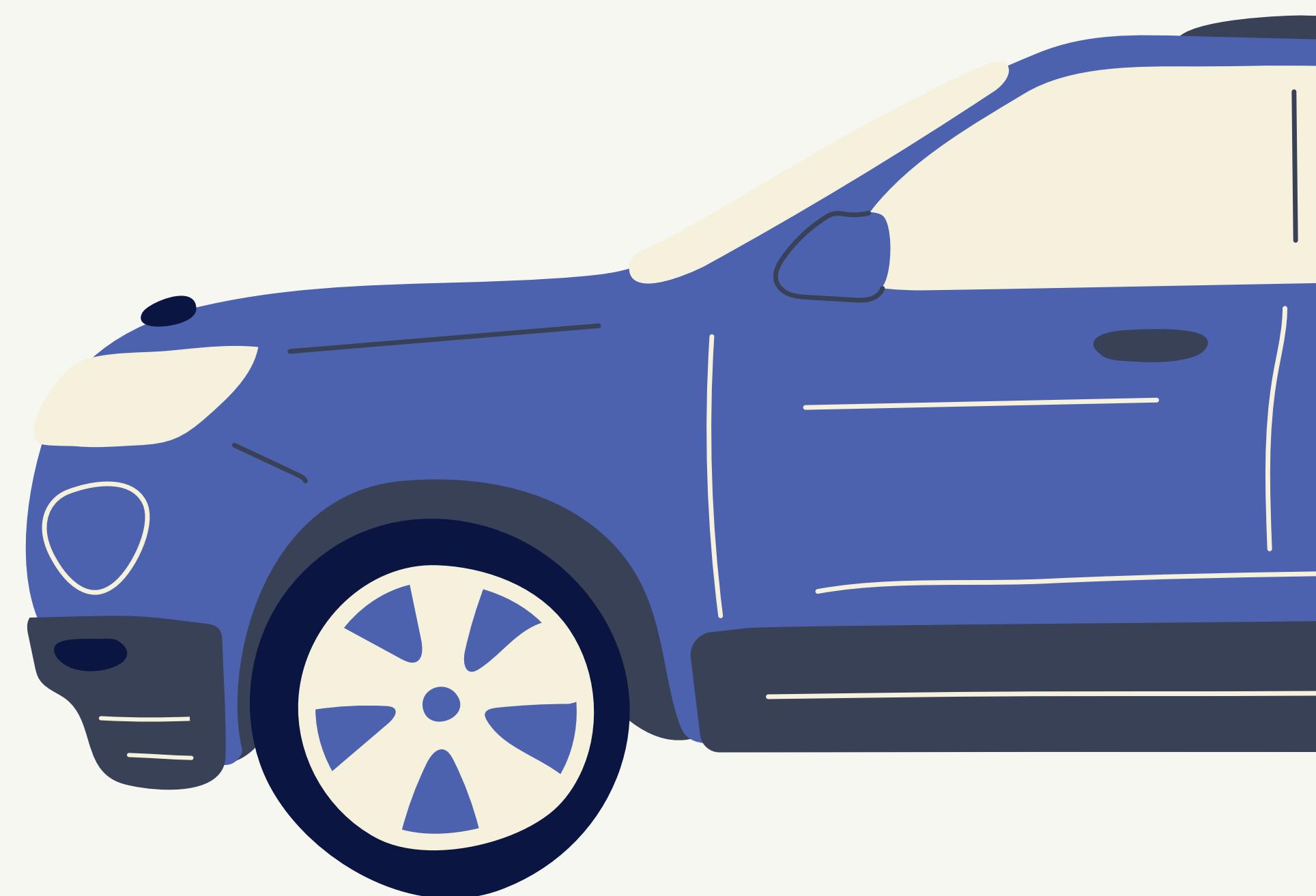


AN APPLICATION FOR HOME-TO-PUBLIC PARKING

แอปพลิเคชันสำหรับการเปลี่ยนบ้านเป็น
ที่จอดรถสาธารณะ

CE66-14 Computer Engineering
King Mongkut's Institute of Technology Ladkrabang



ADVISOR



Asst. Prof. Dr. Chutimet Srinilta

Department of Computer Engineering, KMITL

MEMBERS



Kritsanaphat Phanjaroen
63010040



Kueakun Niyomsit
63010095

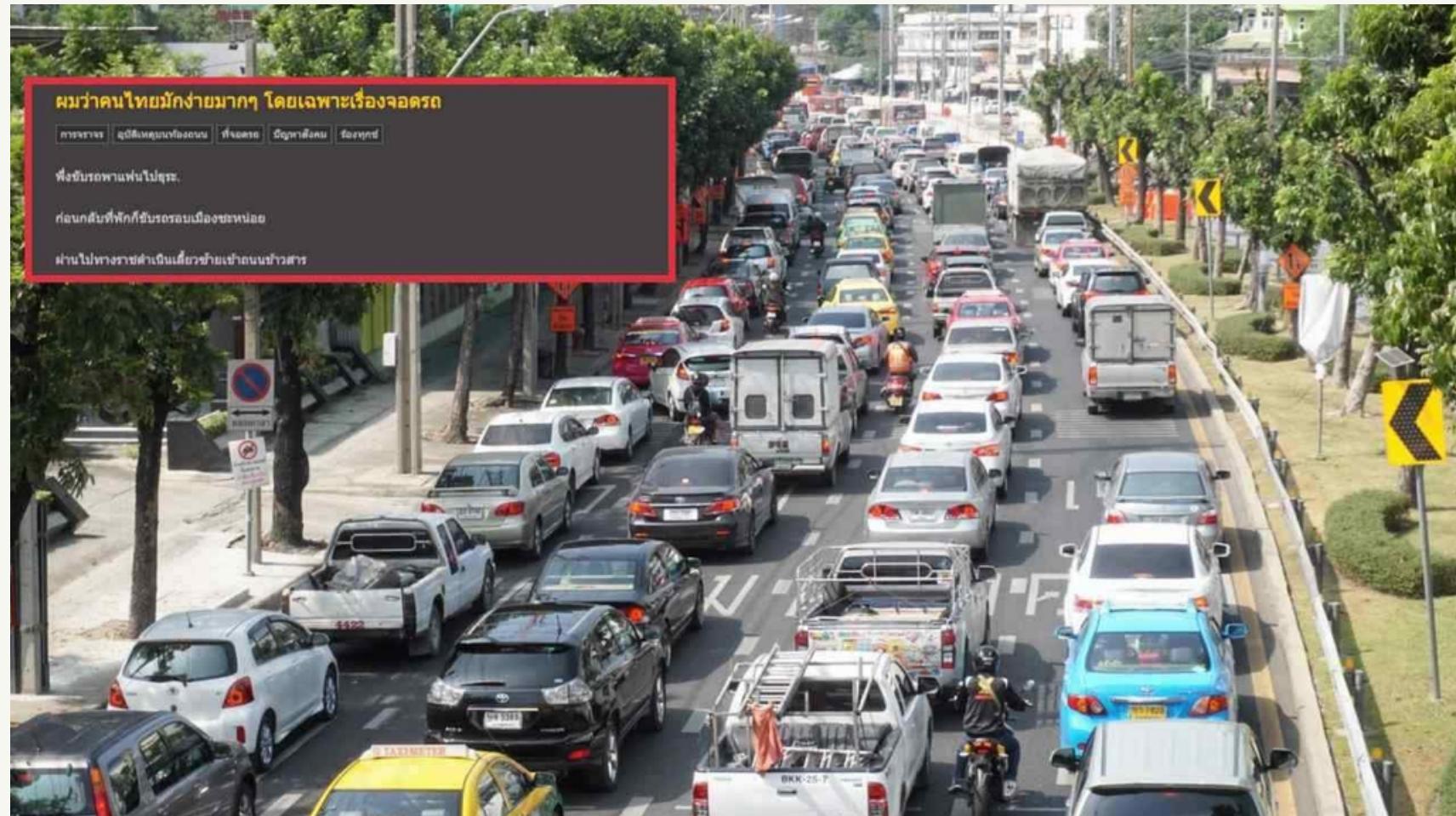


Watcharapol Yotadee
63010870



Introduction

Introduction



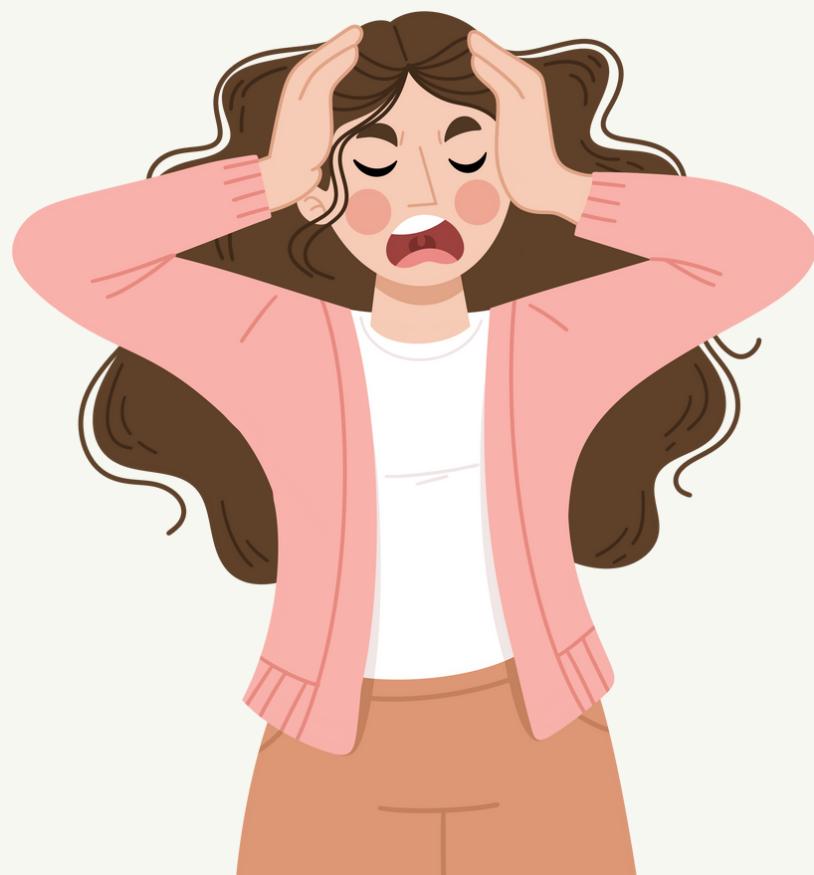
ไปเยาวราช ตอนนี้จอดรถได้ตรงไหนรอค่า

เยาวราช ถนนเยาวราช ตลาดสำเพ็ง การจราจร ที่จอดรถ

ระยะหลัง หาที่จอดรถเยาวราชไม่ได้เลยนะค่ะ ขอบคุณค่ะ

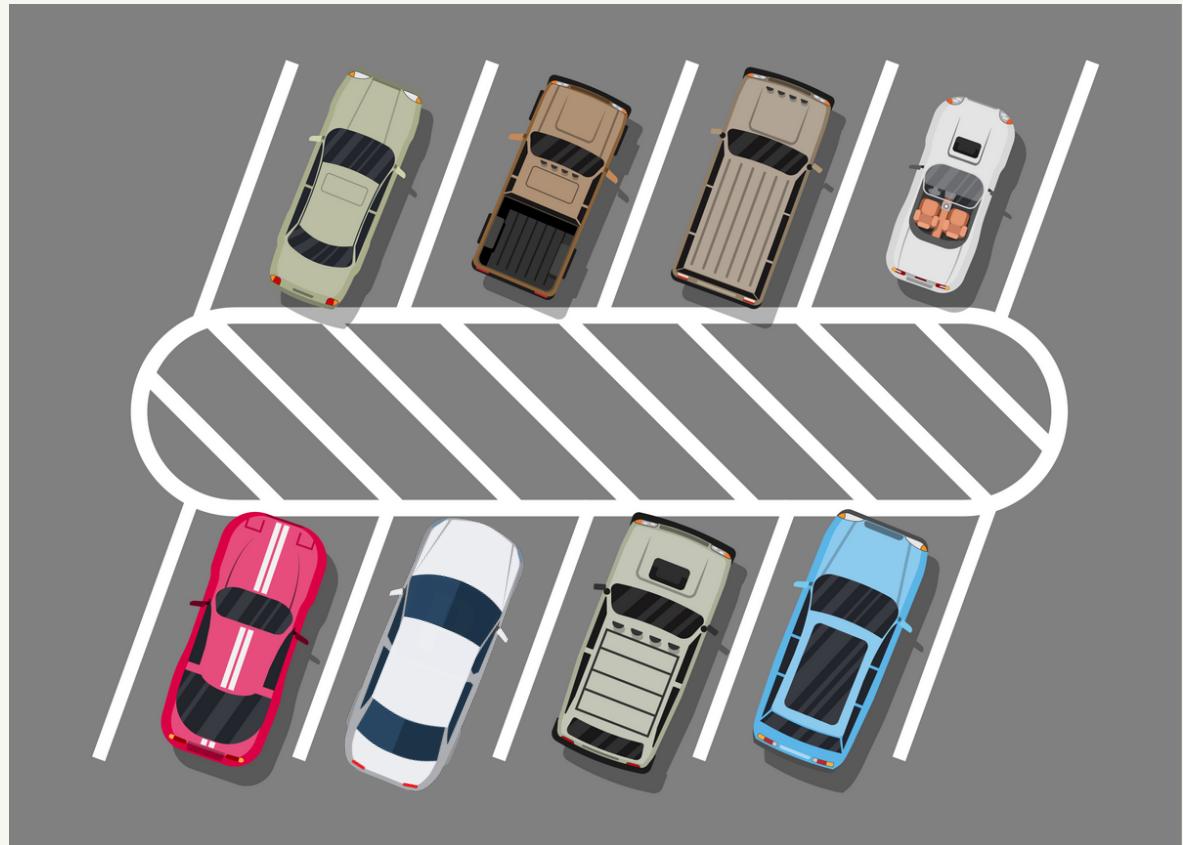
0 + 0 | สมาชิกหมายเลข 6382000
23 ธันวาคม 2564 เวลา 15:02 น.

Problem



จอดไหนดีไม่มีที่
ว่างเลย

มีที่จอดที่ไหน
บ้าง ไม่รู้จักเลย



ประเด็น



เดี่ยวไปเจอกันที่
ห้างเดิมนะ ครับ







ນໍາມັນ



ไม่รู้จะจอดไหนดี

ไม่เคยมาชิดด้วย

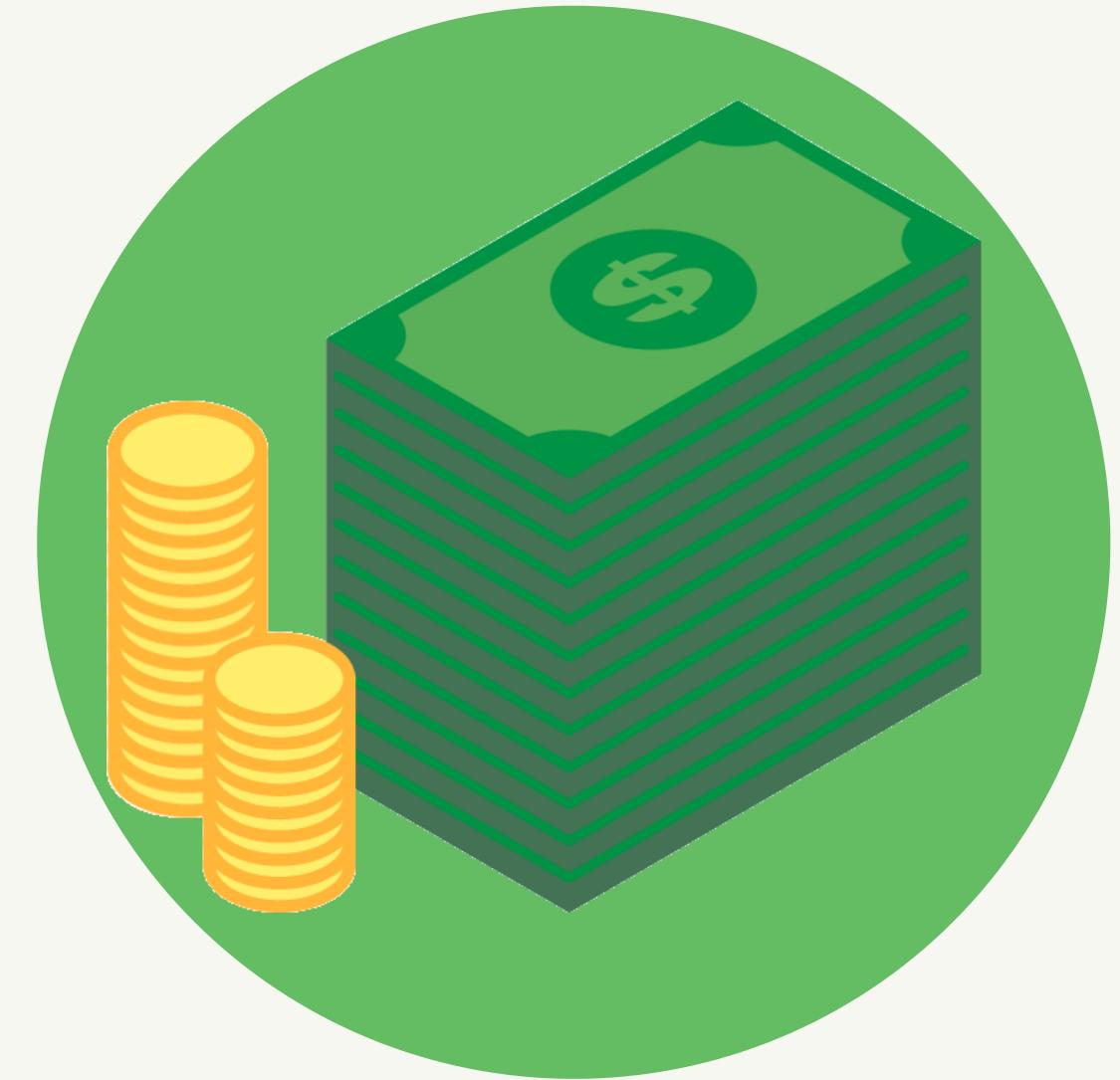
ไม่มีที่ตบแน่นเลย
!!!





Solution







PARKFINDER





เมื่อเทียบกับ คู่แข่ง

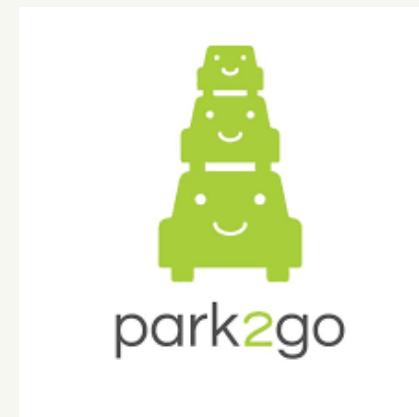
มีระบบให้จองล่วงหน้าเกิน 3 ชั่วโมง

เจ้าของที่จอดลงทุนต่ำ

สามารถตรวจสอบรถได้ตลอดเวลา

focus ที่จอดที่เป็นบ้านคน

ในระบบปกติ จะไม่ใช้คนในการทำงาน



TechStack

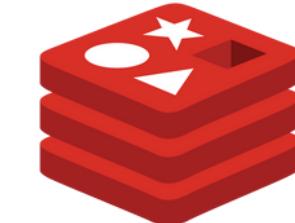
Frontend



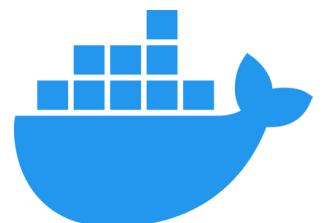
Backend



Database



Services



Feature



รถของฉัน

สามารถเพิ่ม แก้ไข และลบรายละเอียดรถได้



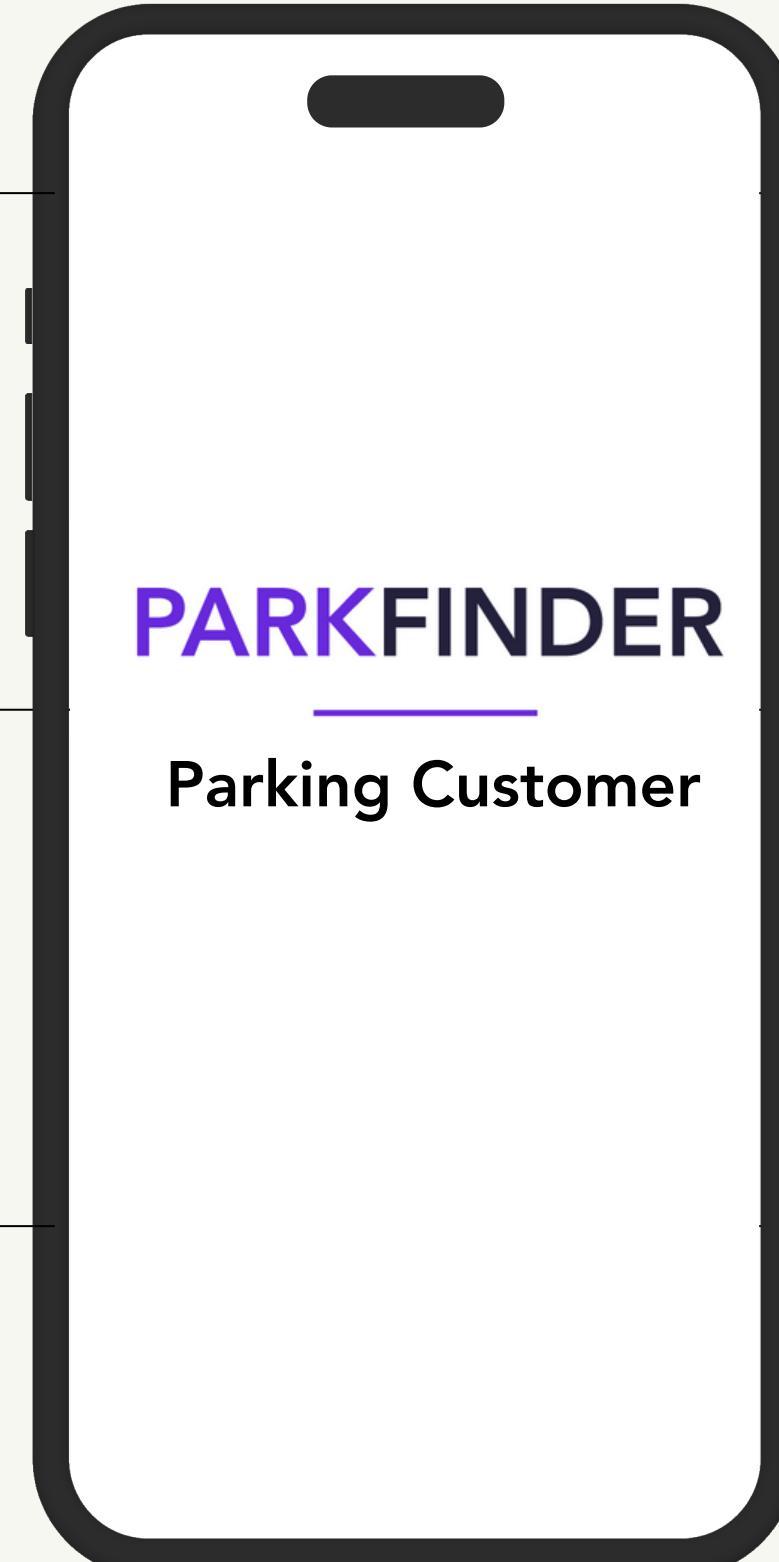
จองที่จอดรถ

จองหรือเช่าที่จอดรถที่ต้องการ โดยสามารถเลือกวัน - เวลา และจำนวนชั่วโมงที่ต้องการได้



