# Meekor


62011286 Unn Jertjamjarat

62011258 Tanachat Mongkolporn

62011193 Patthiya Bundhamcharoen

62011295 Worada Rangsriseaneepitak

Bachelor of Engineering in Software Engineering

Department of Computer Engineering,

School of Engineering

King Mongkut's Institute of Technology Ladkrabang

Academic Year 2022

ISBN XXXXXXXX

**Thesis – Academic Year 2022**

Bachelor of Engineering in Software Engineering
Department of Computer Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang

**Title**: Meekor


**Authors**

| | | |
|---|---|---|
| 1. Unn | Jertjamjarat | Student ID: 62011286 |
| 2. Tanachat | Mongkolporn | Student ID: 62011258 |
| 3. Patthiya | Bundhamcharoen | Student ID: 62011193 |
| 4. Worada | Rangsriseaneepitak | Student ID: 62011295 |


Approved for submission

Isara Anantavrasilp
.....

(Dr. Isara Anantavrasilp)
Advisor

Date 27 / 5 / 2023

# Acknowledgement

We would like to express our deepest gratitude and appreciation to Dr. Isara Anantavrasilp for his invaluable guidance and unwavering support throughout our project. As our project advisor, Dr. Isara has consistently provided us with his extensive knowledge and expertise in the field of Software Engineering, enabling us to navigate complex challenges and achieve remarkable outcomes.

Dr. Isara's dedication and commitment to our project have been truly remarkable. His insightful feedback, constructive criticism, and encouragement have significantly contributed to the success of our endeavor. He has been instrumental in shaping our project's direction, offering valuable suggestions, and pushing us to explore innovative solutions.

We are also grateful to the other professors in the Software Engineering course for their contributions and guidance. Their expertise, teaching, and mentorship have been instrumental in expanding our understanding of software development principles and methodologies. Their commitment to fostering our growth and nurturing our skills has played a vital role in our academic journey.

We would also like to express our appreciation to the entire faculty and staff of the Software Engineering department for providing us with a conducive learning environment and access to the necessary resources. Their dedication to excellence in education has been evident throughout our course, and we are grateful for their ongoing support.

# Abstract

The aim of this software engineering project is to create a Line chatbot called “Meekor” which provides a service to collect the debt from a group of two or more people. Meekor also provides users payment channels and payment verification for users convenience.

Meekor contains four main functions which are create bill, pay bill, verify payment, and remind unpaid bill. Firstly, to use Meekor users must invite Meekor chatbot into the group and type the provided commands to invoke Meekor. Loaners can type “create bill” command to create a bill or monthly bill and include debtors in the created bill. Next, the debtors can pay their debt by tapping the “pay bill” in the button of flex message and attach a payment slip. Meekor then verifies the payment slip and reminds other debtors who haven’t paid the bill in the group. Lastly, the loaner also can manually remind the unpaid bill in the group by typing “remind bill”.

Meekor is mainly developed using Javascript web framework React JS and Express JS. On the frontend component, the LIFF framework is developed using React JS which is JavaScript framework, sending data from Line chat room using provided Line API. The backend component is developed using Express JS, the server side consists of PostgreSQL as a database.

Meekor is a line chatbot which provides a service to collect the debt from a group of two or more people. Meekor also provides users payment channels and payment verification. The Line chatbot can help users who are loaner to collect a bill easily, accurately, and conveniently, while the debtors can pay the bill timely and correctly.

# Table of Contents

# List of Figures

# List Of Tables

# Chapter 1
# Introduction

## 1.1 Motivation

When people go out with each other, either with friends, relatives or colleagues, sometimes they need to share costs of services or products. For example, meals, transportation or groceries. Splitting the bills and keeping track of payments among the members of the groups could be a tedious task, especially when there are many of them, e.g., during a trip. One has to keep track of not only the amount of payment, but also who paid for the bill and when. Furthermore, people may forget to pay back their share of the bill and the debtor might be reluctant to ask for the payment as it may be considered impolite.

Therefore, we aim to create a Line chatbot, Meekor, that acts as a mediator to manage the bill sharing and payments. It is intended to keep track of bills and payment and to prevent uncomfortable situations among the users.

## 1.2 Objectives

1. To create a Line chatbot with a mobile web application that helps users manage bill sharing.
2. To create a back-end system that keeps track of payment records from the chatbot and verify users transactions with bank accounts and Promptpay.

## 1.3 Project Explanation

Meekor is a line chatbot which provides a service to collect the debt from a group of two or more people. Meekor also provides users payment channels and payment verification for users convenience.

The main features of Meekor are creating bills, verifying payment and reminding all debtors of the bill to pay. Firstly, the user has to add Meekor into the Line group, and creating bills in the Line group to let Meekor tracking all members who are included in the bill to pay the bill and verify their payment. Creating a bill requires the user to enter the total of the bill and select members in the group to be a debtor, then input bank account number or promptpay number to be payment channels for debtors. Next, Meekor will repeatedly remind all debtors in the group to pay the bill until the bill is fully paid. To confirm payment debtor has to upload e-slip to Meekor to let Meekor verify the payment.

# Chapter 2

# Related Works

This chapter provides brief details and compares similar applications in Thailand and other countries.

## 2.1 Similar Applications in Thailand

### 2.1.1 Go Dutch Calc Free

The application was developed by a Thai programmer, Mr. Pitsanu Potajan. It aims for dividing the cost of food and service. The application is offline and has the basic necessary features. There are interesting features such as calculating service charges and taxes and creating profiles for debtors. The user can edit the information after creating the bill[1].


*Figure 1. Go Dutch Calc Free Sample Interface*

The downside of this application is that it is an offline application and does not have any feature related to online payment. The user has to check payment by themself and the debtor cannot access or read the information. Unlike meeker which has more complex features such as Not equal splitting and supporting full online payment, the application is more minimal and small in size.

### 2.1.2 Kunthong

KhunThong is a Line Bot, created by KBTG, that acts like a treasurer for your group. For any shared expenses you may have (meal time, holiday trip, service fee, etc.), KhunThong lets you split the bills without a splitting headache [2].

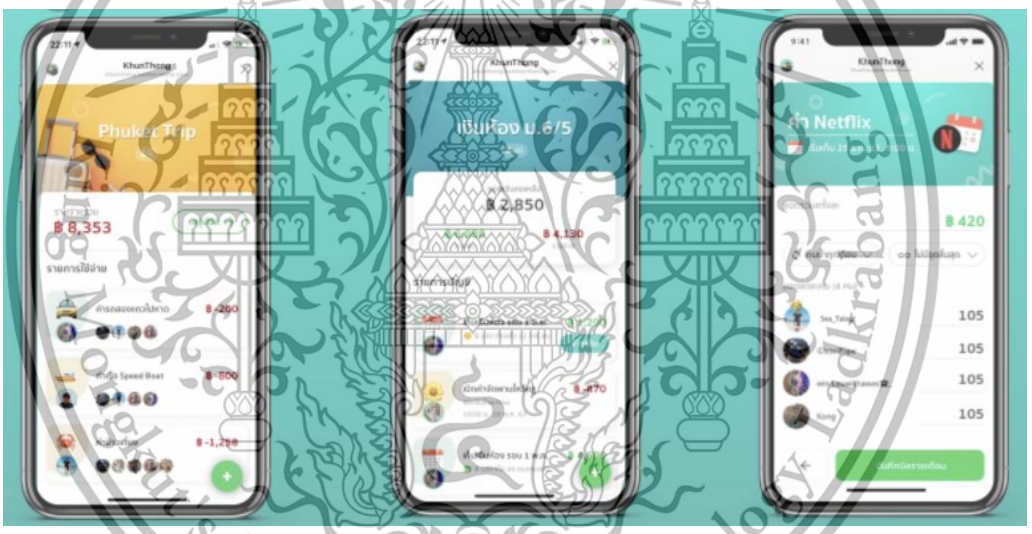



*Figure 2. Kunthong Sample Interface*

Kunthong has all the necessary features and full online payment support which is supported only with Kbank. The software is implemented with the Line application, the most popular chat application in Thailand, so it is easy to access for all ages. Compared to Meekor, Kunthong has many similar features but lacks in calculating service charges and taxes and is unorganized without each user's personal summary of all bills.

### 2.1.3 Jabont

Jabont is a website for splitting bills with basic splitting bill features and can generate a QR code for prompt pay payment. The bill can be accessed by HTTPS link so all users can read and edit information [3].

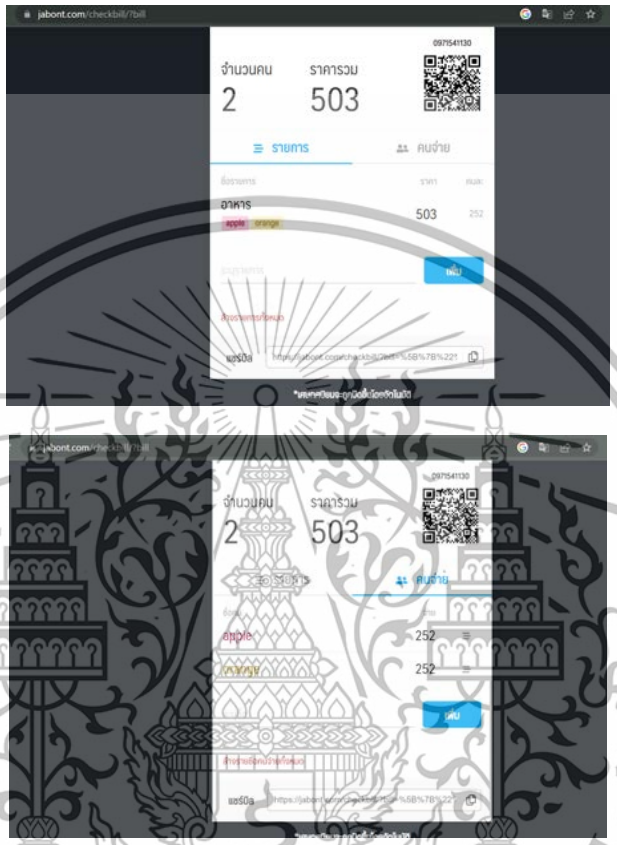



*Figure 3. Jabont Sample Interface*

The website does not have many features such as chatting, Calculating service charges and taxes, Not equal splitting bills, Uploading a bill slip image, and Monthly bill option. It is questionable if it is reliable since the bill can be edited by everyone with the link. Additionally, the website only shows online payment information but does not fully support it. Nevertheless, the website is suitable for rapid cases since the user does not have to download or install it to use.

**Features Comparison of Similar Application in Thailand**

| Feature/ Platform | Go Dutch Calc Free | Kunthong | Jabont | Meekor |
|---|---|---|---|---|
| Type | Application | Application | Website | Application |
| Chatting | | Yes | | Yes |
| Show online payment info | | Yes | Yes | Yes |
| Bill splitting | Yes | Yes | Yes | Yes |
| Monthly bill | | Yes | | Yes |
| Calculating service charges and taxes | Yes | | | Yes |
| Uploading a bill slip image | Yes | | | Yes |
| Scanning bill slip to text | | Yes | | |
| Not equal splitting bill choice | | Yes | Yes | Yes |
| Bill summary | Yes | Yes | Yes | Yes |
| Every user can read data | | Yes | Yes | Yes |
| Each user’s | | | | Yes |

| | | | | |
|---|---|---|---|---|
| personal summary of all bills | | | | |
| Automatic reminding | | Yes | | Yes |
| Adding new users after splitting | Yes | | Yes | |
| Auto inform for successful payment | | Yes | | Yes |

*Table 1. Features Comparison of Similar Application in Thailand*

## 2.1 Similar Applications in Other Countries

### 2.1.1 Splitwise

Splitwise is a free app that allows consumers to split expenses with friends. If a group needs to share the cost of a particular bill, Splitwise ensures that anyone who pays is reimbursed the correct amount and with a minimal number of transactions.

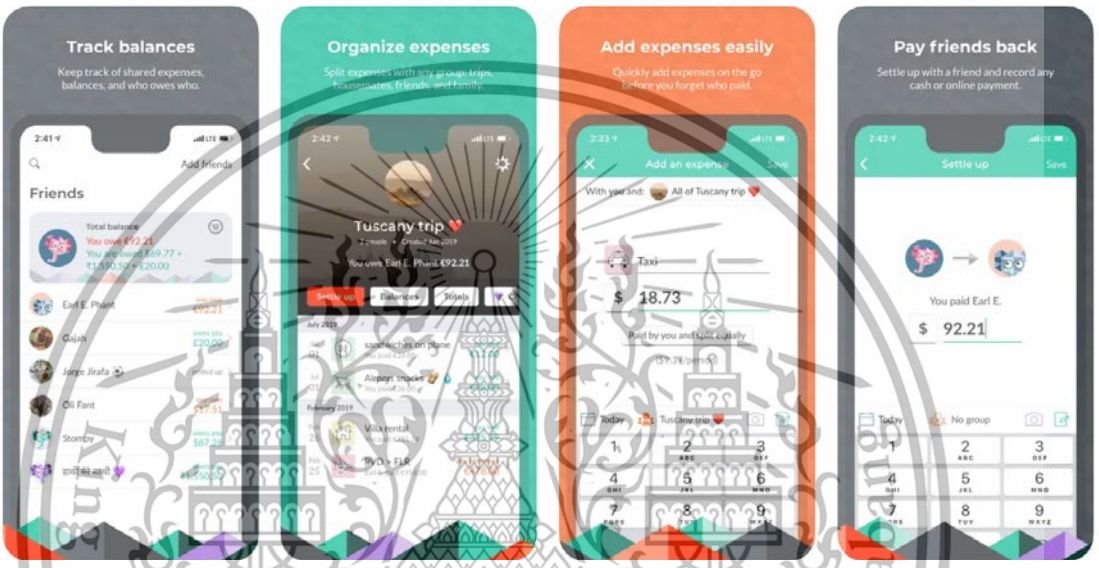



*Figure 4. Splitwise Sample Interface*

Splitwise, users can send an email notification when a bill is due, and the app allows users to send an IOU to someone else in the group. Splitwise does not handle any cash transactions, nor does it link to personal banking details, making it safer than other apps [4].

### 2.1.2 Splid – Split group bills

Splid focuses on money-splitting on group trips. Splid allows you to add in all the expenses of a trip and then split it between each person on the trip. The app is useful for splitting non-trip expenses as well. The application is supporting Not equal splitting. You can settle bills in more than 150 currencies and if you need an offline record of expenses, you can download a PDF or Excel file summary [5].

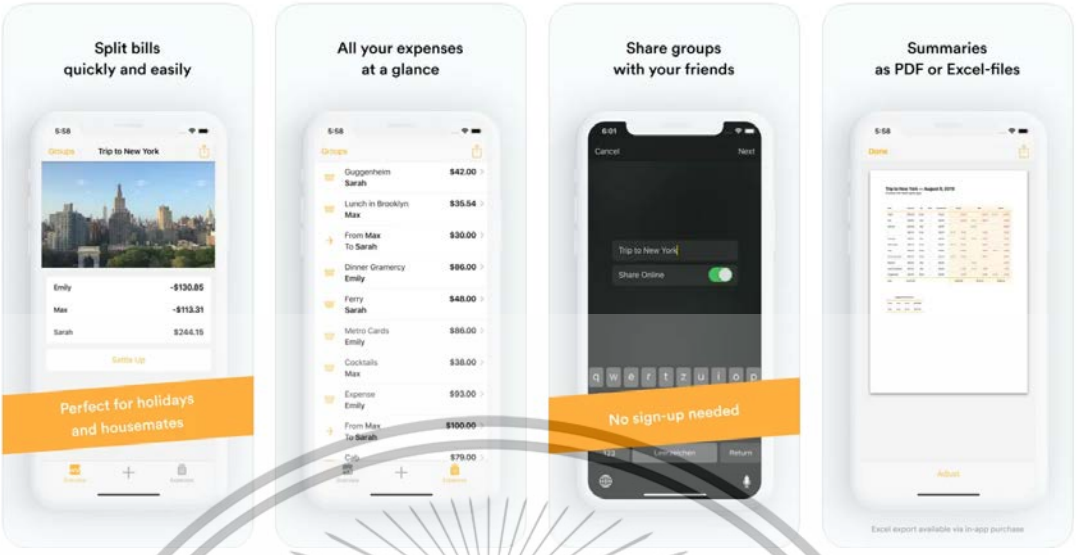



*Figure 5. Splid Sample Interface*

The disadvantage of this program is not having online payment features and a monthly bill option. The application does not handle cash transactions and does not connect to personal banking information.

### 2.1.3 Tab

Tab is for splitting the dinner bill amongst a group of people. You have to upload the scanned copy of the receipt, and all the participants can access the bill on their app. Users can then tap whatever they ordered to settle the payment, or multiple users can share the cost of a single order. The app automatically calculates the taxes and tips and distributes them in proportion [6].

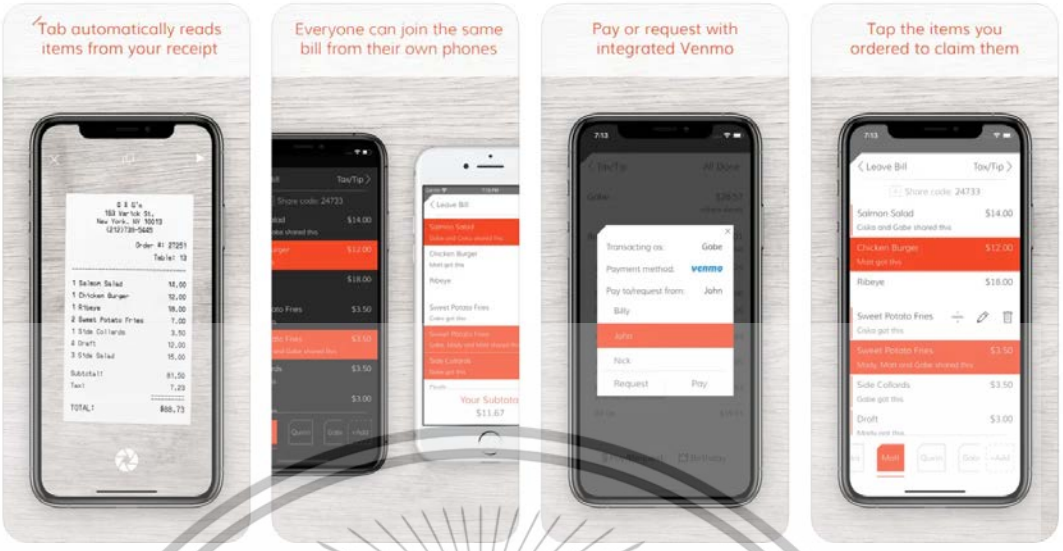



*Figure 6. Tab Sample Interface*

Moreover, the app is only available in the United States owing to the standard receipt format, taxes, and native languages.

