**System Requirements**

1)

Name: Register

Description: A new user creates an account to access the University Car Park application's features.

Operation type: Command

Roles: Guest

Pre-condition:

+User's account does not exist

Post-condition:

+User's account is created

Request:

+password

+email

REST API endpoint:

+POST /register

Request: {

"password": String,

"email": String

}

Response:

200 OK

2)  
Name: Reserve Parking Spot

Description: Users can reserve a parking spot in the University Car Park.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

+Parking spot is available

Post-condition:

+Parking spot is reserved

Request:

+date

+time

+parking spot ID

REST API endpoint:

+POST /reserve

Request: {

"date": String,

"time": String,

"parking\_spot\_id": String

}

Response:

200 OK

3)  
Name: View Parking Availability

Description: Users can view the availability of parking spots in the University Car Park.

Operation type: Query

Roles: Registered User, Guest

Pre-condition:

+none

Post-condition:

+none

Request:

+date

+time

+parking spot ID

REST API endpoint:

+GET /availability

Request: {

"date": String,

"time": String

}

Response:

200 OK

{

"available\_spots": Array

}

4)

Name: Cancel Reservation

Description: Users can cancel their parking spot reservation.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

+Reservation exists for the user

Post-condition:

+Reservation is canceled

Request:

+reservation ID

REST API endpoint:

+DELETE /reservation

Request: {

"reservation\_id": String

}

Response:

200 OK

5)

Name: Modify Reservation

Description: Users can modify their existing parking spot reservation.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

+Reservation exists for the user

Post-condition:

+Reservation is updated

Request:

+reservation ID

+new date

+new time

+new parking spot ID

REST API endpoint:

+PUT /reservation

Request: {

"reservation\_id": String,

"new\_date": String,

"new\_time": String,

"new\_parking\_spot\_id": String

}

Response:

200 OK

6)

Name: View Reservation History

Description: Users can view their past and current reservations.

Operation type: Query

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+none

Request:

+none

REST API endpoint:

+GET /reservations/history

Request: {

}

Response:

200 OK

{

"reservations": Array

}

7)

Name: Rate Parking Experience

Description: Users can provide feedback and rate their parking experience at the University Car Park.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

+User has completed a reservation

Post-condition:

+User's feedback and rating are saved

Request:

+reservation ID

+rating (1-5)

+comment (optional)

REST API endpoint:

+POST /feedback

Request: {

"reservation\_id": String,

"rating": Integer,

"comment": String (optional)

}

Response:

200 OK

8)

Name: View Parking Rates

Description: Users can view the parking rates for the University Car Park.

Operation type: Query

Roles: Registered User, Guest

Pre-condition:

+none

Post-condition:

+none

Request:

+none

REST API endpoint:

+GET /parking-rates

Request: {

}

Response:

200 OK

{

"parking\_rates": Array

}

9)

Name: View Parking Regulations

Description: Users can view the parking regulations for the University Car Park.

Operation type: Query

Roles: Registered User, Guest

Pre-condition:

+none

Post-condition:

+none

Request:

+none

REST API endpoint:

+GET /parking-regulations

Request: {

}

Response:

200 OK

{

"parking\_regulations": Array

}

10)

Name: Report an Issue

Description: Users can report any issues related to the University Car Park, such as parking violations or maintenance needs.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+Issue is reported and recorded

Request:

+issue type

+description

+location (optional)

+image (optional)

REST API endpoint:

+POST /report-issue

Request: {

"issue\_type": String,

"description": String,

"location": String (optional),

"image": String (optional)

}

Response:

200 OK

11)  
Name: View Car Park Map

Description: Users can view a map of the University Car Park, showing available and reserved parking spots.

Operation type: Query

Roles: Registered User, Guest

Pre-condition:

+none

Post-condition:

+none

Request:

+none

REST API endpoint:

+GET /car-park-map

Request: {

}

Response:

200 OK

{

"car\_park\_map": Object

}

12)

Name: Forgot Password

Description: Users can request a password reset link to their registered email address.

Operation type: Command

Roles: Guest

Pre-condition:

+User's account exists

Post-condition:

+Password reset link is sent to the user's email address

Request:

+email

REST API endpoint:

+POST /forgot-password

Request: {

"email": String

}

Response:

200 OK

13)  
Name: Reset Password

Description: Users can reset their password using the link sent to their registered email address.

Operation type: Command

Roles: Guest

Pre-condition:

+User has requested a password reset

+Password reset link is valid

Post-condition:

+User's password is updated

Request:

+token

+new password

REST API endpoint:

+PUT /reset-password

Request: {

"token": String,

"new\_password": String

}

Response:

200 OK

14)  
Name: Update Profile

Description: Users can update their profile information, such as their name, contact number, and vehicle details.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+User's profile information is updated

Request:

+name (optional)

+contact\_number (optional)

+vehicle\_details (optional)

REST API endpoint:

+PUT /update-profile

Request: {

"name": String (optional),

"contact\_number": String (optional),

"vehicle\_details": Object (optional)

}

Response:

200 OK

15)

Name: View Profile

Description: Users can view their profile information, such as their name, contact number, and vehicle details.

Operation type: Query

Roles: Registered User

Pre-condition:

+None

Post-condition:

+None

Request:

+None

REST API endpoint:

+GET /profile

Request: {

}

Response:

200 OK

{

"profile": Object

}

16)

Name: Get Notifications

Description: Users can receive notifications about important updates, such as reservation status changes or car park announcements.

Operation type: Query

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+None

Request:

+None

REST API endpoint:

+GET /notifications

Request: {

}

Response:

200 OK

{

"notifications": Array

}

17)

Name: Add Payment Method

Description: Users can add a payment method to their account for paying parking fees.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+Payment method is added to the user's account

Request:

+card\_number

+expiry\_date

+security\_code

+cardholder\_name

REST API endpoint:

+POST /payment-method

Request: {

"card\_number": String,

"expiry\_date": String,

"security\_code": String,

"cardholder\_name": String

}

Response:

200 OK

18)

Name: Delete Payment Method

Description: Users can delete a payment method from their account.

Operation type: Command

Roles: Registered User

Pre-condition:

+User is logged in

+Payment method exists in the user's account

Post-condition:

+Payment method is added to the user's account

Request:

+payment\_method\_id

REST API endpoint:

+DELETE /payment-method

Request: {

"payment\_method\_id": String

}

Response:

200 OK

19)

Name: List Payment Methods

Description: Users can view the list of payment methods associated with their account.

Operation type: Query

Roles: Registered User

Pre-condition:

+User is logged in

Post-condition:

+none

Request:

+none

REST API endpoint:

+GET /payment-methods

Request: {

}

Response:

200 OK

{

"payment\_methods": Array

}