSAD provides a dependable standard for object-oriented systems, allowing the system design to better convey its features to viewers. UML diagrams such as class diagrams, sequence diagrams, etc, were used to provide details on relationships and interactions between objects, illustrating and clarifying the system model further until all objectives were satisfied. Using OOSAD concepts in conjunction with UML diagrams helps us in the following ways:

- Data encapsulation and information hiding are provided, allowing developers to build systems that are difficult to meddle with by other parts of the system.
- Ease on the scaling of the system, both up and down, with greater efficiency than other analysis methods.
- Unlike structured analysis, which focuses more on procedures, employing UML in analysis and design focuses on data rather than the procedure.
- Allowing effective management of software complexity by modularity.

System design constraint

1. **Operational constraints**
    1. The to-be system is to be a mobile application so the user interface shall be compatible with various sizes of smartphone screen.
    2. The system shall automatically back up its database daily.
    3. The system needs an internet connection.
    4. The system needs admin to maintain and operate the system.
2. **Performance constraints**
    1. The system shall respond in less than 5 seconds for every interaction between system and user.
3. **Security constraints**
    1. The system shall authenticate users and riders using userID and password.
    2. The system shall be able to keep users' transactions confidential.
    3. The user's information shall be visible only to the account owner and the verified actor such as payment authorization.
    4. The system must protect users' financial information such as bank account number, credit card number and CVV.
    5. The rider has to pass an identity verification process before being able to proceed with any other use-cases.
4. **Usability constraints**
    1. The system should be easy to use for both new and experienced users
5. **Cultural & political influence constraint**
    1. The available region for customers to select the destination is strictly only in the Bangkok Metropolitan Region.
    2. The user interface of the system will be displayed in Thai languages.
6. **Legal implications**
    1. The system must comply with the following act:
        - Electronic Transaction Act B.E. 2544 (2001)
        - Direct Sales and Direct Marketing Act B.E. 2545 (2002)
        - Payment System Act B.E. 2560 (2017)
        - Personal Data Protection Act B.E. 2562 (2019)
7. **System design and development time-period constraint**
    1. The design and development of the system is to be completed before the end of February 2022.

Detail Essential Use Case diagram

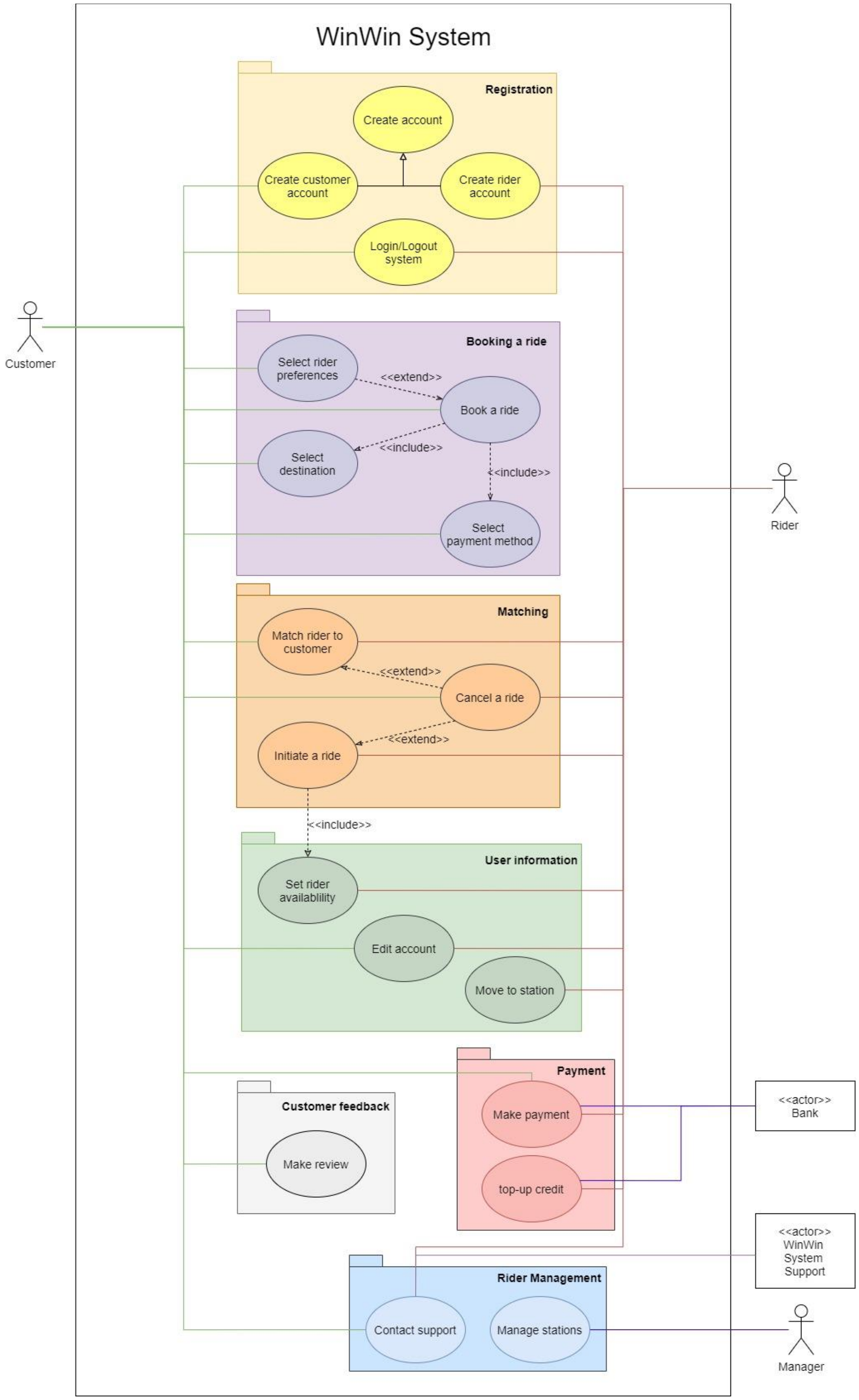


Figure 6: Use case diagram of WinWin System

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.

Class diagram

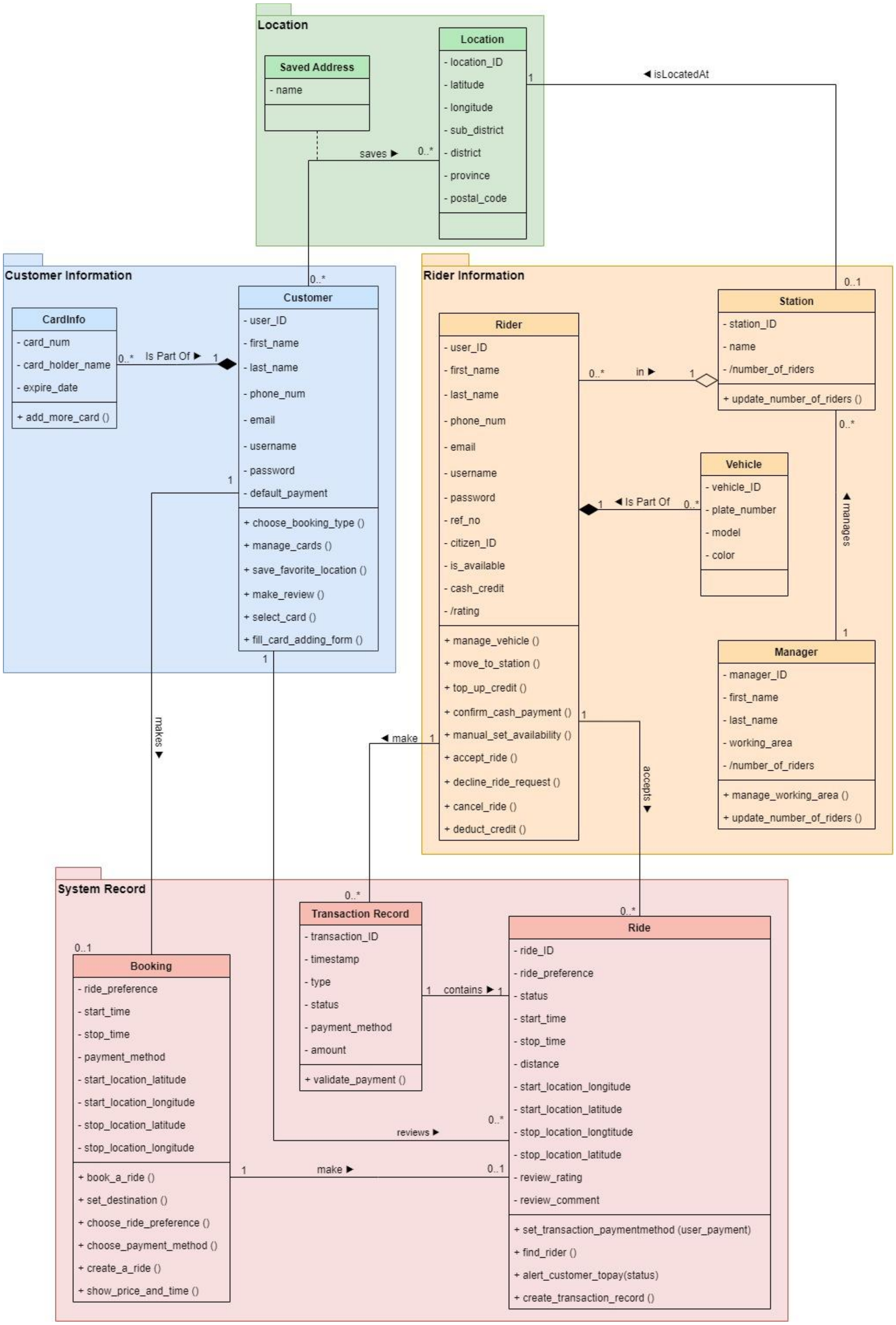


Figure 7: Class diagram of WinWin System

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

CRC Cards

Customer

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

Rider

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])
- first_name (char[50])
- last_name (char[50])
- phone_num (char[10])
- email (char[50])
- username (char[50])
- password (char[20])
- ref_no (char[16])
- citizen_ID (char[13])
- is_available (boolean)
- cash_credit (double)
- rating (double)

Relationships:

Generalization (a-kind-of):	
Aggregation (has-parts):	Vehicle, Station
Other Associations:	Ride, Transaction Record