จองล่วงหน้า

จองที่จอดรถล่วงหน้าได้ ไม่เกิน 3 เดือน



ขอดูรูปรถ

ขณะจอดรถ สามารถขอดูรูปรถได้



ส่งข้อความหาเจ้าของที่

ขณะจอดรถ สามารถส่งข้อความหาเจ้าของที่จอดรถได้



แลกรewards

สามารถดู และแลกรewards ได้

Feature



ที่จอดรถของฉัน

สามารถเพิ่ม แก้ไข และลบรายละเอียดที่จอดรถได้

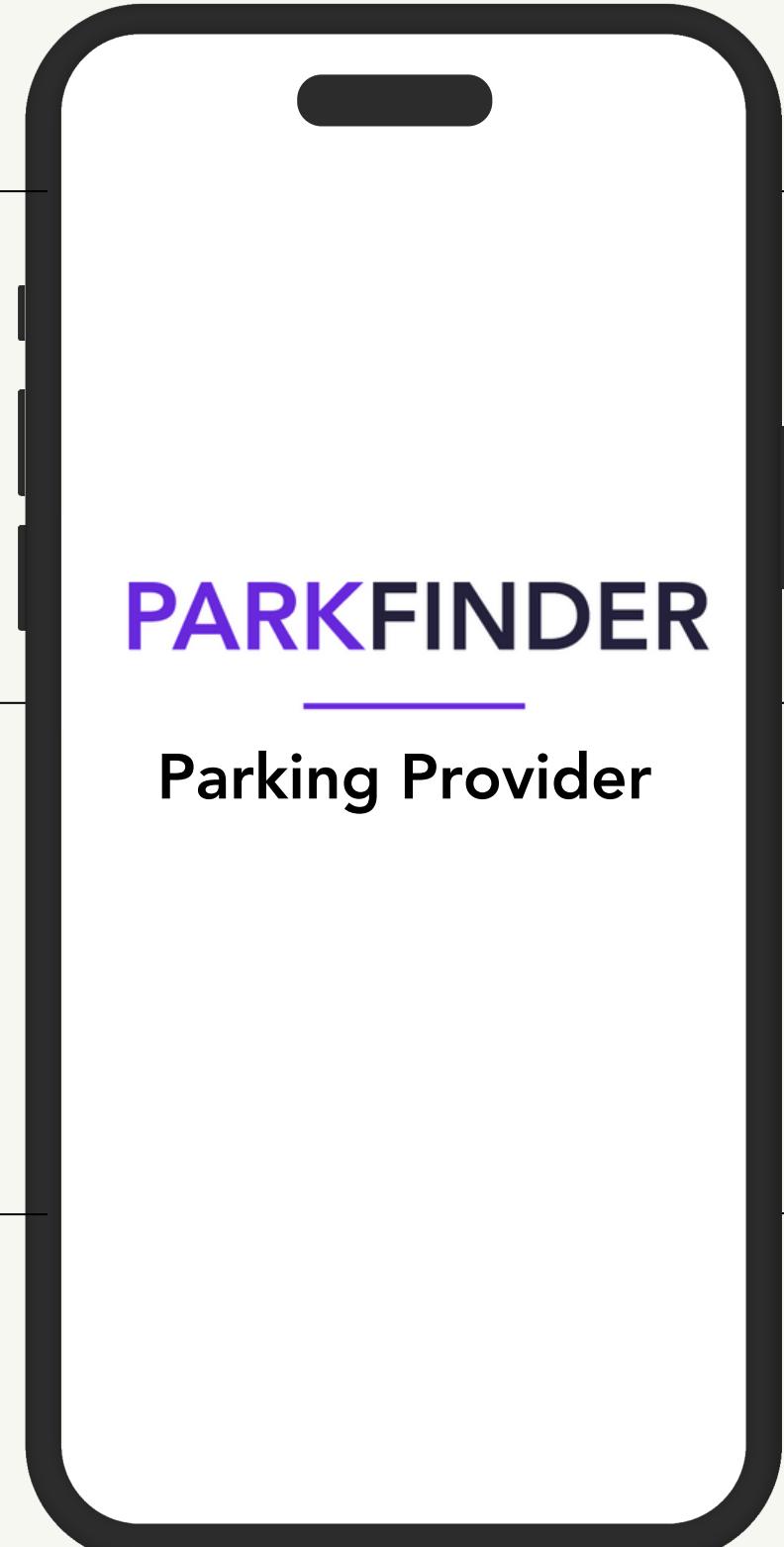
สรุปรายได้

สามารถดูประวัติการให้เช่าและสรุปรายได้ของที่จอดรถได้



ยืนยันการจองล่วงหน้า

สามารถยืนยันการจองล่วงหน้าได้ภายใน 1 วัน



ขอดูรูปรถ

สามารถขอดูรูปถ่ายที่จอดอยู่ได้



ส่งข้อความหาเจ้ารถ

สามารถส่งข้อความหาเจ้าของรถที่จอดอยู่ได้

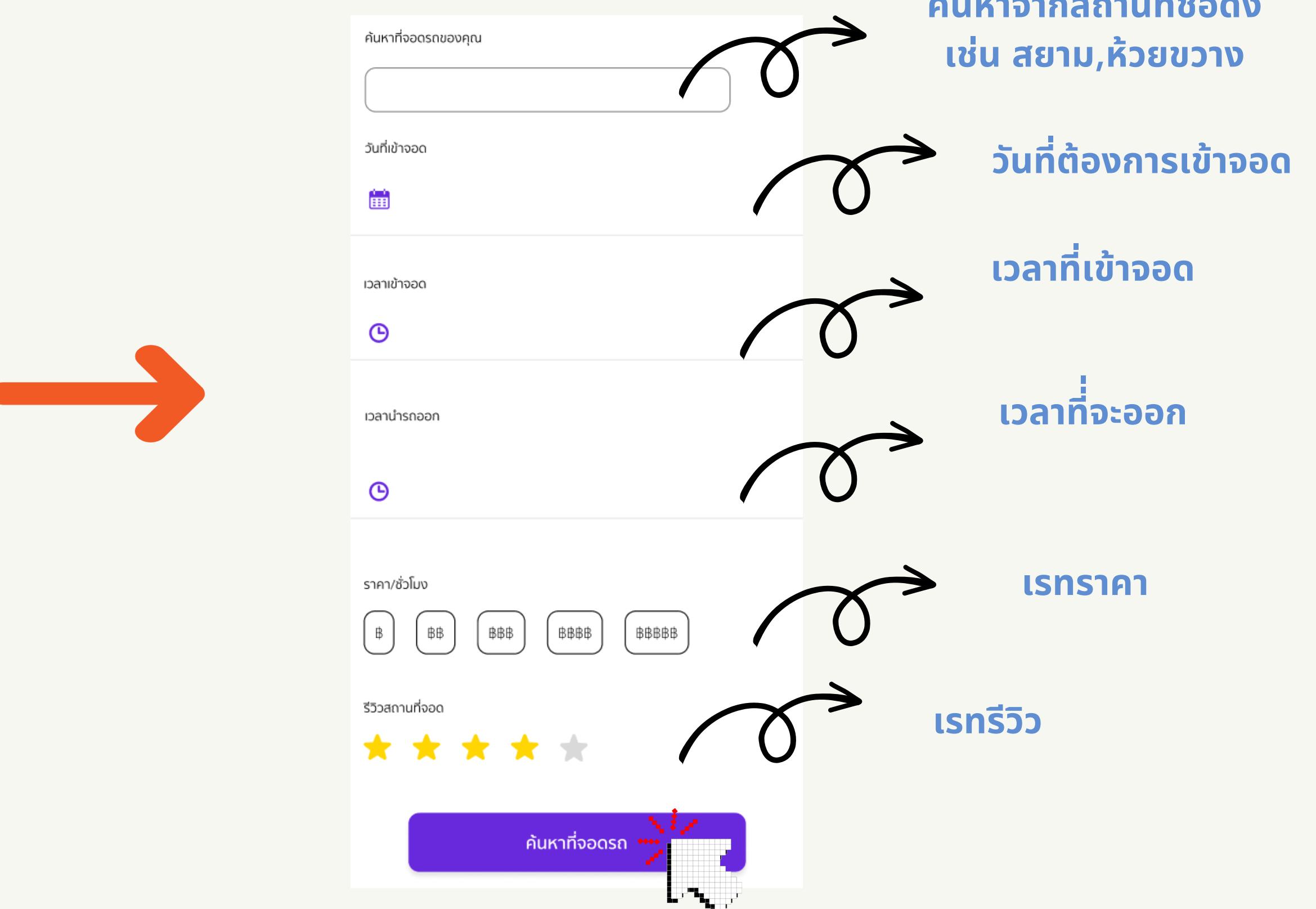
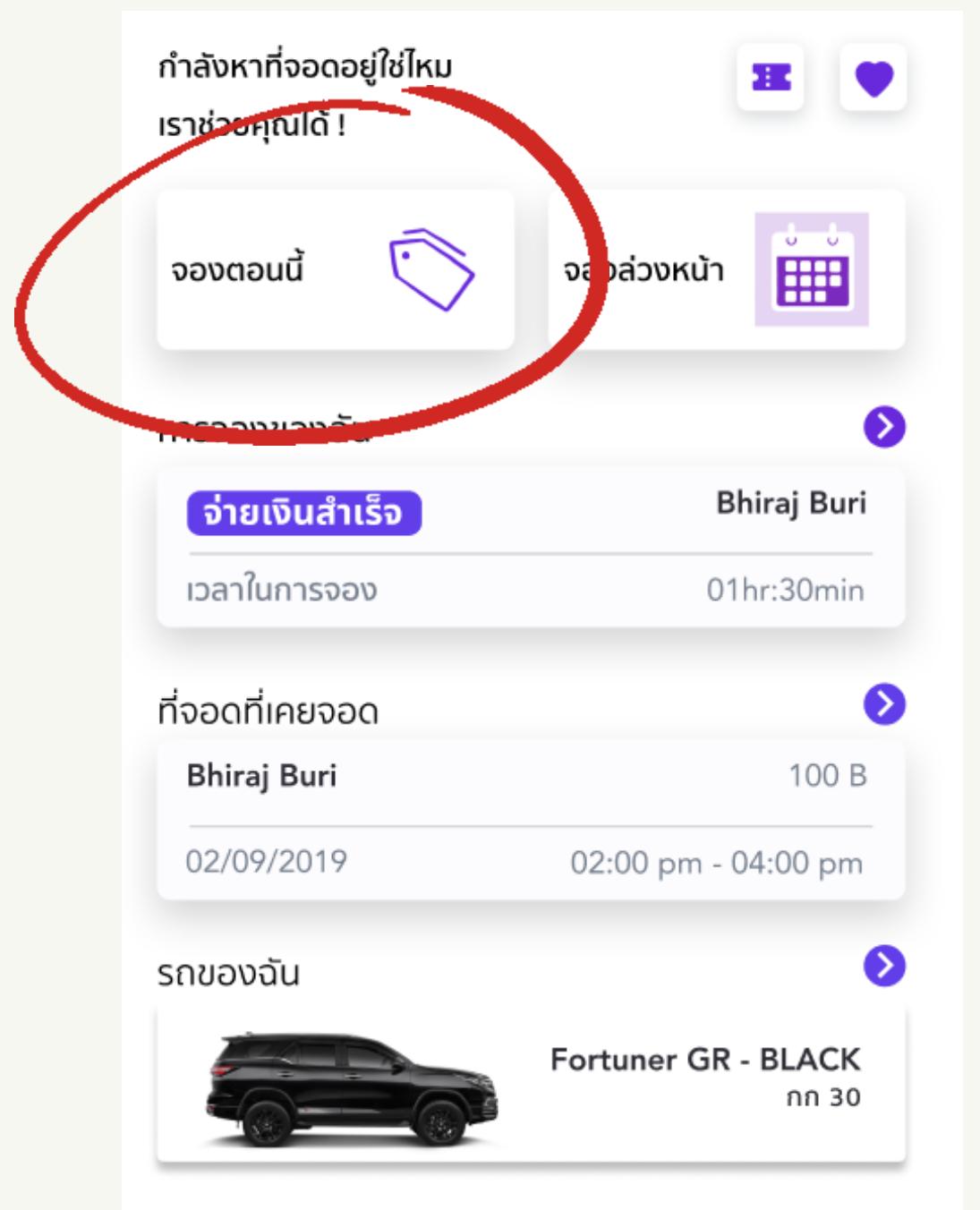


แลกรewards

สามารถดู และแลกรewards ได้



การจองที่จอดรถ



คันหาที่จอดรถที่ซื้อดัง
เช่น สยาม, หัวยขวาง

วันที่ต้องการเข้าจอด

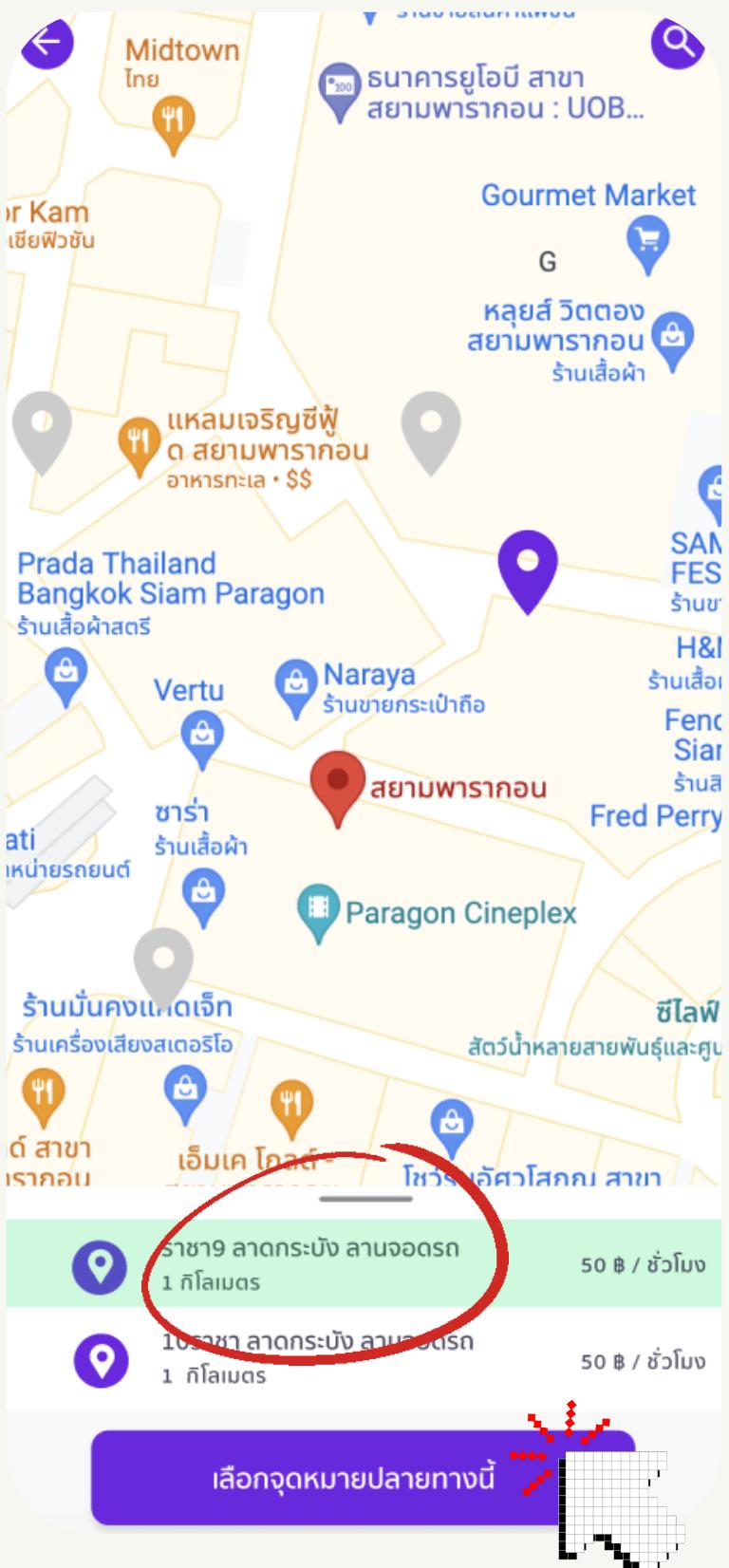
เวลาที่เข้าจอด

เวลาที่จะออก

ราคา

รีวิว

ตรวจสอบความถูกต้อง



รูปที่จอด

ราชาก ลาดกระบัง ลานจอดรถ

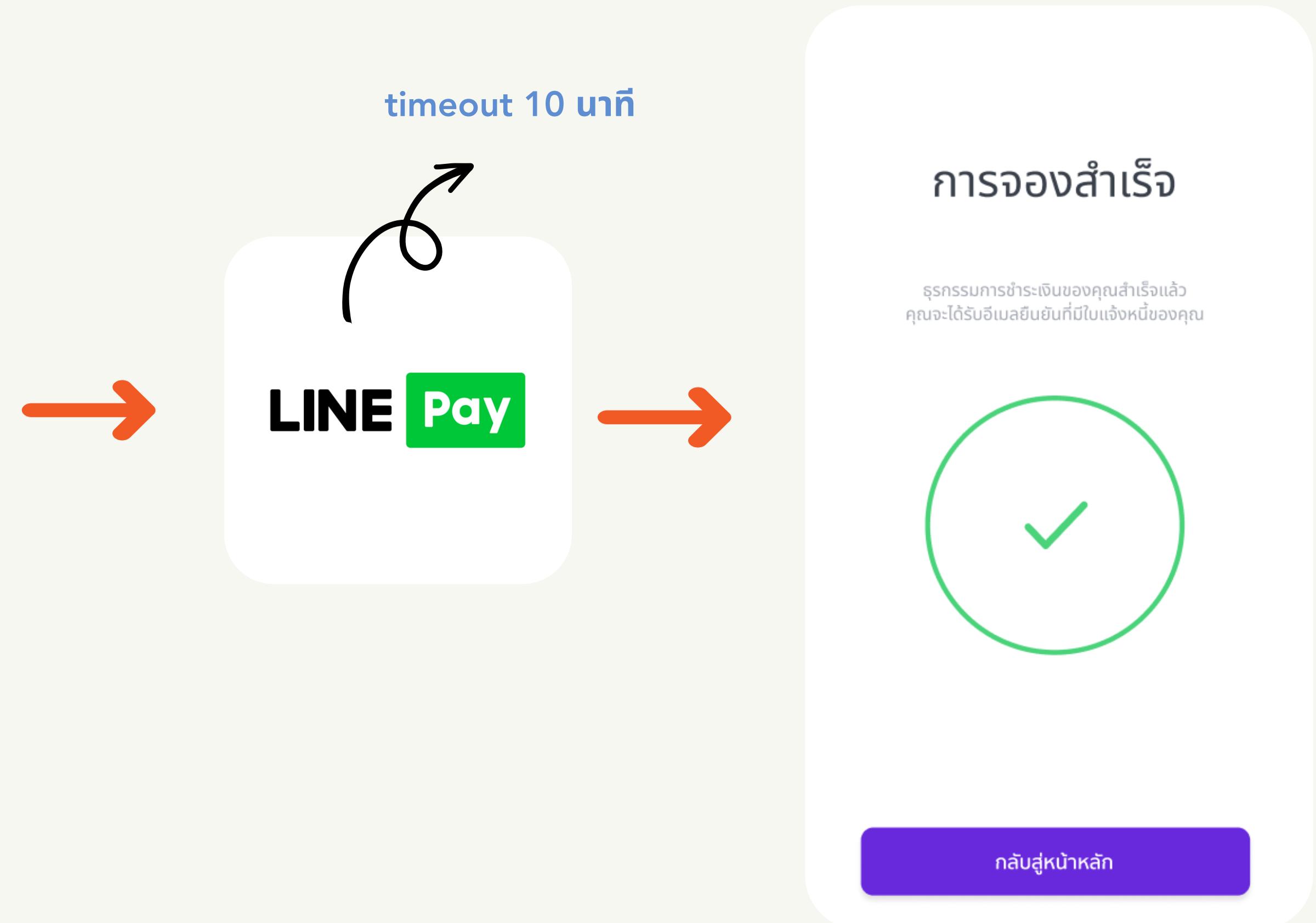
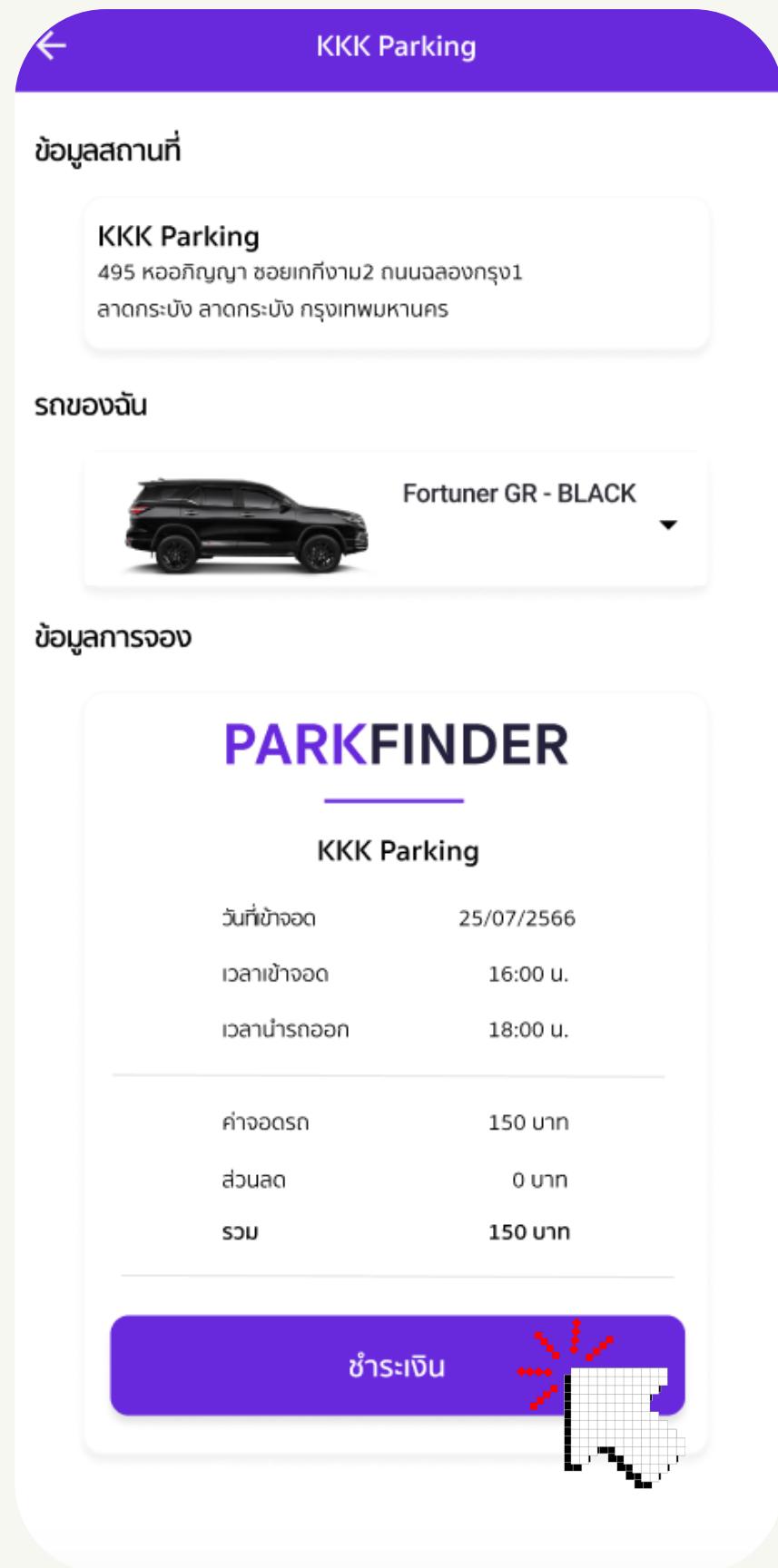
495 หมู่บ้านชุมชน 2 กบล.กรุงเทพฯ
ลาดกระบัง ลาดกระบัง กรุงเทพมหานคร

ราคา 120 / ชั่วโมง ★ 4.2 57 รีวิว

รีวิว (57 รีวิว)