**Features Comparison of Similar Application**

| Feature/ Platform | Splitwise | Splid | Tap | Meekor |
|---|---|---|---|---|
| Type | Application | Application | Application | Application |
| Chatting | | | | Yes |
| Show online payment info | Yes | Yes | Yes | Yes |
| Bill splitting | Yes | Yes | Yes | Yes |
| Monthly bill | | | | Yes |
| Calculating service charges and | | | Yes | Yes |

| | | | | |
|---|---|---|---|---|
| taxes | | | | |
| Uploading a bill slip image | | Yes | Yes | Yes |
| Scanning bill slip to text | | | Yes | |
| Not equal splitting bill choice | Yes | Yes | Yes | Yes |
| Bill summary | Yes | Yes | Yes | Yes |
| Every user can read data | Yes | Yes | Yes | Yes |
| Each user's personal summary of all bills | Yes | Yes | Yes | Yes |
| Automatic reminding | | | | Yes |
| Adding new users after splitting | | | | |
| Auto inform for successful payment | | | | Yes |

*Table 2. Features Comparison of Similar Application in Other Countries*

# Chapter 3

# Background knowledge

## 3.1 REST API

An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other. A REST API is an API that conforms to the design principles of the REST, or representational state transfer architectural style. For this reason, REST APIs are sometimes referred to as RESTful APIs.

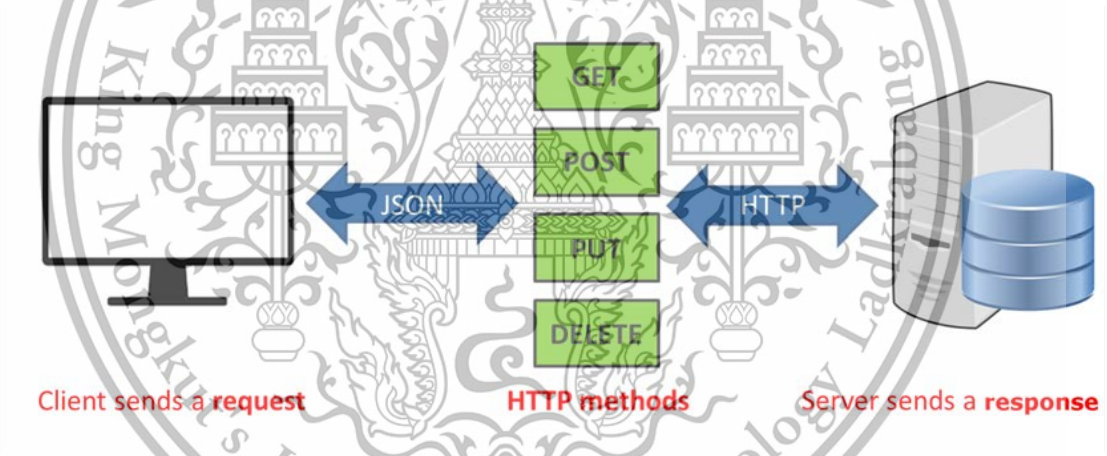



*Figure 7. Rest API Architecture*

REST APIs communicate via HTTP requests to perform standard database functions like creating, reading, updating, and deleting records (also known as CRUD) within a resource. For example, a REST API would use a GET request to retrieve a record, a POST request to create one, a PUT request to update a record, and a DELETE request to delete one. All HTTP methods can be used in API calls. A well-designed REST API is similar to a website running in a web browser with built-in HTTP functionality. The state of a resource at any particular instant, or timestamp, is known as the resource representation. This information can be delivered to a client in virtually any format including JavaScript Object Notation (JSON),

HTML, XLT, Python, PHP, or plain text. JSON is popular because it's readable by both humans and machines—and it is programming language-agnostic.

Request headers and parameters are also important in REST API calls because they include important identifier information such as metadata, authorizations, uniform resource identifiers (URIs), caching, cookies and more. Request headers and response headers, along with conventional HTTP status codes, are used within well-designed REST APIs [7].

It is used in Line Bot as the Bot Act as REST Client and is required to access and process the resources from Line. In Meekor Line Bot also contains the web application part. The web application of Meekor is also used by RESTful Web Service to process the input data.

The key HTTP methods used in REST API interactions are as follows:

- GET: This method is used to retrieve resource representations or information from a server. It is meant for read-only operations and does not modify the resource.
- POST: The POST method is employed to create new subordinate resources within a collection. It is used when a client wants to submit data to be processed by the server, resulting in the creation of a new resource.
- PUT: When a client needs to update an existing resource, the PUT method is utilized. It replaces the entire resource with the updated representation. If the resource does not exist, the API may choose to create a new resource based on the provided representation.
- DELETE: The DELETE method is used to remove a specified resource from the server.
- PATCH: HTTP PATCH requests allow partial updates to be made on a resource. Instead of replacing the entire resource, this method enables modifications to specific fields or properties.

## 3.2 Object relational mapping

An object-relational mapper provides an object-oriented layer between relational databases and object-oriented programming languages without having to write SQL queries. It standardizes interfaces reducing boilerplate and speeding development time.


*Figure 8. ORM Architecture [9]*

Object-oriented programming includes many states and codes in a format that is complex to understand and interpret. ORMs translate this data and create a structured map to help developers understand the underlying database structure. The mapping explains how objects are related to different tables. ORMs use this information to convert data between tables and generate the SQL code for a relational database to insert, update, create and delete data in response to changes the application makes to the data object. Once written, the ORM mapping will manage the application's data needs and you will not need to write any more low-level code [8].

## 3.3 Database management system

Database Management System (DBMS) refers to the technology solution used to optimize and manage the storage and retrieval of data from databases. DBMS offers a systematic approach to manage databases via an interface for users as well as workloads accessing the databases via apps. The management responsibilities for DBMS encompass the information within databases; the processes applied to databases such as access and modification; as well as the logical structure of the database. DBMS also facilitates additional administrative operations such as change management, disaster recovery, compliance and performance monitoring, among others [10].

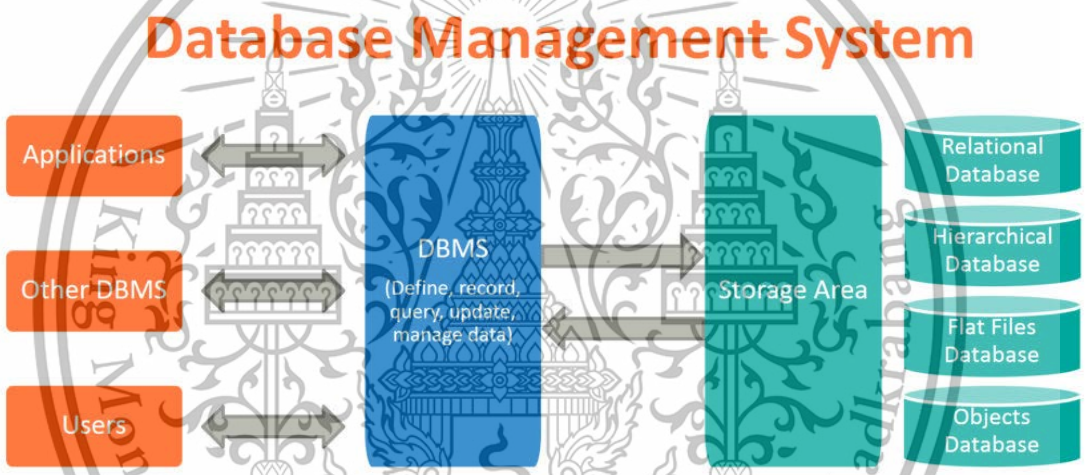



*Figure 9. The schematic of a DBMS system [10]*

## 3.4 Crud Application

The CRUD acronym identifies all of the major functions that are inherent to relational databases and the applications used to manage them, which include Oracle Database, Microsoft SQL Server, MySQL, and other [11]s.

The four CRUD functions can perform different types of operations on selected data within the database.

- Create - The Create operation involves using the INSERT statement to generate a new record in a table. This operation is typically associated with the HTTP methods PUT or POST.
- Read - The Read operation retrieves data from a table based on the provided primary key. It is commonly mapped to the HTTP GET request.
- Update - The Update operation modifies existing data in a table by executing the UPDATE statement with the specified primary key. This operation is typically associated with the HTTP methods PUT, POST, or PATCH.
- Delete - The Delete operation removes a specific row from a table based on the conditions specified in the WHERE clause. It is commonly mapped to the HTTP DELETE request.

These CRUD operations are mapped to corresponding HTTP requests and are utilized during the development phase of an application. They provide a standardized approach for interacting with and manipulating data within a system.

## 3.5 Web Application

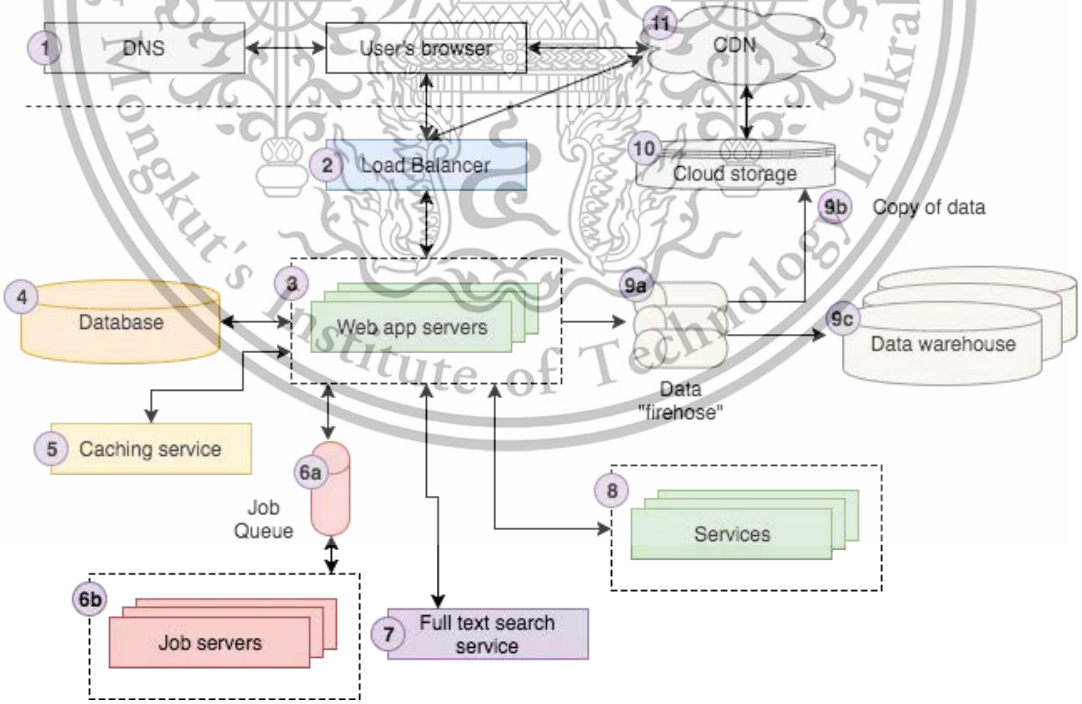



*Figure 10. Modern web application architecture overview [12]*

### 3.5.1 Web Application Servers:

The web application servers serve as a main component of the whole system architecture. At a high level of web servers, they execute the core business logic that handles user's requests and send back HTML to the user's browser. To achieve this, they act as a hub that connects and communicates with all other back-end components such as databases, caching layers, caching services, search services, and other [12]. microservices.

### 3.5.2 Database Servers:

Database servers contain one or more databases to store data and information. The database system provides us the way to define data structures, adding new data, searching for existing data, updating or deleting existing data, and more [12].

### 3.5.3 Cloud Storage:

"Cloud storage is a simple and scalable way to store, access, and share data over the Internet" according to AWS. We can use it to store anything we would store on our local storage. The benefit is being able to interact with the cloud via a RESTful API over HTTP. Also, using the cloud storage provided by vendors means that our storage will not easily be lost as they are 24 hours taken care of by its provider [12].

## 3.6 Line

Line application is the communication application that enables users to send text messages, photos, location, file etc., to their peers or chat groups. Line operates on Smartphone, Personal Computer, iPad and smart watch. Line offers Messaging API and Bot SDK for the developers to add their own functionality to build and maintain their own bots.

### 3.6.1 Line Bot and Official Account

#### 3.6.1.1 Line Official Account

Line Official Account is the special account that enables multiple users to operate and maintain the reachability to its target audience. Line Official Account can have automation in replying to some certain messages, broadcast and manually replies to another user.

#### 3.6.1.2 Line Bot

Line Bot is a part of Line Official Account that uses Line Messaging API and the Bot SDK to develop and enhance the functionality of Line Official Account to always operate on their own without actual humans operating on it.

Some notable of advantages of Line Bot are :

- Performance : No Human required to read and reply to the messages from the users. This reduce human errors
- Cost & Time Effective : Owner does not need to hire the worker to do the manual job which sometimes is required to answer the same question repeatedly.
- Easy to maintain : New functionality can be added directly to the Bot by adding the new lines of code.
- User usability and reach : As Line is very famous and it is one of the main communication platforms. Most of the users are already very comfortable using Line. It is the most effective way to drastically reduce the time to learn the main feature from the user.
- Tools and Library : Line provides the SDK for the Bot development, full documentation and support for development of Bot. Line also provides Line Messaging API which helps the developers to integrate their Bot with Line.
- Data Analysis Compatible : The data from the Bot can be collected and stored in databases.

Some notable of disadvantages of Line Bot are :

- Cost : The deployment of Bot has to have a server which increases the initial cost for some cases.
- Development know-how : The development of Bot has to have a developer who understands how to program
- Offline not-compatible : Cannot operate when the user is offline or the server is down
- Line dependency : Bot cannot operate without using Line service. In the case of Line shutting down or changing the policy, the Bot cannot operate on its own.

## 3.7 Containerization

Containerization helps in the portability and supportability across the development environment. The Containerization software packages the code, libraries, and dependencies required to run the code into a single lightweight executable, referred to as a container. The container can run consistently on any infrastructure.

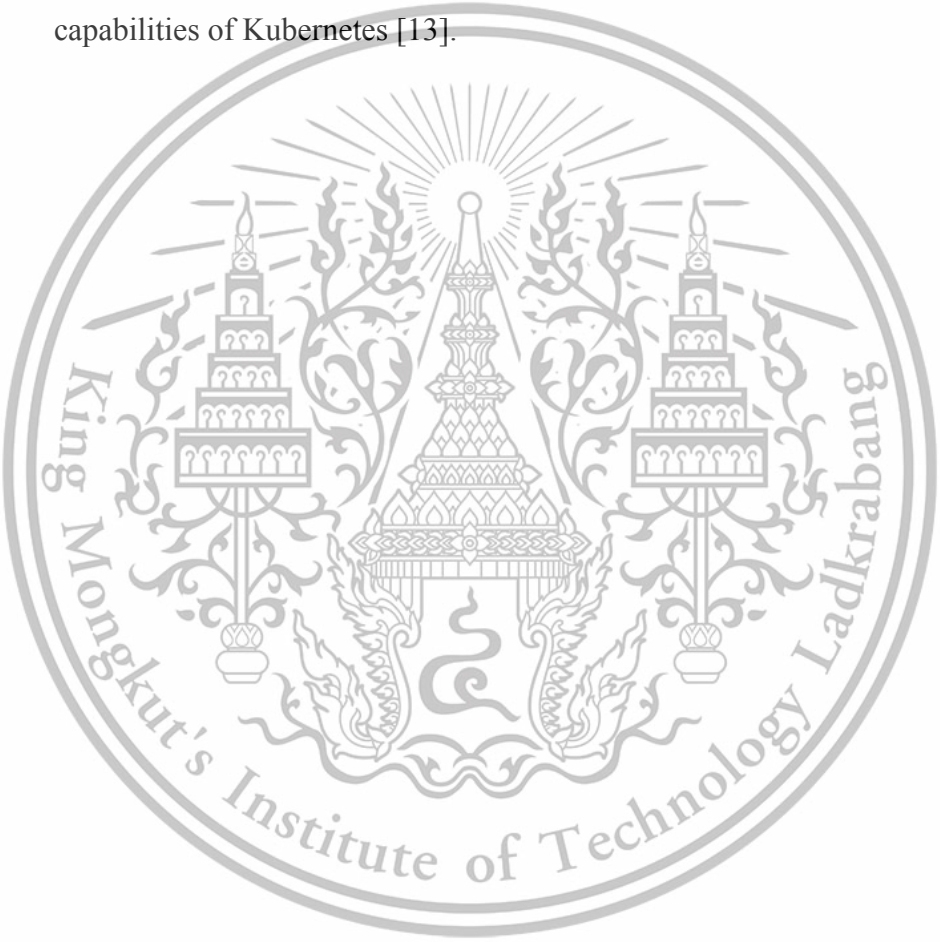
The benefits of using Containerization are listed below:

- Portable: More portable and resource-efficient than virtual machines as the container does not require installing a new virtual machine to run the code in the same environment.
- Development agility: Containerization enhances agility, portability, and speed of development. It eliminates dependency and environment problems on different machines.
- Resource efficiency: As no VM is required, the cost of using Containerization can be relatively low compared to hosting a new VM and setting up the exact environment required for the code to run [13].

### 3.7.1 Container Platforms:

Container platforms provide solutions for managing containerized applications by providing capabilities such as automation, orchestration, and governance. The types of container platforms are listed below:

- Container Engine: Container Engine enables developers to create and manage containers. Docker is the choice for this project.
- Container Orchestrator: Container Orchestrator provides the deployment of containers in the cluster and allows developers to manage and automate containers. The popular choice is Kubernetes.
- Container as a Service: Container as a Service (CaaS) is a third-party cloud service that enables developers to create and maintain containers. For example, Google Kubernetes Engine is a CaaS offering that extends the capabilities of Kubernetes [13].