Component diagram

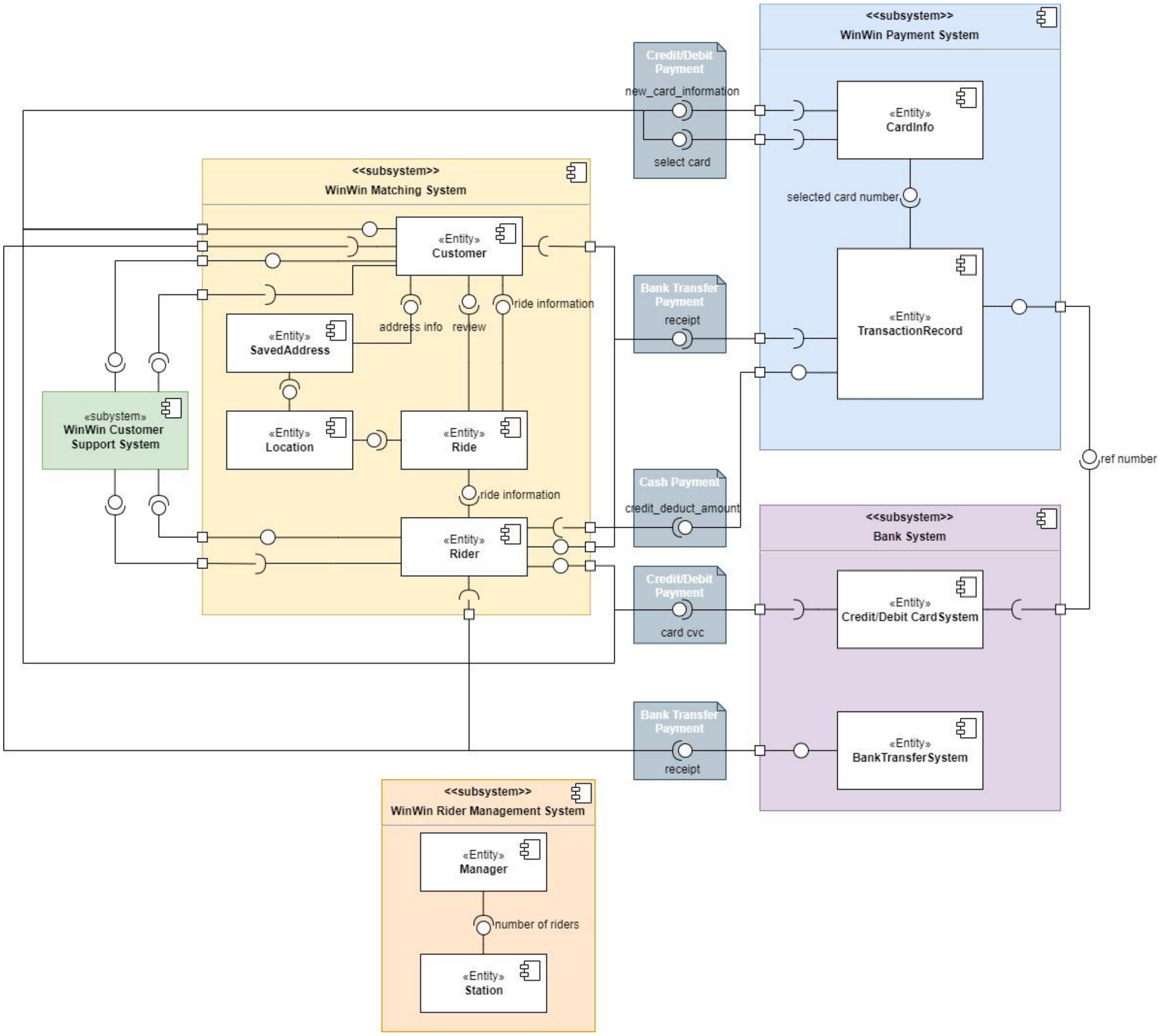


Figure 8: Component diagram

CRUDE Matrix

Table 1: CRUDE Matrix

Make Payment	Customer Actor	Rider Actor	Bank Actor	Customer Class	CardInfo Class	Rider Class	Ride Class	TransactionRecord Class
Customer Actor			R	R E	C R	R	R U	C U
Rider Actor				R			R U	
Bank Actor							R	
Customer Class					E		R U E	C U
CardInfo Class				R				
Rider Class				R				C U
Ride Class			R	R E		R E		
TransactionRecord Class								

Book a Ride	Customer Actor	Customer Class	SavedAddress Class	Location Class	Ride Class	Booking Class
Customer Actor		E	C R	C R		C R U D E
Customer Class			C R	C R		
SavedAddress Class				R		
Location Class						
Ride Class		R		R		R
Booking Class		R	R	R	C E	E

Method specification

Book a ride

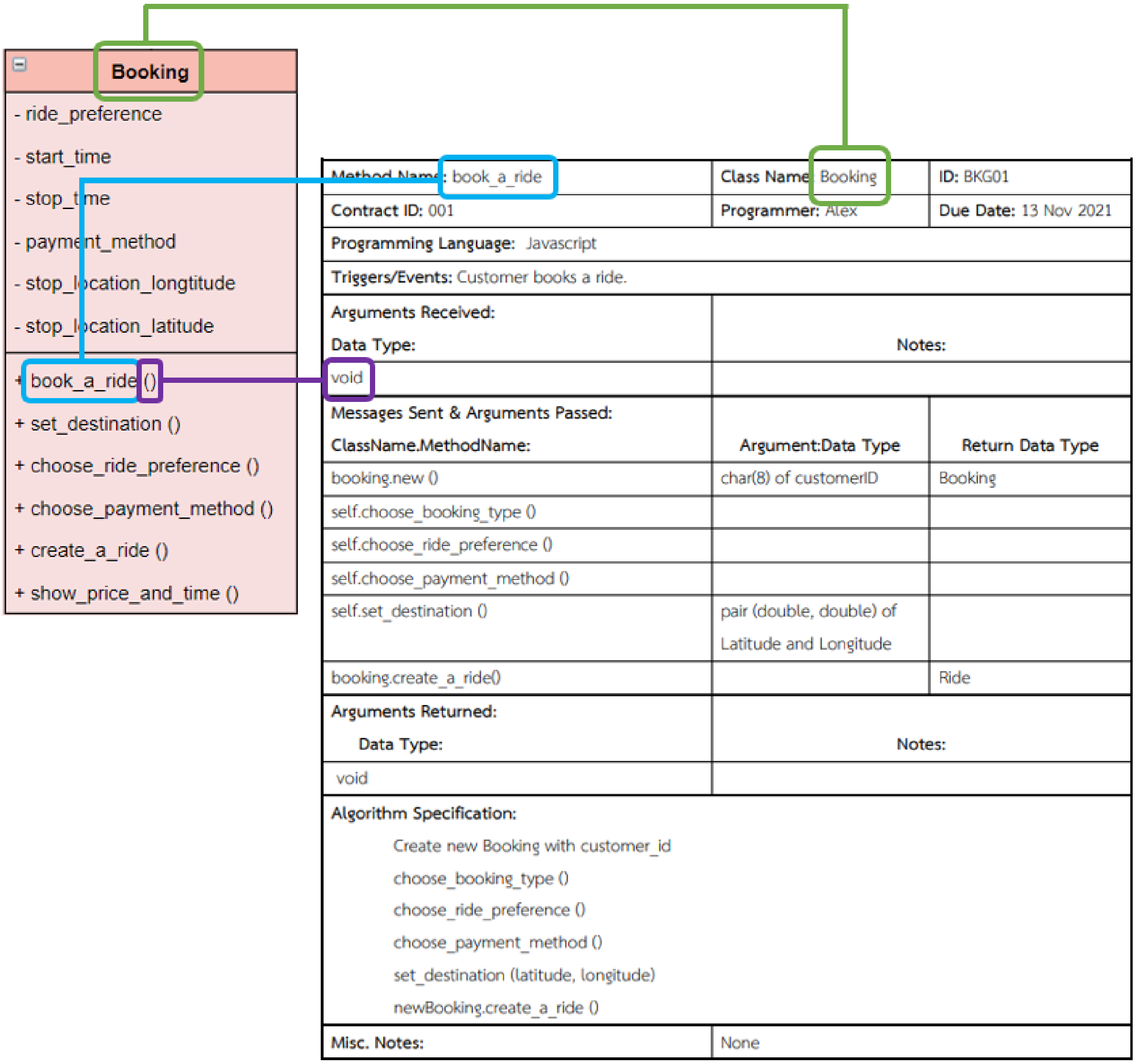
Method Name: book_a_ride	**Class Name:** Booking	**ID:** BKG01
Contract ID: 001	**Programmer:** Alex	**Due Date:** 13 Nov 2021
Programming Language: Javascript		
Triggers/Events: Customer books a ride.		
Arguments Received: **Data Type:**	**Notes:**	
void		
Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument:Data Type**	**Return Data Type**
booking.new ()	char(8) of customerID	Booking
self.choose_booking_type ()		
self.choose_ride_preference ()		
self.choose_payment_method ()		
self.set_destination ()	pair (double, double) of Latitude and Longitude	
booking.create_a_ride()		Ride
Arguments Returned: **Data Type:**	**Notes:**	
void		
Algorithm Specification: Create new Booking with customer_id choose_booking_type () choose_ride_preference () choose_payment_method () set_destination (latitude, longitude) newBooking.create_a_ride ()		
Misc. Notes:	None	

Create Transaction Record

Method Name: create_transaction_record	**Class Name:** Ride	**ID:** RIDE01
Contract ID: 002	**Programmer:** Alex	**Due Date:** 13 Nov 2021
Programming Language: JavaScript		
Triggers/Events: Customer confirms booking.		
Arguments Received: **Data Type:**	**Notes:**	
void		
Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument:Data Type**	**Return Data Type**
TransactionRecord.new ()	char(8) of ride_ID	TransactionRecord
self.set_transaction_paymentmethod ()	varchar(20) of user_payment	
Arguments Returned: **Data Type:**	**Notes:**	
void		
Algorithm Specification: create new TransactionRecord with ride_id set_transaction_paymentmethod (user_payment)		
Misc. Notes:	None	

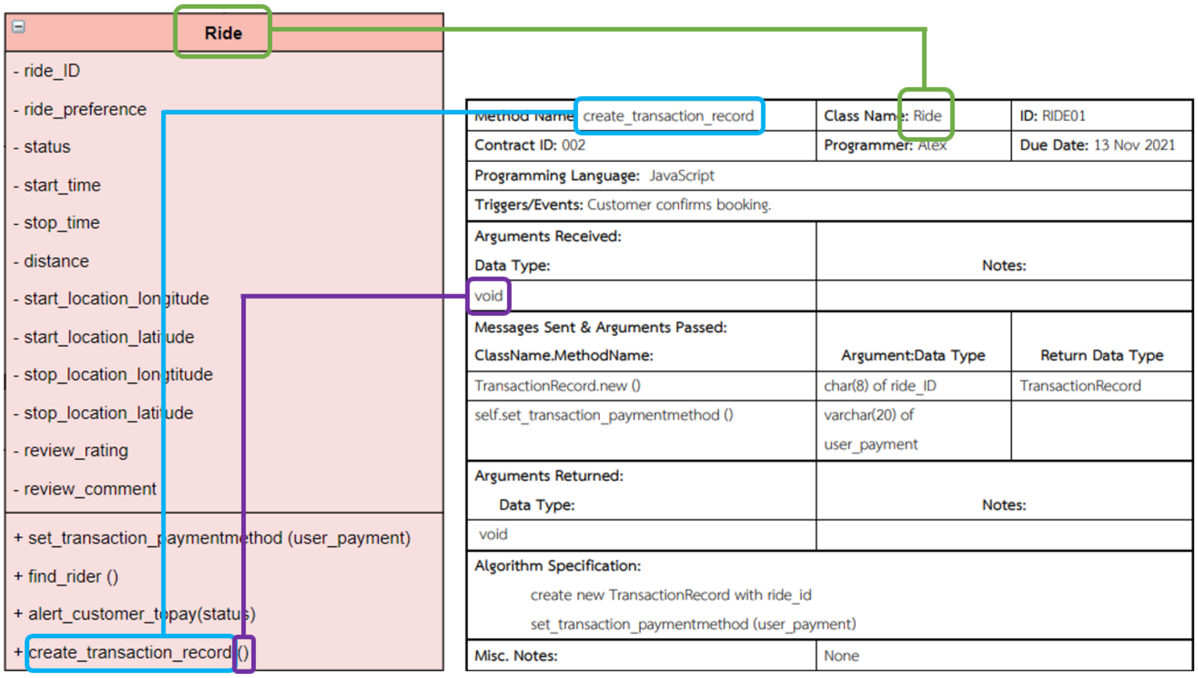
Verifying and class and method design

This section will demonstrate the validation process by reviewing the consistency between class diagram and method specification, checking the method name, the class it belongs to, and the data type of parameter passed in.