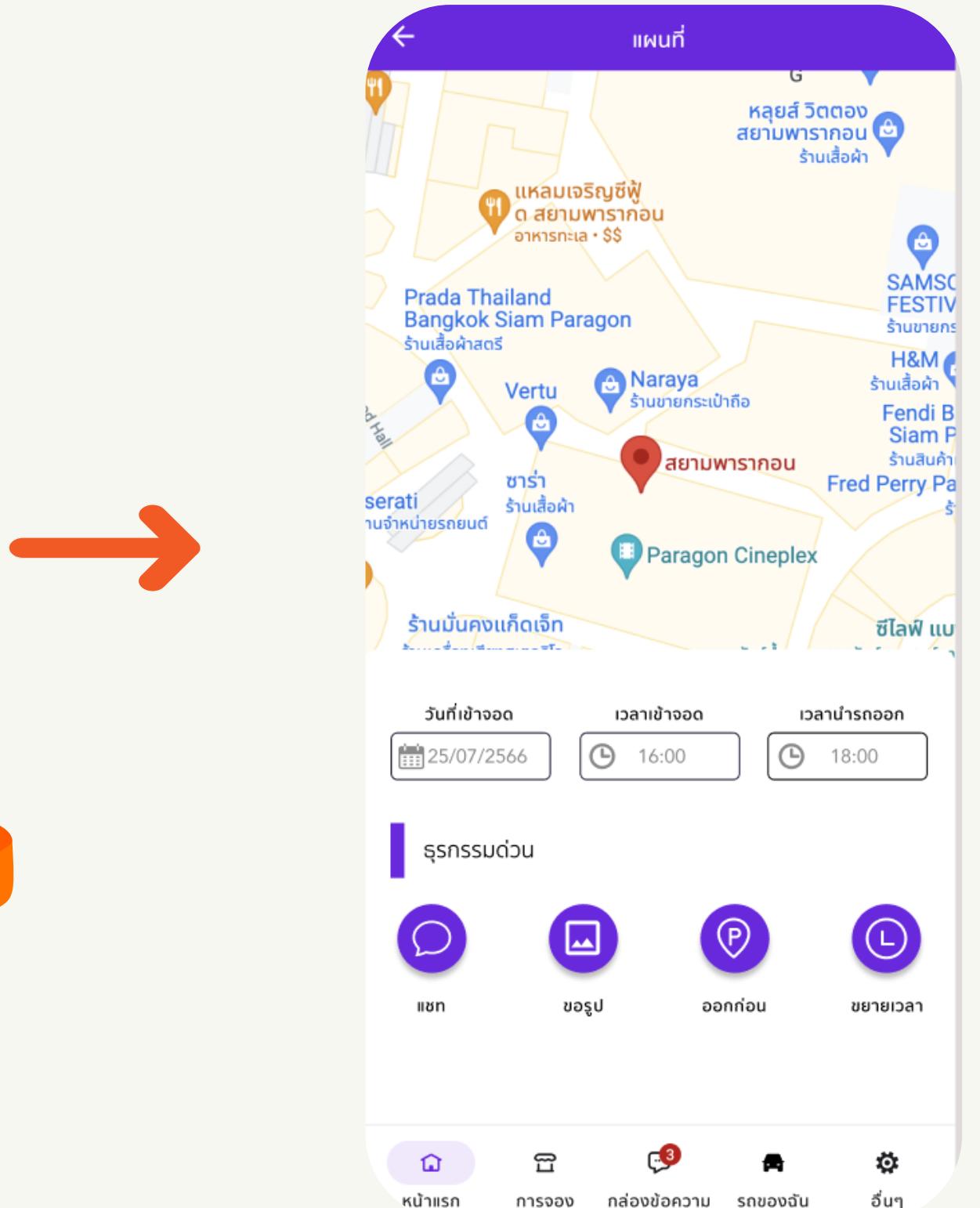
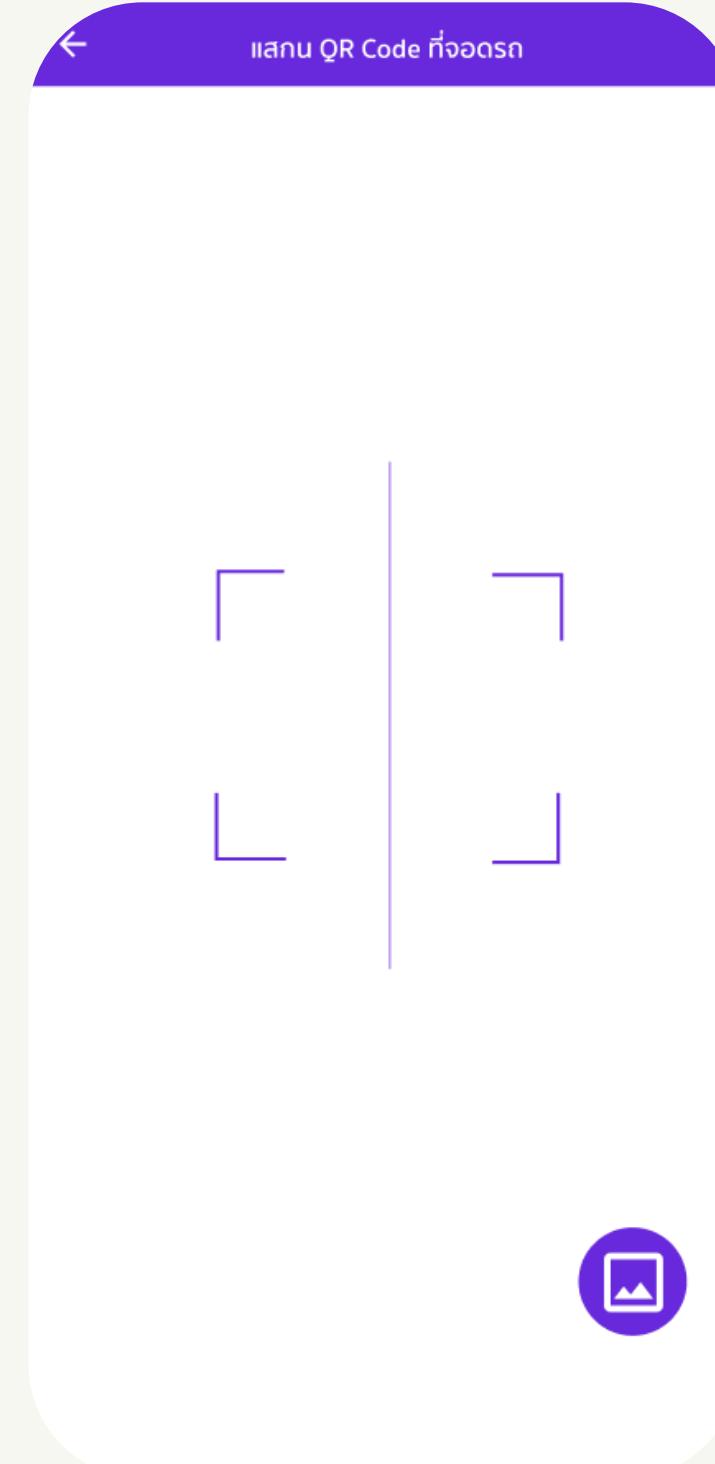
เกือกุล นิยมสิกธ์
★★★★★
เจ้าของที่บริการดีมากครับ แต่แพงไปหน่อย

จองที่จอดรถ





การเข้าที่จอดรถ ของผู้ใช้งาน



⌚ ออกร่องเวลา

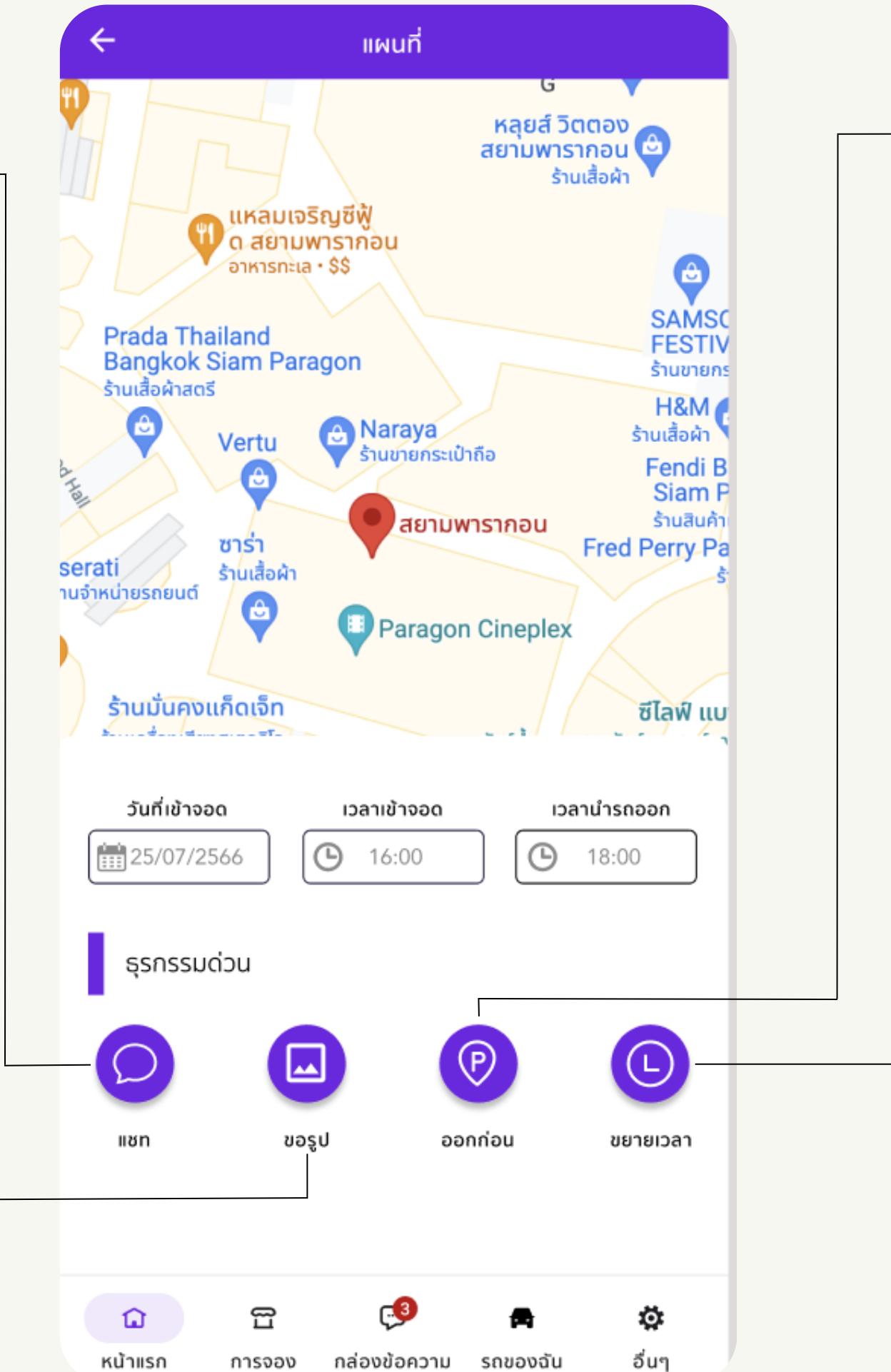
สามารถขออกร่องเวลาที่จะออกตามปกติ

แท็กกับเจ้าของที่จอดรถ ⚡

สามารถคุย สบทนา กับเจ้าของที่จอดรถได้
อย่าง real-time

สามารถขอรูปถ่ายที่จอดอยู่ 📸

สามารถขอรูปถ่ายของตัวเองที่จอดอยู่
ได้ตลอดเวลา



🕒 ขยายเวลา

หากมีนิจว่าไม่สามารถขอตามเวลาที่
กำหนดได้ จะสามารถขยายเวลาโดยไม่มีค่า
ปรับ (กรณีที่มีคนจองต่อจะไม่สามารถ
ขยายเวลาได้)



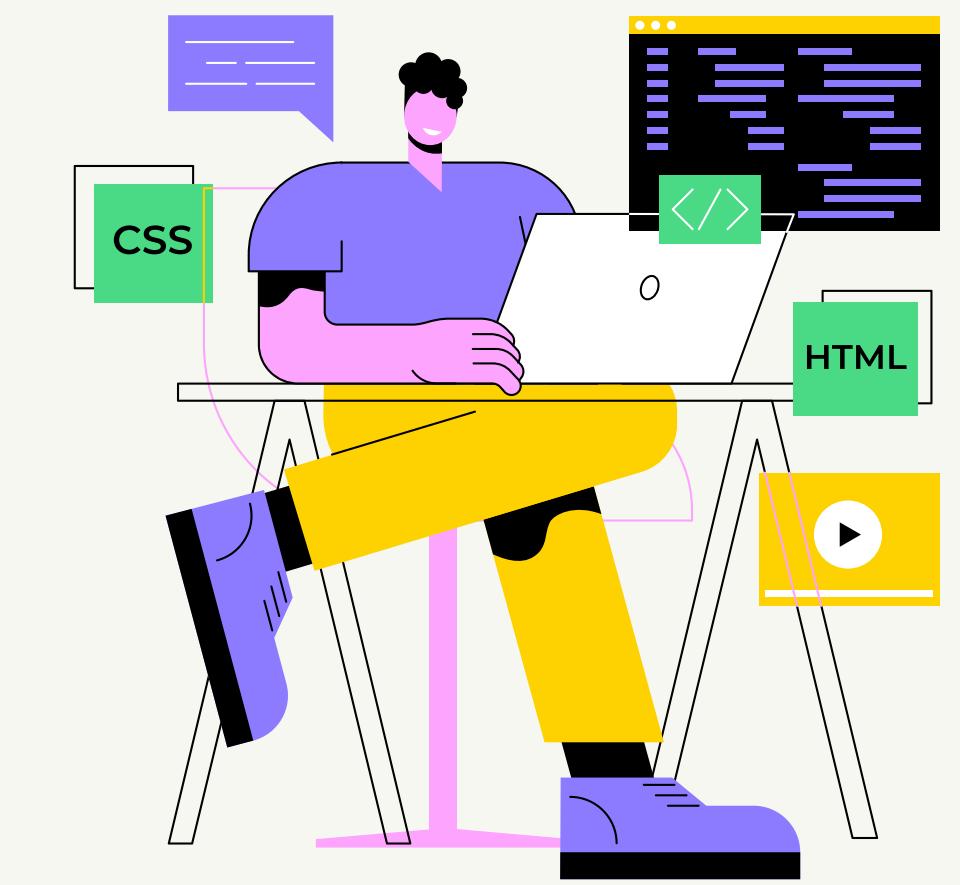
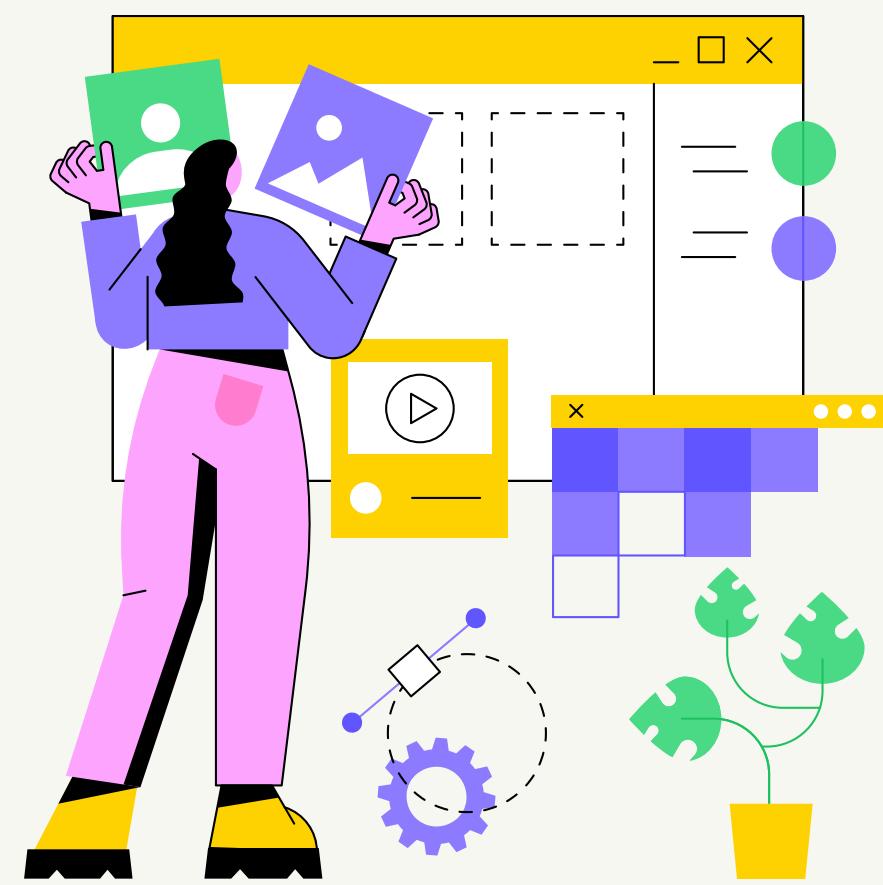
Progress

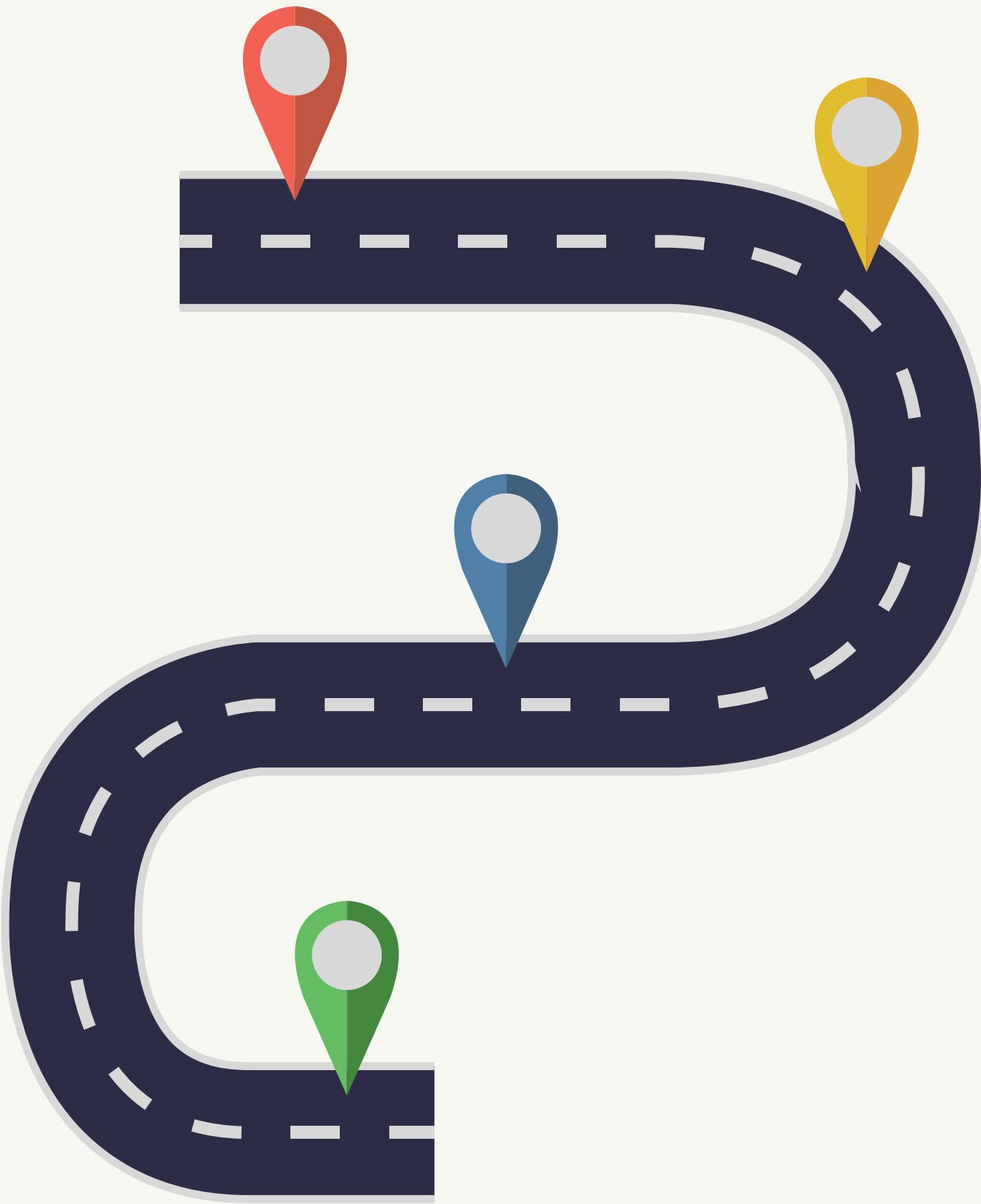


FRONT END
60 %

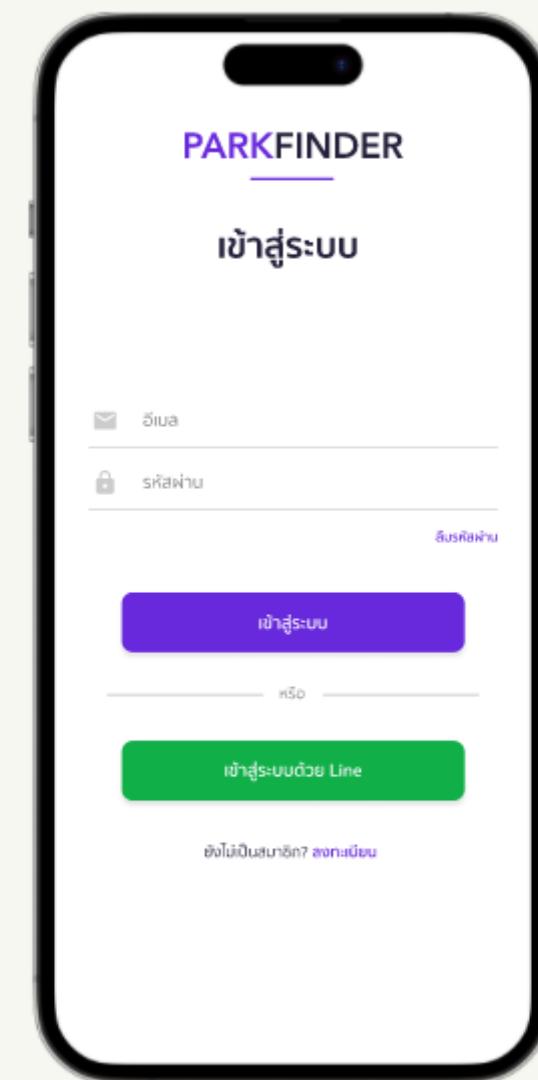
UXUI

40 %

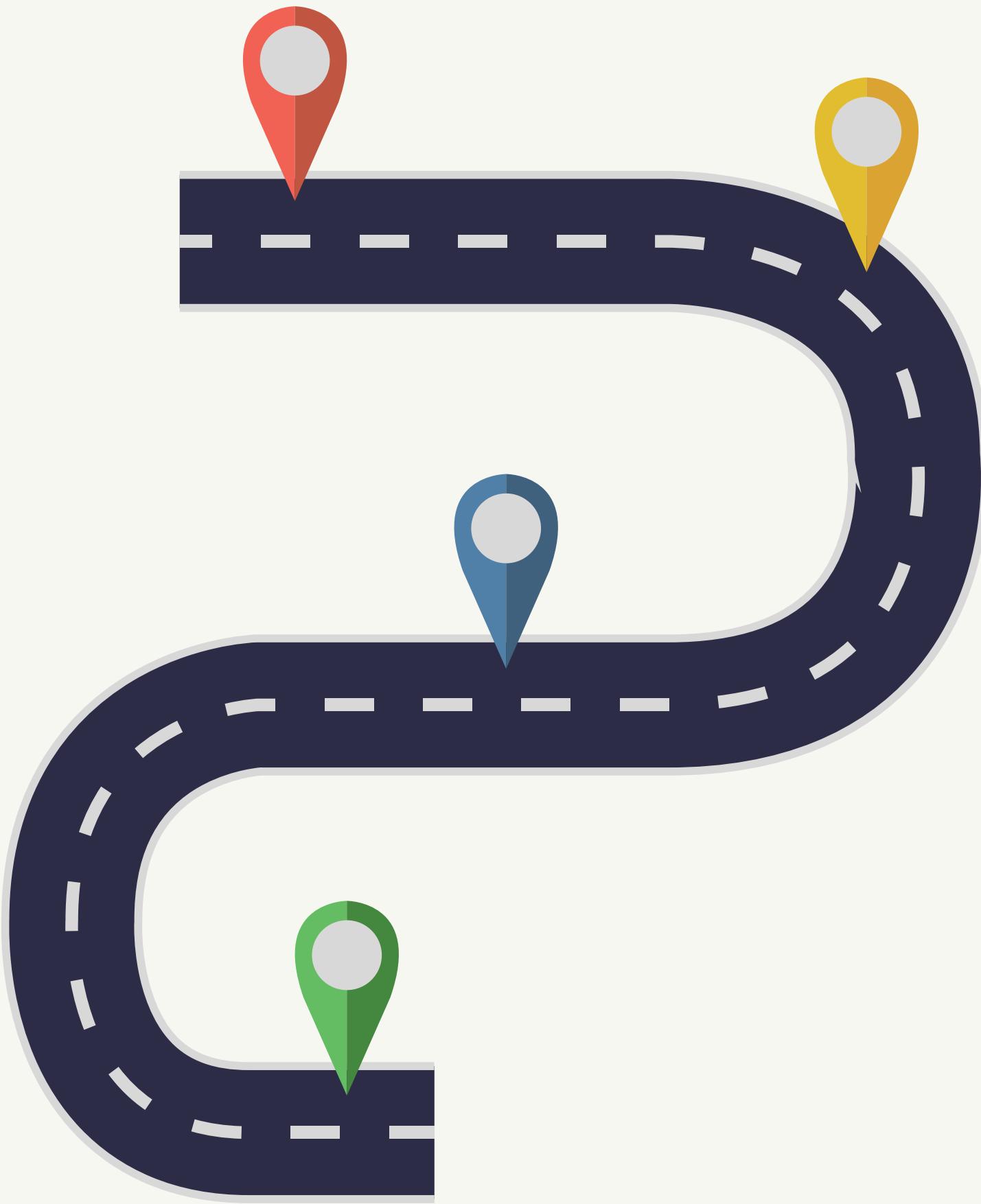




UXUI design



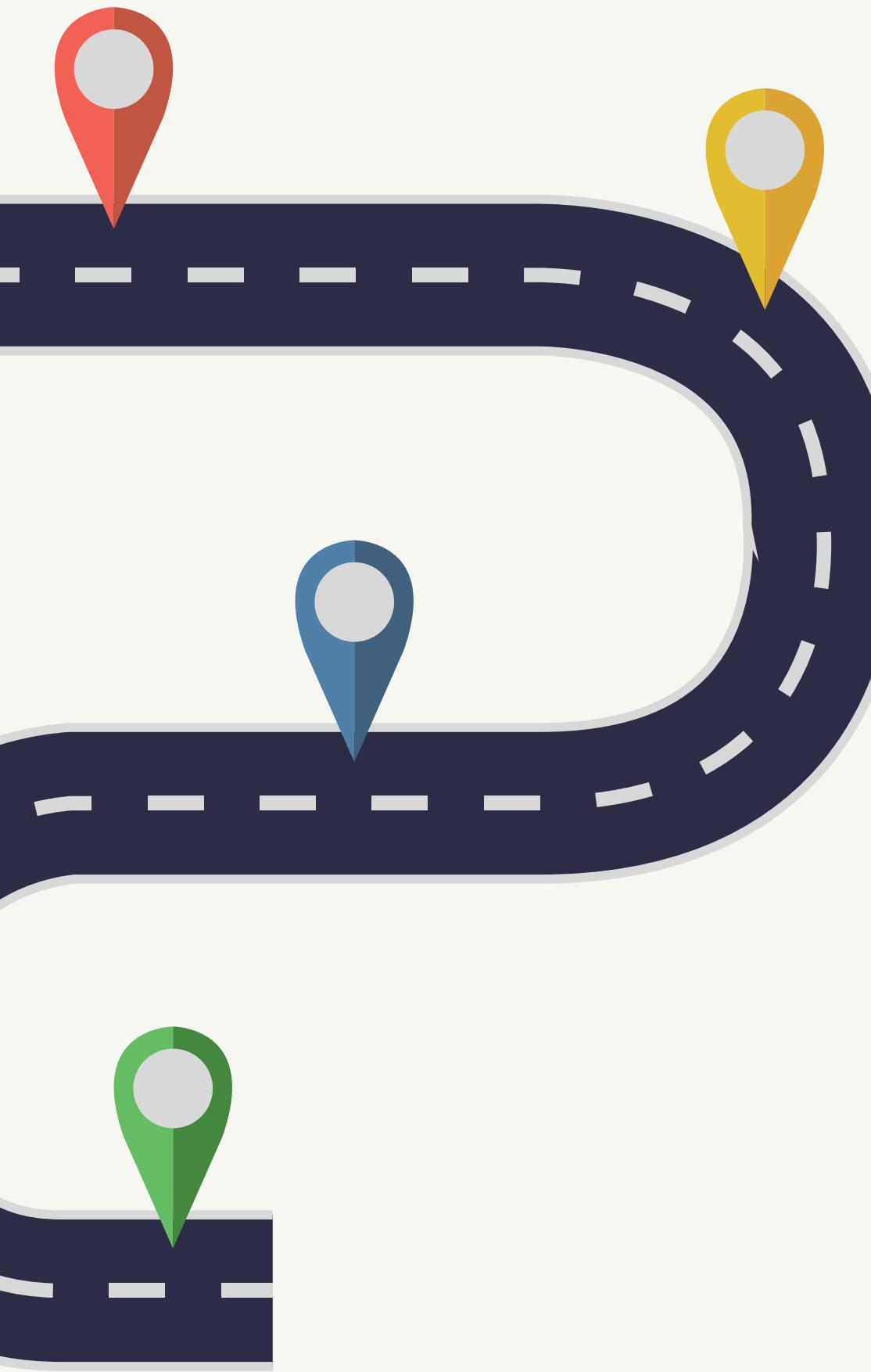
Improve performance by e... Learn more X
Project Edited 2 months ago