# Chapter 4

# Requirements Analysis and System Design

## 4.1 Requirement

**List of defined user types**

| No. | User | Description |
|---|---|---|
| 1. | Loaner | Person who permits something to be borrowed, or is something that is being borrowed temporarily. The loaner usually is the one who created the bill |
| 2. | Debtor | Person who owes a creditor,someone who has the obligation of paying a debt. |
| 3. | Group member | Group members represent both loaner and Debtor. They are both presented in the same group chat on which Meekor operates. |

*Table 3. User Types Table*

### 4.1.1 Functional Requirement

| # | Description |
|---|---|
| 1 | The member in the group can invite Meekor to the Line group |
| 2 | The loaner can input payment information which are, Thailand promptpay number and bank account |
| 3 | The loaner can create a new one-time bill by typing '#สร้างบิล' or interact with flex message from the Meekor |
| 4 | Members can view all bill history of the group by typing '#ดูบิลทั้งหมด' or interact with quick reply box from Meekor. |
| 5 | The loaner can verify transaction amounts and balances to be paid by the debtor of the group from the payment slip. |
| 6 | The loaner can delete their old bill. |
| 7 | The loaner can edit the amount of balance to be paid by every other member. |
| 8 | The loaner in the group can make Meekor to announce the remaining person whom has not pay the bill yet by typing '#ทวงเงิน'. |
| 9 | The loaner has the option to pay by cash. The member did not have to send the payment slip into the group. |
| 10 | The loaner can be able to confirm the payment manually in case of receiving payment by cash. |
| 11 | Meekor can automatically announce the remaining person who has not paid the bill yet at 12.00 pm everyday. |
| 12 | The member in the group can view Meekor's manual. |
| 13 | Meekor have the option to generate the Payment QR code. |

Table 4. Functional Requirement Table

### 4.1.2 Non-Functional Requirement

| # | Description |
|---|---|
| 1 | The system should support both ios and android devices |
| 2 | The system should support Line application |
| 3 | The system should display the currency as Thai Baht |
| 4 | The deployment of the system must use public cloud service |
| 5 | The project must be containerized by using Docker |
| 6 | The system must be easy to understand for the user |
| 7 | The database should be RDBMS ( Relational Database Management System ) |

*Table 5. Non-Functional Requirement Table*

## 4.2. Use case Diagram

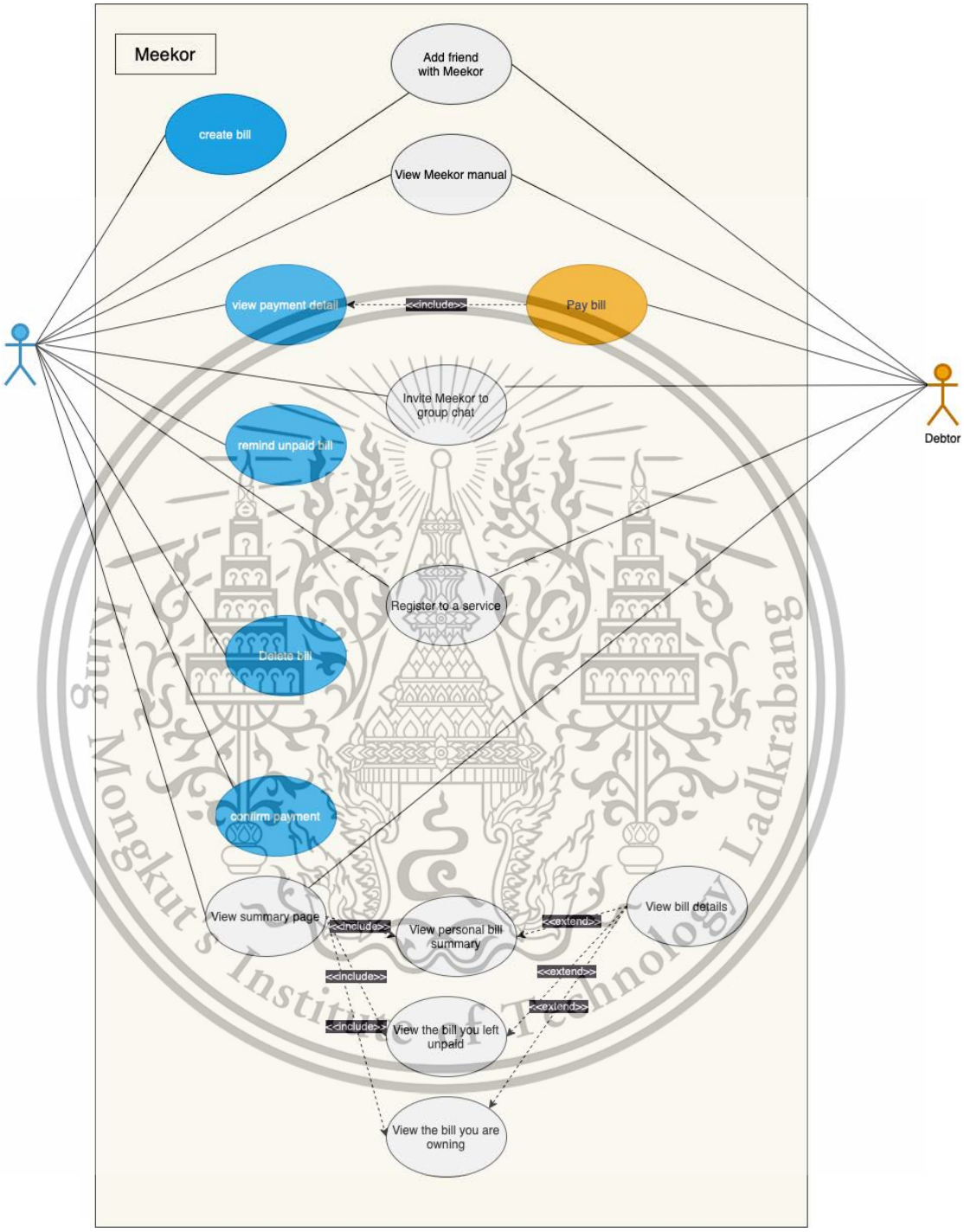



*Figure 11. Use Case Diagram*

### 4.2.1 Brief Use Case

1. **Invite Meekor to group chat**: The user must be able to invite Meekor Line bot to group chat.
2. **Add friend:** The user must be able to add friend with Meekor by interacting with flex message or directly add friend with Meekor via Line add friend feature.
3. **Create Bill:** The user must be able to create a new bill in the group chat.
4. Pay bill: The user must be able to pay a bill which includes them by uploading payment slips or pay with cash.
5. **View personal summary:** The user must be able to view all bills which included them as a debtor.
6. **View bill detail:** The user must be able to view detailed information of every bill created in the group.
7. **View Meekor manual:** The user must be able to view a manual of Meekor in the group.
8. **Confirm bill payment:** The user must be able to view the payment slip which was uploaded by debtors and confirm the bill payment.
9. **Delete bill:** The user must be able to delete the bill which was created by them from the group.
10. **Remind unpaid bill:** The user must be able to send reminders of their unpaid bill in the group.
11. **View summary page:** The user must be able to view the bill summary page.
12. **View the bill you left unpaid:** The user must be able to view the bill which they left unpaid in the group.
13. **View the bill you are owning:** The user must be able to view the bill they are owning in the group.
14. **View payment detail**: The user who owns the bill must be able to view the payment detail of their bills in the group.
15. **Register to service:** The user must be able to register to access Meekor service.

## 4.3 Activity Diagram

### 4.3.1 Create bill feature activity diagram

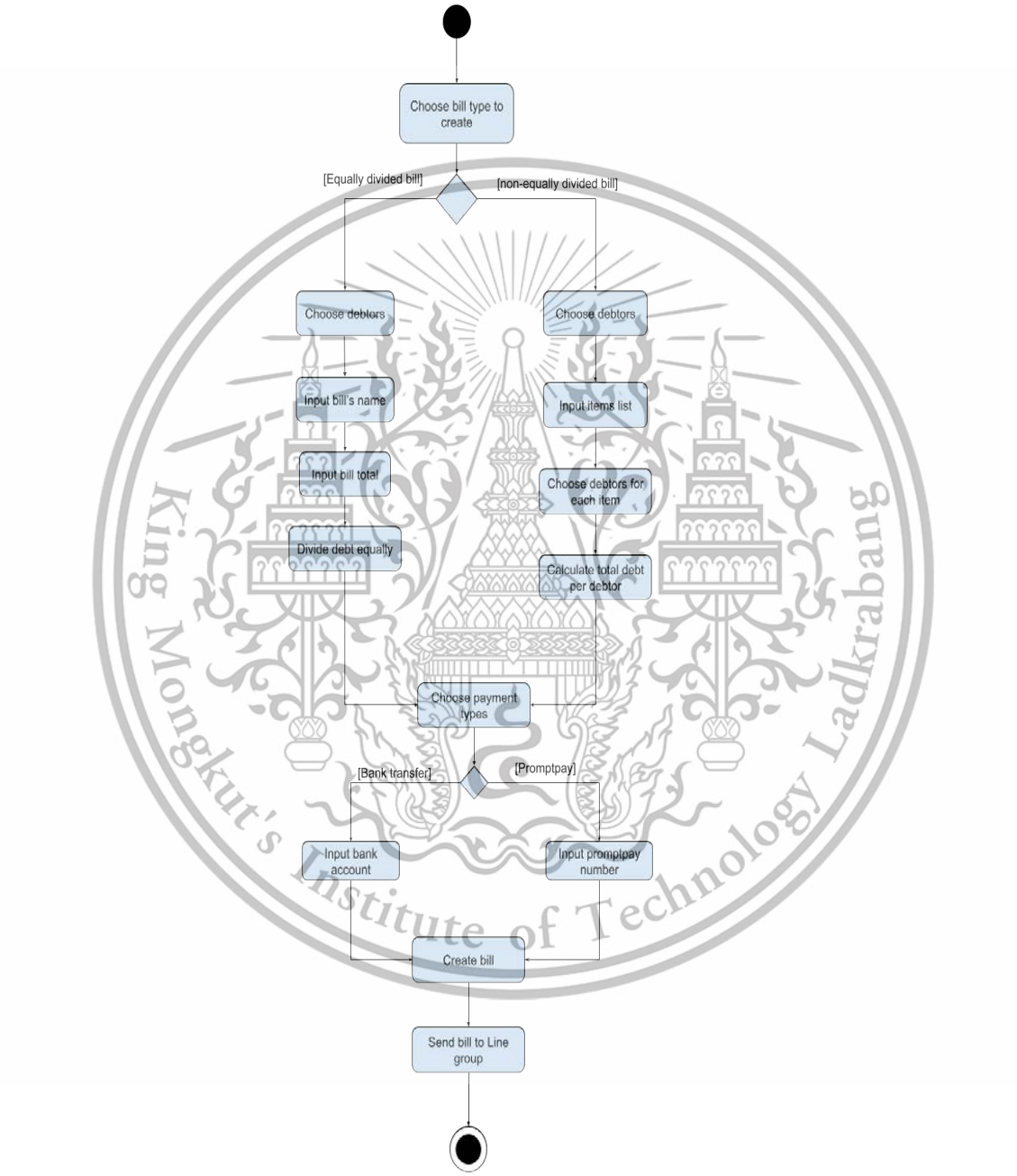



*Figure 12. Create bill activity diagram*

### 4.3.2 Pay bill feature activity diagram

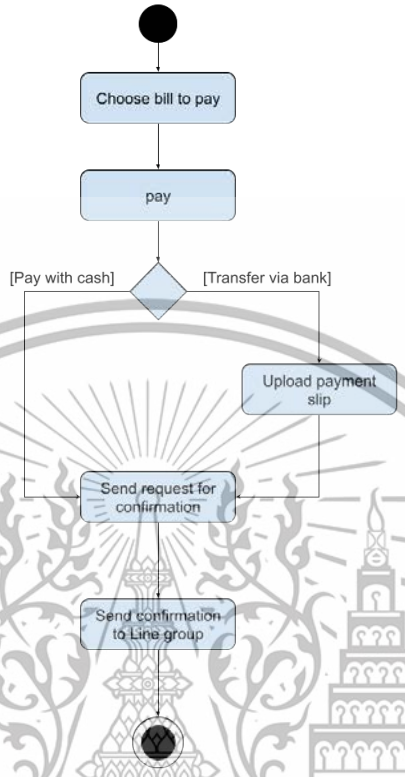



*Figure 13. Pay bill activity diagram*

## 4.4 System Architecture

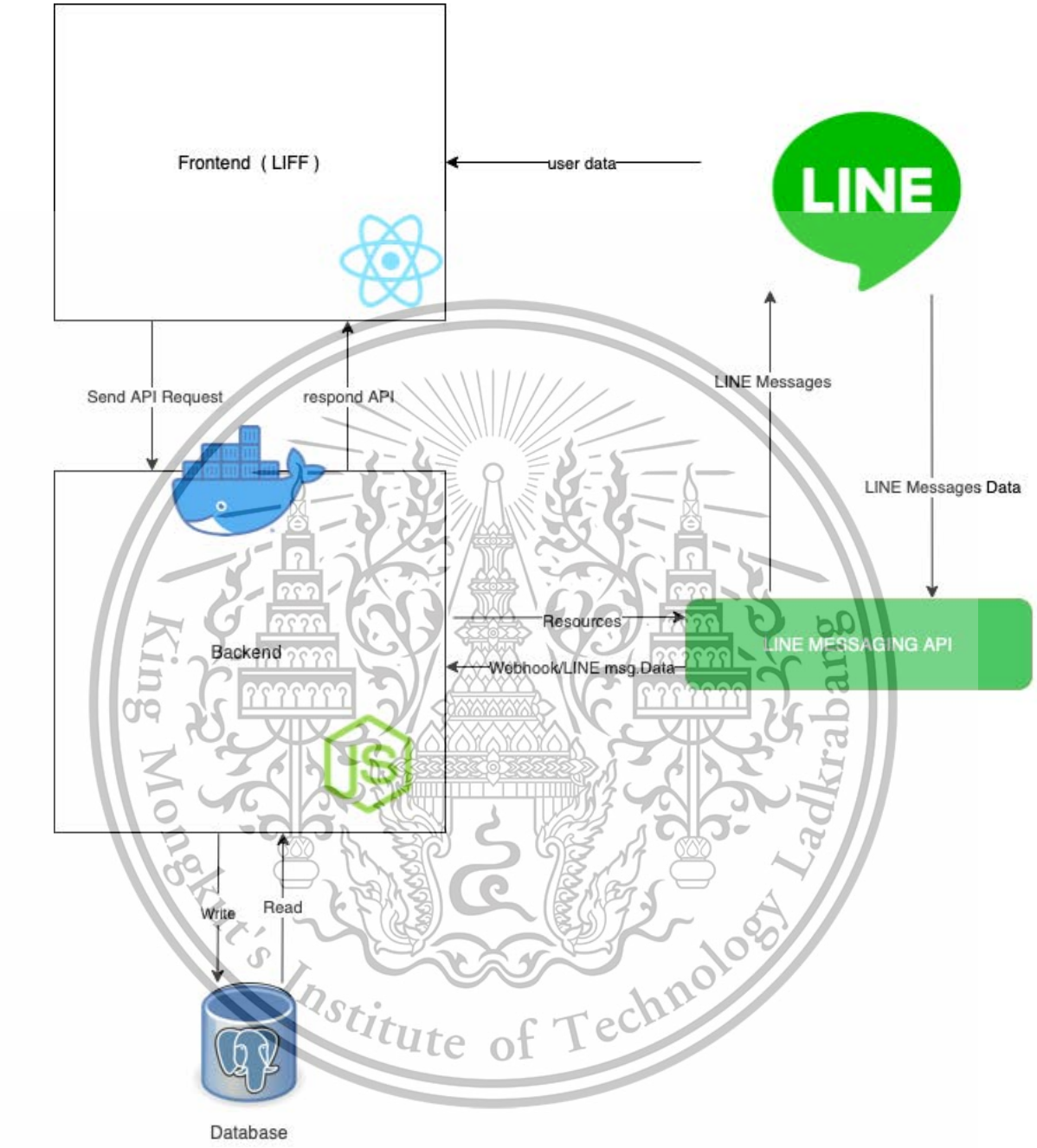



*Figure 14. System Architecture*

### 4.4.1 Frontend Component ( LIFF )

Line Front-end Framework ( LIFF ) is a WebView that is integrated directly with Line application. LIFF provides direct interaction between Line and Web Application by sending user data to the Web Application. By combining LIFF and Web Application we can name the application, LIFF application. LIFF application is written using React.js which consist of the components described below:

- User interface is a component that is responsible for displaying interface and interacting with users. The user interface is built by using React.js along with the plugins such as Tailwind, Bootstrap etc.

### 4.4.2 Backend Component

The backend responsibility is to receive requests from the LIFF application and send responses back to the LIFF application. Backend Component also received webhooks from the Line Messaging API which provides the Line messages data from the chatroom. Backend component consist of another component described below:

- Server is a component that is responsible for handling the client's HTTP request and response. The server also received webhooks along with the Line messages data from the Line messaging API.
- Database is responsible for storing and writing data. Postgresql is selected for handling these tasks due to the information having a fixed structure.

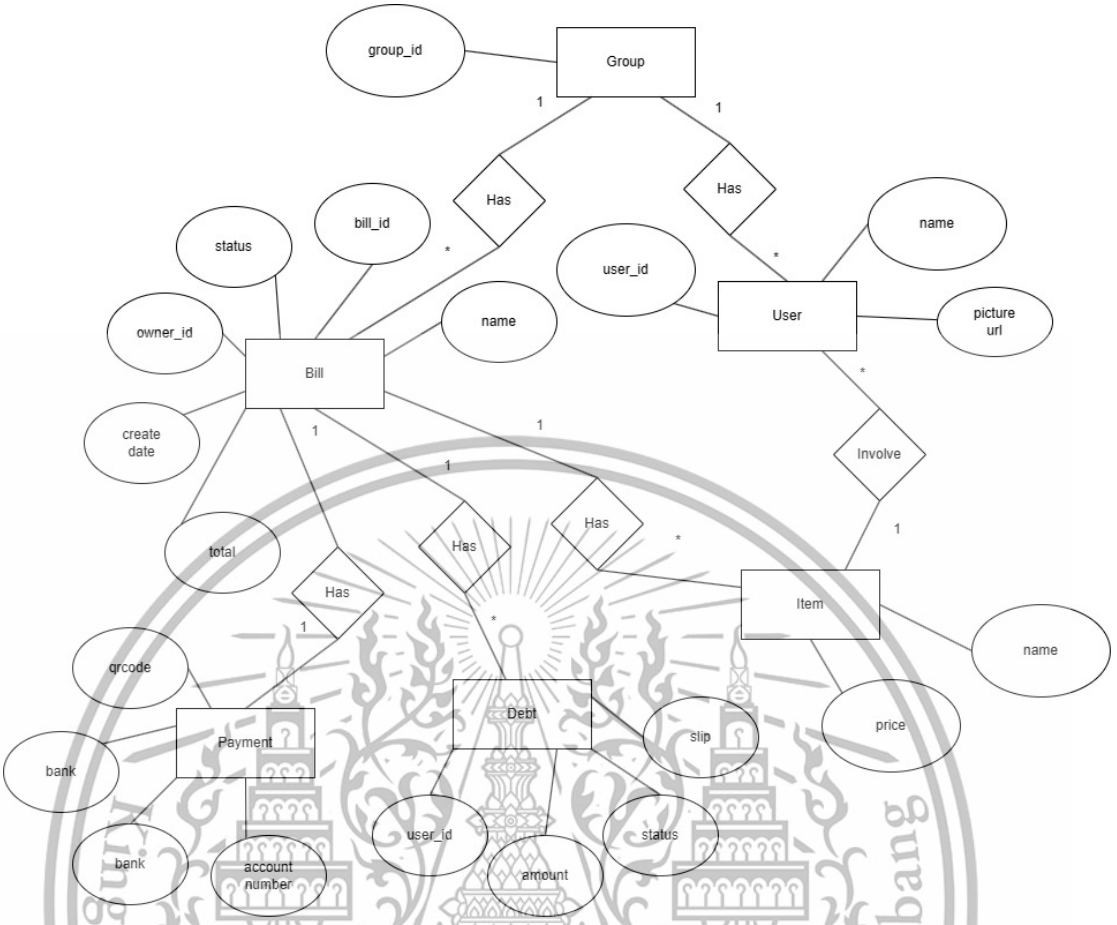



*Figure 15. database ER diagram*

### 4.4.3 Line Messaging API Component

Line Messaging API is an API provided by Line. Line messaging API provides us with necessary API which is mainly used for receiving Line user data and Line messages. Furthermore, Line messaging API gives us a method to directly send the Line messages to the targeted chat room along with sending messages and user data to our Backend server via. Webhook event.