Method Name: book_a_ride	**Class Name:** Booking	**ID:** BKG01
Contract ID: 001	**Programmer:** Alex	**Due Date:** 13 Nov 2021
Programming Language: Javascript		
Triggers/Events: Customer books a ride.		
Arguments Received: **Data Type:**	**Notes:**	
void		
Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument:Data Type**	**Return Data Type**
booking.new ()	char(8) of customerID	Booking
self.choose_booking_type ()		
self.choose_ride_preference ()		
self.choose_payment_method ()		
self.set_destination ()	pair (double, double) of Latitude and Longitude	
booking.create_a_ride()		Ride
Arguments Returned: **Data Type:**	**Notes:**	
void		
Algorithm Specification: Create new Booking with customer_id choose_booking_type () choose_ride_preference () choose_payment_method () set_destination (latitude, longitude) newBooking.create_a_ride ()		
Misc. Notes:	None	

Figure 9: Validating “book_a_ride” method and “Booking” class



Ride

- ride_ID
- ride_preference
- status
- start_time
- stop_time
- distance
- start_location_longitude
- start_location_latitude
- stop_location_longtitude
- stop_location_latitude
- review_rating
- review_comment

+ set_transaction_paymentmethod (user_payment)
+ find_rider ()
+ alert_customer_topay(status)
+ create_transaction_record ()

Method Name: create_transaction_record	**Class Name:** Ride	**ID:** RIDE01
Contract ID: 002	**Programmer:** Alex	**Due Date:** 13 Nov 2021
Programming Language: JavaScript		
Triggers/Events: Customer confirms booking.		
Arguments Received: **Data Type:**	**Notes:**	
void		
Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument:Data Type**	**Return Data Type**
TransactionRecord.new ()	char(8) of ride_ID	TransactionRecord
self.set_transaction_paymentmethod ()	varchar(20) of user_payment	
Arguments Returned: **Data Type:**	**Notes:**	
void		
Algorithm Specification: create new TransactionRecord with ride_id set_transaction_paymentmethod (user_payment)		
Misc. Notes:	None	

Figure 10: Validating “create_transaction_record” method and “Ride” class

As we validated the class and method design, the method name, class name and argument received/parameter passing in method specification and class diagram matched up. For example, “create_transaction_record” Method - “Ride” Class and “book_a_ride” Method - “Booking” Class from *Figure 9: Validating “book_a_ride” method and “Booking” class* and *Figure 10: Validating “create_transaction_record” method and “Ride” class*.

User interface design principles

Layout

Each page is divided into three parts:

- Header Area shows the name of the current page that the user is using
- Body Area shows information and details of that page
- Footer Area is a navigator that can be pressed to go to different pages.

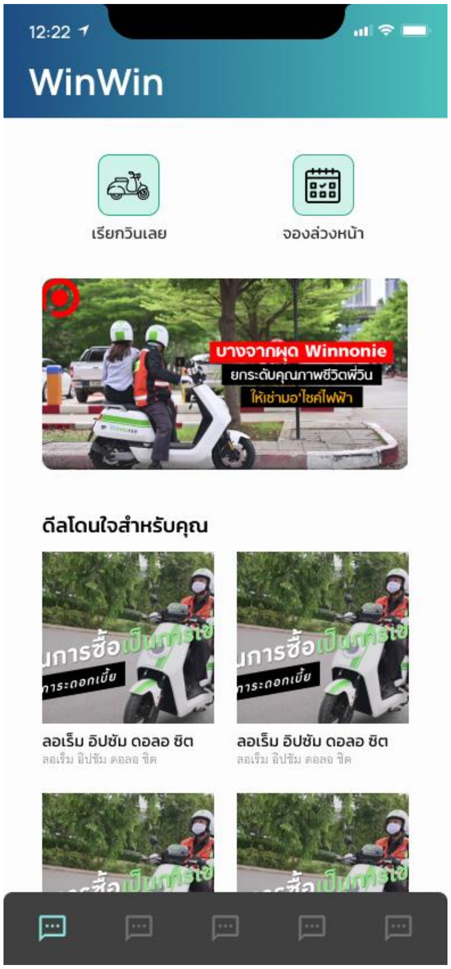


Figure 11: WinWin User Interface – Home page

Content Awareness

- Each page is labeled to tell the user which page is currently on, and each field is clearly labeled.
- Related sections are grouped together and labeled with bold titles for ease of use and to make it easier to separate topics.

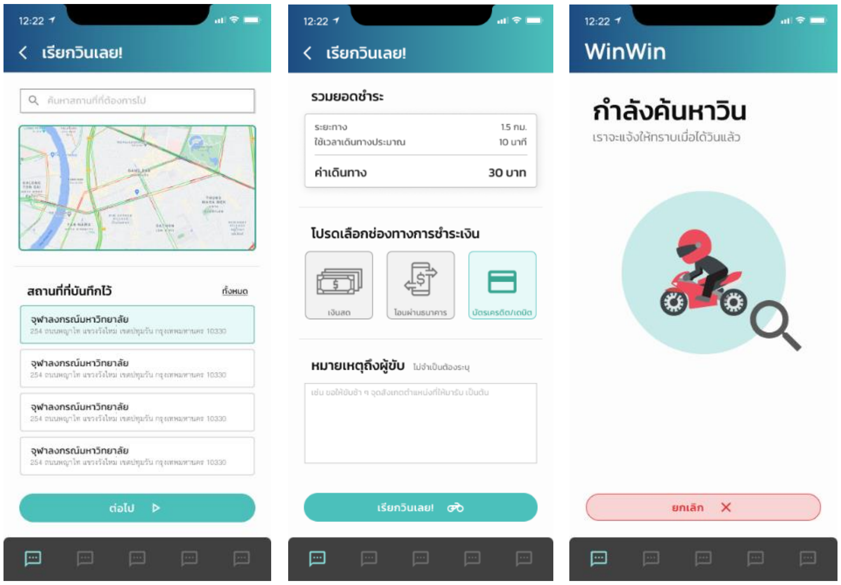

12:22
เรียกวินเลย!
ค้นหาสถานที่ที่ต้องการไป
สถานที่ที่บันทึกไว้
ทั้งหมด
จุฬาลงกรณ์มหาวิทยาลัย
254 ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
จุฬาลงกรณ์มหาวิทยาลัย
254 ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
จุฬาลงกรณ์มหาวิทยาลัย
254 ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
จุฬาลงกรณ์มหาวิทยาลัย
254 ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
ต่อไป
12:22
เรียกวินเลย!
รวมยอดชำระ
ระยะทาง 1.5 กม.
ใช้เวลาเดินทางประมาณ 10 นาที
ค่าเดินทาง 30 บาท
โปรดเลือกช่องทางการชำระเงิน
เงินสด
โอนผ่านธนาคาร
บัตรเครดิต/เดบิต
หมายเหตุถึงผู้ขับ ไม่จำเป็นต้องระบุ
เช่น ขอให้ขับช้า ๆ จุดสังเกตตำแหน่งที่ให้มารับ เป็นต้น
เรียกวินเลย!
12:22
WinWin
กำลังค้นหาวิน
เราจะแจ้งให้ทราบเมื่อได้วินแล้ว
ยกเลิก

Figure 12: WinWin User Interface – Showing page label

Aesthetics

- The design is based on minimalism, by not giving the user too much detail to focus on so that it is easier for the user to proceed.
- Most of the pages have low information density, for the ones in which haven't there's enough white space or divider so that the user can distinguish between objects.
- All texts are easy to read, recognize, and distinguish.

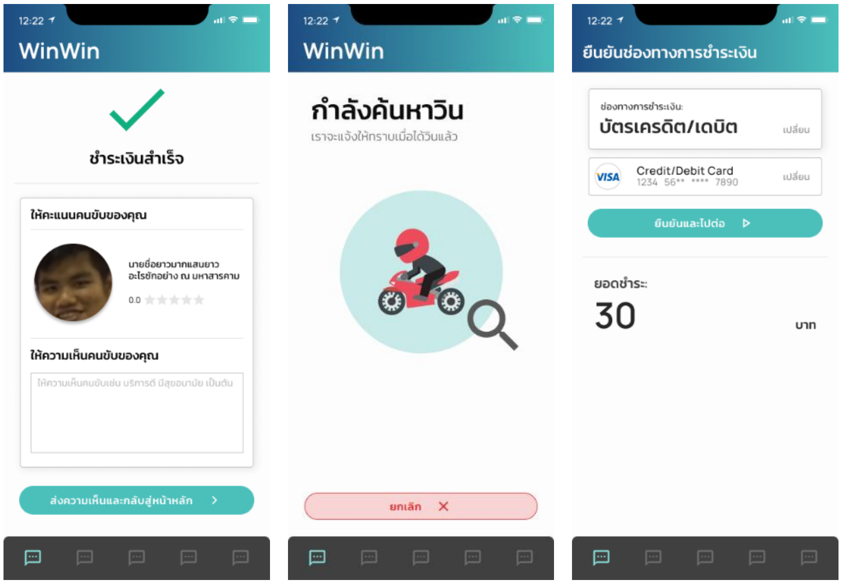


Figure 13: WinWin User Interface – Showing aesthetics

User Experience

- Users can easily learn how to operate the system. Because the systems will guide the user through the process step by step with a clear title of what step is being performed and a button that tells the users what to do in the next step if pressed
- The system is easy to use. Those who already understand the public transport booking system can use it easily and quickly.

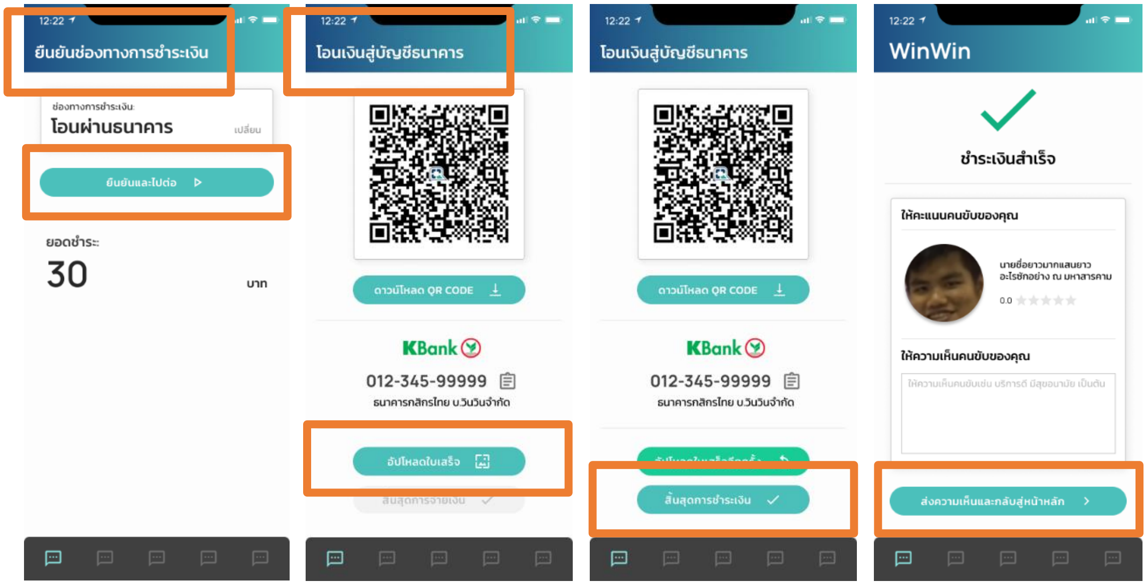


Figure 14: WinWin User Interface – User experience

Consistency

- All objects with the same purpose (e.g., Text box, Text field, Card) have the same style with only adjustment in size (some has a little embellishment to emphasize the detail)
- All buttons in the same manner (e.g., proceed, cancel, the second option) have the same style, so the user is able to predict the purpose of every button.
- The footer is the same on every page that needs one.
- All fonts have been set according to standard HTML font size, e.g., h1, h2, h3, and so on.
- So all texts in the UI have consistency in font and size.