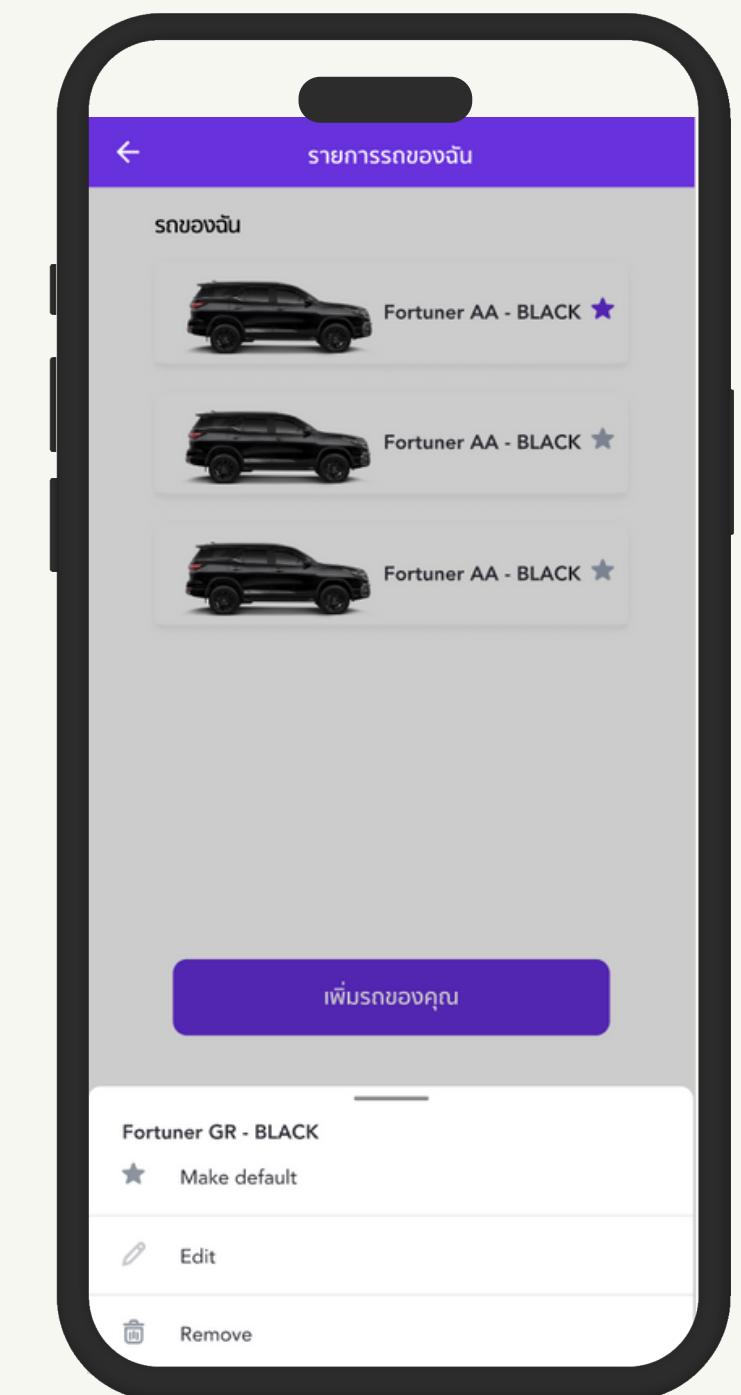
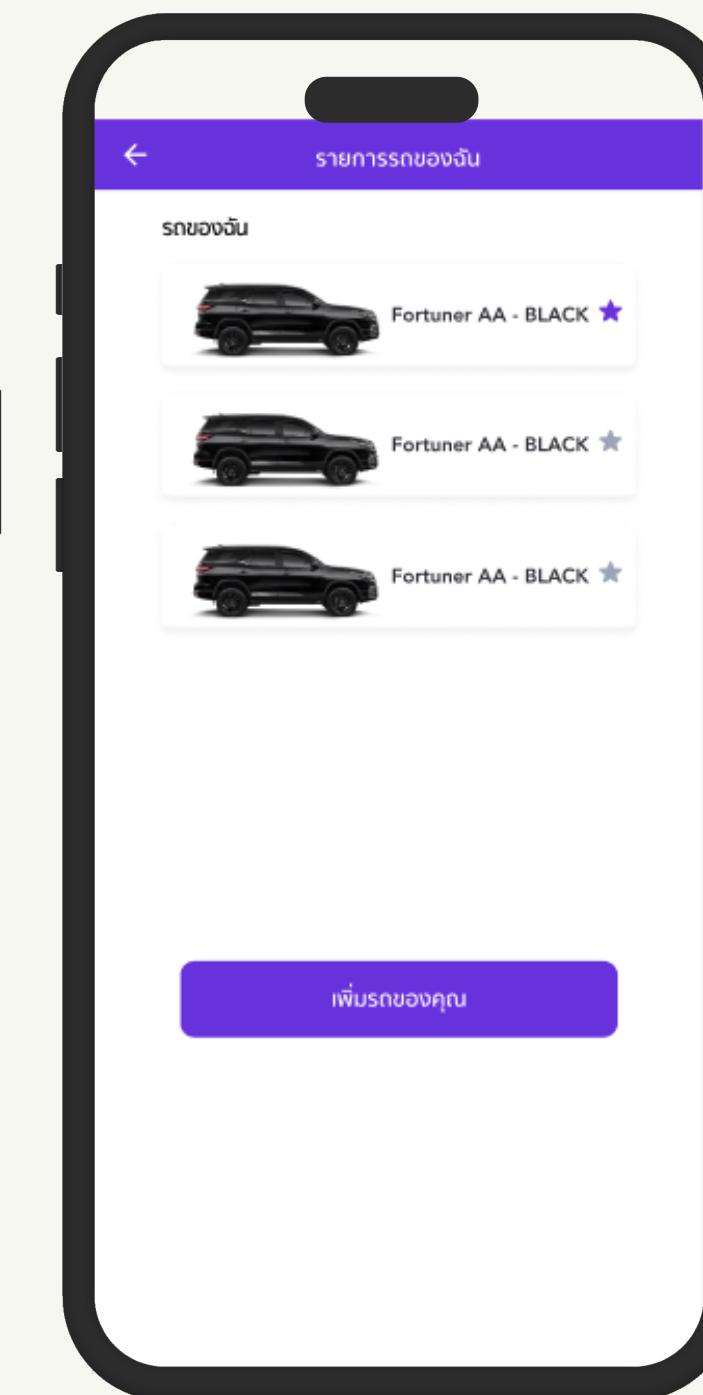
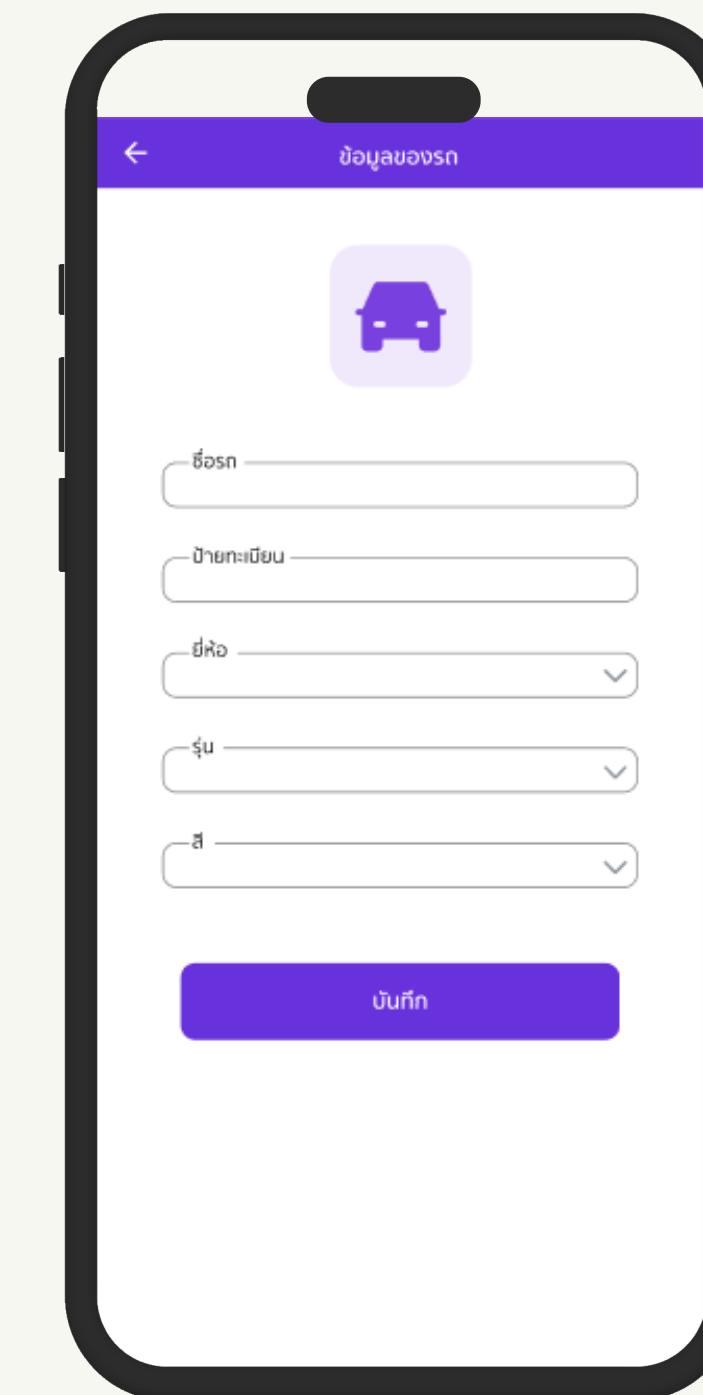
Login Register

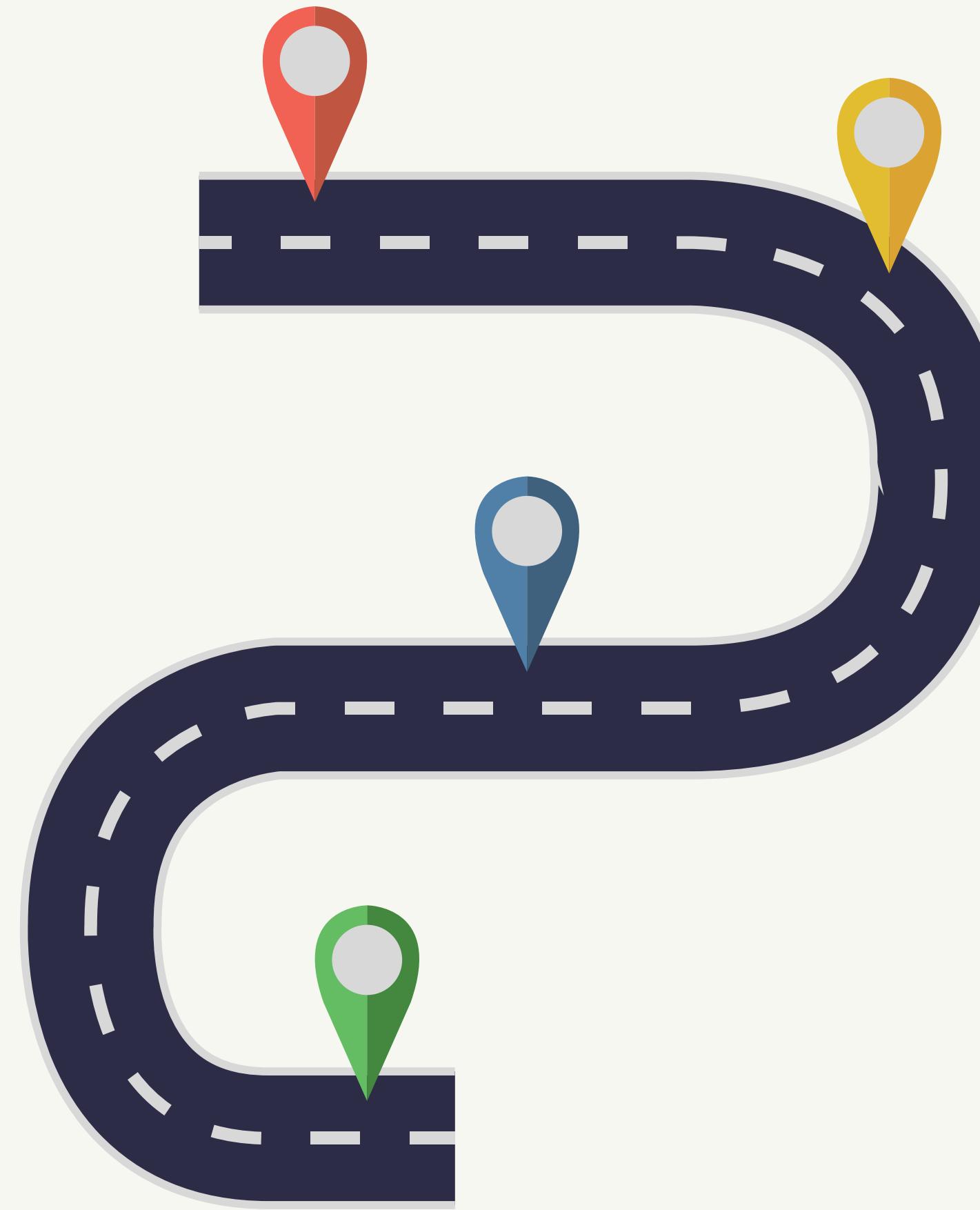
The image displays three mobile phone screens illustrating the ParkFinder app's user authentication process:

- Phone 1 (Left):** Shows the "Login" screen with the title "PARKFINDER" and the text "เข้าสู่ระบบ". It includes fields for "อีเมล" (Email) and "รหัสผ่าน" (Password), and a "เข้าสู่ระบบ" (Log In) button.
- Phone 2 (Middle):** Shows the "Register" screen with the title "PARKFINDER" and the text "ลงทะเบียน". It includes fields for "ชื่อ" (Name), "นามสกุล" (Surname), "เบอร์โทรศัพท์" (Phone Number), "อีเมล" (Email), "รหัสผ่าน" (Password), and "ยืนยันรหัสผ่าน" (Confirm Password). A "ลงทะเบียน" (Register) button is at the bottom.
- Phone 3 (Right):** Shows a "Two-step verification" screen with the title "การยืนยันขั้นที่ 2" (Step 2 Verification). It features a purple padlock icon, the text "กรุณากรอกรหัส OTP ที่ส่งไปในอีเมล" (Please enter the OTP sent to your email), and a numeric input field with numbers 1-6. A "ยืนยัน" (Verify) button is at the bottom.

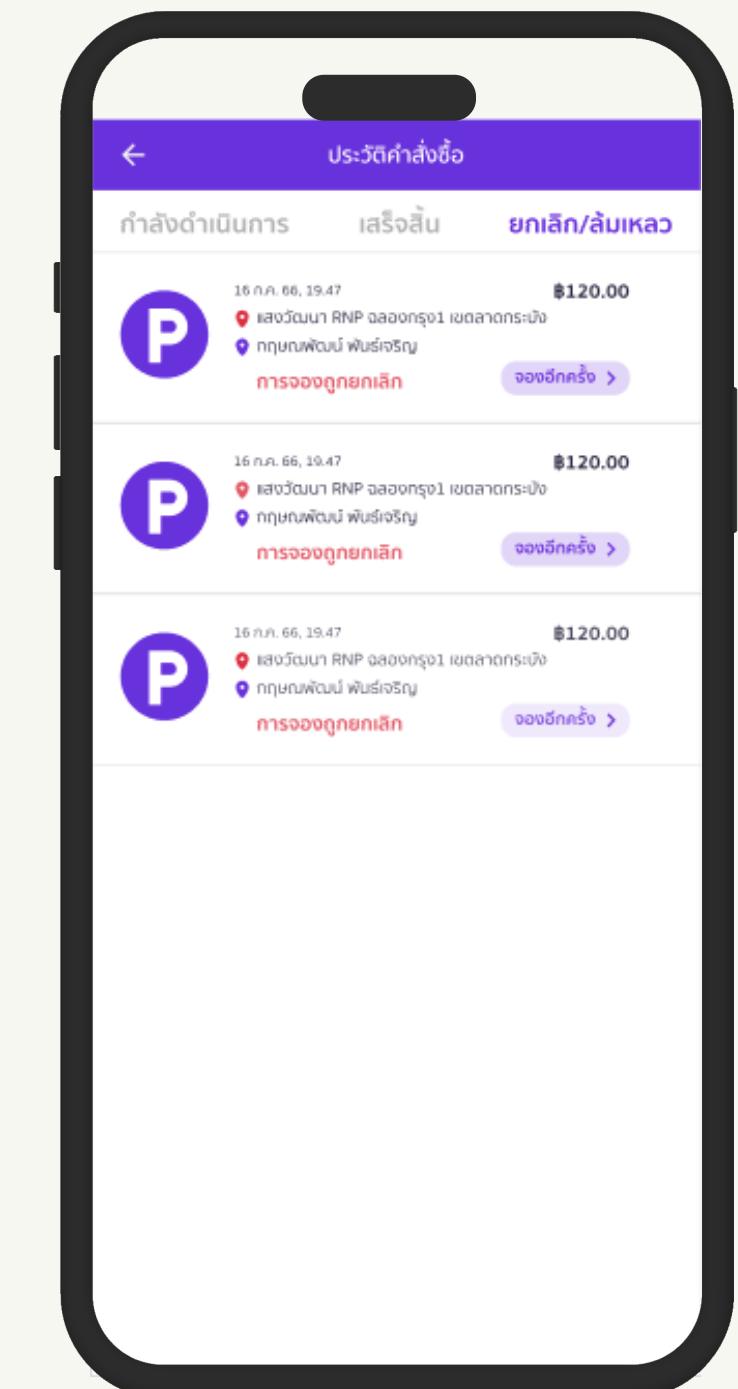
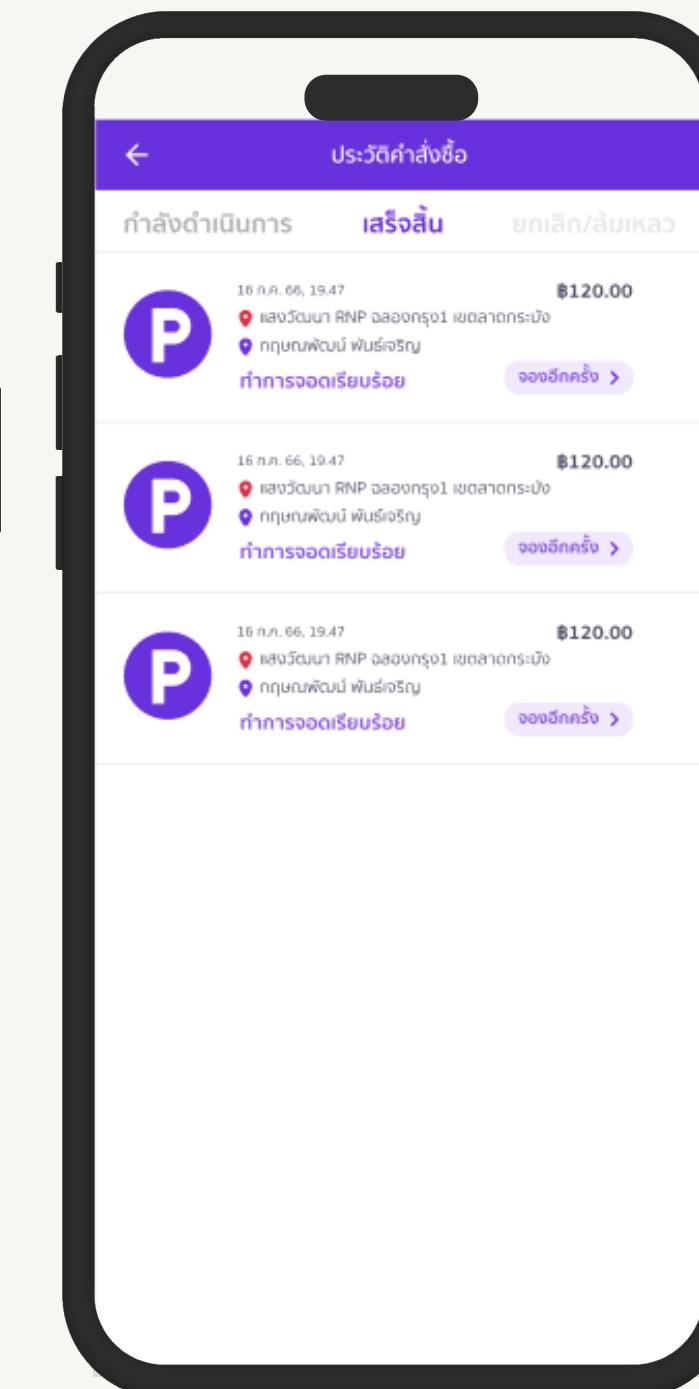
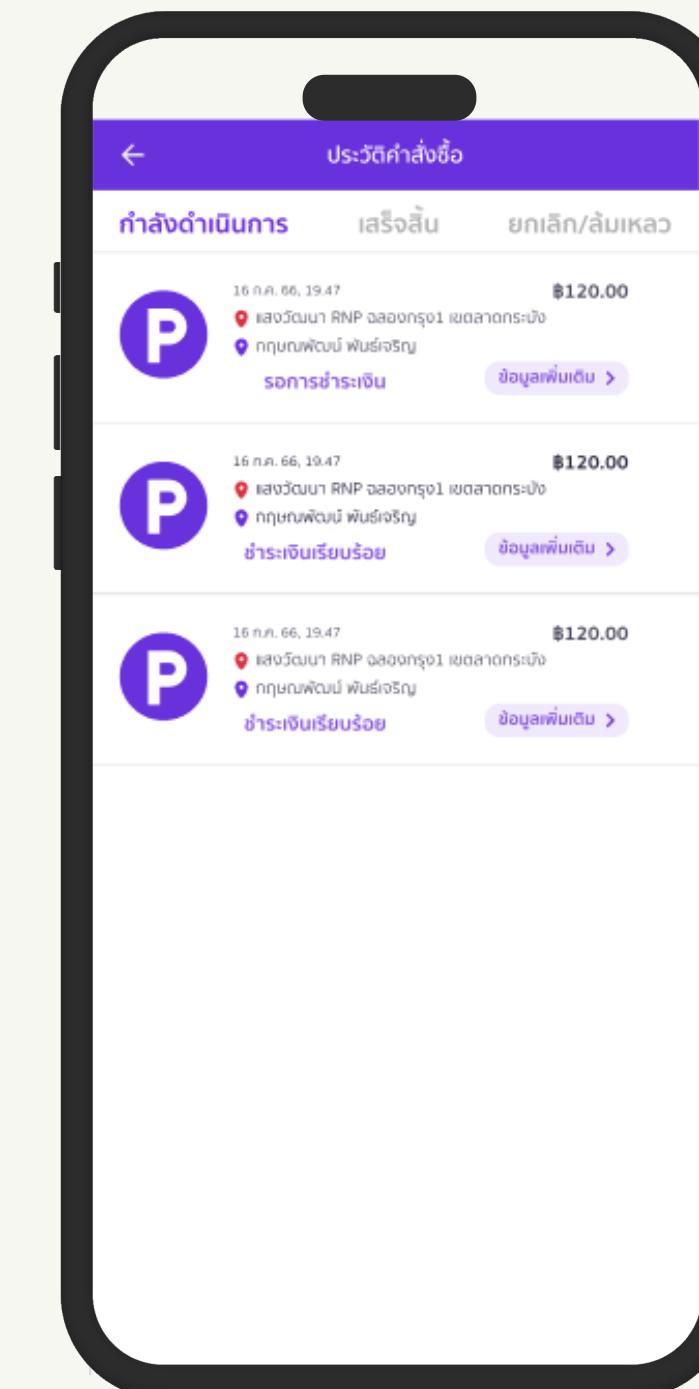