### 4.4.4 Line Application Component

Line application is an application that directly acts as the interface between the user and our application. Users can open a LIFF application by interacting with it on Line Application. When our bot Meekor sending the messages, the user can read or interacting with the message on Line Application. Line application consists of additional components described below :

- Flex messages : a custom made message provided by Line. It usually consists of text, buttons and images packed in one message.
- Chat room : Chat room is a room where a group of users of Line applications can interact with each other by sending messages or images. Also, Chat Room is the main method for our application to interact with users.

## 4.5 Frontend Design

In our project, we begin with Miro as the initial wireframing tool to create a concise foundation for our design concept. Miro's Collaborative Features facilitate team ideation and the generation of initial design concepts. However, to achieve refined and production-ready designs, we transition to Figma. Figma's comprehensive tools and capabilities enable meticulous refinement, ensuring pixel-perfect alignment, typography, and interactivity. This integration of Miro and Figma in our design workflow optimizes collaboration, creativity, and precision, resulting in exceptional frontend experiences.

### 4.5.1 Color Palette

The choice of a palette featuring pink, cream, and brown for the UI of the "Meekor" debt collecting application, accompanied by a bear mascot, serves multiple purposes. Firstly, the palette's soft and warm tones, including pink and cream, create a visually pleasing and approachable interface, which helps alleviate the often negative connotations associated with debt collection. By employing these colors, Meekor aims to establish a more empathetic and user-friendly environment, fostering a sense of trust and understanding for its users. Additionally, the inclusion of brown complements the softer shades, adding a touch of grounding and reliability to the overall design. The bear mascot further enhances Meekor's character, representing strength, protection, and a reassuring presence. This combination of colors and mascot helps to reframe the debt collection experience, emphasizing a supportive and helpful approach rather than a confrontational one, ultimately improving user engagement and fostering a more positive user experience.[14]

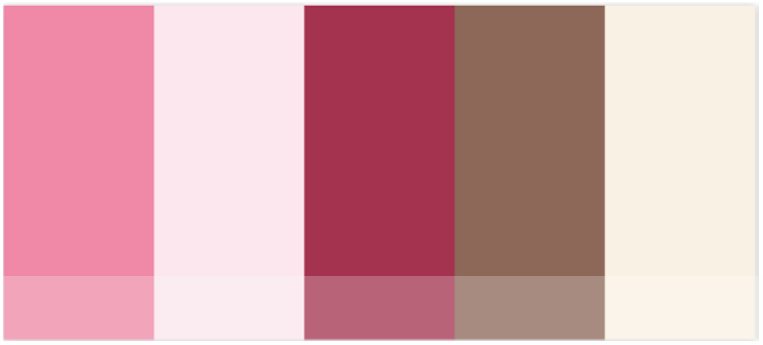


*Figure 16. Color Palette*

### 4.5.2 Wireframe

#### 4.5.2.1 Miro

Miro is a whiteboarding platform, which played a pivotal role in the initial stages of LIFF's user interface design. It was employed to create a concise wireframe showcasing a specific use case and the application's features. The sketched UI design in Miro served as a guiding framework for the actual UI development.[15] As the development process progressed, our team made a collective decision to embark on a UI redesign endeavor aimed at enhancing user-friendLiness and visual appeal. This involved incorporating captivating colors and making the interface more intuitive and user-centric.

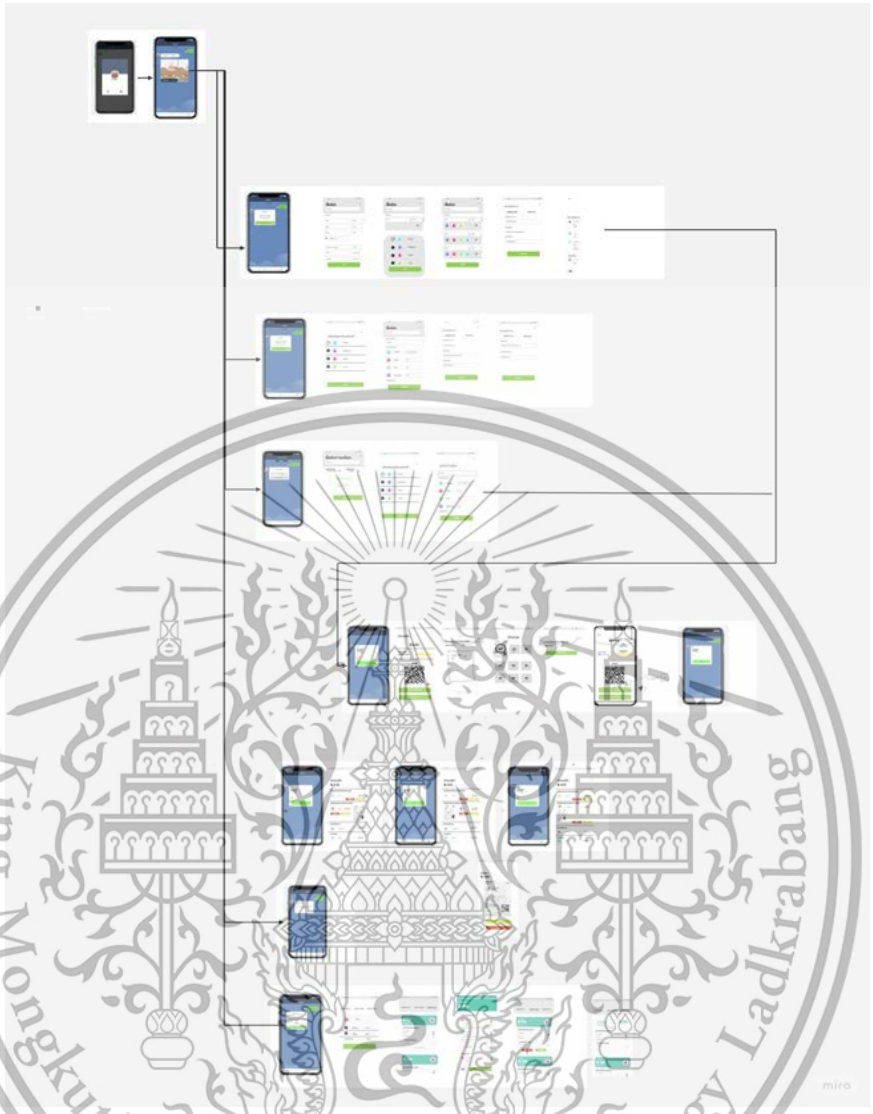


*Figure 17. Miro Wireframe*

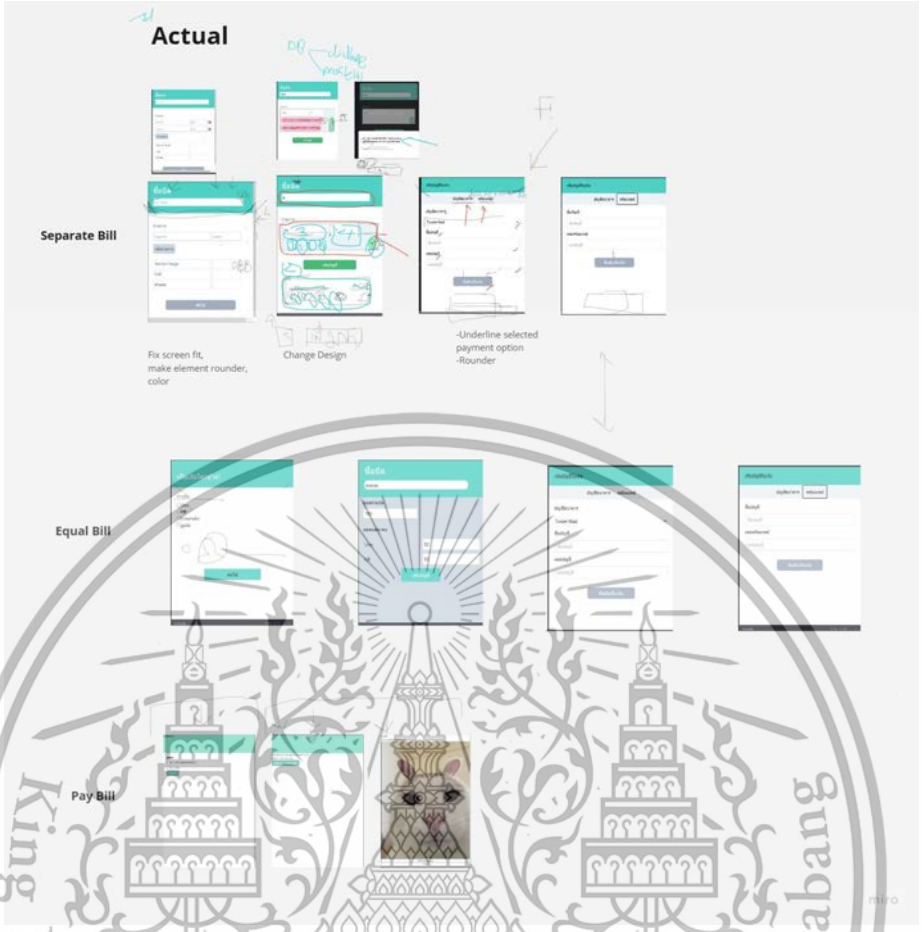



*Figure 18. 1st Edition of User Interface*

#### 4.5.2.2 Figma

After careful consideration of the previous UI design developed in Miro, it was determined that a change in the color palette for the application was necessary. To accomplish this, the decision was made to transition to Figma for the creation of the final production. Figma was chosen for its ability to generate more precise and aesthetically pleasing wireframes.[16] This transition was deemed essential to ensure the highest level of accuracy and visual appeal in the final user interface.


*Figure 19. Create bill page wireframe*

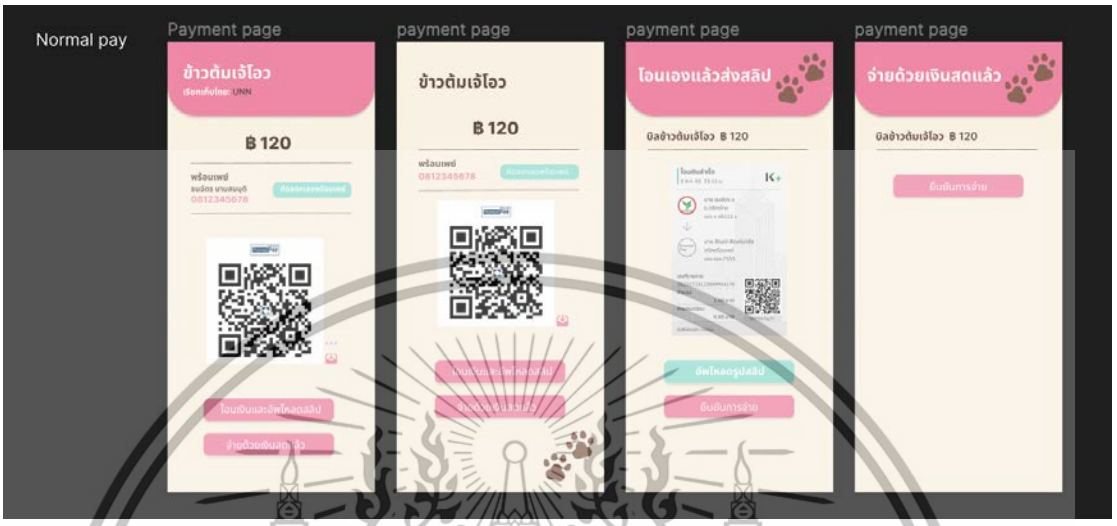



Figure 20. Payment page wireframe

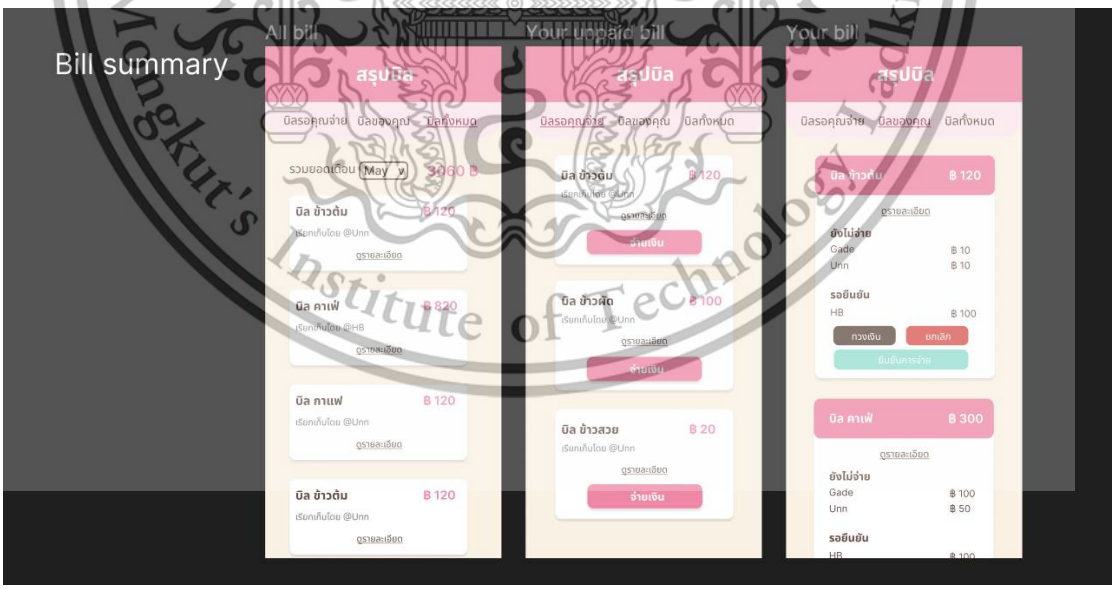



*Figure 21. Bill summary page wireframe*

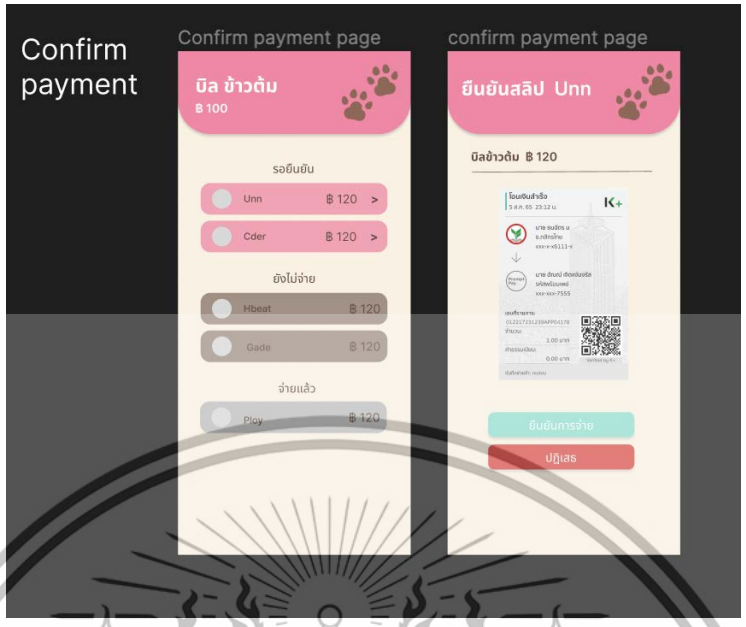



*Figure 22. Confirm payment page wireframe*

### 4.5.3 User Interface Design

When it comes to user interface design, incorporating Gestalt Principles can greatly enhance the overall user experience. One key principle is closure, which suggests that users tend to perceive incomplete or fragmented elements as a whole. By strategically utilizing this principle, designers can create intuitive interfaces that allow users to effortlessly perceive and comprehend the underlying structure. Another important principle is proximity, which states that objects that are close to each other are perceived as a group. Leveraging this principle helps designers organize related elements in a way that facilitates easy recognition and understanding of their relationships. Additionally, the common region principle suggests that elements contained within a common boundary are perceived as a group. By employing this principle, designers can visually group related elements and differentiate them from unrelated ones, thereby enhancing overall clarity and ease of use. Lastly, the principle of similarity asserts that elements sharing similar attributes, such as shape, color, or size, are perceived as related. By employing this principle judiciously, designers can establish visual hierarchies, convey meaning, and guide users' attention effectively. In conclusion, by integrating closure, proximity, common region, and similarity principles into the user interface design of our project, we can create a visually

cohesive and intuitive experience that optimizes user engagement and satisfaction.[17][18]

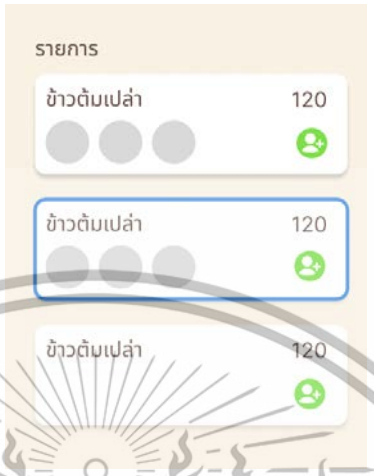



*Figure 23. Example of Common region principle*

From figure 17. For example, the common region principle of Gestalt was applied in Web application user interface design by grouping the debtors who shared the item in the same card. Thus, the user could distinguish each item and recognize the debtors of each item.