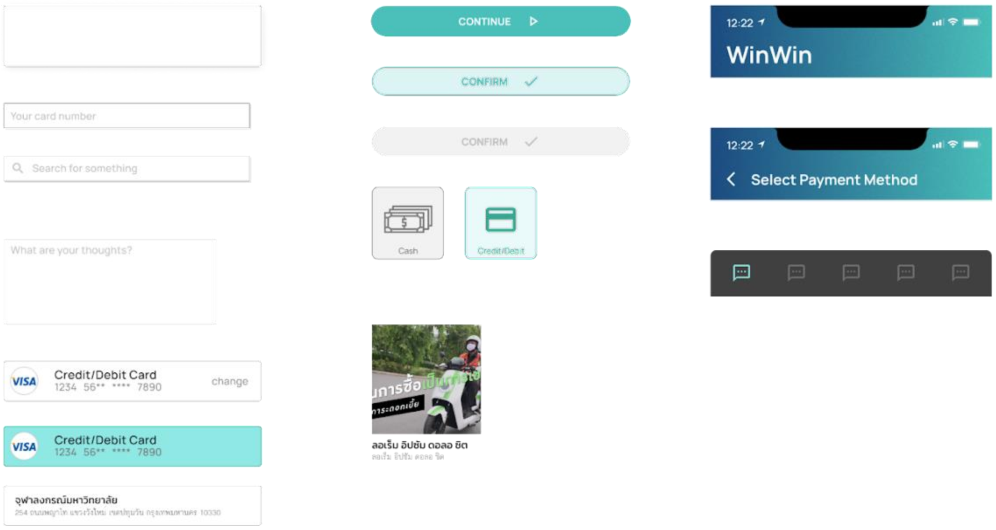


Figure 15: WinWin User Interface – Consistency

Minimal User Effort

- For users to pay the fee, in any method possible, they need to push on buttons at most three times (on the app alone, and assuming there's no change in payment method) to finish the transaction.
- For users to book a ride they need to push around 4-6 buttons to initiate a ride.
- As the objects are easy to distinguish, the user needs only a little effort to recognize what to do next and hence, save time and effort.

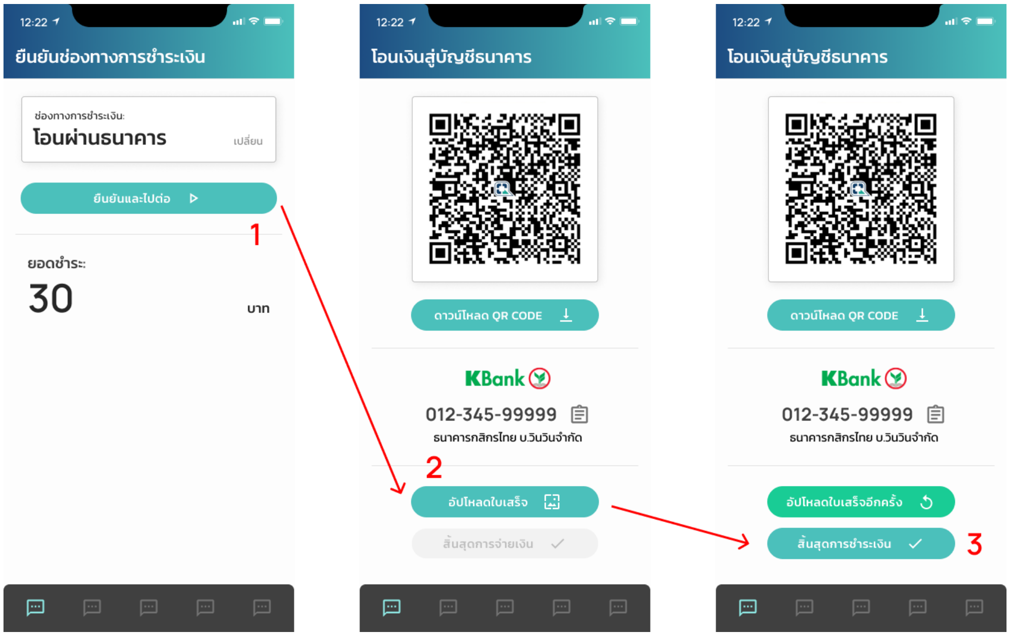


Figure 16: WinWin User Interface – Minimal User Effort

Detail Real Use Case description

Book a ride

Use Case Name: Book a ride	**ID:** 4	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Real	
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 clicks “Booking right-away” button or “Booking in advance” button 2. The customer sets their destination for the ride. Include flow: Select Destination 3. The customer specifies their preferences for the ride. Extension points: Select Rider Preferences 4. The system shows the price rate and the predicted amount of time of the requested ride. 5. The customer chooses their method of payment. Include flow: Select Payment Method 6. The system saves the record of the booking.		
Subflows: -		
Alternate/Exceptional Flow: 1-a1: If the customer does not want to book ride anymore at anytime, the customer can clicks return button to go back to home menu 2-e1: If the user books another ride at the same riding time as another ride request, the system notifies the user of the prevention of duplicated ride request.		

Make payment

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

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 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
3. The customer and the rider clicks “Return to main menu” button to return to main menu

Subflows:

S-1: Make cash payment

1. The system show “Confirm Payment” screen
2. The customer pays to the rider directly
3. The rider clicks on the “Confirm” button to verify their reception of the customer's cash payment
4. The system deducts the rider’s cash credit

S-2: Make bank transfer payment

1. The system show “Confirm Payment” screen
2. The customer clicks on “Pay” button
3. The system shows the bank account of WinWin system and QR code
4. The customer goes to mobile banking application and make a transfer
5. The customer goes back to WinWin application
6. The customer clicks on the “Upload Receipt” button to upload the receipt.
    1. The customer can re-upload the receipt by clicking “Re-upload receipt” button
7. The customer clicks on the “Submit” button.
8. The system checks the submitted receipt

S-3: Make card payment

1. The system show “Confirm Payment” screen with default credit/debit card shown
    1. If the customer clicks on change credit/debit card button, the system shows the added cards of the customer and “add more card” button
        1. If the customer clicks “add more card” button, the customer fills the card adding form and click “add card” button
    2. The customer select card to use
    3. The system show “Confirm Payment” screen with chosen credit/debit card
2. The customer clicks on pay button
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 click change button then the system will show the “Select Payment Method” part at the bottom of the screen (N-1)