My Car

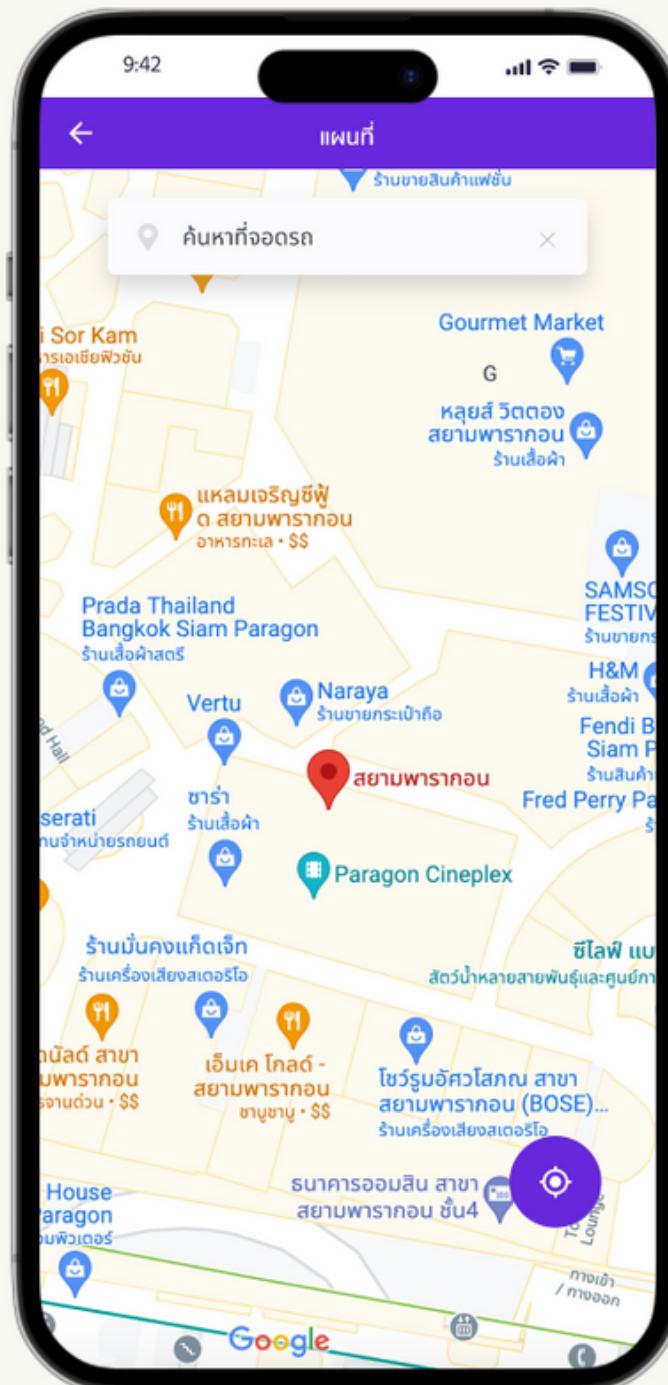




History



Google Map API



Google Maps Platform Keys & Credentials All Google Maps Platform APIs ▾ + CREATE CREDENTIALS LEARN

To view all credentials visit [Credentials in APIs & Services](#)

API Keys

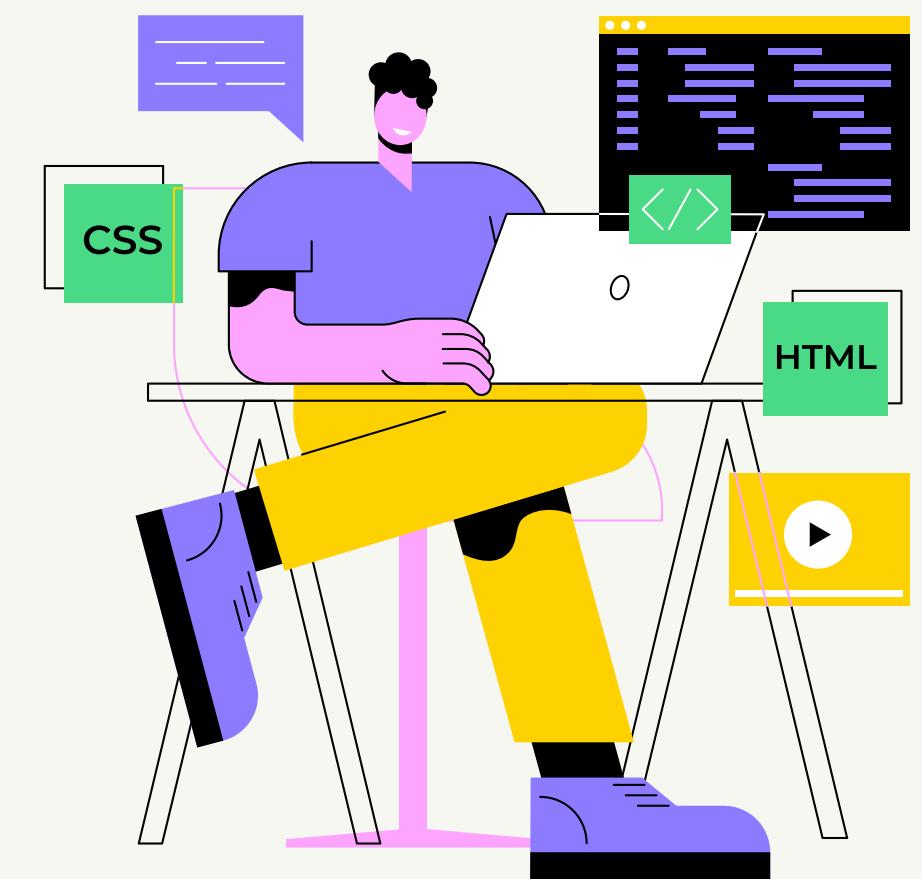
Name	Creation date	Restrictions	Actions
Places API	Aug 26, 2023	Places API ...	SHOW KEY ⋮
Maps API Key	Aug 24, 2023	Android apps, 1 API ...	SHOW KEY ⋮

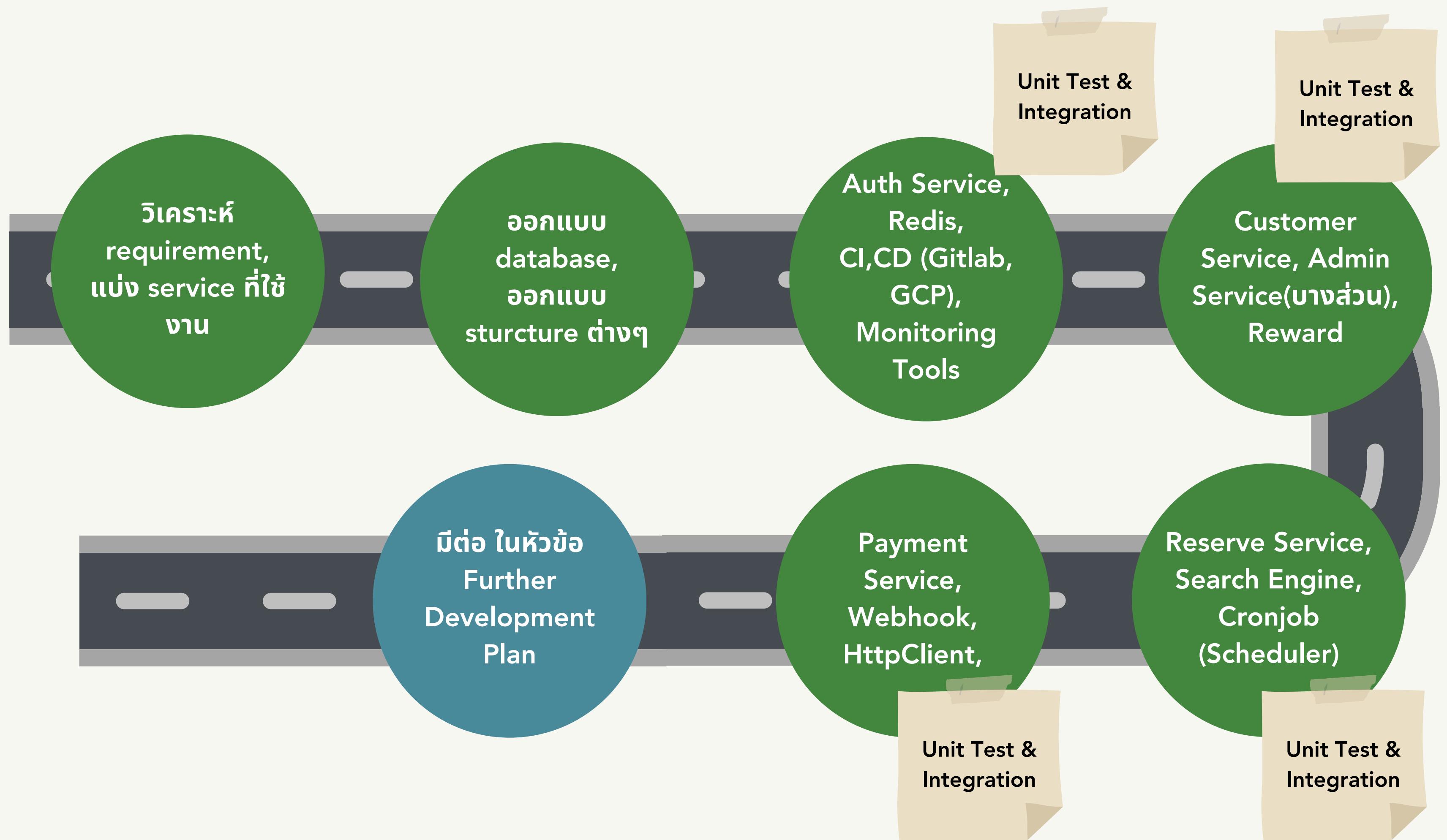
DEPLOYMENT

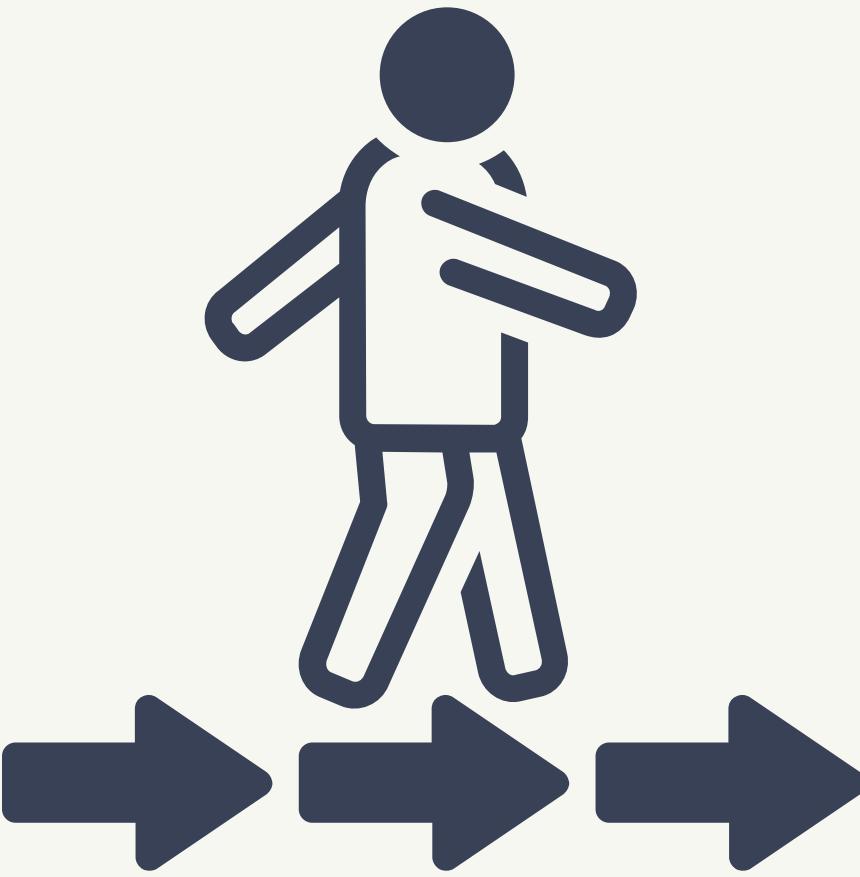
30 %



BACK END
70 %







Further Development Plan



UXUI

- Parking Provider
- Admin Web App



Integration

- Provider App
- Customer App
- Admin Web App

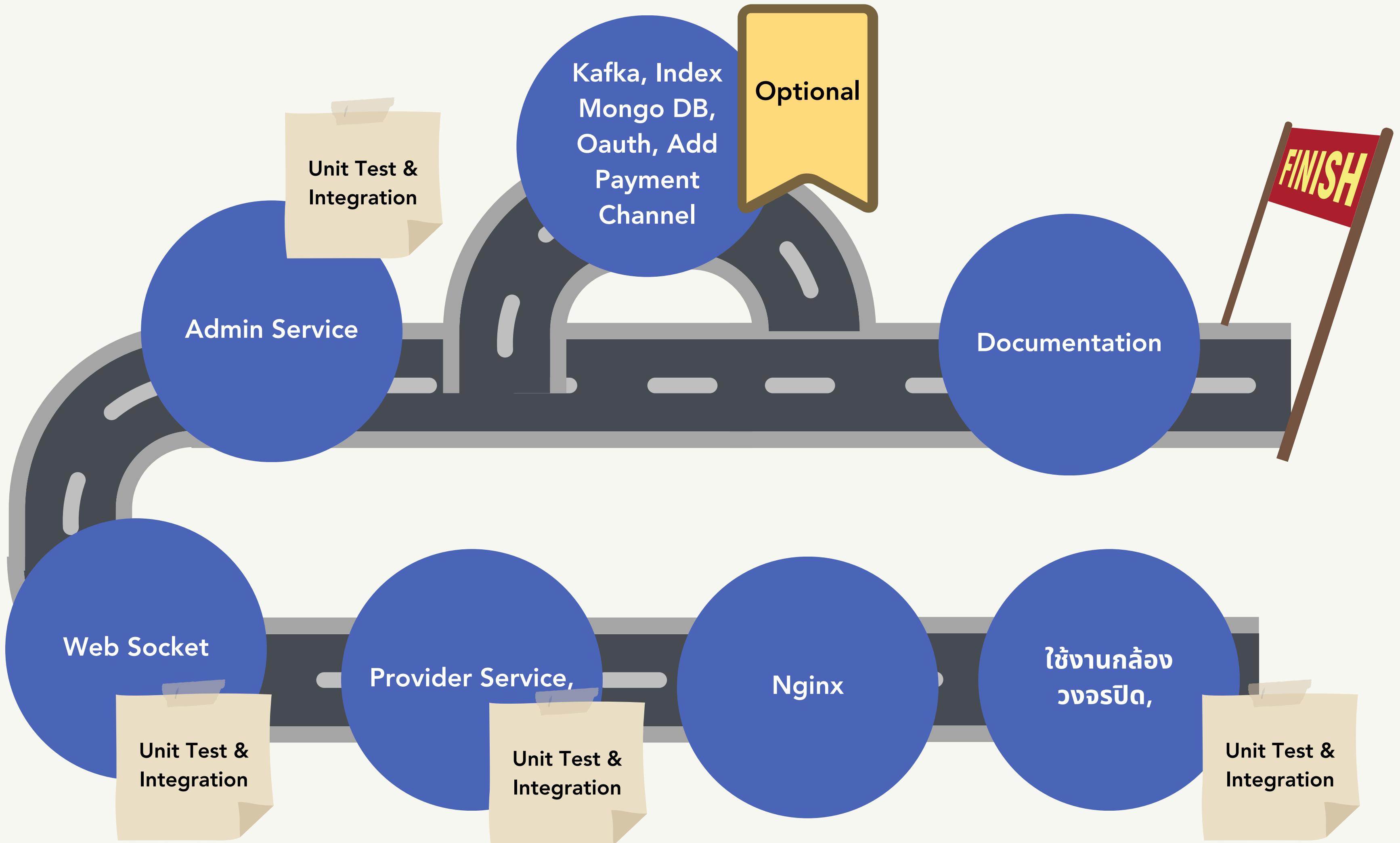


Front end

- Provider App
- Admin Web App



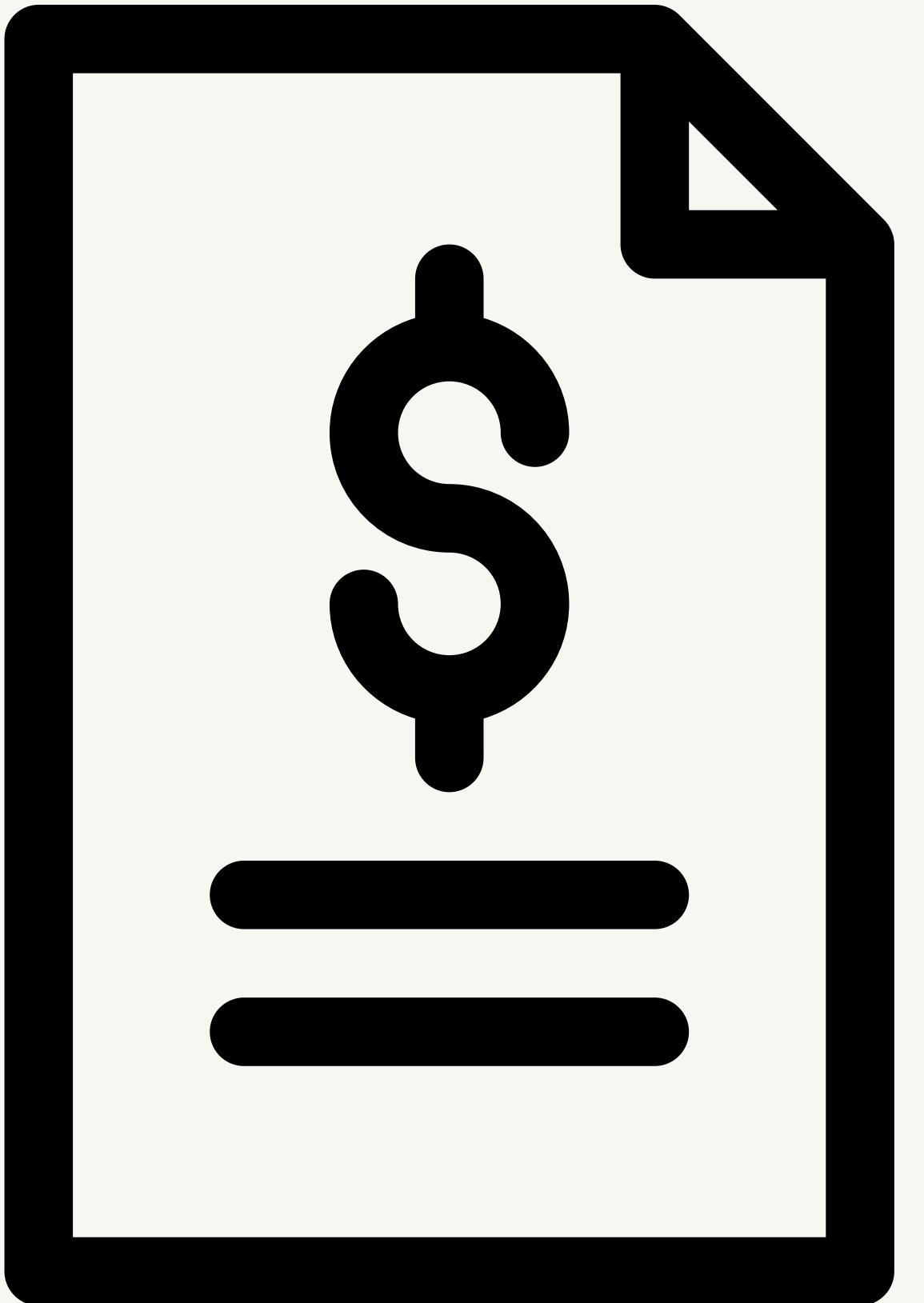
Automated test



**Thanks for your
attention!**

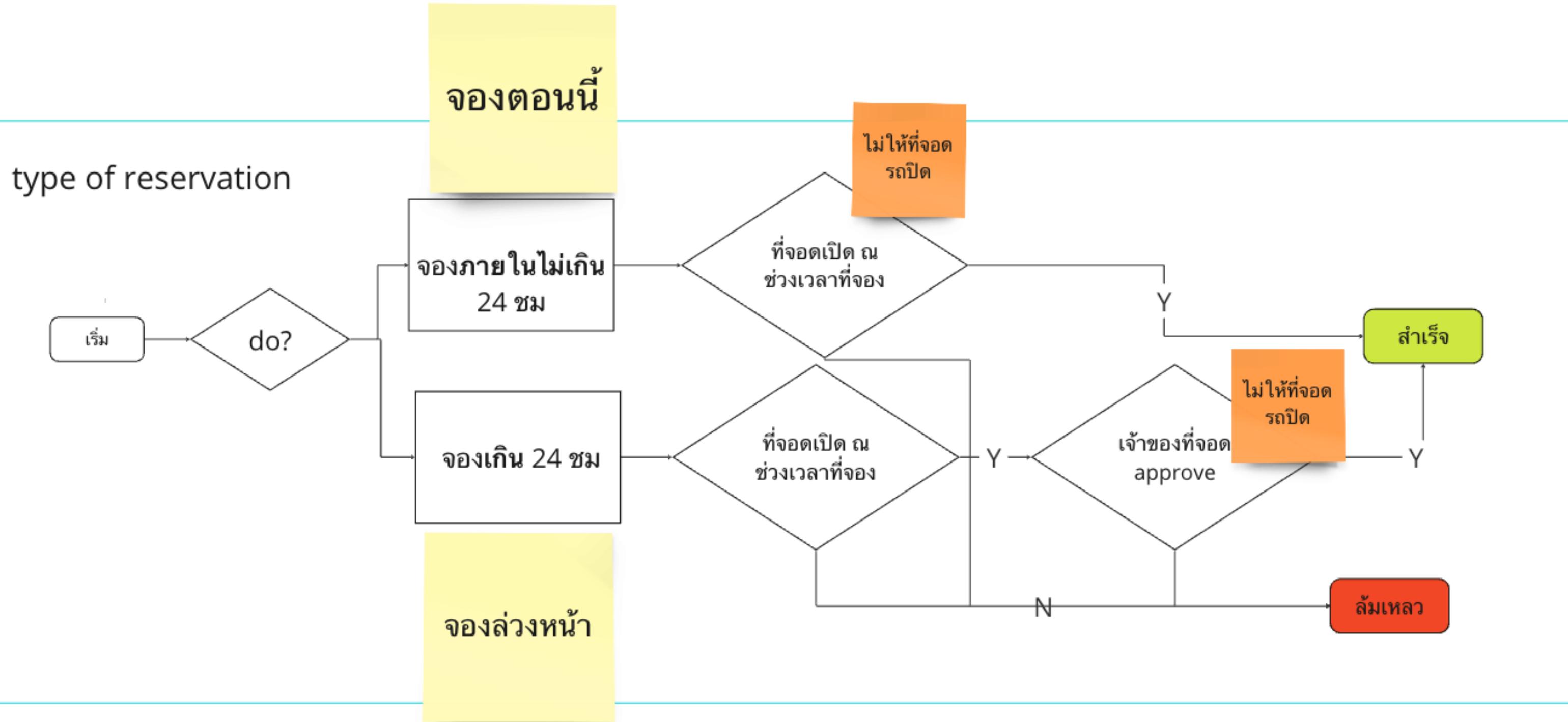


Appendix



GENERAL

จองต่อนี่ กับ จองล่วงหน้าต่างกันอย่างไร



จะแก้ปัญหากรณีคนจอดรถคันเก่าออกไม่ตามเวลา และมีคนจองต่ออยังไง?