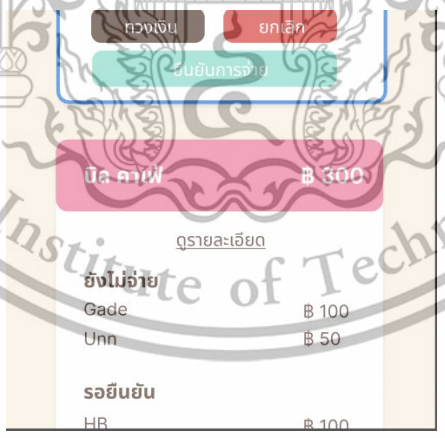



*Figure 24. Example of Closure principle*

Figure 18 exemplifies the use of the closure principle of Gestalt in user interface design. By overlaying a card on the screen, it indicates to the user that there is more content to be discovered by scrolling down. This implementation enhances user experience, promotes engagement, and ensures a seamless browsing experience.

### 4.5.4 Line Bot Design

#### 4.5.4.1 Line bot functionality

The functionalities of the Meekor Line chatbot can be classified into two distinct parts. The first part encompasses the Line chat bot itself, while the second part involves the utilization of the Line front-end framework (LIFF), which is a web application accessible through the Line application.

The primary functionality of the Line chat bot entails facilitating user interaction through messaging within Line group chats. When a user sends a message, the bot responds by employing a flex message, allowing users to perform various tasks such as creating bills, viewing all bills within the group, collecting debt, and deleting bills. Moreover, the bot provides a user manual for reference and offers the convenience of executing commands through the employment of quick reply functions.

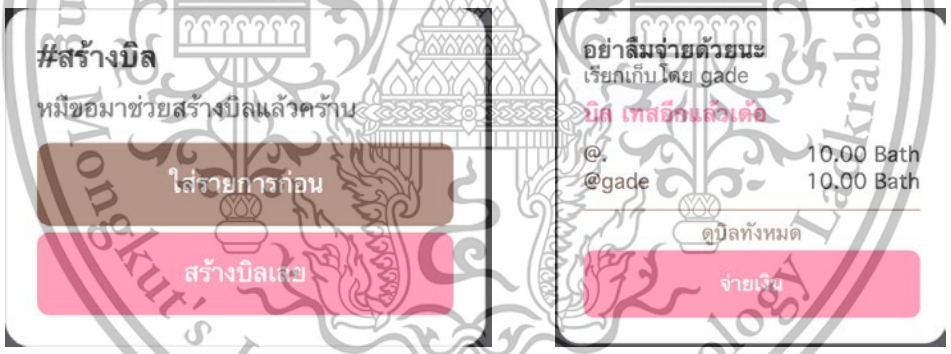



*Figure 25. Example of Crate bill function in Line bot*

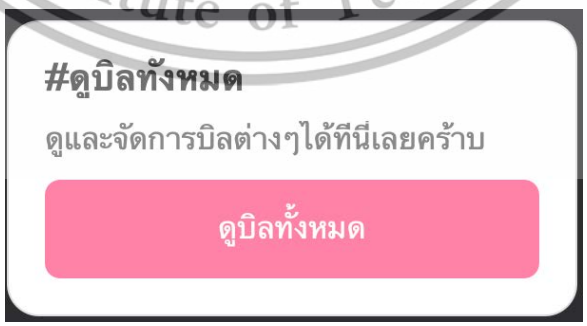



*Figure 26. Example of View all bill function in Line bot*

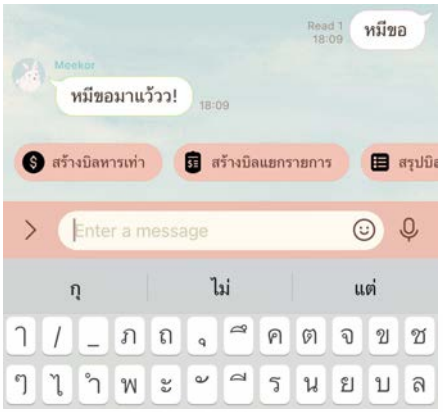



Figure 27. Example of Line bot quick reply

#### 4.5.4.2 Line Bot Characteristic

The Meekor Line chatbot features a charming mascot, personified as a pink bear, exuding a distinct playful essence. Deliberate considerations were made in crafting the character of Meekor to embody a lighthearted and comical demeanor, evident in the carefully chosen language employed in its messages. This deliberate approach serves to cultivate a friendly and amiable atmosphere within the user community.

It is worth noting that the primary purpose of the Meekor Line chatbot revolves around debt collection. Recognizing the potential tension and apprehension often associated with this attempt, the incorporation of Meekor's playful characteristic assumes a vital role. By infusing a sense of playfulness into the bot's interactions, users are encouraged to adopt a more relaxed and less confrontational stance as they navigate the task of debt collection.

The playful aspect of Meekor acts as a valuable mechanism to alleviate the inherent stress and aggression frequently encountered in debt collection processes. Through the incorporation of light-heartedness and humor, Meekor fosters an environment that enables users to approach their debt collection obligations with a greater sense of ease and diminished tension.

To cultivate a playful characteristic for Meekor, we employed Thai slang expressions, such as "คร้าบ," and adopted endearing pronunciations, such as "น้อง" or "หมีขอ," to enhance the charm of the chatbot.

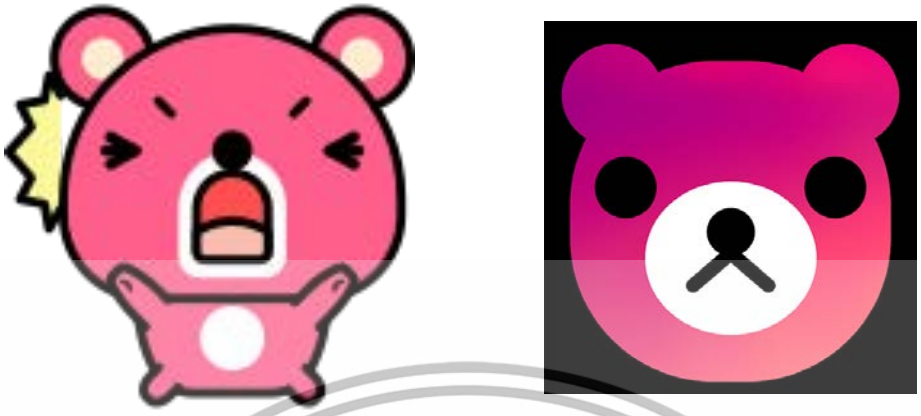


*Figure 28. Meekor mascot*

# Chapter 5

# Development

## System Development

This chapter describes the system development of Meekor. The topics will cover how each development tool is involved in the development process.

## 5.1 Frontend Development

The client-side components are user interfaces of the website which are used to display and interact with users.

### 5.1.1 User Interface


*Figure 29. react logo*

We use React as a frontend development framework. React is a popular and widely used frontend framework developed by Facebook. It is designed to facilitate the creation of interactive user interfaces for web applications. React follows a component-based architecture, where the UI is divided into reusable and self-contained components.

React utilizes JavaScript's capabilities to manage application state, handle user events, and perform data manipulation. React enables us to use JavaScript's features, such as variables, functions, loops, conditionals, and object-oriented programming, within the React code to implement complex logic and functionality.

The bullet points below show the advantage of choosing react over other frontend frameworks.

1. Component-Based Architecture : React follows a component-based architecture, where the UI is broken down into reusable components.
2. Virtual DOM : React utilizes a virtual DOM (Document Object Model) to efficiently manage and update the user interface. The virtual DOM is an abstraction of the actual DOM, and React uses it to perform efficient updates by comparing the previous and current states of the components.
3. Declarative Syntax : React promotes a declarative approach to building user interfaces. Developers describe the desired outcome, and React takes care of updating the UI to reflect the desired state.
4. Rich Ecosystem and Community Support : React boasts a vast and active community of developers, which has resulted in a rich ecosystem of libraries, tools, and resources.
5. Reusability and Code Sharing : React's component-based architecture promotes reusability, allowing developers to create UI components that can be shared across different projects.

### 5.1.2 Line Front-end Framework

Line Front-end Framework (LIFF) is also included in our user interface development, it is a platform for web apps provided by Line. LIFF benefits us in getting data from the Line Platform for example the Line user ID, provides features that utilize user data and send messages on the user's behalf. The application made by LIFF is called LIFF apps.

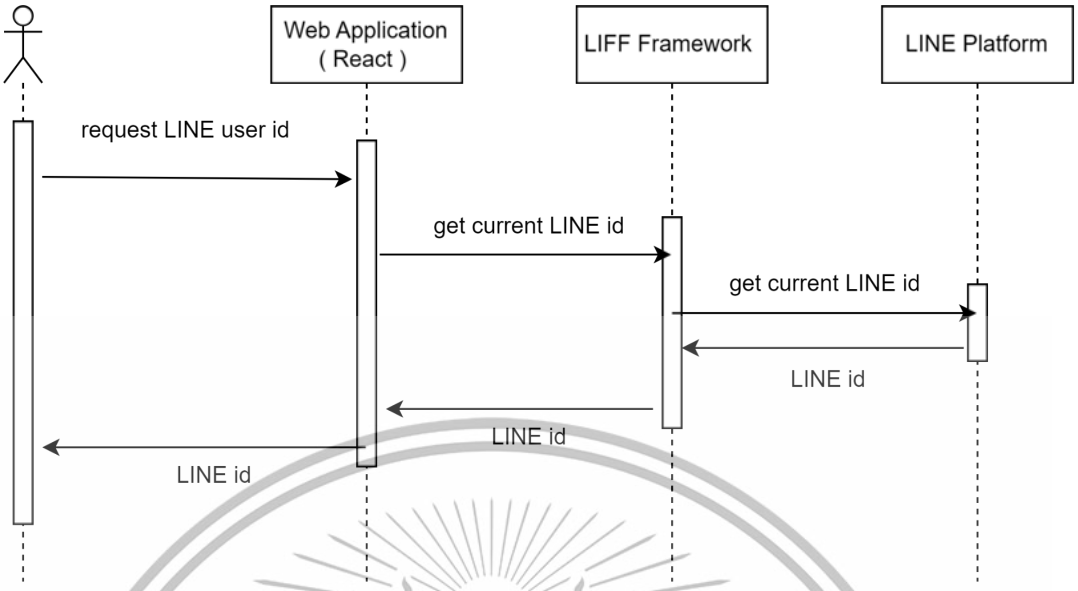



*Figure 30. Example of getting current user Line Id*

LIFF apps must be run in the LIFF browser in order to retrieve the data from Line Platform. Since LIFF browser runs within Line, the LIFF app can access user data without having to prompt users to log in. The LIFF browser also provides features that are specific to Line, such as being able to share the LIFF app and sending a message to a friend.

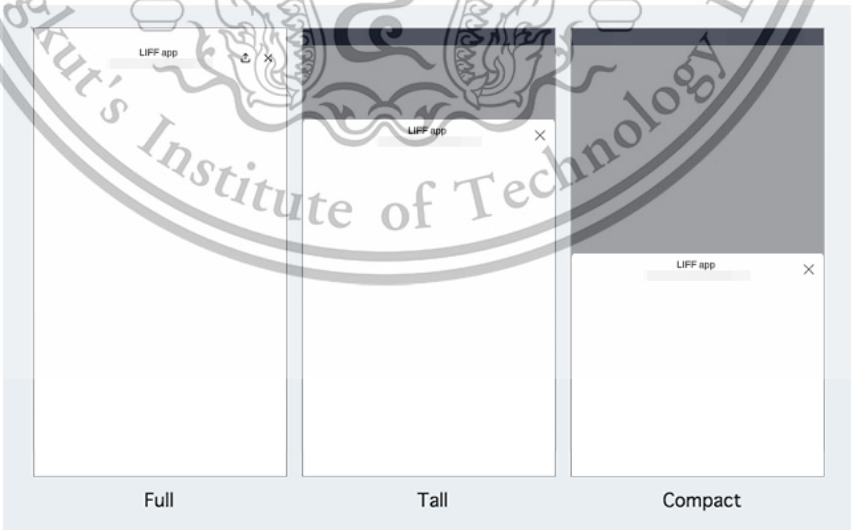



*Figure 31. Example of LIFF Browser*

### 5.1.3 Line Flex Message

Flex Messages are messages that can be freely customized based on CSS Flexible Box (CSS Flexbox). We can adjust the size of the message, allocate text, images, and icons in specific locations, and add interactive buttons according to our specifications.


*Figure 32. Example of flex message*

Line messages are created using Line Flex Message Simulator which provides customization of the message layout freely based on the specification for CSS Flex Box. By using Line Flex Message Simulator, it provides us JSON format objects to be used for our server to send to the user.

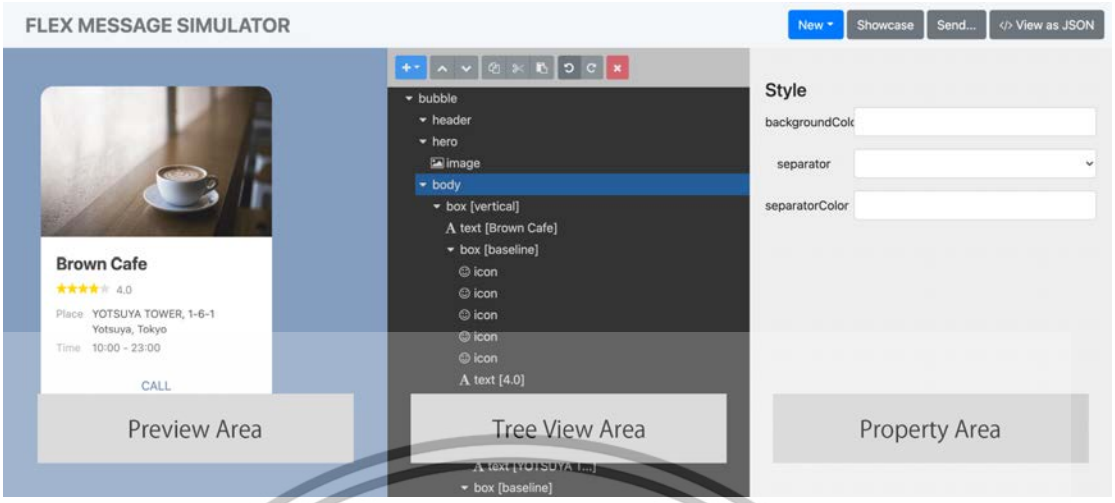



*Figure 33. Line Flex Message Simulator*

```
{
  "type": "bubble",
  "hero": {
    "type": "image",
    "url":
"https://scdn.line-apps.com/n/channel_devcenter/img/fx/01_1_cafe.png",
    "size": "full",
    "aspectRatio": "20:13",
    "aspectMode": "cover",
    "action": {
      "type": "uri",
      "uri": "http://linecorp.com/"
    }
  },
  "body": {
    "type": "box",
    "layout": "vertical",
    "contents": [
      {
        "type": "text",
        "text": "Brown Cafe",
        "weight": "bold",
        "size": "xl"
      },
...
...
...
```

*Figure 34. Example of JSON format object of the flex message*

As we finish designing our Flex Message in the Flex Message Simulator. We use **Line Messaging API** to send Flex messages to the group chat.

## 5.2 Backend Development

The backend consists of the server, database and 3rd party APIs that communicate to handle requests from the client.

### 5.2.1 Server

The server is developing by using Node JS with Express as the back-end web framework for creating RESTful APIs.

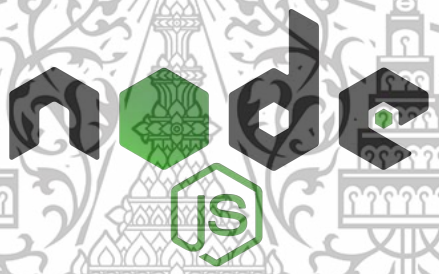


*Figure 35. nodeJS logo*

Express provided features that allow us to set up middlewares to integrate with Line bot SDK, define API routes which will be used to perform different actions based on certain HTTP methods and implement CRUD operations (GET, POST, PUT, and DELETE) on the API, which will run the corresponding database commands. The code is written in TypeScript which is a strict syntactical superset of JavaScript and adds optional static typing to the language. It is designed for the development of large applications and transpiled to JavaScript.*[9]* There are 3 main services of the server that is responsible for each feature:

1. Bill service: Responsible for handling requests regarding bill management, which includes retrieving bill details, updating bill details and deleting bills.
2. Group service: Responsible for handling requests regarding group handling, which includes retrieving group details and creating groups.

3. Promptpay-QR code service : Responsible for generating payment QR code.

#### 5.2.1.1 Bill Service

The APIs and their description provided by bill service are listed in the following table.

| Path | HTTP Method | Action |
|---|---|---|
| /bill/{bill_id} | GET | Get specific bill detail |
| /bill/{bill_id} | PUT | Update bill detail |
| /bill/{bill_id} | DELETE | Delete bill |

*Table 6. Bill Service API Table*

#### 5.2.1.2 Group Service

The APIs and their description provided by group service are listed in the following table.

| Path | HTTP Method | Action |
|---|---|---|
| /group | GET | Get all groups |
| /group/{group_id} | GET | Get group members |
| /group/{group_id}/bill | GET | Get all bills according to {group_id} |
| /group/{group_id} | POST | Create group |
| /group/{group_id}/bill | POST | Create bill for {group_id} |

*Table 7. Group Service API Table*

### 5.2.1.3 Promptpay-QR

Promptpay QR Code is the QR Code that contains the owner's bank information and can be scanned by a bank application such that the payment transaction can be made without directly typing the bank information.

The payment QR code is generated by this service. In order to generate the QR we need to understand what the string in the QR code means.

**Reading QR Code information**

Suppose the raw text from QR code is : "00020101021129370016A000000677010111011300660000000005802TH53037646 3048956" (The example above use the mobile number of : 000-000-0000) [19]

1. 00020101021129370016A000000677010111011300660000000005802TH5303764 63048956
    a. Version Number : Field 00 having length 02 which data is 01 Hence the first set of value is '000201'
2. 00020101021129370016A000000677010111011300660000000005802TH5303764 63048956
    a. QR Type : Field 01 having length 02 which data is 11 Hence the set of value is '010211'
    b. '11' means this QR payment is used in multiple transactions and if it is '12' it means for one transaction only.
3. 00020101021129370016A000000677010111011300660000000005802TH5303764 63048956
    a. Merchant account information : Field 29 having length 37 and the set of value is 0016A000000677010111011300660000000 and can be divided into 2 subfield.
    b. It is the application Id. It refers to the type of the smart cards e.g., credit cards, citizen ids, etc. Field 00 having length 16 with the value of A000000677010111

c. 0016A000000677010111011300660000000000 : Field 01 having length 13 and the value is 0066000000000 which '01' means it is a mobile number and it is 00000000

d. If the value is '02' it means the number is citizens Id.