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

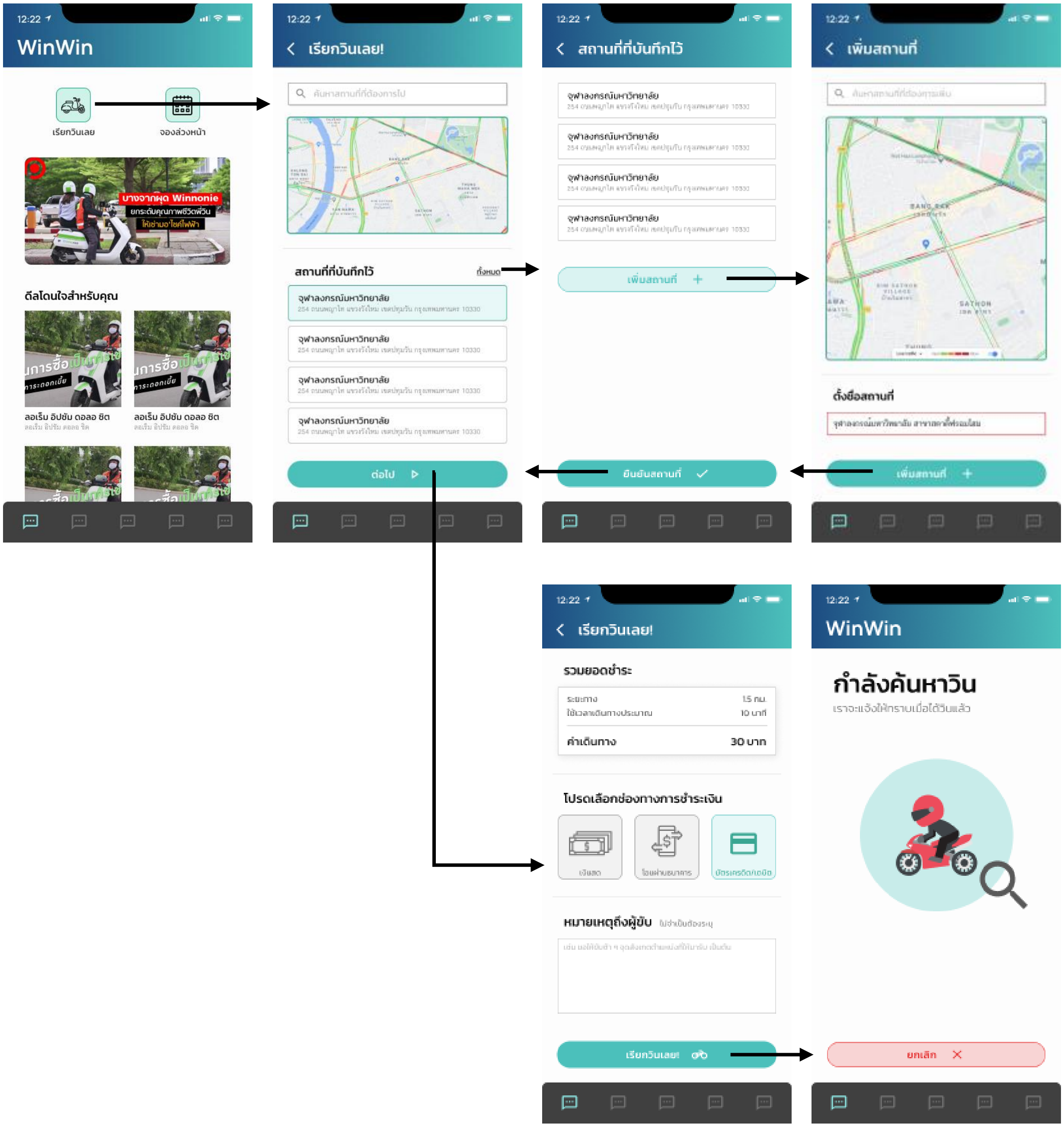
User interface design

Interactive user interface is available at

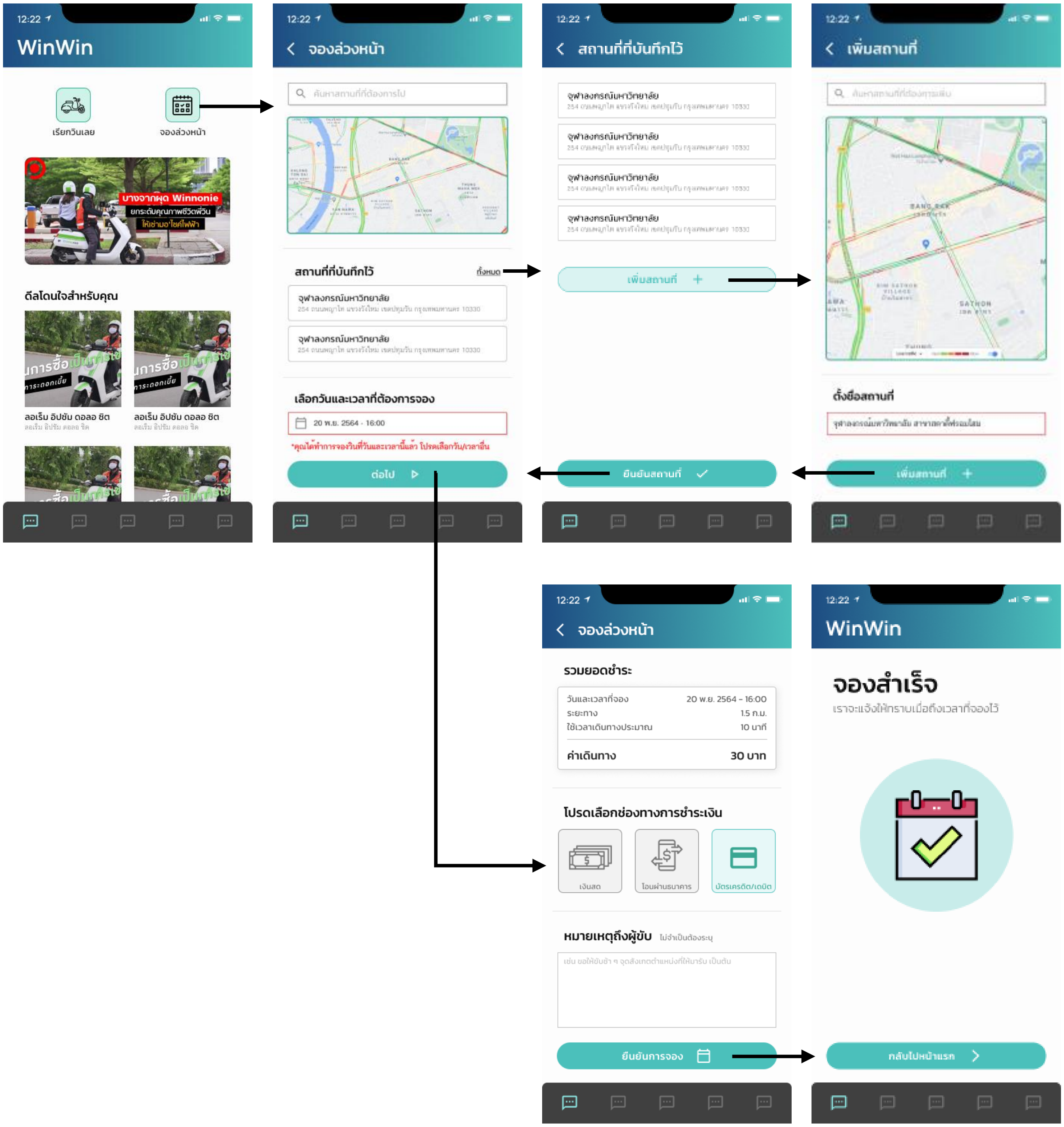
https://www.figma.com/file/t4T5hITYcZCSgz7ajLSWXO/Waterfall?node-id=119%3A12163

Book a ride

Book right away



Book in advance



*คุณได้ทำการจองวินที่วันและเวลานี้แล้ว โปรดเลือกวัน/เวลาอื่น

ตั้งชื่อสถานที่: จุฬาลงกรณ์มหาวิทยาลัย สาขา[illegible]

รวมยอดชำระ

วันและเวลาที่จอง	20 พ.ย. 2564 - 16:00
ระยะทาง	1.5 ก.ม.
ใช้เวลาเดินทางประมาณ	10 นาที
ค่าเดินทาง	30 บาท