1

มีการป้องกันไว้เบื้องต้นคือ ในการจองจะไม่สามารถ
จองติดกันได้เป็นเวลา 1 ชั่วโมง
เช่น A จอง 12.00-13.00 B จะเริ่มได้ตั้งแต่ 14.00
หรือ จองได้ถึงแค่ 11.00

2

ซึ่งระบบได้เพื่อเวลาให้นาย A ในการเรทเป็นเวลา
ทั้งสิ้น 15 นาทีโดยไม่เสียค่าปรับ

จะแก้ปัญหากรณีคนจอดรถคันเก่าออกไม่ตามเวลา และมีคนจองต่ออยังไง?(ต่อ)

3

แต่หากนาย A รู้ว่าไม่สามารถออกก่อนเวลาได้ และ
ขยายเวลาไม่ได้เนื่องจากนาย B จองไว้ต่อน 14.00
นาย A ต้องยอมเสียค่าปรับเพื่อให้ระบบนำเงินไป
ชดเชยให้นาย B

4

การที่เว้นเวลาไว้ 1 ชั่วโมงทำให้เราสามารถมีเวลาให้
นาย B ทราบถึงปัญหาและมีเวลาหาที่จอดใหม่ถึง
45 นาทีเลยทีเดียว

CAMERA

วิธีที่จะทำงานร่วมกับกล้องที่ research ไว้เบื้องต้น

ขั้นตอน: หา API ดึงรูปจากกล้อง

ในบทความนี้จะใช้ภาพนิ่งจากกล้อง Hikvision ซึ่งหลายๆ model ที่ขายในบ้านเรา จะใช้ Pattern ของ url ประมาณนี้:

<http://{Ip Address}/Streaming/channels/{CameraId}/picture>

Note: ยื่ห้ออื่นๆที่น่าจะเจอกันบ่อย
dahua:

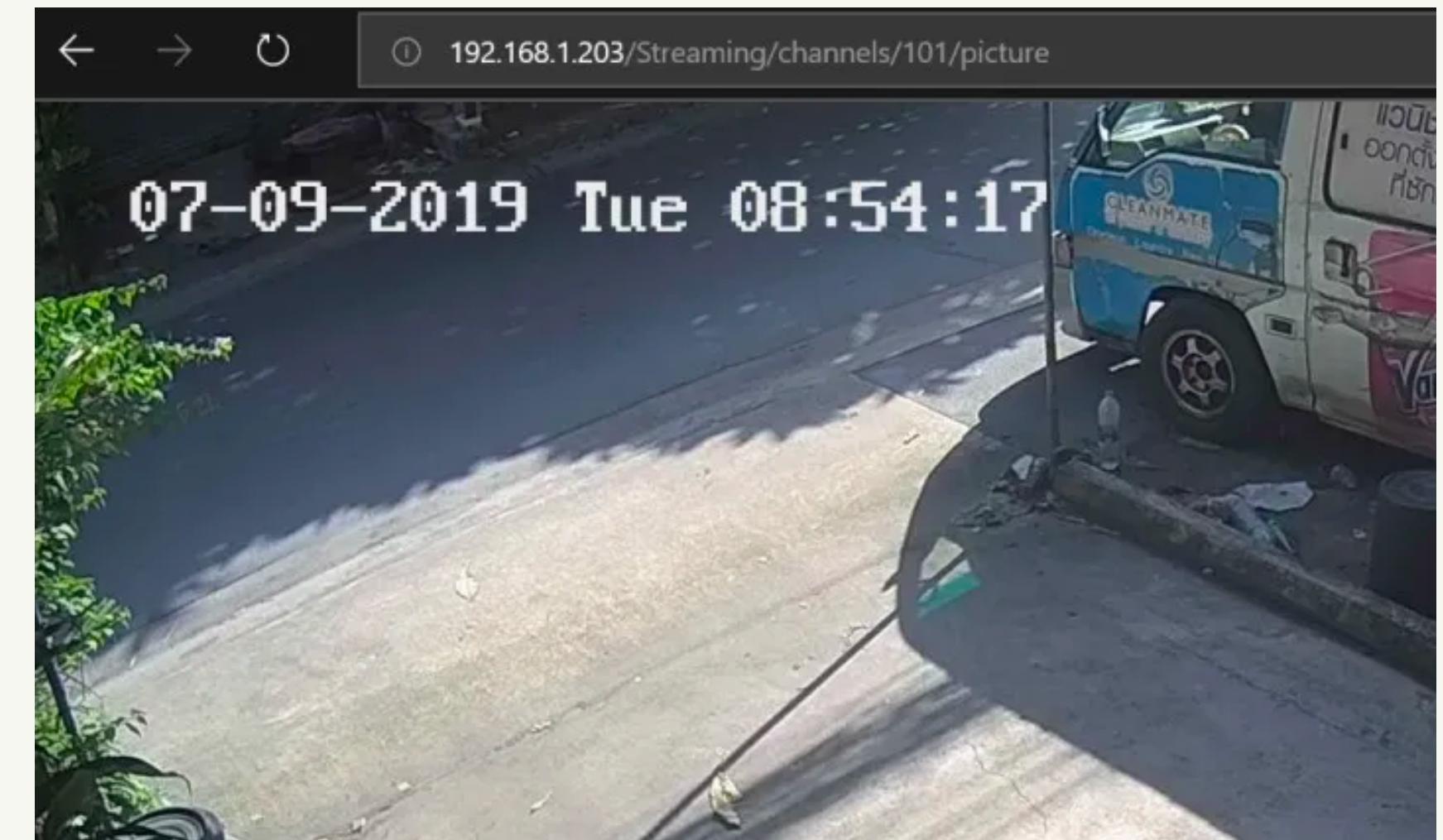
<http://{Ip Address}/cgi-bin/snapshot.cgi>

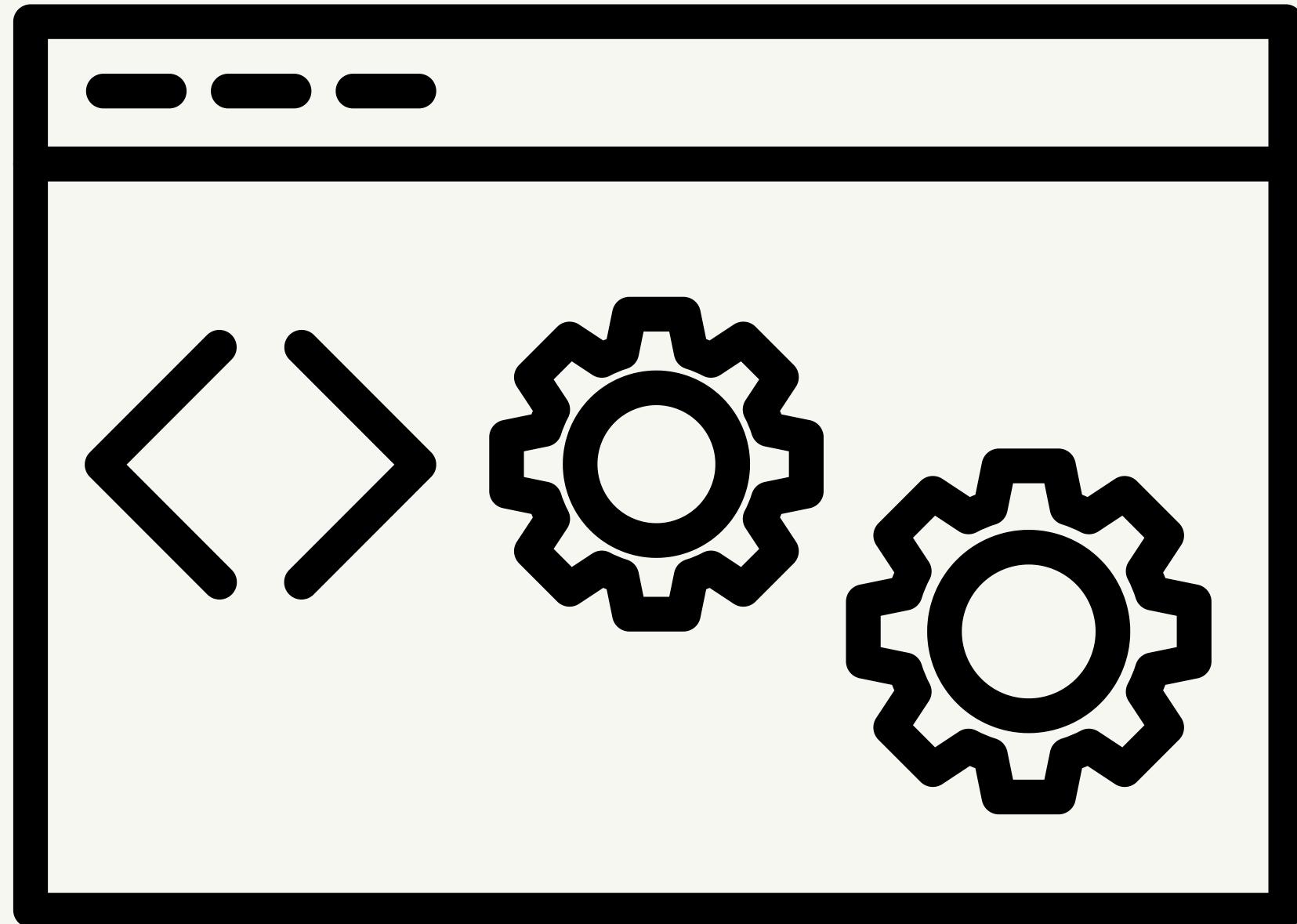
Axis:

<http://{Ip Address}/axis-cgi/jpg/image.cgi?camera={CameraId}&resolution={width}x{height}>

สำหรับยื่ห้อหรือรุ่นอื่นๆเพิ่มเติม [ที่นี่](#)

หา Ip ของกล้องเราให้เจอแล้วลองเรียกดูภาพ จาก url ที่ระบุไว้ผ่าน web browser ดู





BACKEND

Unit Test

API Spec Document
Park Finder

PARKFINDER

Documentation
Version 1.0.0

{HOST} = 35.240.149.20:5009

Login Register Service

API Register

Description : สำหรับทำการลงทะเบียนผู้ใช้งาน

POST

{HOST}/customer/register

Header

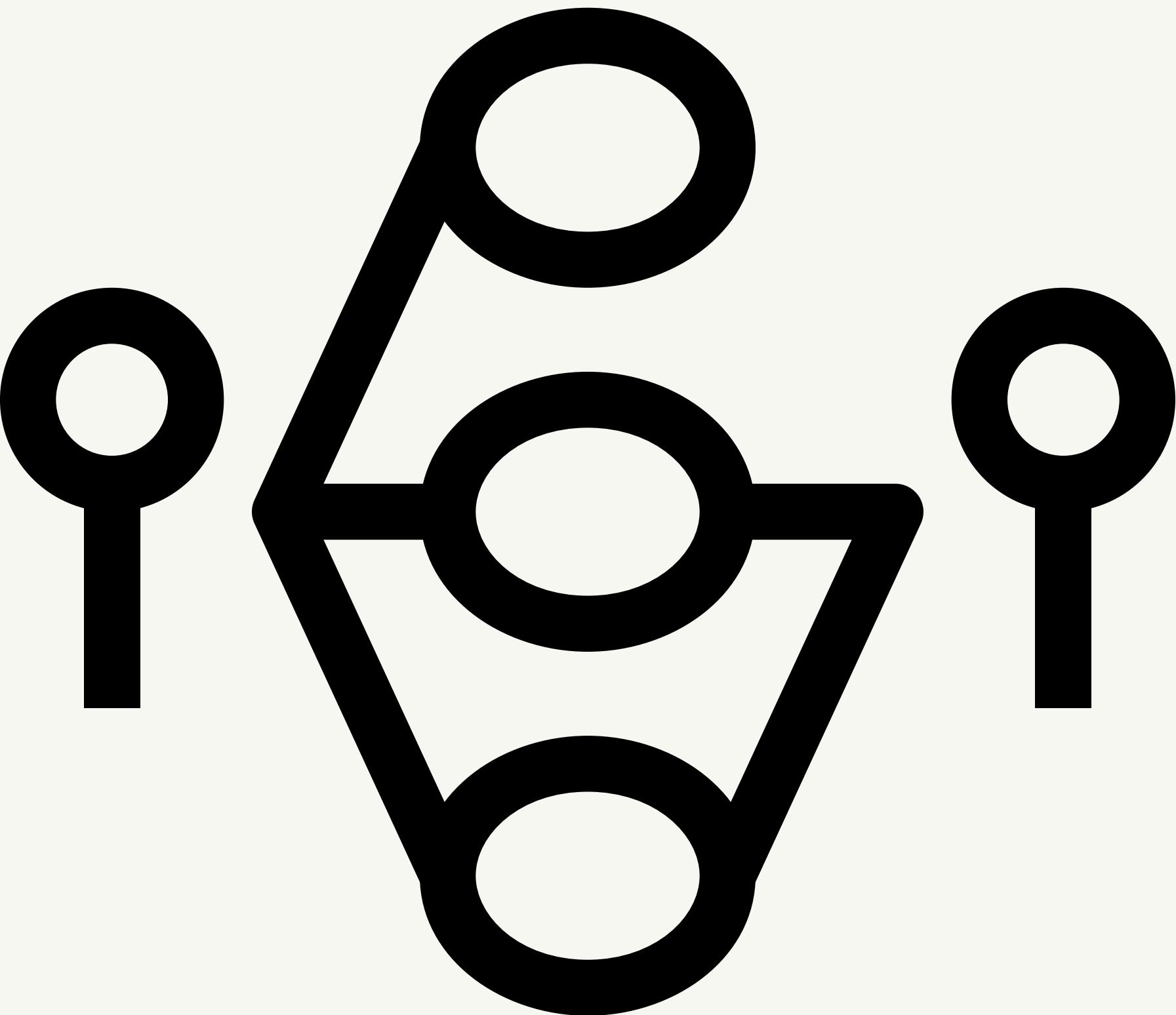
Content-Type: application/json

Request Body :

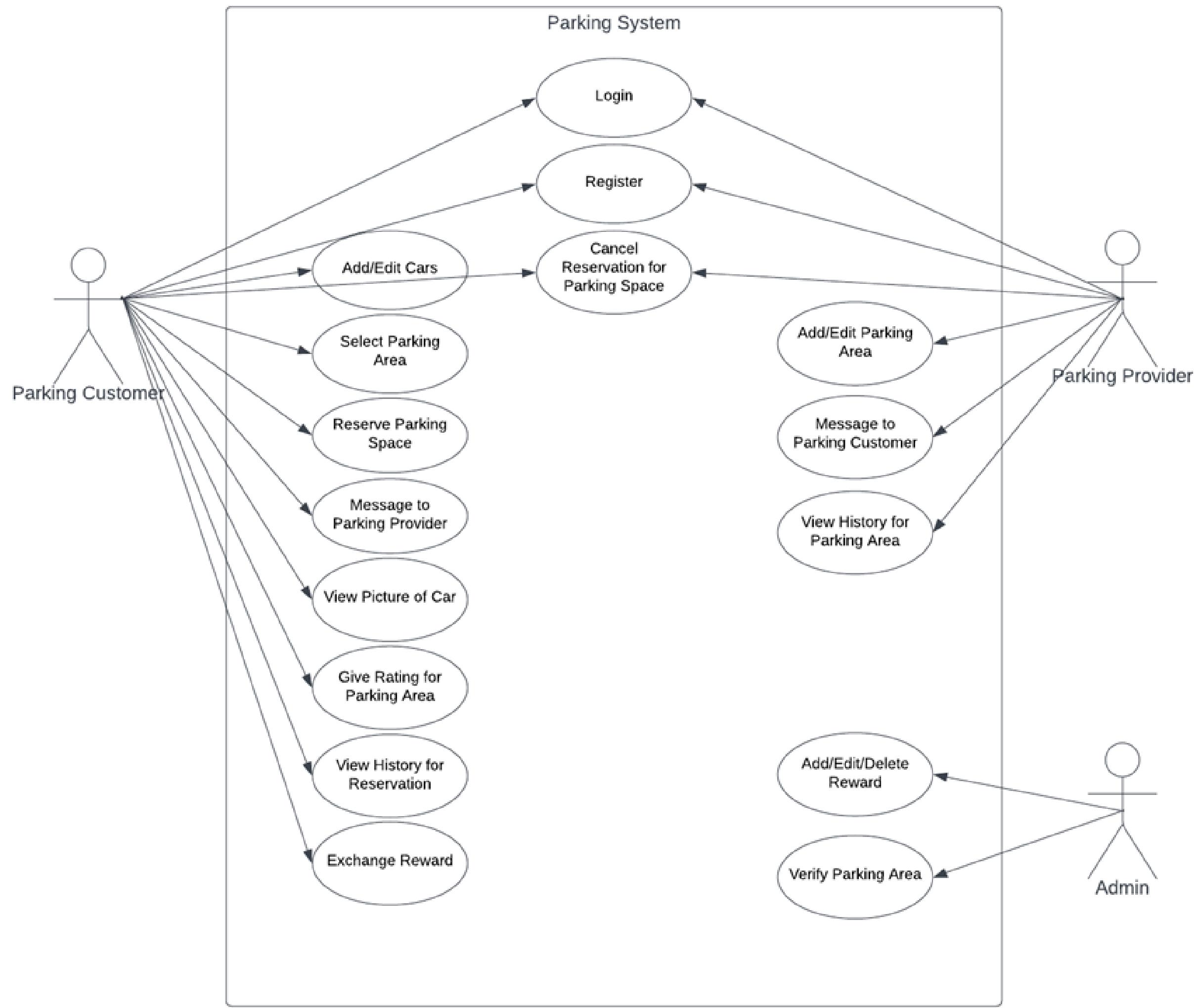
Field name	Data types	Require	Description
first_name	String	Y	ชื่อจริงที่ต้องการลงทะเบียนไว้
last_name	String	Y	นามสกุลที่ต้องการลงทะเบียนไว้
email	String	Y	email ของผู้ใช้งาน
password	String	Y	password ของผู้ใช้งาน

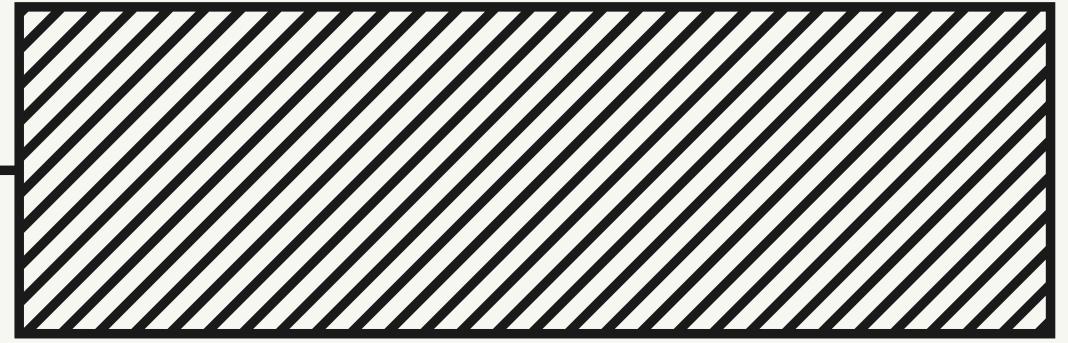
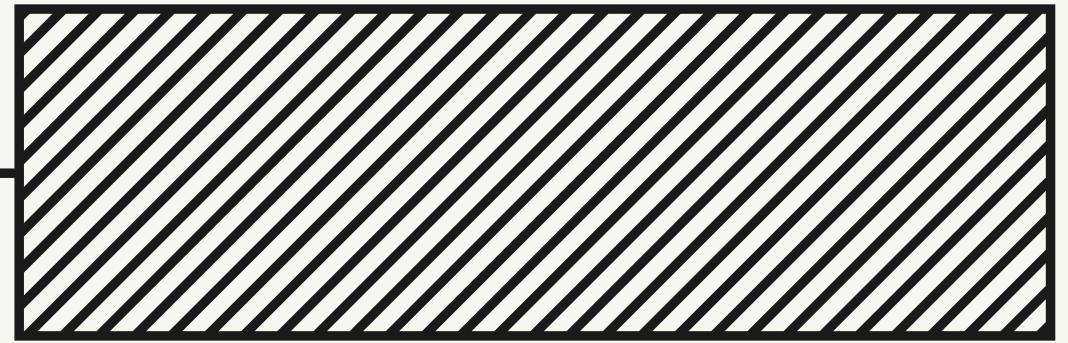
Body example :

```
{  
    "first_name": "develop",  
    "last_name": "develop",  
    "email": "developer@easyparkfinder.com",  
    "phone": "0804560908",  
    "password": "EasyParkFinder"  
}
```