4. 000201010211293700 16A0000006770101110113006600000000058 02TH5303764 63048956

a. Country : Field 58 having length of 02 which data is TH

5. 000201010211293700 16A0000006770101110113006600000000058 02TH5303764 63048956

a. Currency ( ISO Standard ) : Field 53 having length of 03 and the value is 764

6. 000201010211293700 16A0000006770101110113006600000000058 02TH5303764 63048956

a. Checksum value : Field 63 having 04 length and the value is 8956. The checksum value obtain by using CRC-16 standards. The value is used up to the length of the field of checksum.

```
const crc = require('crc-itu');
const value =
'000201010211293700 16A0000006770101110113006600000000058 02TH5
3037646304';
// Calculate CRC-16 checksum
const checksum = crc(Buffer.from(value,
'ascii')).toString(16);
console.log('Checksum:', checksum);
```

*Figure 36. Applying checksum in javascript*

*Figure 37. Generated Promptpay QR Code*

### 5.2.2 Database

#### 5.2.2.1 PostgreSQL


*Figure 38. PostgreSQL logo*

The relational database management system (DBMS) that we have selected is PostgreSQL. In order to automatically migrate the schema and map to models, we use PRISMA, an node.js and TypeScript Object-relational Mapping (ORM) for PostgreSQL, MySQL, SQL Server, SQLite, MongoDB, and CockroachDB. ORM allows querying and manipulating data from a database using an object-oriented paradigm, reducing the need to write. ORM will increase the development productivity, speed and flexibility, as it reduces boilerplate code, handles migration,

force to write MVC code (makes the code a little cleaner) and is applicable for many database brands.

#### 5.2.2.2 Prisma


*Figure 39. Prisma Logo*

Prisma is an open-source Object-Relational Mapping (ORM) tool that simplifies database management and data access in modern web applications. It provides a type-safe and intuitive API for interacting with databases, allowing developers to work with databases using a more structured and declarative approach.

Prisma uses a schema definition language (SDL) to describe the structure and relationships of database tables and fields. The schema defines the entities, their attributes, and the relationships between them, serving as a blueprint for the underlying database.

```
model User {
  user_id      String
  display_name String
  picture_url  String @default("")
  group        Group  @relation(fields: [group_id], references: [id])
  group_id     String

  @@unique([user_id, group_id])
}

model Bill {
  id         Int      @id @unique @default(autoincrement())
  name       String
  created_at DateTime @default(now())
  total      Float
```

```
  status      String
  owner_id    String

  monthly     Monthly? @relation(fields: [monthly_id], references: [id])
  monthly_id  Int?     @unique
  // monthly Monthly?

  payment     Payment @relation(fields: [payment_id], references: [id])
  payment_id  Int     @unique
  // payment Payment?

  debts Debt[]

  items Item[]

  group    Group  @relation(fields: [group_id], references: [id], onDelete: Cascade,
onUpdate: Cascade)
  group_id String
}
```

*Figure 40. Prisma schema Example*

We use Prisma Studio to visualize the database. Prisma studio is a graphical user interface (GUI) tool provided by Prisma for exploring and managing the database. It allows us to interact with your database tables, data, and relationships in a visual and intuitive way, making it easier to understand and work with your database schema.

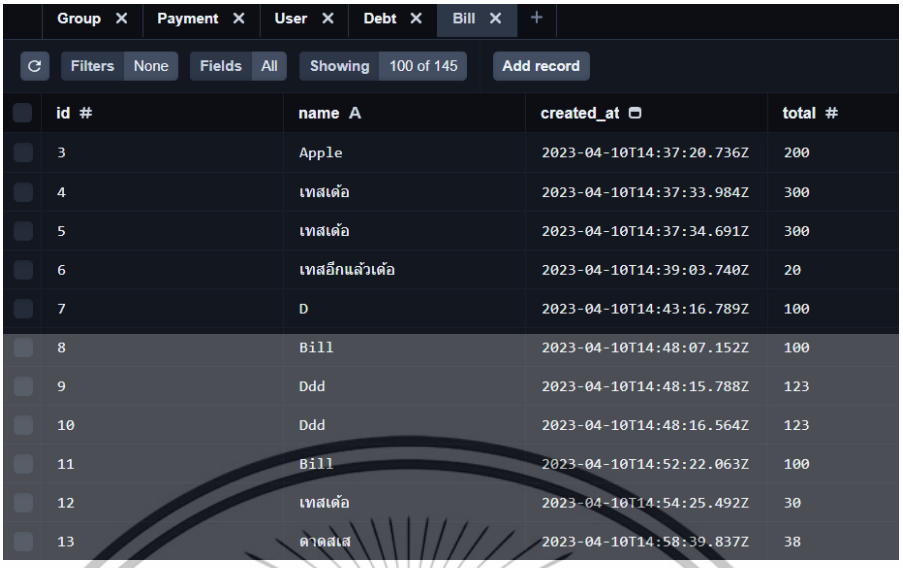



*Figure 41. Prisma studio Example*

### 5.2.2.3 Firestore


*Figure 42. Firestore logo*

We use Firestore in image hosting as it is not required to set up an image server. Firestore is a fully managed NoSQL document database provided by Google Cloud Platform. It is designed to store, sync, and query data for web, mobile, and server applications. Firestore offers a flexible and scalable solution for managing structured data, making it a popular choice for building real-time applications, collaborative platforms, and mobile apps.

## 5.3 Line Bot

Line Bot is a part of Line Official Account that uses Line Messaging API and the Bot SDK to develop and enhance the functionality of Line Official Account to always operate on their own without actual humans operating on it.
We use Line Bot to send messages as it is integrated with the Line ***Messaging API.***

### 5.3.1 Interaction with Line Bot

Line Bot operates by capturing the messages in the chatroom and interacting with the application via. Webhook events. We have to set up our webhook url in the Line developer console.

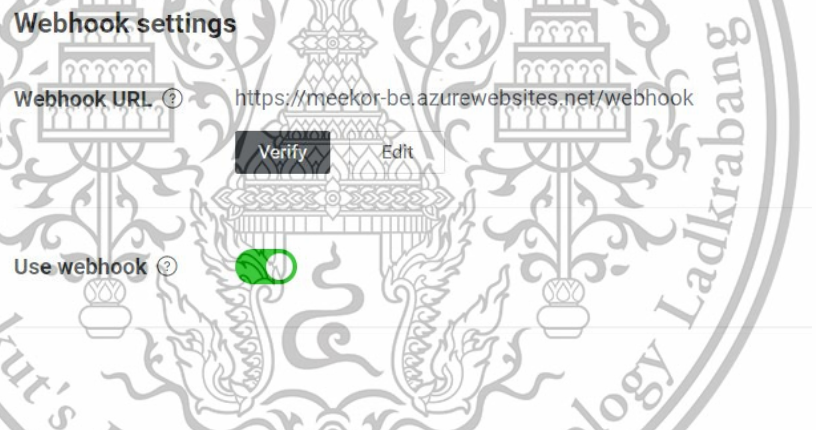



*Figure 43. Line developer console example*

When the webhook are received, the bot will interact according to the messages in these formats

| Command | Action |
|---|---|

| | |
|---|---|
| หมีขอ | Display Menu |
| #สมัครสมาชิก | Register to the application |
| #ช่วยเหลือ | Send infographic about the application |
| #สร้างบิล | Send create bill flex message |
| #ดูบิลทั้งหมด | Send view all bill message |
| #ทวงหน่อย | Send reminder message |

*Table 8. Command list for Meekor bot*

### 5.3.2 Sending Flex Messages

After receiving a webhook event object, the bot can use replyToken to send back the Flex message immediately to the correct chatroom. User can then interacts with the button inside the flex message to open our LIFF Application




```
"events": [
  {
    "type": "message",
    "message": {
      "type": "text",
      "id": "14353798921116",
      "text": "Hello, world"
    },
    "timestamp": 1625665242211,
    "source": {
      "type": "user",
      "userId": "U80696558e1aa831..."
    },
    "replyToken": "757913772c4646b784d4b7ce46d12671",
    "mode": "active",
    "webhookEventId": "01FZ74A0TDDPYRVKNK77XKC3ZR",
    "deliveryContext": {
      "isRedelivery": false
```


*Figure 44. Webhook event object example*

```
if (event.message.text == "#ช่วยเหลือ"){
  return client.replyMessage(event.replyToken, {
    "type": "image",
    "originalContentUrl": "https://i.ibb.co/Q9Hjkg2/image.jpg",
    "previewImageUrl": "https://i.ibb.co/Q9Hjkg2/image.jpg"
  })
}
```

*Figure 45. Sending flex message example*

From the Fig. [45] we are sending the infographic on our application by using the command client.replyMessage(event.replyToken,{<Flex Message>})

### 5.3.3 Line Messaging API

Line Messaging API acts like the middle that connects a Server to a Line Official Account so it is possible to write code to make services that developers want by message and conversation with users in a kind of Chatbot. The Messaging API allows for data to be passed between your bot server and the Line Platform. Requests are sent over HTTPS in JSON format. The steps of how it working are as follows:


*Figure 46. Line messaging API*

1. The user sends a message to the Line Official Account.
2. The Line Platform sends a webhook event to the webhook URL of the bot server.
3. According to the webhook event, the bot server responds to the user through the Line Platform.

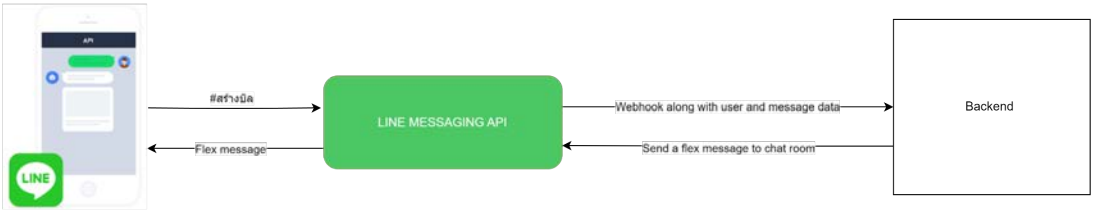



*Figure 47. Receiving Webhook event*

## 5.3 Infrastructure

### 5.3.1 Github Action


*Figure 48. Pipeline Example*

GitHub Actions is a workflow automation and continuous integration/continuous deployment (CI/CD) platform provided by GitHub. It allows us to automate various tasks and processes within your software development workflow. We decided to use Github Action as our project is using Github as a version control.

Automating Continuous Integration/Continuous Deployment (CI/CD) processes through GitHub Actions brings numerous advantages to software development and deployment workflows. Firstly, using CI/CD tools enhances productivity and efficiency by automating repetitive tasks. We can focus on coding and feature development rather than spending time on manual build and deployment activities.

This automation reduces human error and accelerates the software development lifecycle.

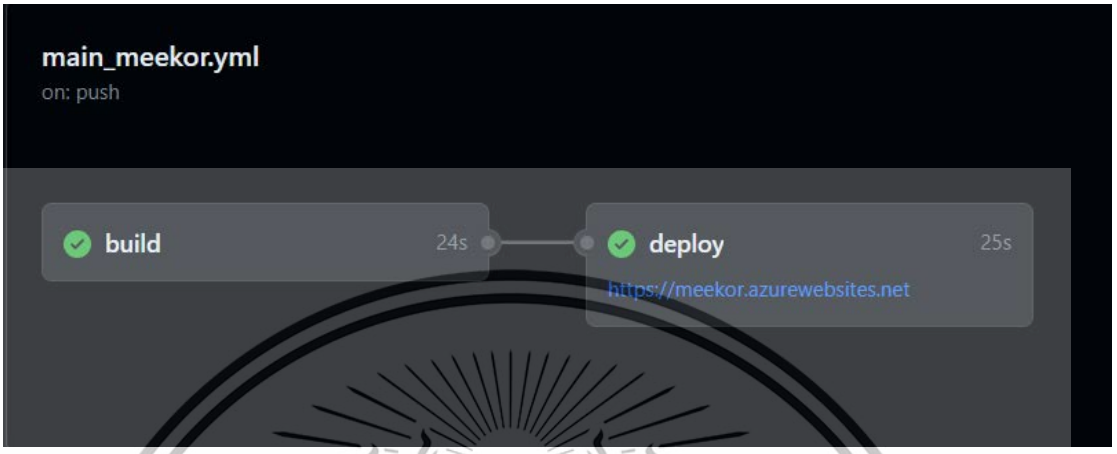



*Figure 49. Build and deploy pipeline*

The process show in workflow in the figure above is involved in two jobs : build and deploy

- Build : This is included in the Setting up Node.js version and run npm install to add the required node packages.
- Deploy : This includes downloading artifacts from build jobs and then deploying to Azure Web App.

The above workflow will trigger when we push the code into 'Production' branch in our GitHub version control system.

### 5.3.2 Docker container

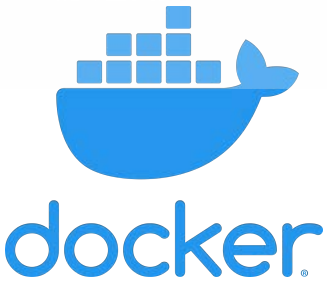



*Figure 50. Docker logo*

Dockerfile and Docker-compose.A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings [20]

Docker-compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. Compose works in all environments: production, staging, development, testing, as well as CI workflows. [21]

### 5.3.3 Stackblitz

*Figure 51. Stackblitz logo*

StackBlitz is an instant full stack web IDE for the JavaScript ecosystem. It's powered by WebContainers, the first WebAssembly-based operating system which boots Node.js environment in milliseconds, securely within your browser tab. It provides you with the use of the web to build the web.

We decided to go with Stackblitz as Line has to bind the static web url in order to use the LIFF application, this problem arises when collabing together as it has to always change the static link.

### 5.3.4 Ngrok


*Figure 52. ngrok logo*

For development purposes we use Ngrok to expose our backend server to the internet. Ngrok is a globally distributed reverse proxy fronting your web services running in any cloud or private network, or your machine. [22]

### 5.3.5 Microsoft Azure cloud


*Figure 53. Azure cloud logo*

Azure Cloud, also known as Microsoft Azure, is a comprehensive cloud computing platform offered by Microsoft. It provides a wide range of cloud services and solutions, including infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). We use Microsoft Azure cloud to host our frontend and backend server.

# Chapter 6

# Result

## 6.1 Frontend

This section will discuss the frontend aspects of Meekor, specifically the development of a web application implemented using LIFF (Line Frontend Framework) and the design of Flex Messages within the Line platform. LIFF enables Meekor to offer users a consistent interface directly within the Line app. The Flex Messages, designed to align with Meekor's theme and user interface, effectively present information such as bill details, summaries, and payment information.

The web application has been under development throughout both semesters. In the first semester, the focus was on building the web application, while in the second semester, efforts were directed towards improving its theme and user experience. The development and design of Flex Messages took place in the second semester, once most of the basic features had been implemented.

The work will be discussed by feature, showcasing both the web frontend and Flex Message design for each feature with use case. The features to be covered are as follows:

- Subscribe: Enables users to subscribe to Meekor's services.
- Create Bill: Allows users to create new bills.
- Create Equal Bill: Creates bills with equal amounts to be split among multiple users.
- Create Bill with Item List: Creates bills with detailed item lists.
- Payment: Facilitates bill payments.
- View all bills: View all bills and the details
- Confirm: Confirms payments or transactions.
- Notification: Sends notifications to users regarding bill payment reminders.

### 6.1.2 Subscribe

1. [New User] add Meekor as a friend in line and subscribe to Meekor so that I can create bills and be included in other people's bills.
    - Subscribe - If a user is already friends with Meekor, they can click on 'subscribe.' The system will save the user's data into the database and send a successful subscription message to the Line group. If the user is already subscribed, the system will send an 'already subscribed' message to the Line group.
    - Guiding - There are well instruction for new user. If a user subscribes before adding Meekor as a friend, the bot will send a message instructing them to add Meekor as a friend first.
    - Interface design - The interface is designed to be simple and informative, with a color theme and logo that help users recognize it more easily. The button helps reduce unnecessary or confused action leads to improve user experience.


*Figure 54. Registration message*

### 6.1.3 Create equal bill

1. [User] Call Meekor on Line group and select 'Create Equal Bill' to create a bill that will be automatically divided by the system.
2. [Owner] Adding bill name for distinguish bills in the group and serves as a brief description or label for the bill's purpose or contents.
3. [Owner] Adding amount and the amount in the bill and adjust specific amount of each loaner
    - Automated partitioning - The system will automatically divide the amount among the bill members and display it in real-time.
4. [Owner] Checking member checkbox so that the owner can add a divider of the bill.
5. [Owner]Providing the loaner with a payment method for bill payment, two options are available: Bank Transfer and QR code.
6. [Owner] Confirming bill so that the system will save the bill in the database and send the bill summary flex message into the Line group.
    - The bill summary flex message - includes the bill name, owner, loaners, and the respective amounts owed by each loaner. The flex message also contains two buttons for payment and viewing the bill.