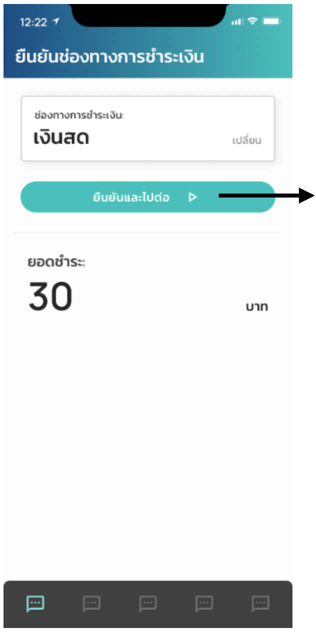
จองสำเร็จ

เราจะแจ้งให้ทราบเมื่อถึงเวลาที่จองไว้

Make payment

Payment method: Cash

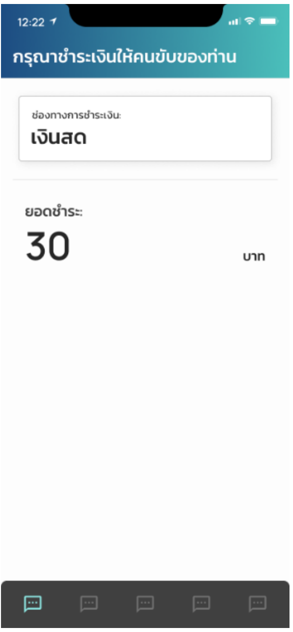
Customer



ยืนยันช่องทางการชำระเงิน

ช่องทางการชำระเงิน:
เงินสด
เปลี่ยน

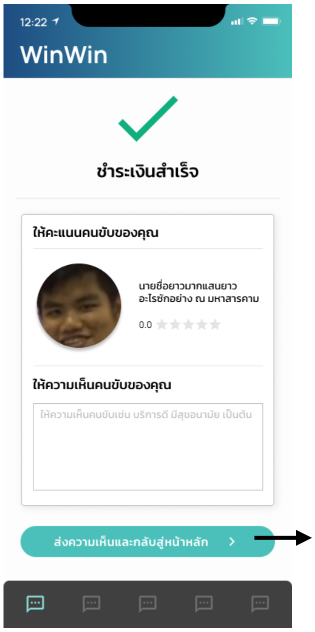
ยอดชำระ:
30
บาท



กรุณาชำระเงินให้คนขับของท่าน

ช่องทางการชำระเงิน:
เงินสด

ยอดชำระ:
30
บาท



WinWin

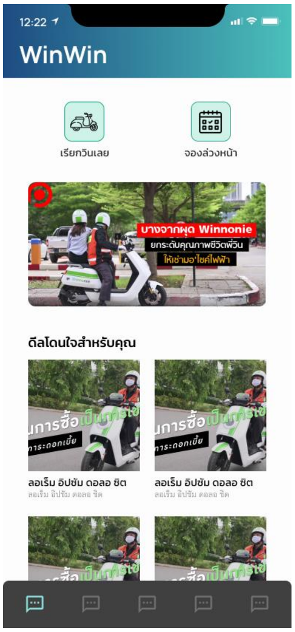
ชำระเงินสำเร็จ

ให้คะแนนคนขับของคุณ

นายชื่อยาวมากแสนยาว
อะไรซักอย่าง ณ มหาสารคาม
0.0 ★★★★★

ให้ความเห็นคนขับของคุณ

ให้ความเห็นคนขับเช่น บริการดี มีสุขอนามัย เป็นต้น



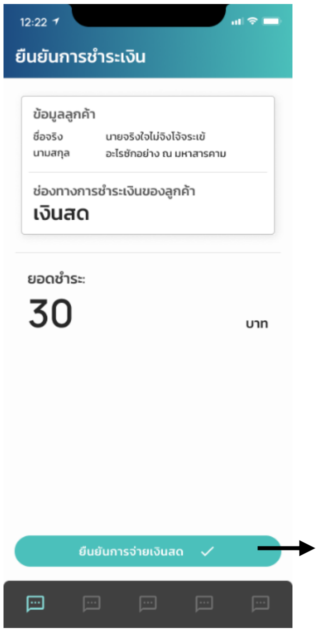
WinWin

ดีลโดนใจสำหรับคุณ

ลอเร็ม อิปซัม ดอลอ ซิต
ลอเร็ม อิปซัม ดอลอ ซิต

ลอเร็ม อิปซัม ดอลอ ซิต
ลอเร็ม อิปซัม ดอลอ ซิต

Rider



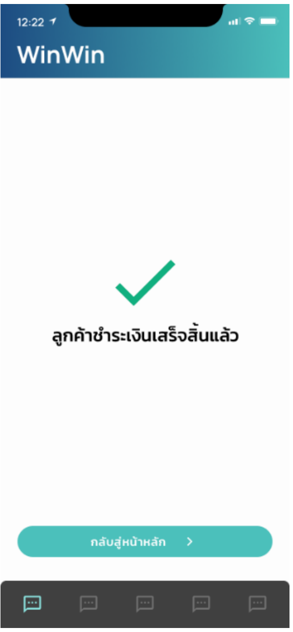
ยืนยันการชำระเงิน

ข้อมูลลูกค้า

ชื่อจริง	นายจริงใจไม่จิงโจ้จระเข้
นามสกุล	อะไรซักอย่าง ณ มหาสารคาม

ช่องทางการชำระเงินของลูกค้า
เงินสด

ยอดชำระ:
30
บาท

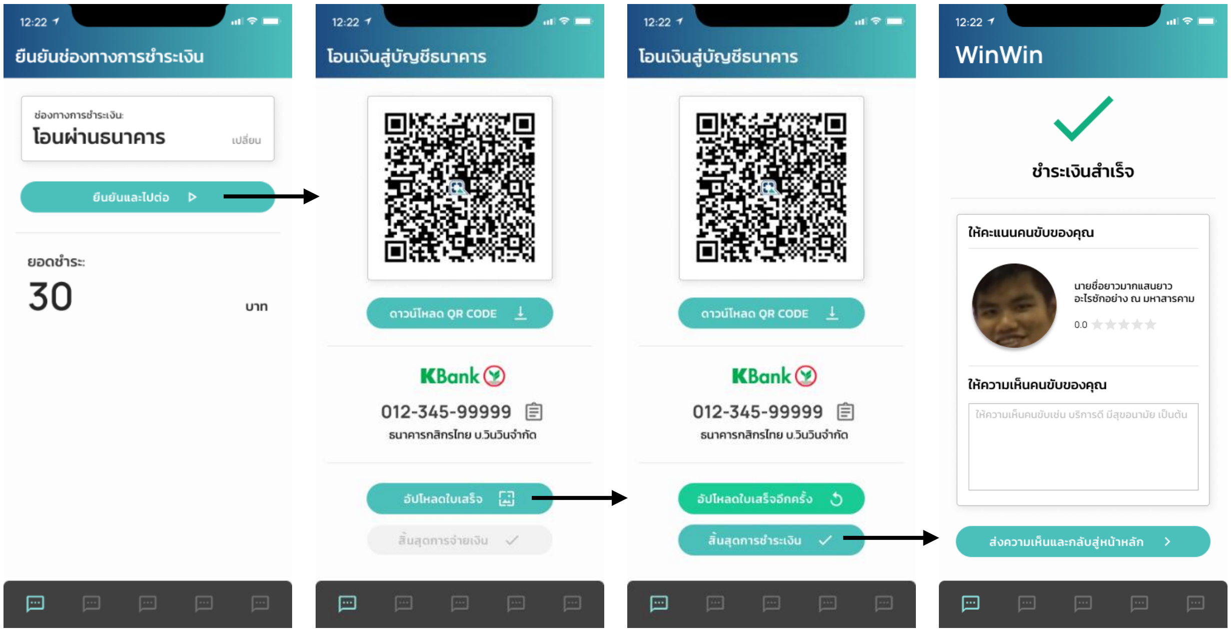


WinWin

ลูกค้าชำระเงินเสร็จสิ้นแล้ว

Payment method: Bank Transfer

Customer



12:22

ยืนยันช่องทางการชำระเงิน

ช่องทางการชำระเงิน:
โอนผ่านธนาคาร

ยอดชำระ:
30 บาท

12:22

โอนเงินสู่บัญชีธนาคาร

KBank
012-345-99999
ธนาคารกสิกรไทย บ.วินวินจำกัด

12:22

โอนเงินสู่บัญชีธนาคาร

KBank
012-345-99999
ธนาคารกสิกรไทย บ.วินวินจำกัด

12:22

WinWin

ชำระเงินสำเร็จ

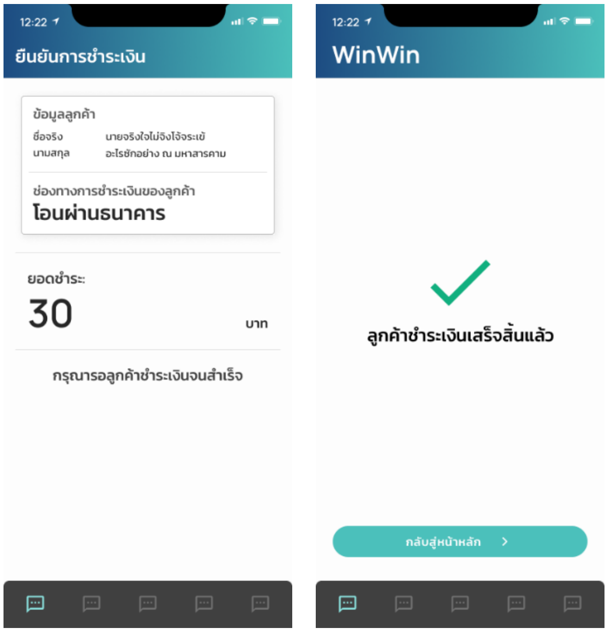
ให้คะแนนคนขับของคุณ

นายชื่อยาวมากแสนยาว
อะไรซักอย่าง ณ มหาสารคาม
0.0 ☆☆☆☆☆

ให้ความเห็นคนขับของคุณ

ให้ความเห็นคนขับเช่น บริการดี มีสุขอนามัย เป็นต้น

Rider



12:22

ยืนยันการชำระเงิน

ข้อมูลลูกค้า

ชื่อจริง	นายจริงใจไม่จิงโจ้จระเข้
นามสกุล	อะไรซักอย่าง ณ มหาสารคาม

ช่องทางการชำระเงินของลูกค้า
โอนผ่านธนาคาร

ยอดชำระ:
30 บาท

กรุณารอลูกค้าชำระเงินจนสำเร็จ

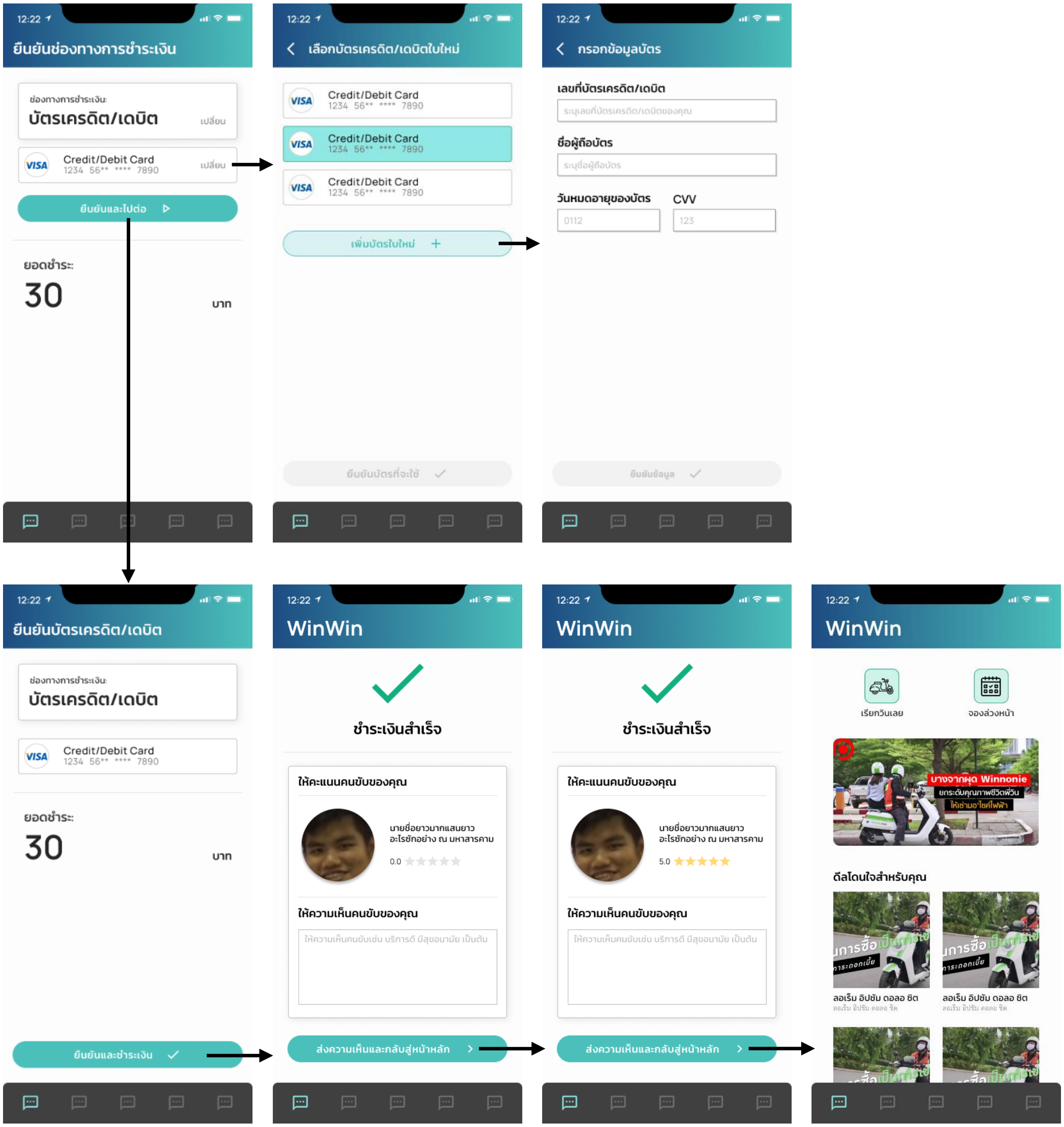
12:22

WinWin

ลูกค้าชำระเงินเสร็จสิ้นแล้ว

Payment method: Credit/Debit Card

Customer



12:22

ยืนยันช่องทางการชำระเงิน

ช่องทางการชำระเงิน:
บัตรเครดิต/เดบิต
เปลี่ยน

Credit/Debit Card
1234 56** **** 7890
เปลี่ยน

ยืนยันและไปต่อ ▷

ยอดชำระ:
30 บาท

12:22

เลือกบัตรเครดิต/เดบิตใบใหม่

Credit/Debit Card
1234 56** **** 7890

Credit/Debit Card
1234 56** **** 7890

Credit/Debit Card
1234 56** **** 7890

เพิ่มบัตรใบใหม่ +

ยืนยันบัตรที่จะใช้ ✓

12:22

กรอกข้อมูลบัตร

เลขที่บัตรเครดิต/เดบิต
ระบุเลขที่บัตรเครดิต/เดบิตของคุณ

ชื่อผู้ถือบัตร
ระบุชื่อผู้ถือบัตร

วันหมดอายุของบัตร: 0112
CVV: 123

ยืนยันข้อมูล ✓

12:22

ยืนยันบัตรเครดิต/เดบิต

ช่องทางการชำระเงิน:
บัตรเครดิต/เดบิต

Credit/Debit Card
1234 56** **** 7890

ยอดชำระ:
30 บาท

ยืนยันและชำระเงิน ✓

12:22

WinWin

✓
ชำระเงินสำเร็จ

ให้คะแนนคนขับของคุณ
นายชื่อยาวมากแสนยาว
อะไรซักอย่าง ณ มหาสารคาม
0.0 ☆☆☆☆☆

ให้ความเห็นคนขับของคุณ
ให้ความเห็นคนขับเช่น บริการดี มีสุขอนามัย เป็นต้น

ส่งความเห็นและกลับสู่หน้าหลัก ›

12:22

WinWin

✓
ชำระเงินสำเร็จ

ให้คะแนนคนขับของคุณ
นายชื่อยาวมากแสนยาว
อะไรซักอย่าง ณ มหาสารคาม
5.0 ★★★★★

ให้ความเห็นคนขับของคุณ
ให้ความเห็นคนขับเช่น บริการดี มีสุขอนามัย เป็นต้น

ส่งความเห็นและกลับสู่หน้าหลัก ›

12:22

WinWin

เรียกวินเลย
จองล่วงหน้า

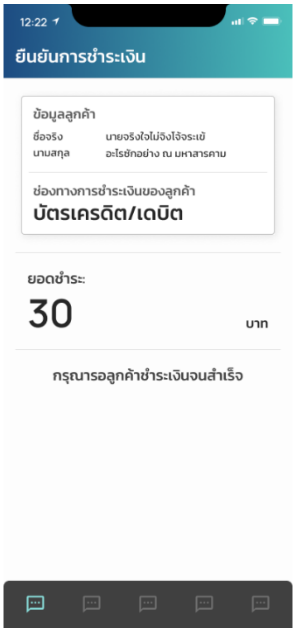
บางจากผุด Winnonie
ยกระดับคุณภาพชีวิตพี่วิน
ให้เช่ามอ'ไซค์ไฟฟ้า

ดีลโดนใจสำหรับคุณ

ลอเร็ม อิปซัม ดอลอ ซิต
ลอเร็ม อิปซัม ดอลอ ซิต

ลอเร็ม อิปซัม ดอลอ ซิต
ลอเร็ม อิปซัม ดอลอ ซิต

Rider


12:22

ยืนยันการชำระเงิน

ข้อมูลลูกค้า

ชื่อจริง	นายจริงใจไม่จิงโจ้จระเข้
นามสกุล	อะไรซักอย่าง ณ มหาสารคาม

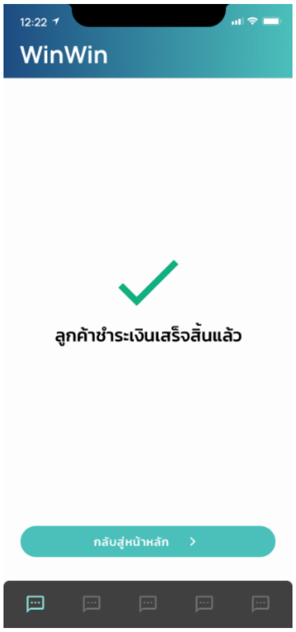
ช่องทางการชำระเงินของลูกค้า

บัตรเครดิต/เดบิต

ยอดชำระ:

30 บาท

กรุณารอลูกค้าชำระเงินจนสำเร็จ

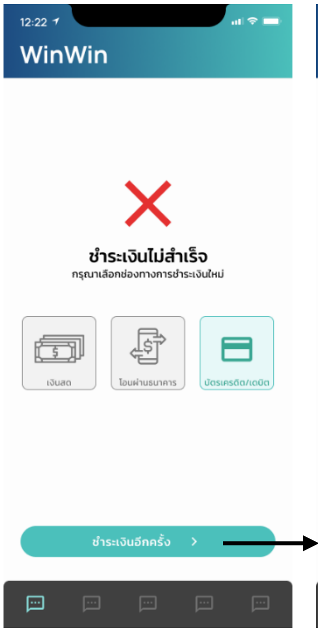

12:22

WinWin

✓

ลูกค้าชำระเงินเสร็จสิ้นแล้ว

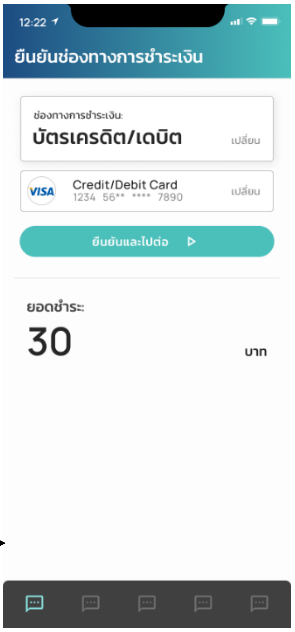
Payment Fail


12:22

WinWin

ชำระเงินไม่สำเร็จ

กรุณาเลือกช่องทางการชำระเงินใหม่


12:22

ยืนยันช่องทางการชำระเงิน

ช่องทางการชำระเงิน:

บัตรเครดิต/เดบิต เปลี่ยน

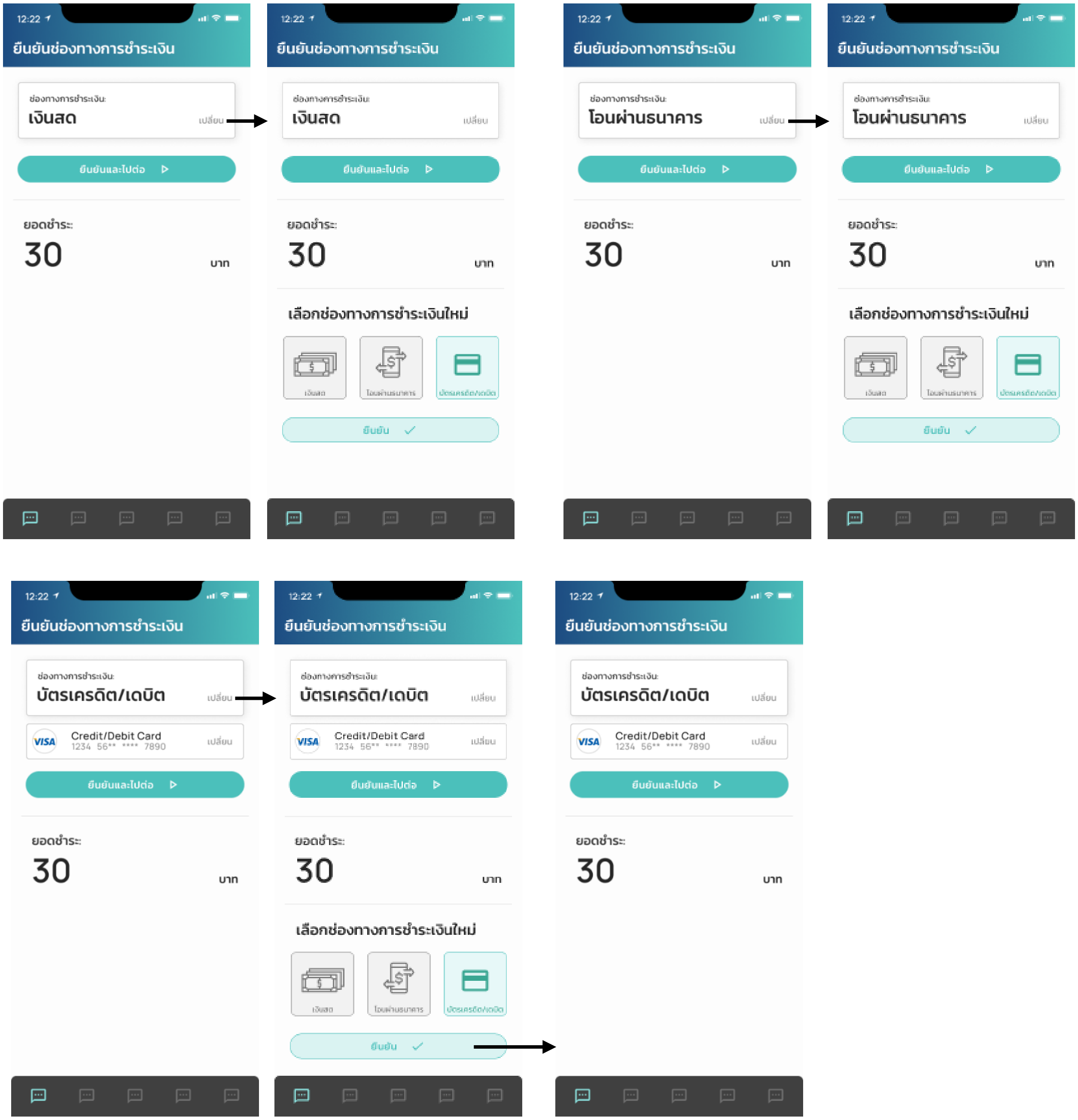
VISA Credit/Debit Card
1234 56** **** 7890 เปลี่ยน

ยอดชำระ:

30 บาท

Change payment method

Customer



Non-functional requirements and their effect on physical architecture design

Table 2: Non-functional requirements and their effect

Type of Nonfunctional Requirement	Requirement	Effect for designing architectural model
Operational requirements	The system shall operate in a mobile device environment.	Client server has to support mobile devices.
	The system shall automatically back up its database daily.	Have at least two databases and snapshot to backup database daily
Performance requirements	The system shall respond in less than 5 seconds for every interaction between system and user.	Use thin client-server architecture with multiple redundant servers that can scalable easily
Security requirements	The system shall authenticate users and riders using userID-password.	Use thin client-server architecture that internet-based authentication tool is more advanced
	The system shall be able to keep users' transactions confidential.	Use an external banking system that is secure and reliable.
Usability requirements	The system should be easy to use for both new and experienced users.	-

An overview of the system infrastructure design

Network diagram

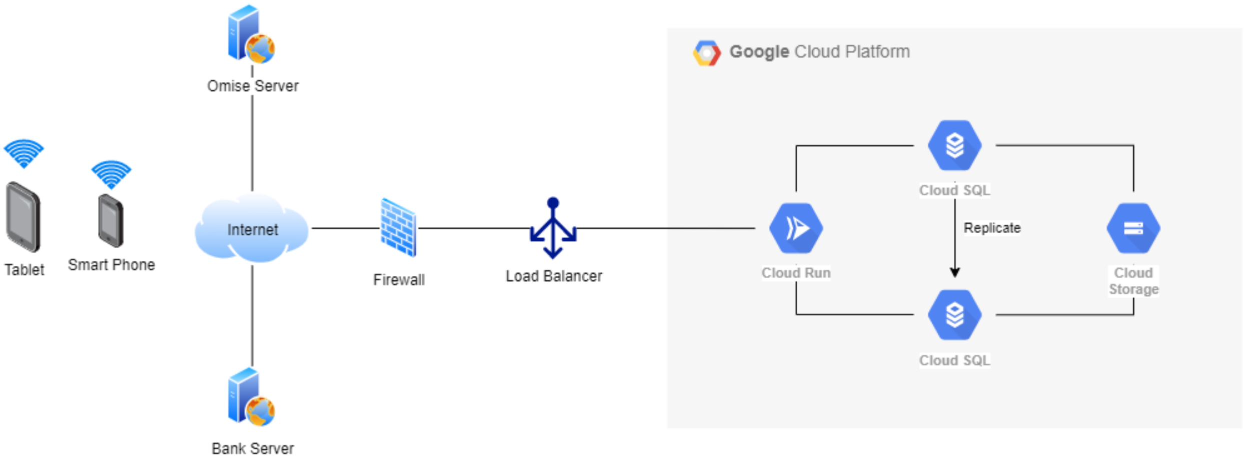


Figure 17: Network diagram

Deployment diagram

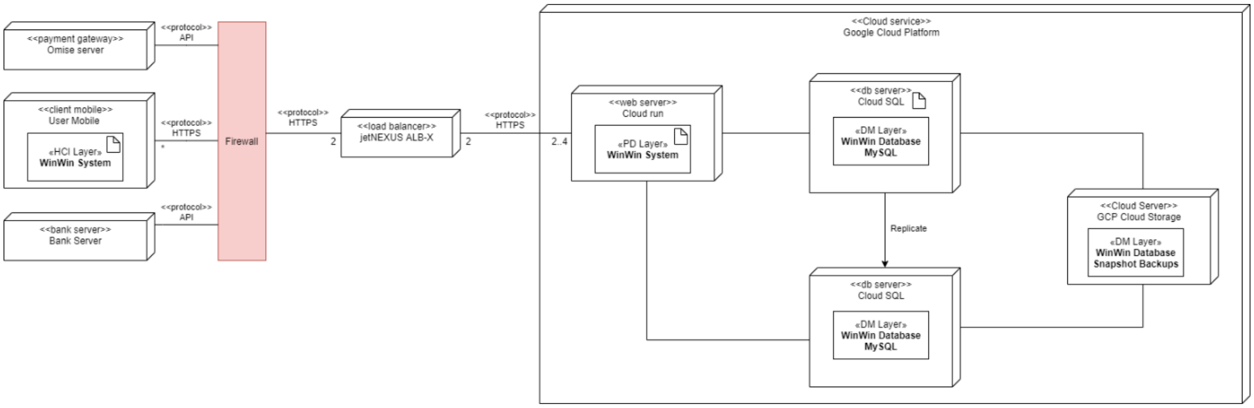


Figure 18: Deployment diagram

Hardware and software specification

From the deployment diagram, we used three-tier architecture. It can be divided into 3 parts which are client, application server and database server. The lists provided in the table are minimum requirements of hardware and software specifications that we considered from several factors such as costs of maintenance and performance of hardware. Therefore, the system will not crash and perform good performance with this hardware and software specifications.