Use Case Diagram





DFD

context diagram

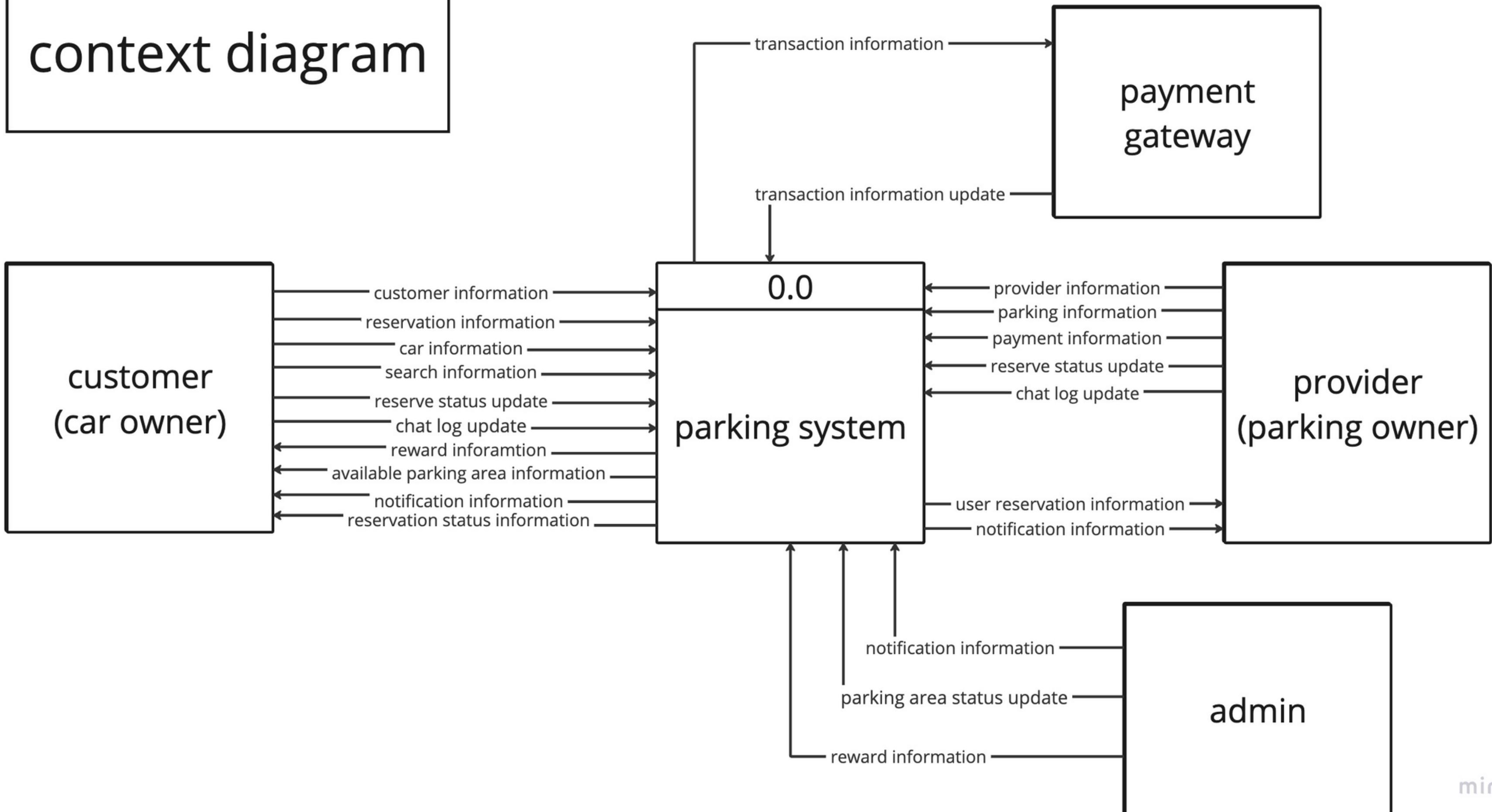
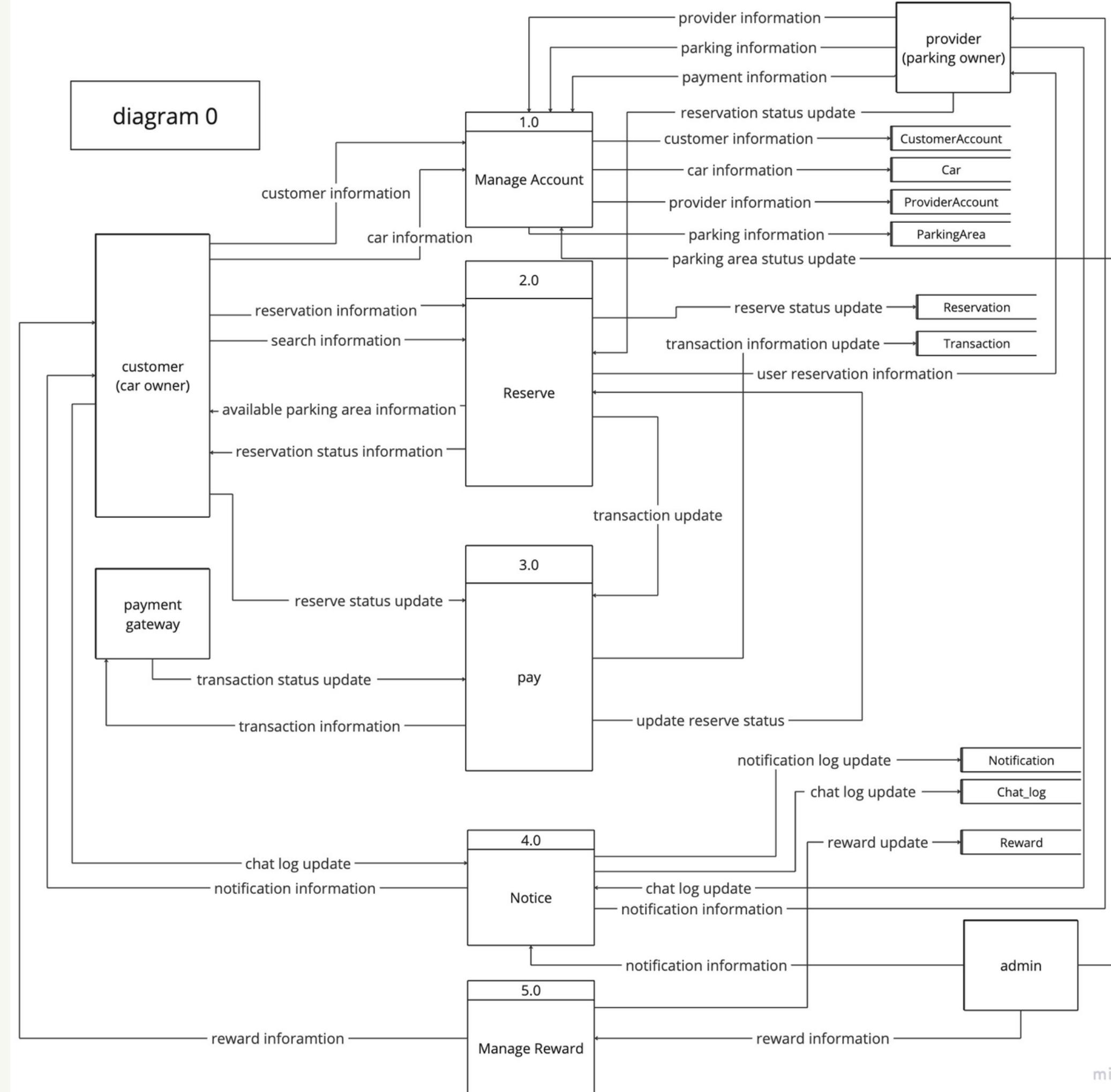
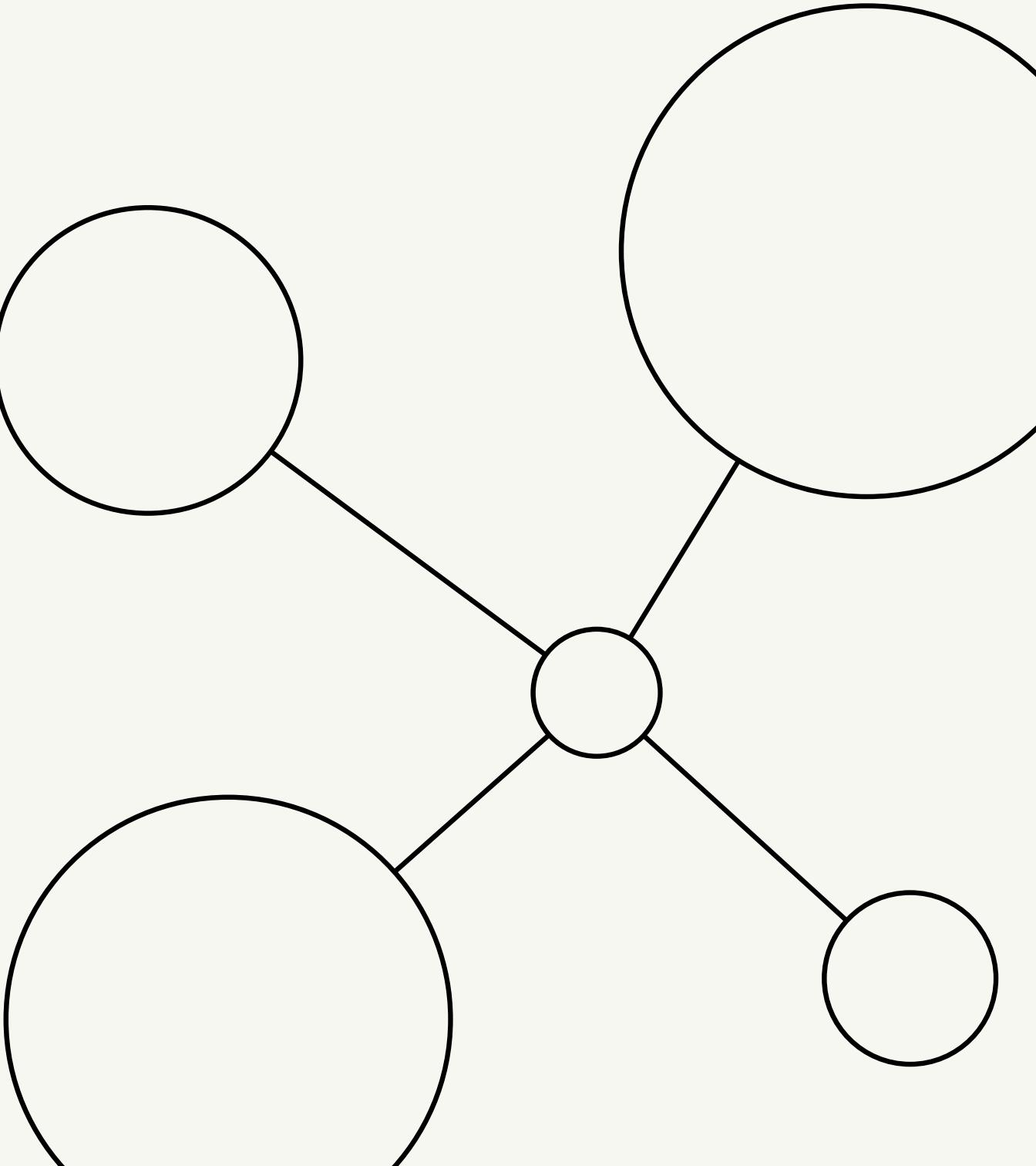
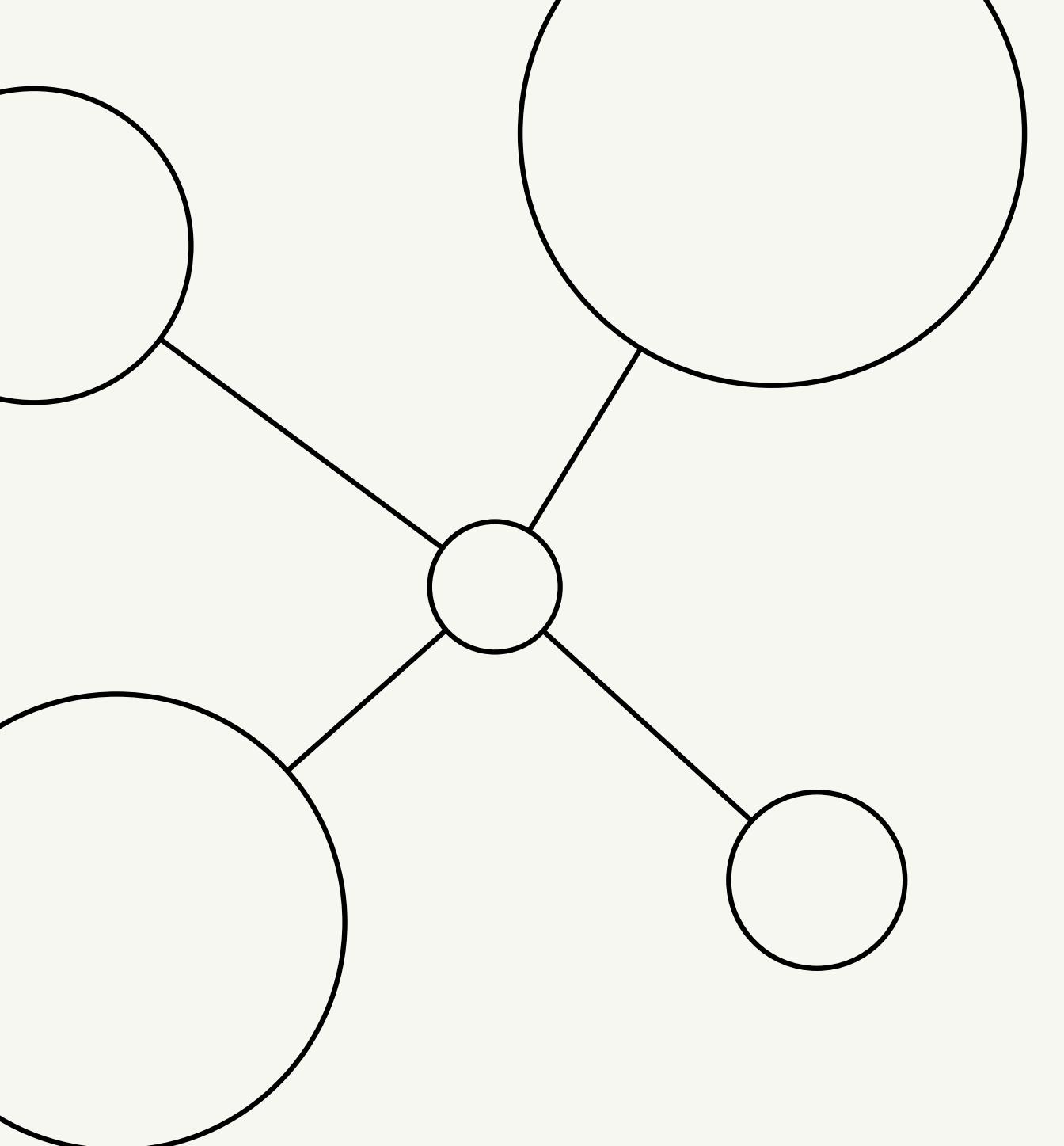


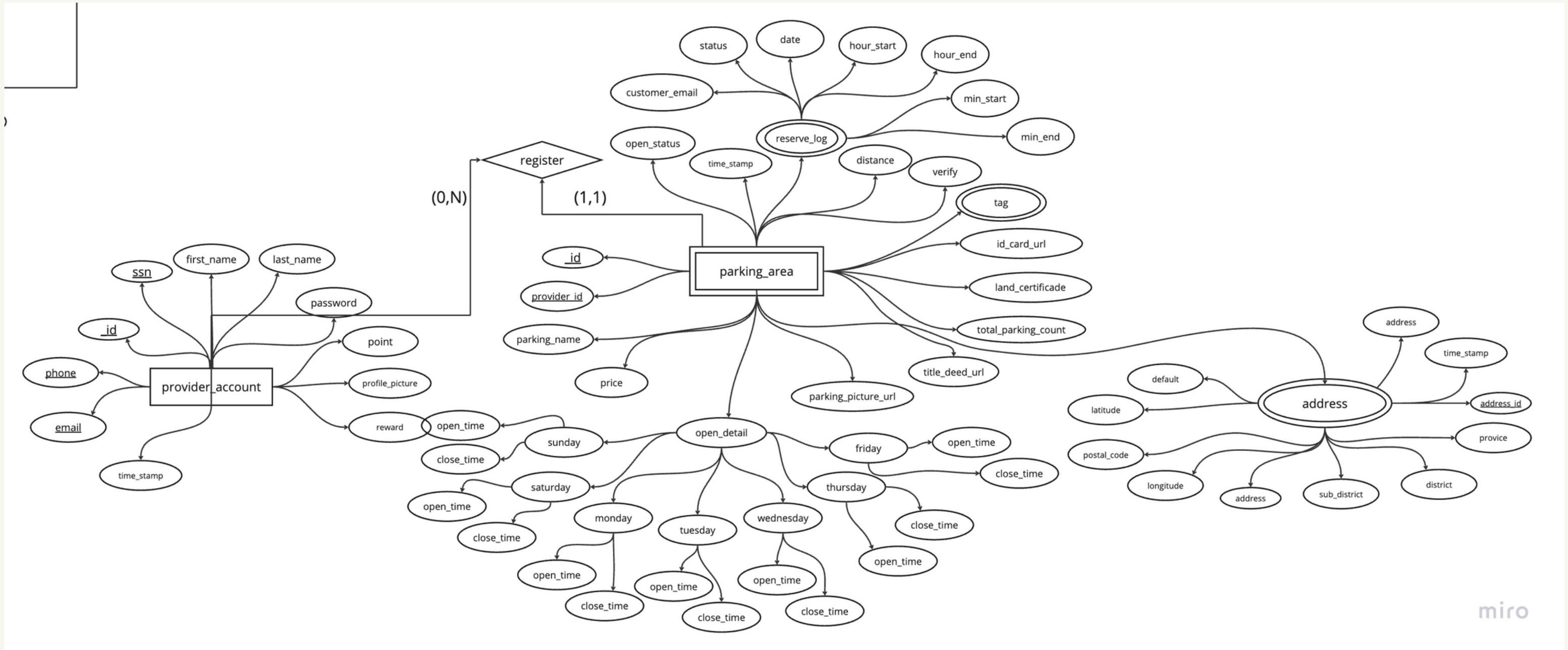
diagram 0



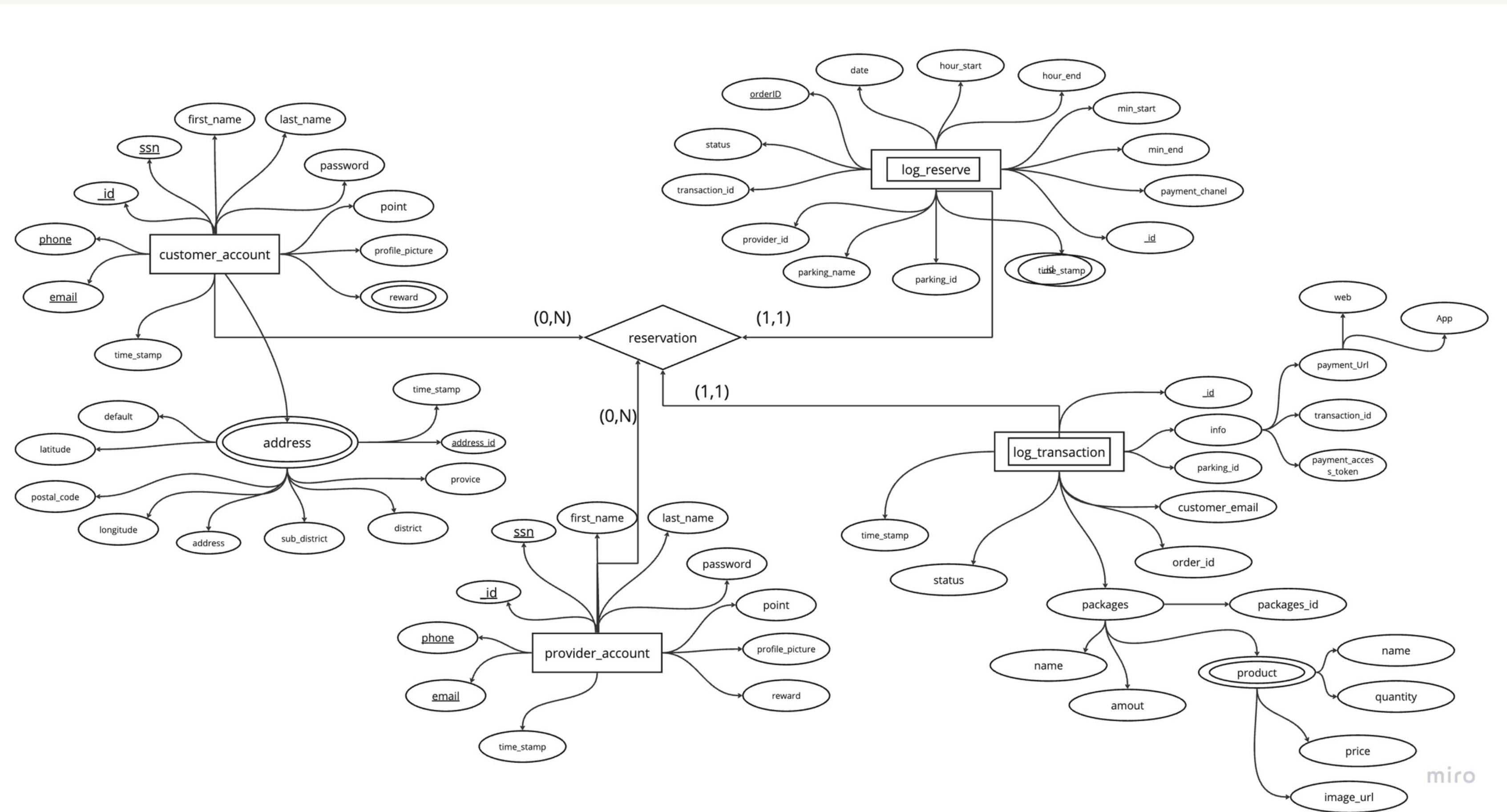


ER

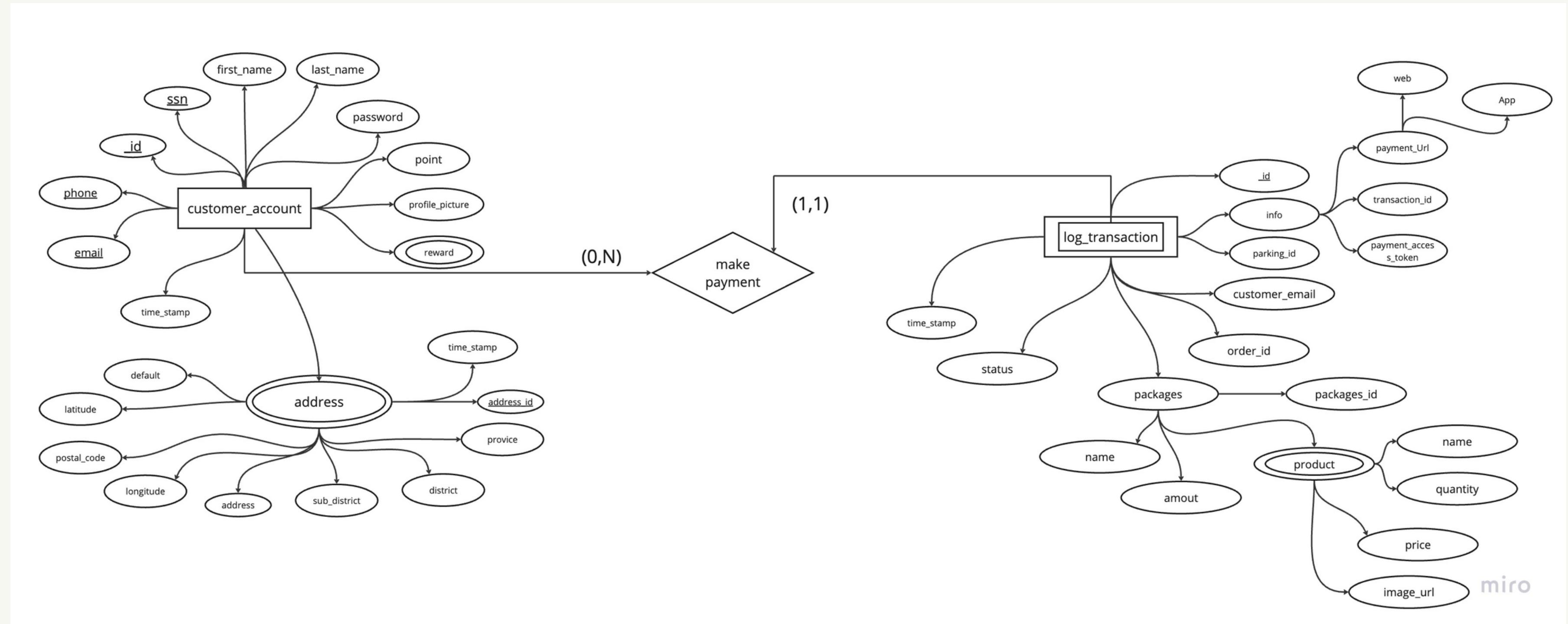
Register (ລົງທະເບີຍນິ້ວດຣອ)



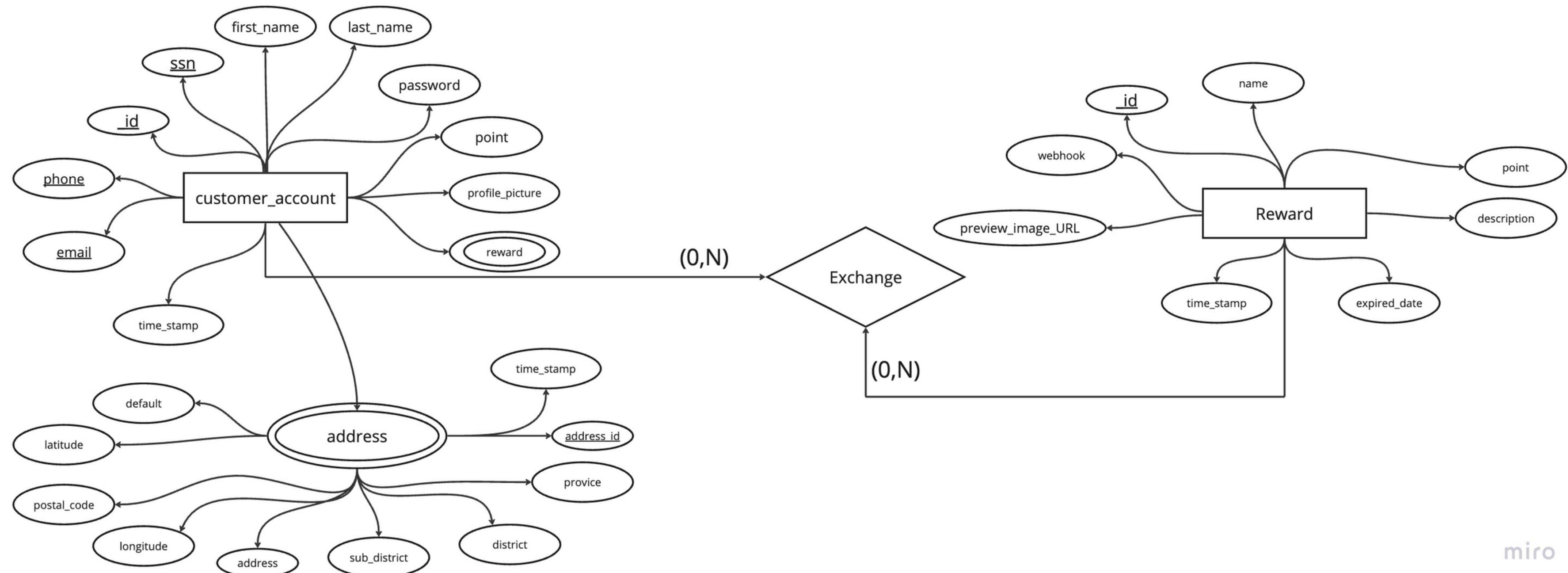
Reservation (ກາຣຄອງ)



Make Payment(សរាយ transaction)



Exchange(ແລກ Reward)

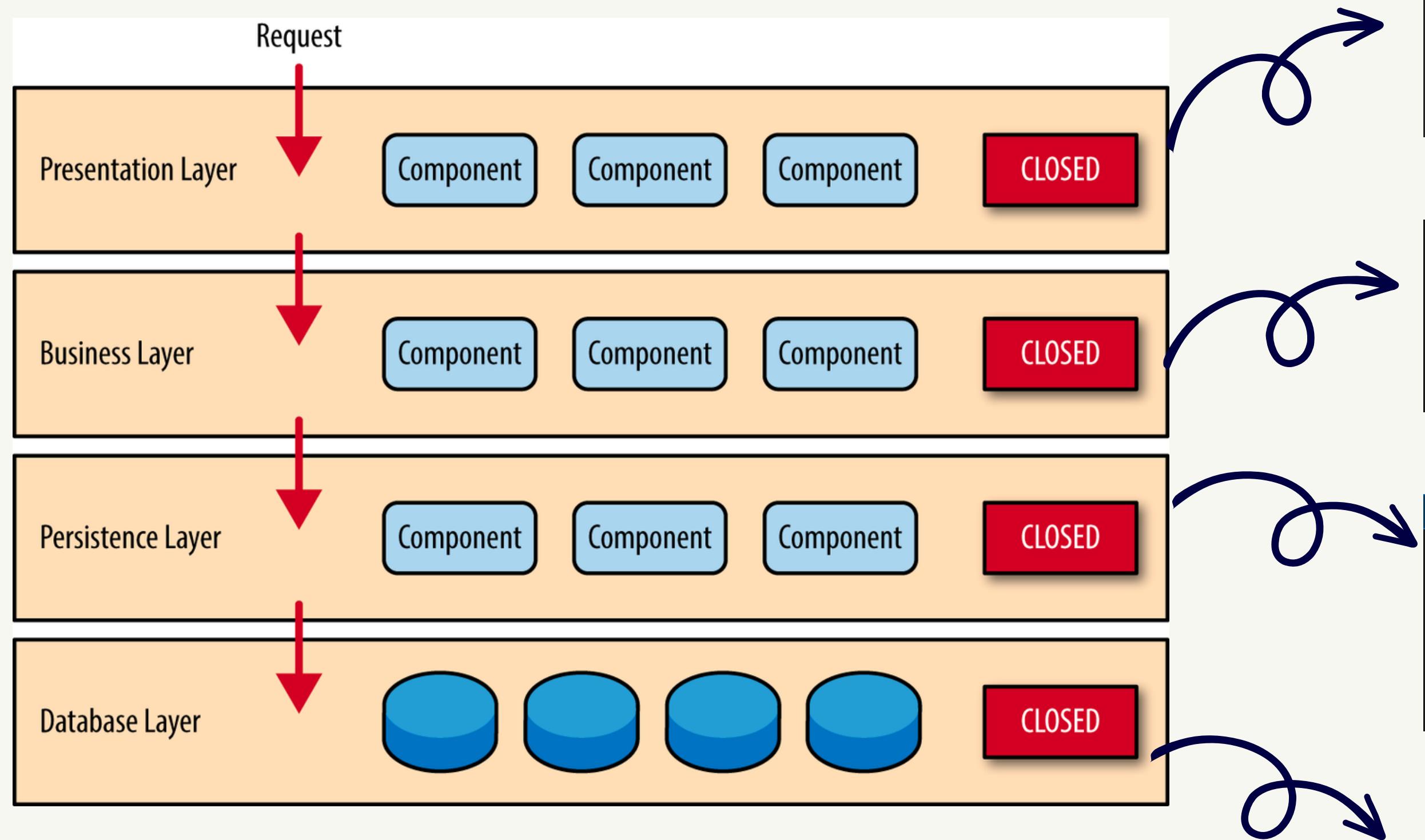




Architecture & Design Pattern

Architecture

Layered Architecture:



```
✓ routers
-auth_router.go
-customer_router.go
-factory.go
-payment_router.go
-reserve_router.go
```

```
✓ services
-auth_service.go
-customer_service.go
-factory.go
```