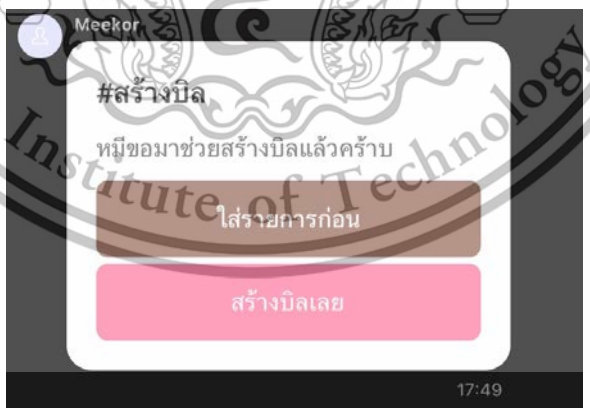



*Figure 55. Create bill message*

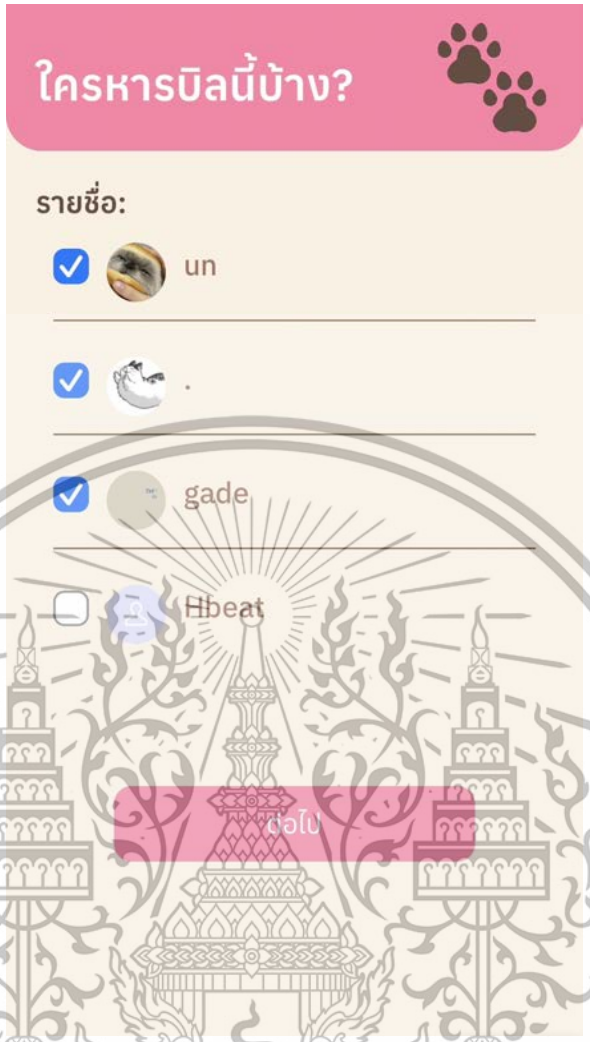



*Figure 56. Create equal bill page*


*Figure 57. Create equal bill page (2)*

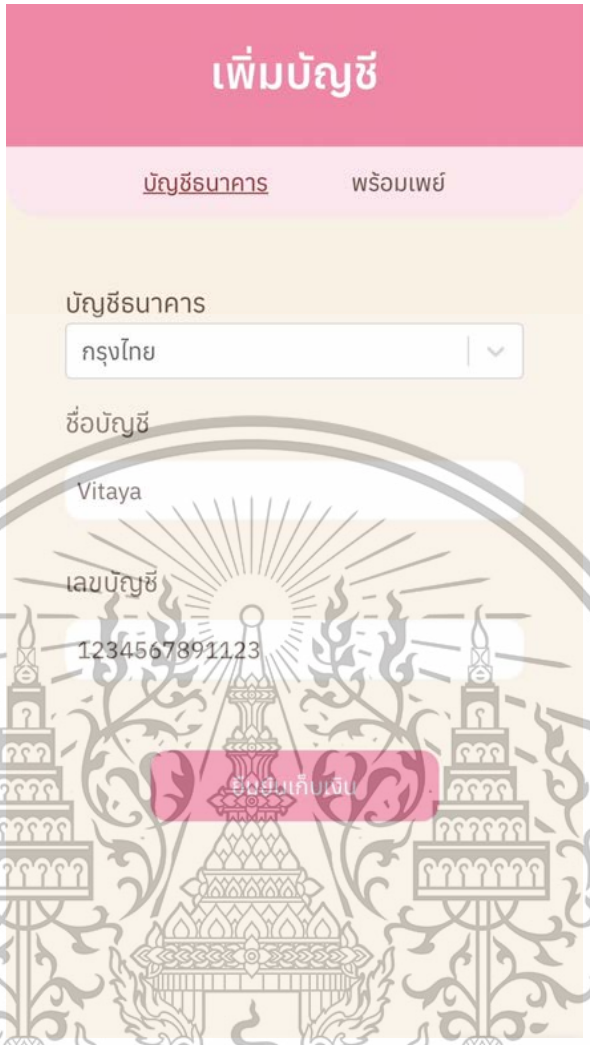



*Figure 58. Add account page*


*Figure 59. Add Promptpay account page*


*Figure 60. Created bill message*

### 6.1.4 Create bill with item list

1. [User] Call Meekor on Line and select 'Create Bill with item list' to create a bill that have option for adding items first.
2. [Owner] Adding bill name for distinguish bills in the group and serves as a brief description or label for the bill's purpose or contents.
3. [Owner] Add items and their respective prices to the bill. This will provide the necessary data for accurate price calculations.
4. [Owner] Assign members to each item, Owner can select or pick the respective members for each item on the bill
    - Debt management system - This system is responsible for computing the total price of the bill and determining the specific amounts owed by each individual loaner.
5. [Owner]Providing the loaner with a payment method for bill payment, two options are available: Bank Transfer and QR code.
6. [Owner] Confirming bill so that the system will save bill in database and send the bill summary flex massage into the Line group.
    - The bill summary flex message - includes the bill name, owner, loaners, and the respective amounts owed by each loaner. The flex message also contains two buttons for payment and viewing the bill.

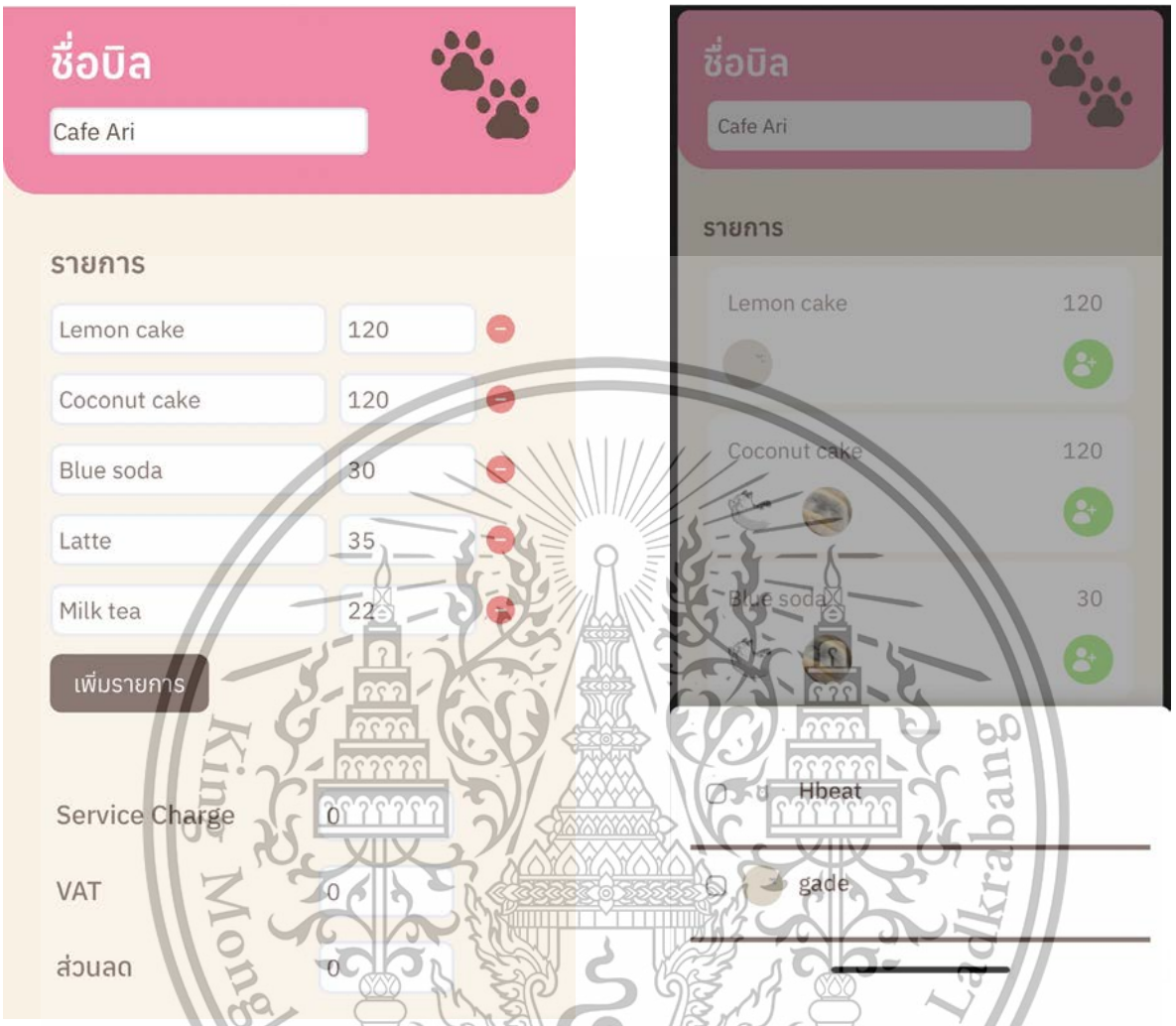



*Figure 61. Create separate bill page*

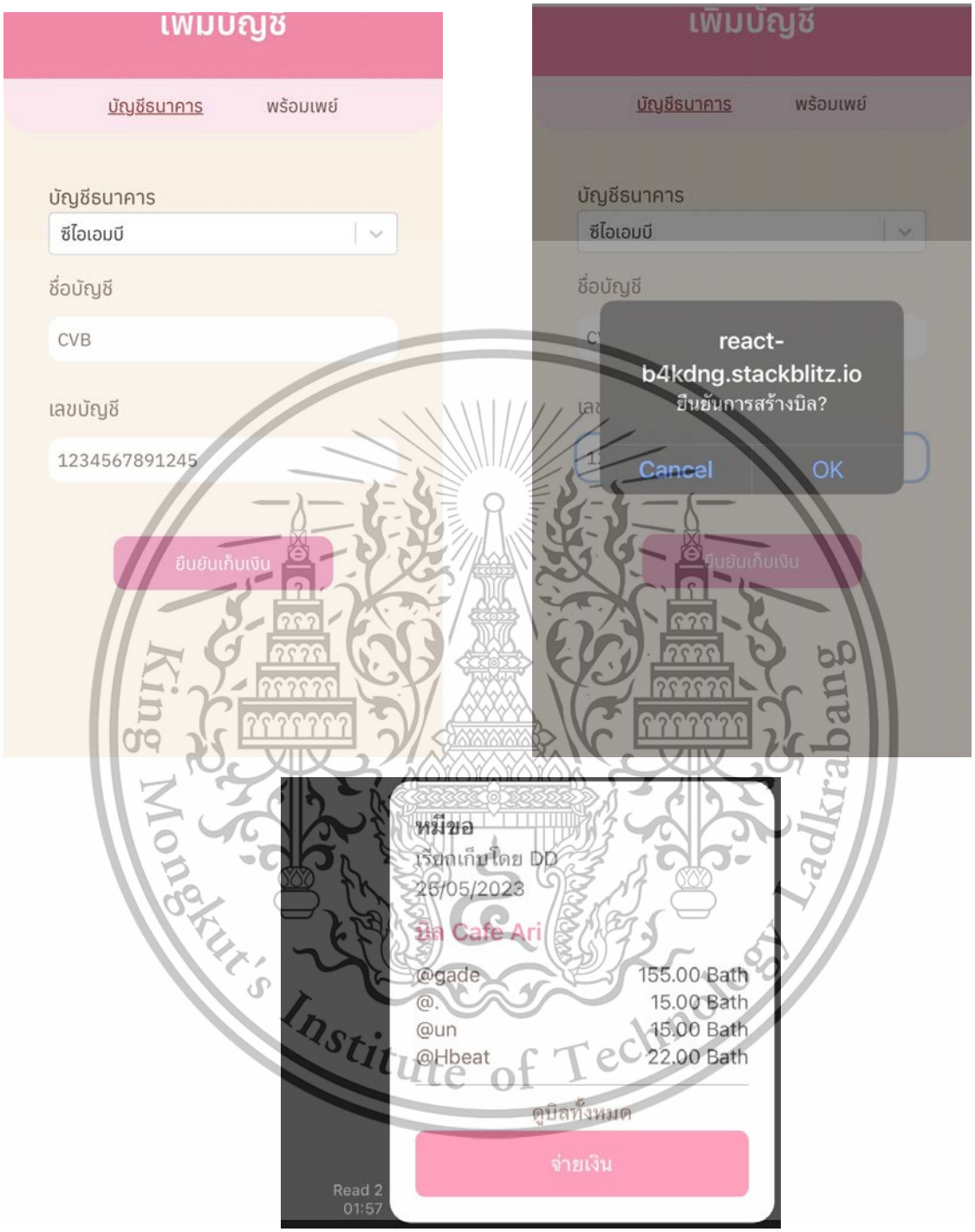



*Figure 62. Create separate bill page(2)*

### 6.1.5 View all bills

1. [User] View all bills within a group, categorized by month. This feature enhances the organization and accessibility of bill management within the application.
    - The simple and organize design of each bill by month provide a user-friendly and efficient interface for users to manage their bills and track their financial progress within group settings
2. [Owner] View only own bill. The owner will only see the bills that they have created and easier to focus on.
3. [Owner] Delete bill functionality enables users to remove unwanted or no longer relevant bills from their records. By accessing the bill section and selecting the specific bill to be deleted, users can easily initiate the deletion process. This feature provides users with control over their bill history, allowing them to keep their records updated
4. [User] View only the bills they need to pay. This allows users to focus on and prioritize their payments when dealing with multiple bills, making it easier to identify their specific payment obligations.

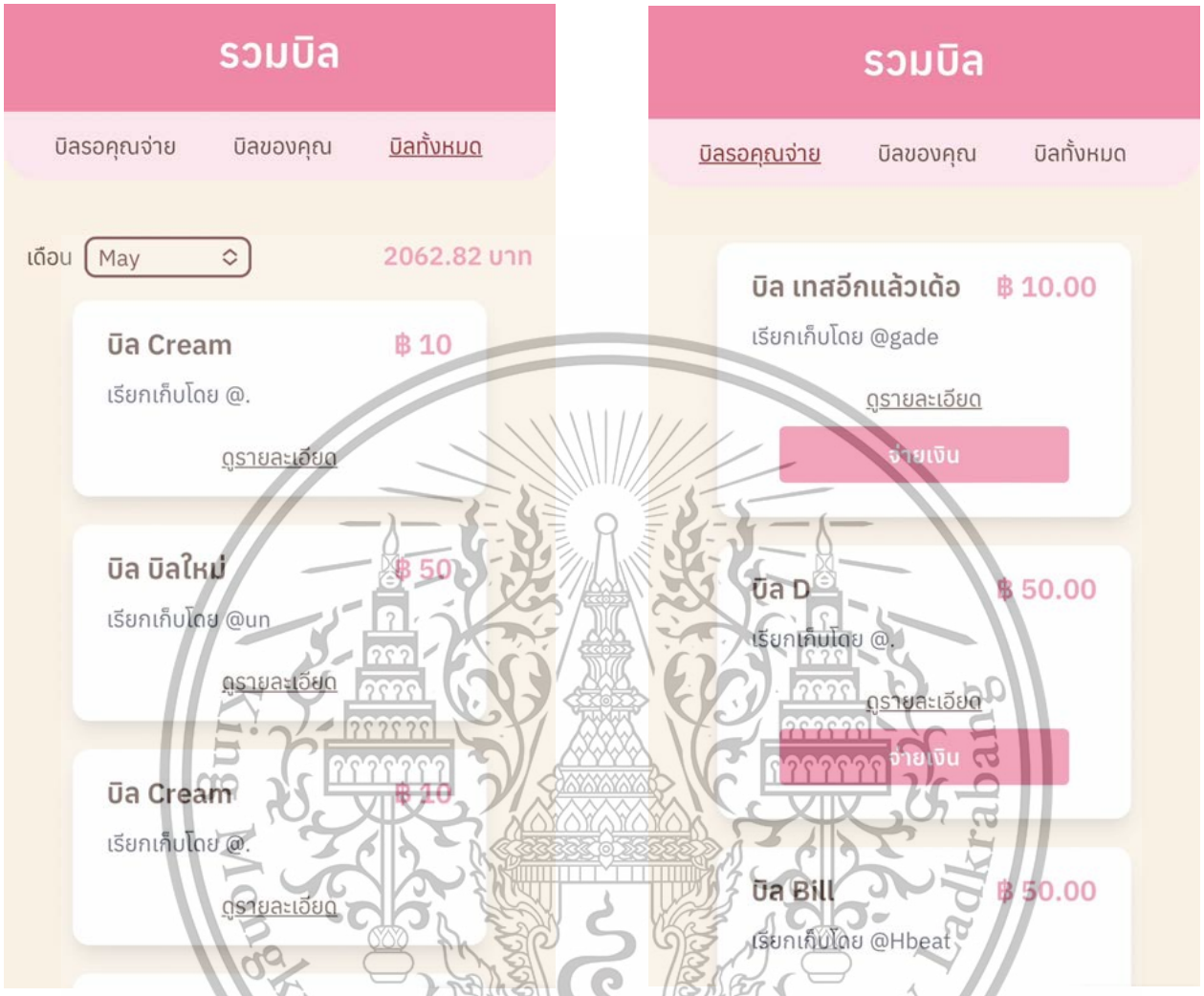



*Figure 63. View all bills pages*


*Figure 64. View your bill page*

### 6.1.6 Payment

1. [User] View bill summary flex and able to press paying button
2. [User] View the payment information according to the owner payment method so that the loaner can pay the bill. The basic information include:
    - Bill Name - A descriptive name or title assigned to the bill for easy identification and reference.
    - Debt Amount: The total amount owed or payable for the bill.
    - QR Code - A Quick Response (QR) code that contains encoded information related to the bill, such as payment details or a reference number. The QR code can be scanned using a mobile device for convenient bill payment.
    - Name - The name of the owner account.
    - Payment Method Details:
        - PromptPay - In the case of PromptPay, it refers to the registered mobile phone number or national identification number associated with the PromptPay account.
        - Account Number- For traditional bank transfers or direct deposits, the account number to which the payment should be made is provided.
3. [Loaner] Copy payment method with button, the loaner can easily copy the payment method number to facilitate their payment process
4. [Loaner] Send payment confirmation request by pressing the button.The first option is for bank transfer and uploading the payment slip. The second option is for cash payment. The system will send payment confirmation requests and flex messages into the group.
5. [Loaner] Uploading the slip picture and pressing the confirm button, the slip picture will be saved to the database.