Table 3: Hardware and software specification

Specification	Standard Client	Standard Application Server	Standard Database Server
Operating System	• Android 5.0 • iOS 11.0	• Linux	• Linux
Special Software	• WinWin Application	• Apache	• MySQL
Hardware	• 4GB Memory • 128 GB disk drive	• 16 GB Memory • 1 TB disk drives • Dual-core 3.0 GHz processor	• 32 GB Memory • 1-4 TB disk drives • Dual-core 3.0 GHz processor
Network	• Ethernet/WiFi/ 3G	• Ethernet / WIFI	• Ethernet

Verifying and validating the physical architecture layer

Verifying and validating deployment diagram and network diagram

All hardware and communication paths needed to satisfy functional and non-functional requirements are included in both deployment diagram and network diagram i.e. every element within one diagram can be mapped into an element within another diagram.

In the deployment diagram, there are 3 main servers (Client, Bank and Omise), consequently there are 3 main servers in the network diagram (tablet and smartphone depicts customer's mobile).

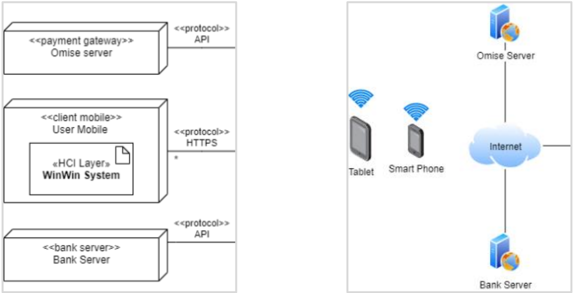


Figure 19: Validating deployment diagram and network diagram – Clients and External Server

The requests are then passed through a firewall before entering load balancer and, consequently, the WinWin system's server which is hosted entirely on the Google Cloud Platform. This is shown in both deployment diagram and network diagram.

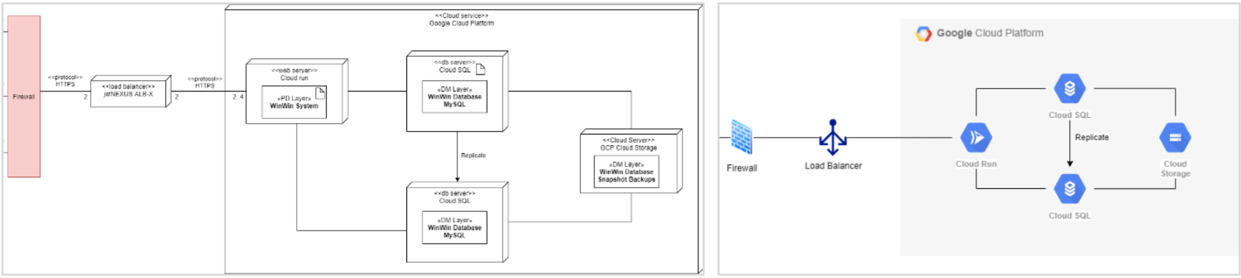


Figure 20: Validating deployment diagram and network diagram – Load balancer and Servers

Focusing at the WinWin system server in each diagram, it is hosted on Google Cloud Platform as indicated by the encapsulating box. Cloud Run is used as a web server, 2 Cloud SQL instances are used that replicates another as well as Cloud Storage that backs up the Cloud SQL.

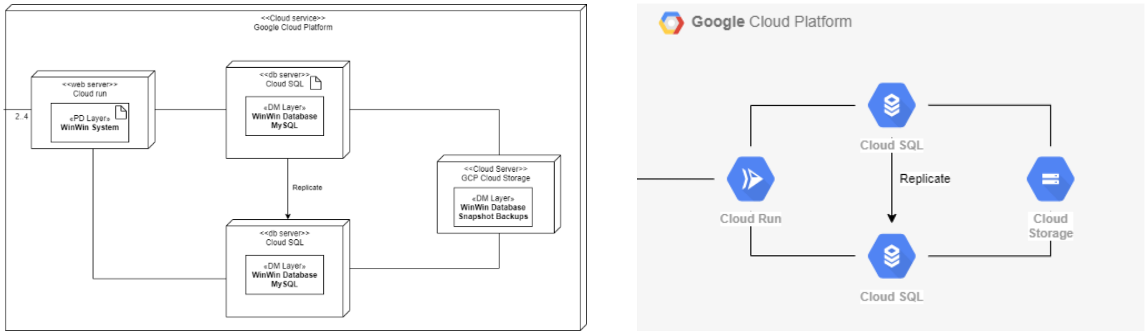


Figure 21: Validating deployment diagram and network diagram – Cloud Servers and Databases

Verifying and validating hardware and software specification

From non-functional requirements, the system shall operate in a mobile device environment. Therefore, hardware and software specification must support all mobile OS (IOS, Android).

From the network diagram, we have SQL databases. Therefore, software specification of database servers must support them, shown in network diagram.

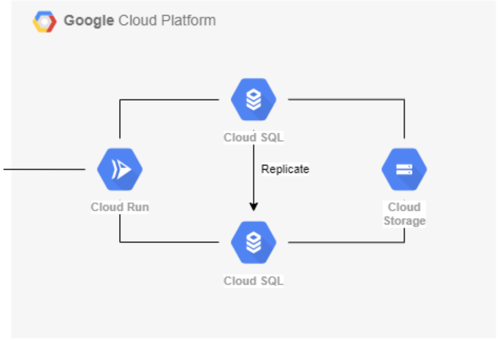


Figure 22: Validating hardware and software specification – Database support

From network diagram and deployment diagram, we used a three-tier architecture. There are 3 parts which are client, web server and database server. Therefore, hardware and software specification must specify specifications for client, web server and database server.

The impact of your analysis and design solution

Impact on the global context

- Due to the spread of COVID-19, customers should not wait in a group for motorcycle taxis at the station. As a result, if customers use WinWin's application, the spread may be reduced because they no longer need to wait at the station.
- The system supports the use of smartphones and technology in order to drive future civilization.

Impact on the economic context

- Customers would have more options for transportation, and motorcycle taxis would have more jobs, resulting in a shift in the economy.

Impact on the societal context

- Using the service from local motorcycle taxis supports increasing work employment and cash flow in the community.
- Reduce problems from conflict between customer and rider about price rate that is uncertain.
- People can transport easily.

Impact on the environmental context

- The system supports the cashless society, so it will reduce the usage of paper.

Conclusion of the design process

Prior to the design process, behavioral model, structural model, and functional model have been constructed and validated to ensure the consistency of the software product.

Before moving on to the design process, a package diagram was constructed on top of the class diagram, and component diagrams were also constructed to illustrate the communications between components. After the validation of diagrams mentioned earlier, the preparation before the design process was done.

Initially, we constructed method specifications for methods we need to implement (in this case, book_a_ride and create_transaction_record), then verified classes and method design to ensure the consistency between classes and methods.

Then we had created detail-real use case descriptions for *book a ride* and *make payment* use cases and proceeded on designing the user interface for those two use cases on Figma (which is available on the User interface design section). Consequently, we did a user interface design principle analysis and altered the design to match all the principles.

Lastly, we designed the physical layer architecture layer.

We started by stating all non-functional requirements and their effects on designing architectural models. Then we constructed a network diagram and a deployment diagram to show the relationship between hardware and software and how the system's software components would fit together.

After knowing the scope of hardware and software used in the project, we created a hardware and software specifications to specify their minimum requirements to get the tasks done and ended the design process with verifying and validating the physical architecture layer to ensure the integrity of all the components and non-functional requirements in the system.

System Installation

and Operation

Migration plan

Conversion Plan

Hardware installation

As the system that we developed uses servers and databases based on cloud service, so no hardware is needed.

Software installation

For the software installation of the system. It is necessary to install software such as the IDE as well as the environment and framework that are involved.

Data conversion

As our system is a whole new developed system to replace the old one. So we don't need to have the data conversion part.

Conversion dimension

Conversion Strategy: Parallel, Phased, Whole system

Conversion style

We chose parallel because the development of our new system took time and to keep our old customers and to make sure that in the early days when our new system is not stable. We can continue to serve our service to our customers. So, we choose to keep the old system while developing a new system that will replace the old one in the future.

Conversion location

We choose the phased type as the conversion location. The first group of areas that we will serve is the area in Bangkok. Because in Bangkok, people choose to use motorcycle taxis the second most, after MRT. That's why we choose to use our system in these locations first. If there is a good response, we will expand the service area in the future.

Conversion modules

The reason why we choose the whole system is because the new system we developed has almost no parts that can be used with the old system. That's why we choose the whole system to make the system we developed as efficient as possible.

Change Management Plan

Revising Management Policies

Management policies about human resources need to be revised in order to support the new system. Since our company is a start-up company, at present, there are only a few employees in the company. Therefore, our working culture should be flexible which could be done by using agile development to help our employees talk about the process and when there is a problem, they can help each other to solve it in time.

Assessing Costs & Benefits

As our company is new to the market so our budget is low. We, therefore, need to be careful about the costs. We will refer to the budget for the development of our system in the feasibility analysis and try to use the existing budget in accordance with the plan to achieve our system development goals.

Motivate adoption

In order to encourage the users and staff to accept the new system, there are two adoption strategies, Informational strategy and Political strategy. In informational strategy, we need to clarify in detail how our system can support local motorcycle taxis. In political strategy, we may use organization power when the target adopters set their own personal cost more than the benefits.

Conduct training

Since our system is new, we should not assume that our users will be able to use our system without training. For the admins, we will have the training session to make them get familiar with the system. For users, some users may be familiar with the system as it is similar to other platforms, creating an infographic on how the system works should be sufficient.

Post-implementation activities

System Support

- Online support
  - The system has online support in several channels such as user guides that are tutorial for new users and FAQ that users can read in application.
- Call center
  - If the problem of the user can't be solved by reading FAQ or that problems don't be in FAQ. Users can contact the call center and staff will answer and take care of users to solve their problem.

System Maintenance

We have system maintenance when we get change requests such as problem reports from users, user feedbacks, and senior management. Support teams will create meetings to create plans and process to maintenance system.

Project Assessment

- Project team review
  - Project team review will be conducted immediately after the system is installed. Members will list what worked and mistakes from this project. Project manager will summarize all lists to be lessons learned documents.
- System review
  - System review will be conducted every month to compare estimated system and to-be system and analyze what we should do in future.

Contributions

Table 4: Contributions

Name	Contributed part	Level of Achievement
6230123921 Thitaree Setwipattanachai	Objective of the project, Objective of the design document, Design criteria and design activities, Introduction of an overview of system design modeling, System design constraint, Verifying and class and method design	5
6230252121 Tarm Kalavantavanich	Class diagram, CRUDE Matrix, Verifying and validating the physical architecture layer	5
6231301421 Kanokpich Chaiyawan	To-be system overview, CRUDE Matrix, Method specification, Migration plan	5
6231304321 Kittipong Deevee	User interface design principles, Detail Real Use Case description, User interface design, Conclusion of the design process	5
6231307221 Jirawat Kusalangkurwat	Non-functional requirements and physical architecture, Hardware and software specification, Verifying and validating the physical architecture layer, The impact of your analysis and design solution, Post-implementation activities	5
6231333521 Nopdanai Sayamnet	Design criteria and design activities, Introduction of an overview of system design modeling, System design constraint, Method specification, Verifying and class and method design	5
6231353021 Raviporn Akekunanon	Document correction and organization, Reference document, Component diagram, An overview of the system infrastructure design	5
6231372021 Atiwat Deepo	Detail Essential Use Case diagram, Class diagram, User interface design principles, Detail Real Use Case description, User interface design, An overview of the system infrastructure design	5