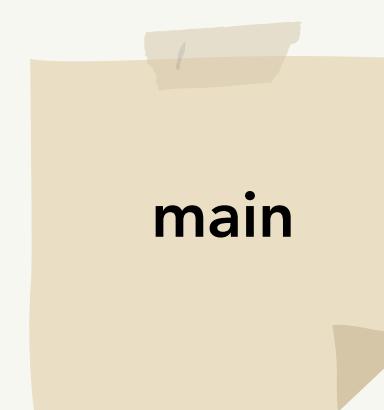
```
✓ storage
-account_storage.go
-car_storage.go
-log_storage.go
-parking_area_storage.go
-reward_storage.go
-token_storage.go
```



Design Pattern

Dependency Injection

cS := cs.NewCustomerServices(db, redis)



```
result, err := cs.CustomerAccoutStorage.UpdateAccountInterface(ct)
if err != nil {
    return err
}

func (cs CustomerServices) SaveOTP(email string, otp string) error {
    key := fmt.Sprintf(redisOTPKeyFormat, email)

    // Cache for 10 minute
    result := cs.Redis.Set(key, otp, 120)
    err := result.Err()
    if err != nil {
        return err
    }
    fmt.Println("Cache user otp for email:", email)

    return nil
}
```

service

Factory Method Pattern

ນໍາ service ໄປໃສ່ເປັນ parameter ຂອງ Router

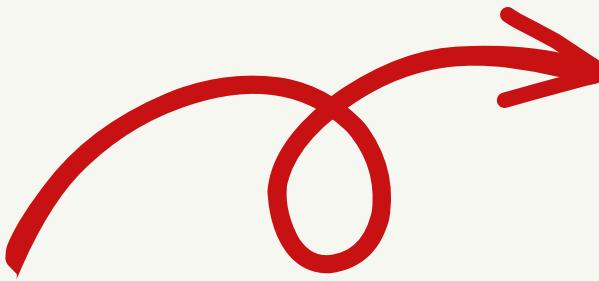
main

```
// Setup new router
cr.NewCustomerRouter(gCustomer, customerServices, searchServices, paymentServices, reserveServices)
```

router

```
func NewCustomerRouter(g *echo.Group, cs cs.ICustomerServices, ss ss.ISearchServices, pms pms.IPaymentServices, rss rss.IReserveServices) {
    cr := CustomerRouters{
        CustomerServices: cs,
        SearchServices:   ss,
        PaymentService:   pms,
        ReserveService:   rss,
    }
}
```

Factory Method Pattern



ประกาศไว้ที่ service
(Business Layer)

```
type ICustomerServices interface {
    CheckExistEmail(ctx context.Context, email string) *models.CustomerAccount
```

```
func (cr CustomerRouters) customerRegisterHandler(c echo.Context) error {
    context := c.Request().Context()

    request := new(models.RegisterAccountRequest)
    if err := c.Bind(request); err != nil {
        return c.JSON(http.StatusBadRequest, models.MessageResponse{Message: err.Error()})
    }

    account := cr.CustomerServices.CheckExistEmail(context, request.Email)
    if account != nil {
        return c.JSON(http.StatusConflict, models.MessageResponse{Message: "This email is already exists"})
    }
}
```

router
(presentation layer)
ที่ใช้พารามิเตอร์
Customer Service ไว้
ทำให้สามารถใช้งาน
ฟังก์ชันของ interface
Customer Service ได้

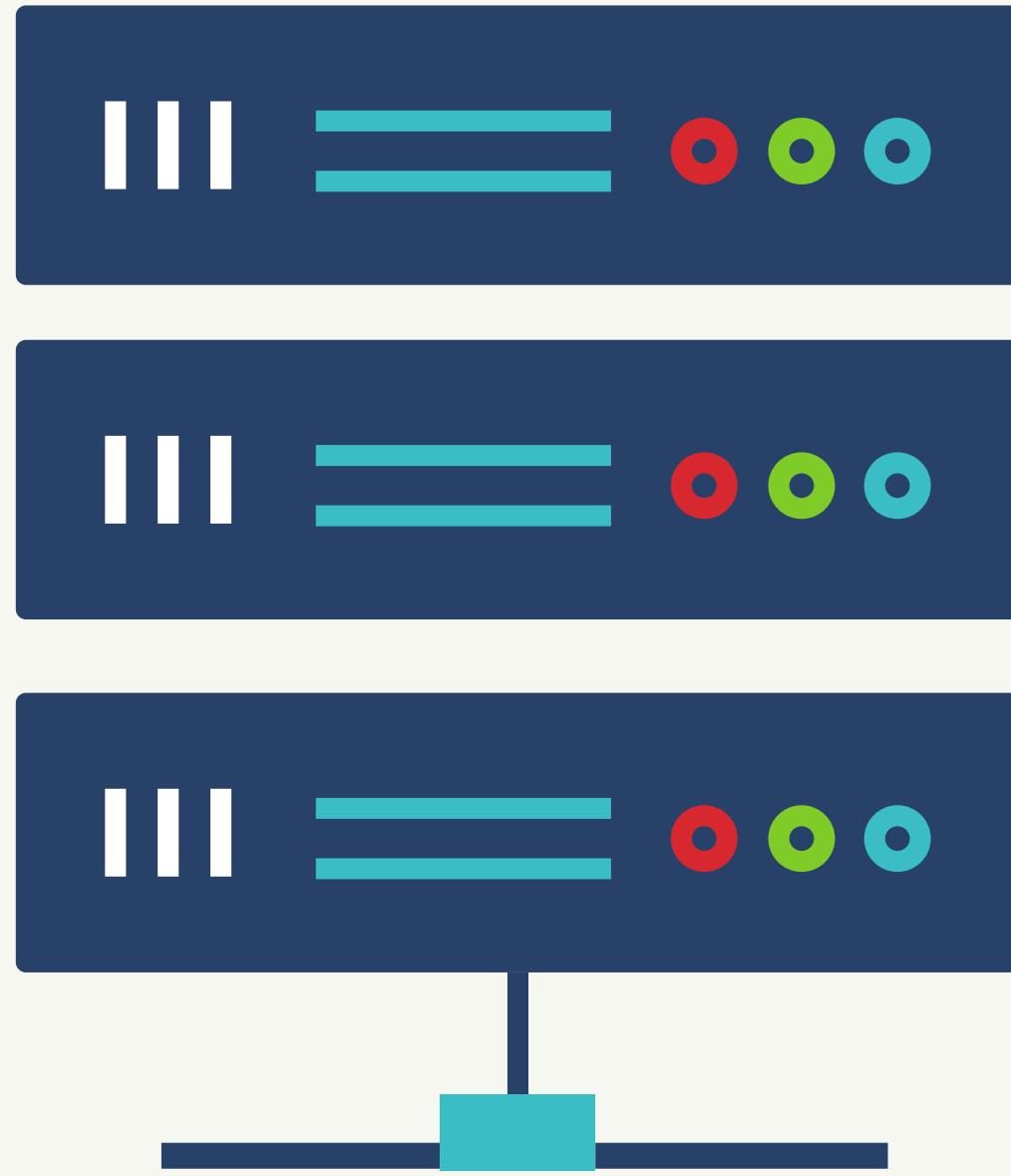
service

Repository Pattern

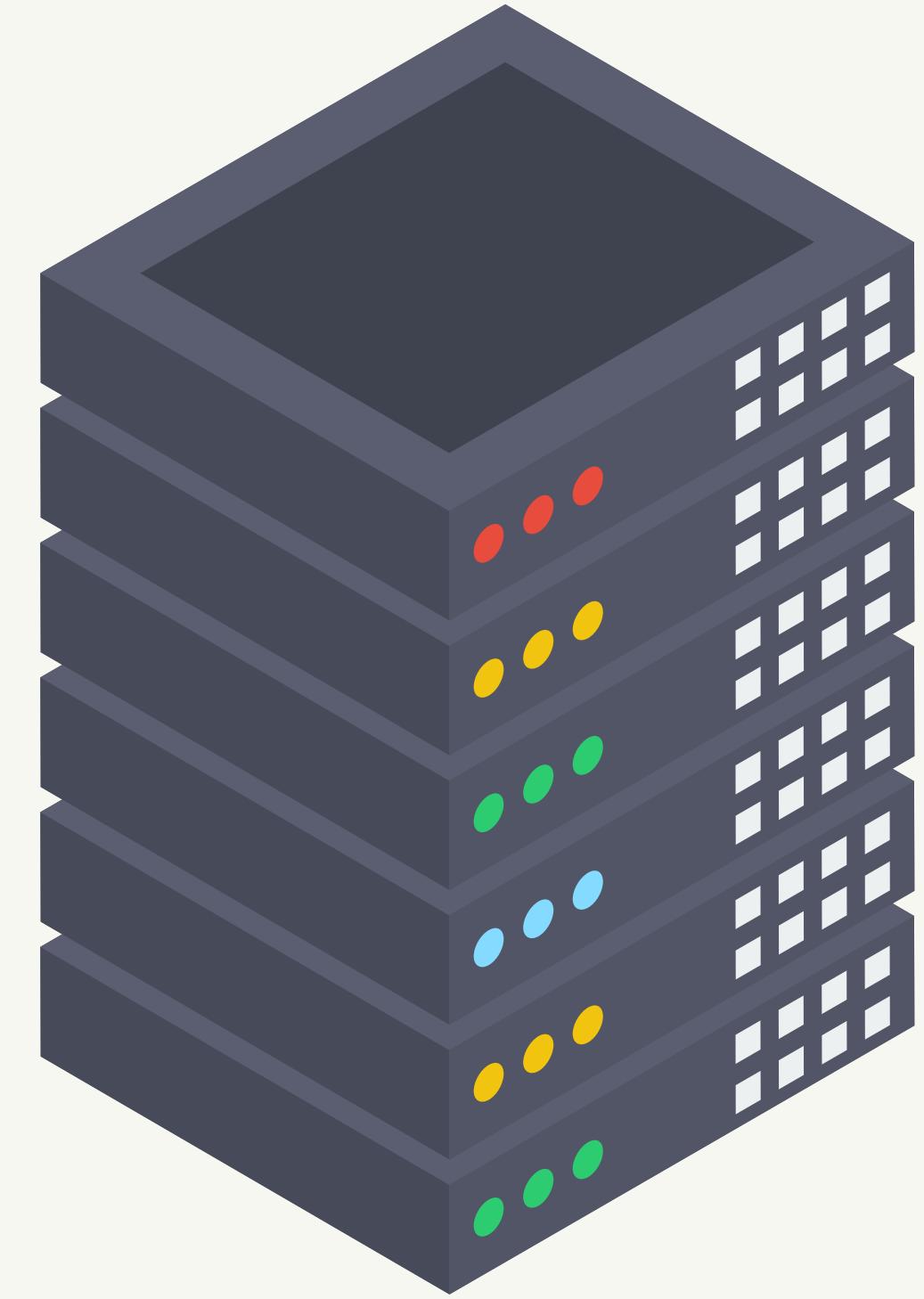
```
func NewCustomerServices(  
    db *mongo.Database,  
    rd *connector.Redis,  
) ICustomerServices {  
    customer_account_storage := storage.NewCustomerAccoutStorage(db)  
    token_storage := storage.NewTokenStorage(db)  
    car_storage := storage.NewCarStorage(db)  
    return CustomerServices{  
        CustomerAccoutStorage: customer_account_storage,  
        TokenStorage: token_storage,  
        CarStorage: car_storage,  
        Redis: rd,  
    }  
}
```

```
✓ storage  
-GO account_storage.go  
-GO car_storage.go  
-GO log_storage.go  
-GO parking_area_storage.go  
-GO reward_storage.go  
-GO token_strorage.go
```

```
func NewAdminAccoutStorage(db *mongo.Database) *AccoutStorage {  
    return &AccoutStorage{  
        Collection: db.Collection(os.Getenv("COLLECTION_ADMIN_ACCOUNT_NAME")),  
    }  
}  
  
func (cs AccoutStorage) InsertAccount(ctx context.Context, data interface{}) (*mongo.InsertOneResult, error) {  
    result, err := cs.Collection.InsertOne(ctx, data)  
    return result, err  
}  
  
func (cs AccoutStorage) UpdateAccountInterface(ctx context.Context, filter interface{}, update interface{}) (  
    result, err := cs.Collection.UpdateOne(ctx, filter, update)  
    if err != nil {  
        fmt.Println(err)  
    }  
    fmt.Println("Modified count:", result.ModifiedCount)  
    return result, err  
}
```



CI&CD



1

เขียน Dockerfile เพื่อสร้าง image ของแต่ละ service

 Dockerfile.cronjob

 Dockerfile.api

2

เขียน .gitlab-ci.yml โดยแบ่ง stages เป็น 2 ส่วนคือ build และ deploy

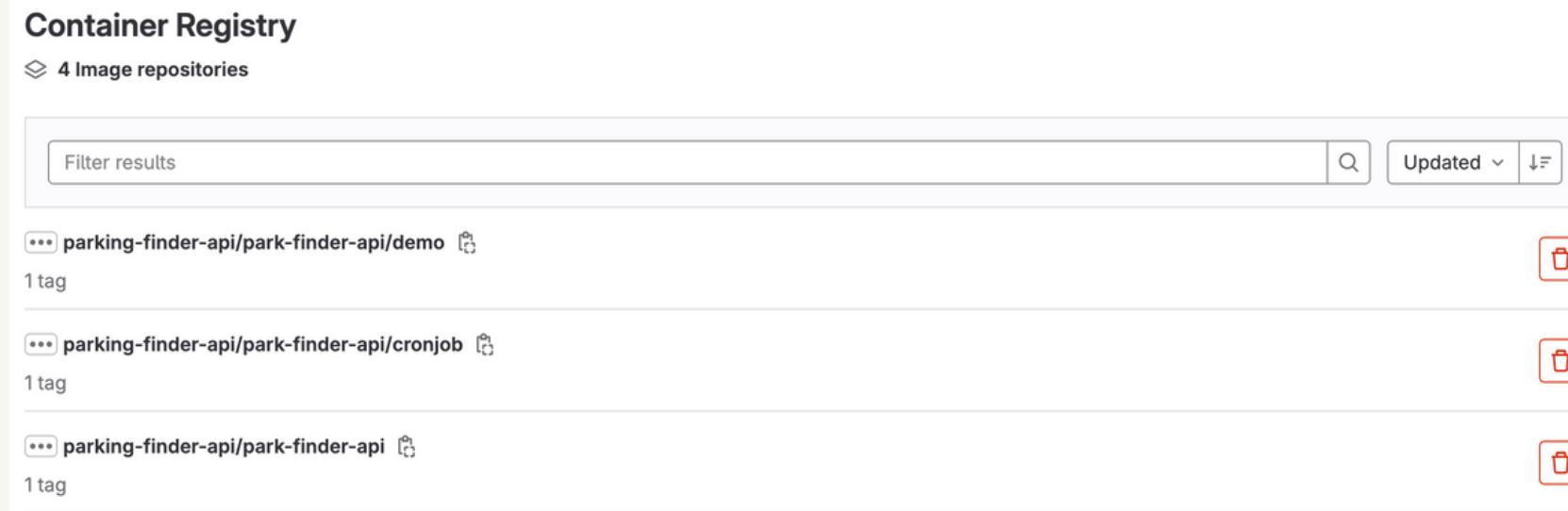
```
↳ .gitlab-ci.yml
1 stages:
2   - staging-build
3   - staging-deploy
4
5 > staging-build: ...
29
30 > staging-deploy: ...
49
```

- build
เพื่อ build docker
image ลงใน
gitlab

- deploy

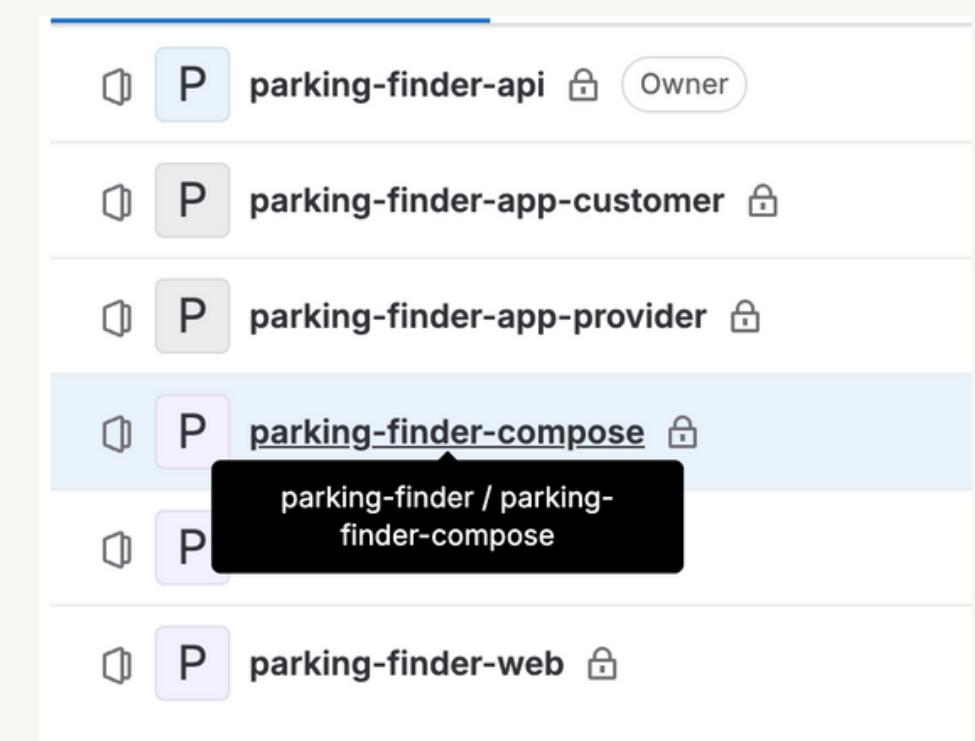
3

หลังจากนั้นจะเห็นได้ว่ามี docker image ที่พึ่ง build ไปใน gitlab



4

สร้าง repo compose แยกสำหรับให้ vm clone มาใช้งาน
ใน repo นี้จะประกอบไปด้วย file docker-compose
ที่ทำการสร้าง container ที่ build docker image ใน
gitlab และ deploy script ที่กำหนดการลบ old
container และสร้าง container ใหม่จาก file docker-
compose



5

สร้าง instances ใน compute engine ของ gcp และทำการติดตั้ง docker และ docker compose ให้เรียบร้อย และ clone repo compose มา

6

นำค่า target ของ instances และ ทำการ generate ssh key เพื่อนำไปใส่ใน Variables ของ gitlab

CI/CD Variables </> 2			
Key	Value	Environments	Actions
SSH_PRIVATE	*****	All (default)	 
TARGET_STAGING	*****	All (default)	 

7

เมื่อเริ่ม deploy(push code ลง branch staging) ตัว gitlab จะดำเนินการตาม stages ที่เรากำหนดไว้ และเมื่อถึง stages deploy จะเรียกใช้จาก variables ที่เรากำหนดไว้ ในการเข้าถึง instance เพื่อเรียกใช้ deploy script

8

เมื่อ deploy ผ่านจะปรากฏ container ทั้งหมดที่เราเขียนไว้ในไฟล์ docker compose

NAMES						
e4fd10032568	amir20/dozzle:latest		"/dozzle"	5 days ago	Up 5 days	0.0.0.0:9999->8080/tcp, :::
9999->8080/tcp	dozzle		"python app.py"	5 days ago	Up 5 days	0.0.0.0:4500->4500/tcp, :::
b49ff5d66fe2	registry.gitlab.com/parking-finder/parking-finder-api/park-finder-api/demo:staging		"/parking-finder-api"	5 days ago	Up 5 days	0.0.0.0:5009->5009/tcp, :::
4500->4500/tcp	parking-finder-demo		"python app.py"	5 days ago	Up 5 days	0.0.0.0:4200->4200/tcp, :::
d17c38b53ed2	registry.gitlab.com/parking-finder/parking-finder-api/park-finder-api:staging		"docker-entrypoint.s..."	5 days ago	Up 5 days	6379/tcp
5009->5009/tcp	parking-finder-api					
08fb69729a9b	registry.gitlab.com/parking-finder/parking-finder-api/park-finder-api/cronjob:staging					
4200->4200/tcp	parking-finder-cronjob					
38318c8df408	redis:alpine					
	parking-finder-api_redis_1					

port 5009

เป็น main process ที่เขียนโดย golang ที่ใช้ Echo

port 4200

```
Start scheduler...
All routes
{'GET', 'OPTIONS', 'HEAD'} /static/<path:filename>
{'GET', 'OPTIONS', 'HEAD'} /job/
{'OPTIONS', 'POST'} /job/cancel_reserve
{'OPTIONS', 'POST'} /job/timeout_reserve
{'OPTIONS', 'POST'} /job/before_timeout_reserve
{'OPTIONS', 'POST'} /job/extend_reserve
{'OPTIONS', 'POST'} /job/remove_job
* Serving Flask app 'src' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:4200
* Running on http://172.17.0.2:4200
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 141-942-053
```

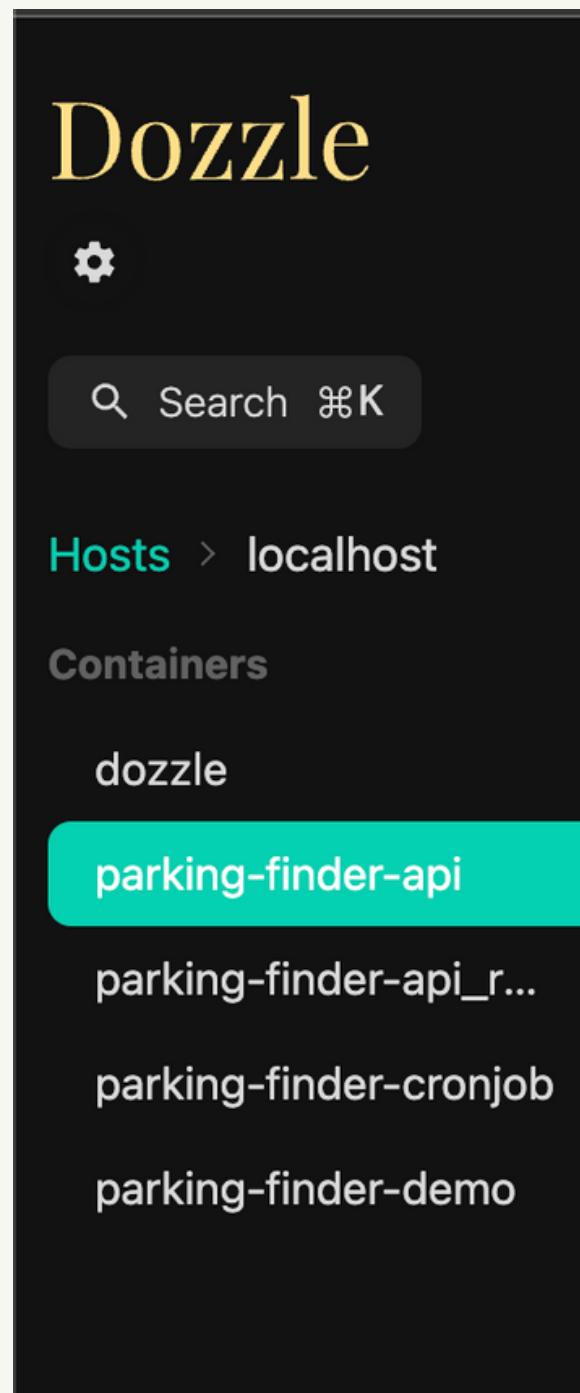
ເປັນ background process ທີ່ເຂົ້າໃຈໂດຍ python ທີ່ໃຊ້
Flask

port 6379

```
kritsanaphat@park-finder:~$ docker logs 38318c8df408
1:C 06 Nov 2023 11:41:43.622 * o00Oo00Oo00Oo Redis is starting o00o000o000o
1:C 06 Nov 2023 11:41:43.624 * Redis version=7.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 06 Nov 2023 11:41:43.624 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 06 Nov 2023 11:41:43.625 * monotonic clock: POSIX clock_gettime
1:M 06 Nov 2023 11:41:43.640 * Running mode=standalone, port=6379.
1:M 06 Nov 2023 11:41:43.640 * Server initialized
1:M 06 Nov 2023 11:41:43.641 * Ready to accept connections tcp
kritsanaphat@park-finder:~$
```

ສໍາຮຽ redis

port 9999



เป็น service ของ dozzle เพื่อดู log การทำงาน

ทำไมถึงใช้ NOSQL

- เนื่องจากตอนออกแบบระบบ คำถึงถึงว่าการระบบของที่จอดรถ ต้องมีการเข้าถึงอัพเดตค่า status ต่าง ๆ อยู่บ่อย ๆ รวมถึงมีระบบสนับนา real-time ซึ่งเข้าใจว่าจะเหมาะสมกว่า
- เคยใช้บ่อยกว่า SQL