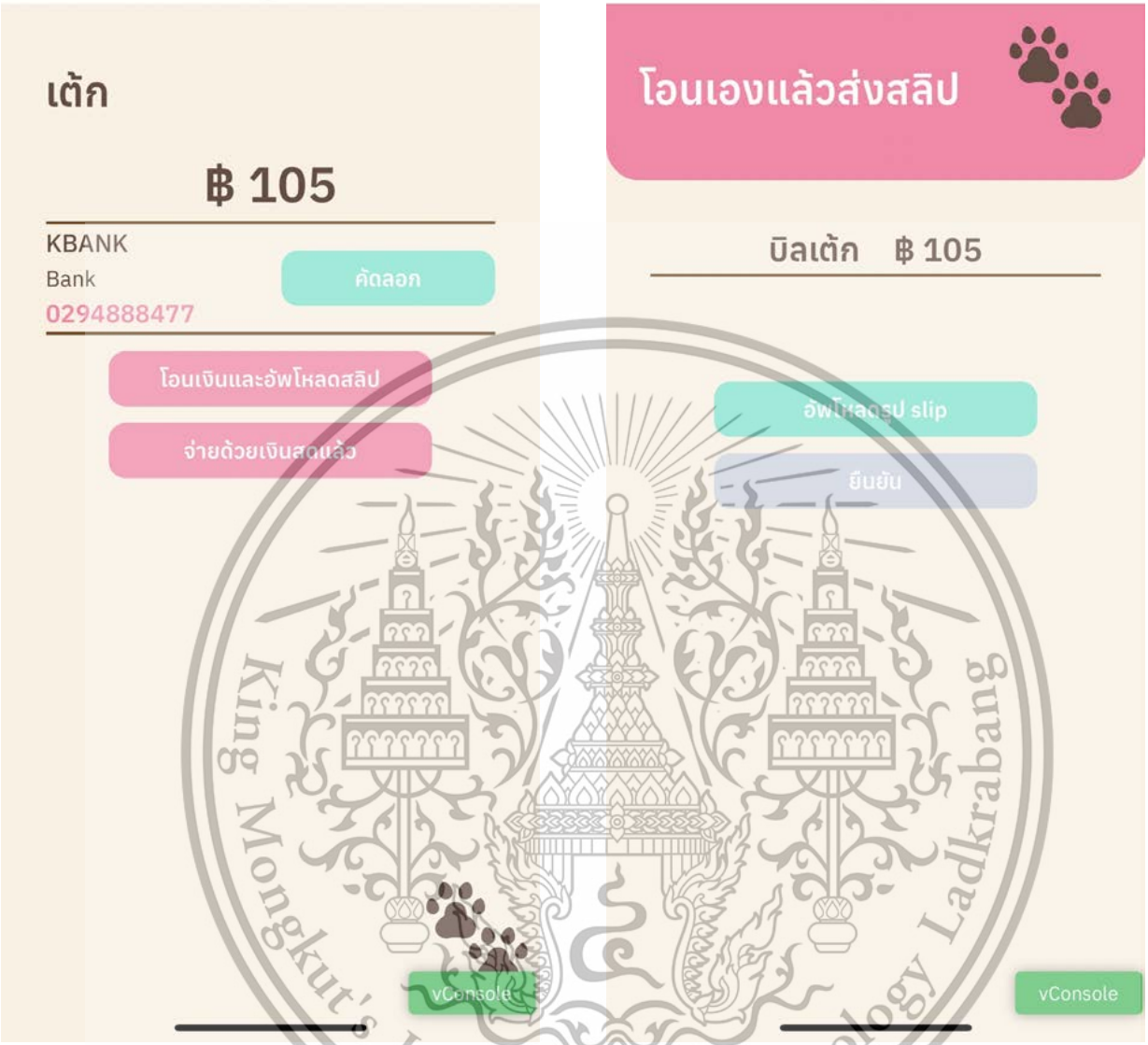



*Figure 65. Pay bill page*

### 6.1.7 Confirm

1. [User] View the real-time payment status of their bills. This functionality allows users to easily track and monitor the bill payments, providing them with real-time information about the status of their payment confirmation request.
2. [Owner] Verify the authenticity and accuracy of the slip, the owner can view the payment slip in the system's interface. The interface will provide information such as the name and amount, allowing the owner to ensure that the slip corresponds correctly.
3. [Owner] Confirming the payment, the debt status will be updated to "closed," and a successful confirmation message will be sent to the Line group.
4. [Owner] Not confirming the payment, the debt status will be updated to “opened” again and a not confirmed message will be sent to the Line group which will provide an instruction to send payment confirmation.

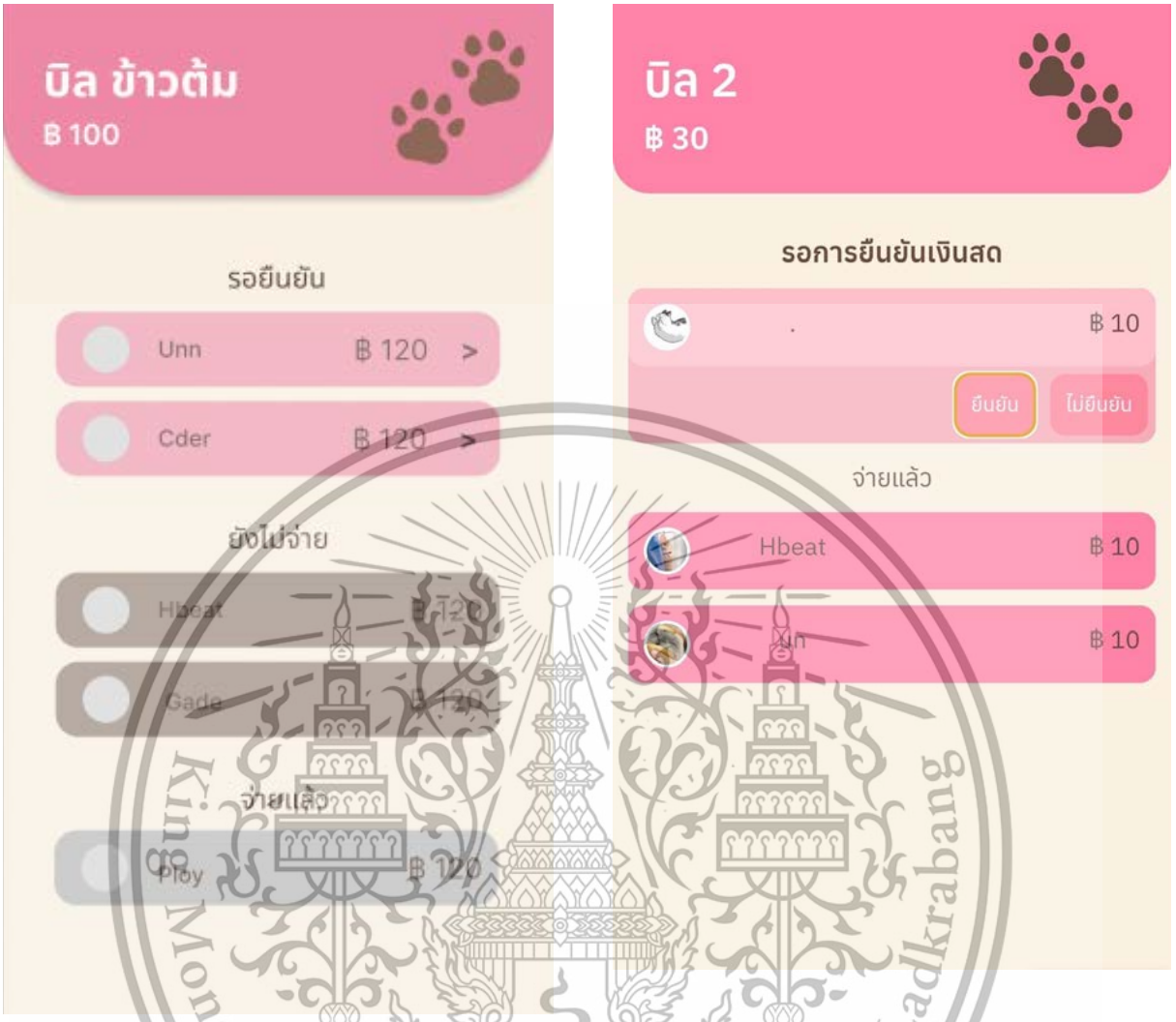



*Figure 66. Confirm paid bill pages*


*Figure 67. Confirm paid bill page(2)*

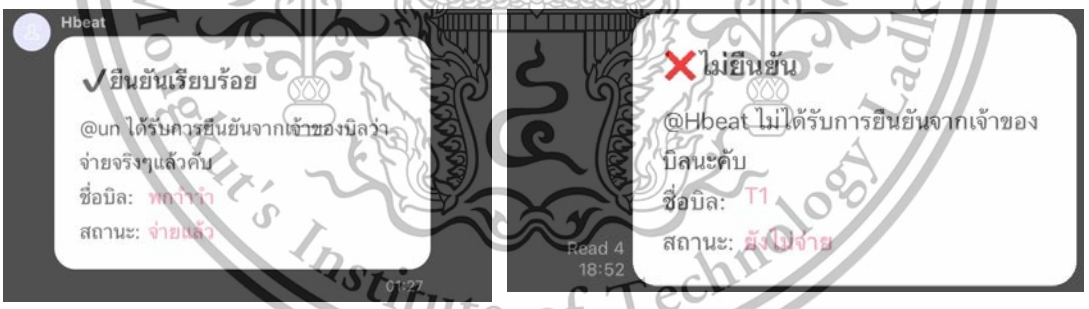



*Figure 68. Confirm message in Line group chat*


*Figure 69. Bill is fully paid message*

### 6.1.8 Notification

1. [Owner] Sending reminder messages through the Line group, ensuring effective communication with borrowers. This feature allows loaners to stay updated on their borrowers' payment status and take necessary actions.


*Figure 70. Notification message*

### 6.1.9 Help

1. [User] View information on how to use Meekor effectively to maximize its benefits and features.

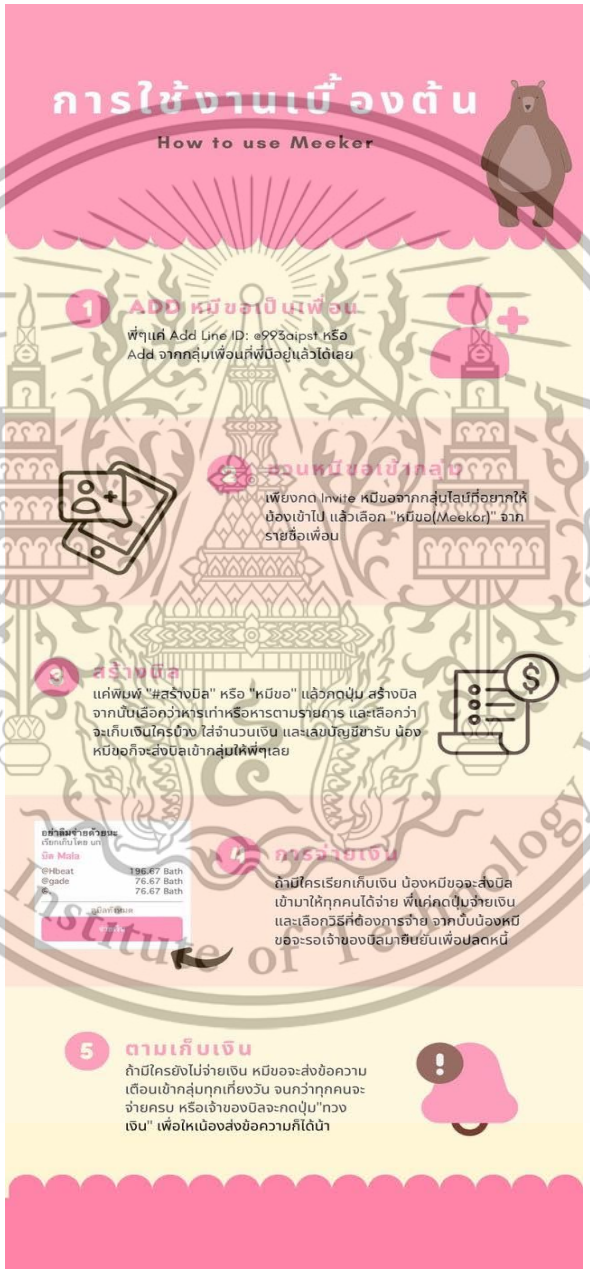



*Figure 71. User manual*

## 6.2 Backend

### 6.2.1 Server

The server development for Meekor is centered around two main technologies: Node.js and Express. These tools help create the backend of the Meekor application. Node.js is a powerful runtime environment that allows JavaScript code to run on the server-side. This means that it can handle the server-side logic and processing required for the application. It works well with the front-end components of the application.

### 6.2.2 API services

Through the Bill Service API, users can retrieve, update, and delete specific bill details, ensuring organization and control over their debts. The Group Service API facilitates group-related functionalities, enabling users to access group information, get associated bills and debts. Additionally, Meekor's Promptpay-QR service generates payment QR codes containing bank information, convenient payment transactions for loaners. Meekor aims to enhance user experience, collaboration, and financial organizing by providing comprehensive features and services within the application.

### 6.2.3 Database

Meekor uses a PostgreSQL database as its main system for managing relational data. To make it easier to handle database changes and manipulate data, Meekor incorporates Prisma, a tool that works with Node.js and TypeScript to map objects to the database (ORM). Prisma provides a user-friendly and type-safe API, reducing repetitive code and improving interactions with the database. It also includes Prisma Studio, a visual interface that helps visualize and interact with the database schema. For efficient image hosting, Meekor relies on Firestore, a scalable NoSQL document database provided by Google Cloud Platform. Firestore allows real-time management of data and seamlessly meets the dynamic needs of Meekor's image hosting feature.

# Chapter 7
# Conclusion

**Meekor** is a Line chatbot which provides a service to collect the debt from a group of two or more people. Meekor also provides users payment channels and payment verification for users convenience. The name **Meekor** came from the two words in Thai which are **"Mee" (หมี)** and **"Kor" (ขอ)** Which means Bear and Asking. The bear can be sometimes viewed as cute and trembling at the same time. When the two words come together it means the bear is asking. The user might feel both responsibility to pay the debt on due.

**Meekor** operates on Line platform which is the most widely used communication platform for Thai people. Thai people usually specify their debts into the chat group manually. Meekor will come in to automate this process as it helps the people to keep track of their loan and debt more easily.

This project is still open for a new ideas and board selection for improvements, given that there are enough resources for further development :

- Provides automatic verification by bank.
- OCR for the original payment slip.
- Integrate the project with a more powerful hosting platform and better project management tools.
- Expanding the user scope from the people of Thailand to other foreign countries.

With the new ideas and improvement mentioned above, it would make Meekor become an impactful application to more people of Thailand. We hope that more people will gain knowledge and benefit from our work.

# Bibliography


**[1]** *Potajan, P. (2015, December 4). Go Dutch Calc Free. Retrieved May 26, 2023, from App Store website: https://apps.apple.com/us/app/go-dutch-calc-free/id1062846697*

***[2]*** *ขุนทอง - เพื่อนรู้ใจในการเรียกเก็บเงิน. (n.d.). Retrieved May 27, 2023, from ธนาคารกสิกรไทย website: https://www.kasikornbank.com/th/personal/Digital-banking/Pages/khun-thong.aspx*

**[3]** *CheckBill. (n.d.). Retrieved Dec 14, 2022, from https://jabont.com/checkbill/*

**[4]** *Splitwise. (2011, August 24). Splitwise. Retrieved May 26, 2023, from App Store website: https://apps.apple.com/us/app/splitwise/id458023433*

**[5]** *Jersch, N. (2016, May 14). Splid – Split group bills. Retrieved May 26, 2023, from App Store website: https://apps.apple.com/us/app/splid-split-group-bills/id991473495*

***[6]*** *bring10. (2013, February 17). Tab - The simple bill splitter. Retrieved May 27, 2023, from App Store website: https://apps.apple.com/us/app/tab-the-simple-bill-splitter/id595068606*

***[7]*** *What is a REST API? (n.d.). Retrieved May 26, 2023, from IBM website: https://www.ibm.com/topics/rest-apis*

***[8]*** *MyConsultAdmin. (2020, April 3). Introduction to database management systems (DBMS). Retrieved May 26, 2023, from Breitech website: https://breitech.co.za/introduction-to-database-management-systems-dbms/*

***[9]*** *JPA. (n.d.). Retrieved May 26, 2023, from ORM Components website: https://www.tutorialspoint.com/jpa/jpa_orm_components.htm*

***[10]*** *Fulton, J. (2018, December 12). Web architecture 101 - The storyblocks tech blog - Medium. The Storyblocks Tech Blog. Retrieved from*

*https://medium.com/storyblocks-engineering/web-architecture-101-a3224e126947*

***[11]*** *Editor. (2021, March 11). Understanding object-relational mapping: Pros, cons, and types. Retrieved May 26, 2023, from AltexSoft website: https://www.altexsoft.com/blog/object-relational-mapping/*

***[12]*** *CRUD - Definition & overview. (2019, October 9). Retrieved May 26, 2023, from Sumo Logic website: https://www.sumologic.com/glossary/crud/*

***[13]*** *Containerization explained. (n.d.). Retrieved May 26, 2023, from IBM website: https://www.ibm.com/topics/containerization*

***[14]*** *R, M. (2022, May 10). Color psychology. UX Planet. Retrieved from https://uxplanet.org/colour-psychology-to-empower-and-inspire-you-3424dae70f2*

***[15]*** *Casey. (2020, March 17). What is Miro and How Can You Use it For Collaboration? Retrieved May 26, 2023, from Innovation Training | Design Thinking Workshops website: https://www.innovationtraining.org/what-is-miro-and-how-to-use-miro-for-collaboration/*

***[16]*** *Kopf, B. (2018, July 31). The power of figma as a design tool. Toptal. Retrieved from https://www.toptal.com/designers/ui/figma-design-tool*

***[17]*** *Chapman, C. (2018, March 27). Exploring the gestalt principles of design. Toptal. Retrieved from https://www.toptal.com/designers/ui/gestalt-principles-of-design*

***[18]*** *Cherry, K. (2006, March 15). What are the gestalt principles? Verywell Mind. Retrieved from https://www.verywellmind.com/gestalt-laws-of-perceptual-organization-2795835*

***[19]*** *มีอะไรอยู่ใน PromptPay QR แกะสเปค QR ที่จะใช้จ่ายผ่าน mobile banking ได้ทุกธนาคารในอนาคต. (n.d.). Retrieved May 26, 2023, from Blognone website: https://www.blognone.com/node/95133*

**[20]** *What is a Container? (2021, November 11). Retrieved May 27, 2023, from Docker website: https://www.docker.com/resources/what-container/*

**[21]** *Docker Compose overview. (2023, May 26). Retrieved May 27, 2023, from Docker Documentation website: https://docs.docker.com/compose/*

***[22]*** *ngrok. (n.d.). ngrok. Retrieved May 27, 2023, from Online in One Line website: https://ngrok.com/*