Blockchain-based parking app (dApp)

Requirements engineering, analysis and design
Stakeholders and their dependencies

- Owner
  - Owner of the parking spot
    - Register parking spot and specify its availability
- Renter
  - User of the parking spot
    - Search and reserve parking spot
    - Start/end parking session
    - Extend parking session
    - Payment
Stakeholders and their dependencies

- **Owner**
  - Owner of the parking spot
  - Register parking spot and specify its availability

- **Renter**
  - User of the parking spot
  - Search and reserve parking spot
  - Start/end parking session
  - Extend parking session
Stakeholders and their dependencies
Stakeholders goals

- Goal #1: Register parking spot
- Goal #2: Parking session started
- Goal #3: Parking session ended
- Goal #4: Payment handled
Goal #1: Register parking spot
Goal #2: Parking session started
Goal #3: Parking session ended

Parking dApp

- Parking session ended
- End parking session
- Check availability of parking spot

Renter

- Request to end parking
- Request to extend parking
- Parking extended
Goal #4: Payment handled
Use case diagram
<table>
<thead>
<tr>
<th>Use case ID: name:</th>
<th>UC#1: Save Parking Spot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date created:</td>
<td>15.09.2021</td>
</tr>
</tbody>
</table>

**Actors:** Owner

**Description:** The owner creates the parking spots, enters its information and availability.

**Trigger:** Owner wants to create and save a parking spot for renting out in the parking dApp

**Precondition:** Physical parking spot is ready for the use. Owner has log in to Parking dApp.

**Postcondition:** Information about parking spot and its availability is entered to the Parking dApp.

**Main flow:**
1. Owner provides information about parking spot.  
2. Parking dApp registers parking spot.  
3. Owner provides information about parking spot availability.  
4. Parking dApp specifies parking spot availability.  
5. Owner confirms the entered information.

**Alternative flow:** None

**Priority:** Must

**Assumptions:** Owner should have access to Parking dApp
<table>
<thead>
<tr>
<th>Use case ID: name:</th>
<th>UC#2: Start Parking Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date created:</td>
<td>15.09.2021</td>
</tr>
<tr>
<td>Actors:</td>
<td>Renter</td>
</tr>
<tr>
<td>Description:</td>
<td>Renter arrives to the place and intends to park the car. He used the parking dApp to reserve the parking spot.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>Renter wants to park the car.</td>
</tr>
<tr>
<td>Precondition:</td>
<td>Renter has arrived to the place and wants to park his car.</td>
</tr>
<tr>
<td>Postcondition:</td>
<td>Renter’s car is parked and parking session has started</td>
</tr>
</tbody>
</table>
| Main flow:        | 1. Renter queries for parking spot.  
                      2. Parking dApp reserves parking spot.  
                      3. Renter confirms the received parking spot. |
<p>| Alternative flow: | None                       |
| Priority:         | Must                       |
| Assumptions:      | Renter should have access to Parking dApp |</p>
<table>
<thead>
<tr>
<th><strong>Use case ID:</strong> UC#3: End Parking Session</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name:</strong> UC#3: End Parking Session</td>
</tr>
<tr>
<td><strong>Date created:</strong> 15.09.2021</td>
</tr>
<tr>
<td><strong>Actors:</strong> Renter</td>
</tr>
<tr>
<td><strong>Description:</strong> Renter finishes using the parking spot.</td>
</tr>
<tr>
<td><strong>Trigger:</strong> Renter wants to finish the parking.</td>
</tr>
<tr>
<td><strong>Precondition:</strong> Renter wants to finish the parking.</td>
</tr>
<tr>
<td><strong>Postcondition:</strong> Parking session has ended.</td>
</tr>
</tbody>
</table>
| **Main flow:** 1. Renter requests to end parking.  
2. Parking dApp ends parking session.  
3. Renter receives notification about session end. |
<p>| <strong>Alternative flow:</strong> None               |
| <strong>Priority:</strong> Must                      |
| <strong>Assumptions:</strong> Renter should have access to Parking dApp. |</p>
<table>
<thead>
<tr>
<th>Use case ID: name:</th>
<th>UC#3.1: Extend Parking Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date created:</td>
<td>15.09.2021</td>
</tr>
<tr>
<td>Actors:</td>
<td>Renter</td>
</tr>
<tr>
<td>Description:</td>
<td>Renter has finished using the parking spot. If the parking spot is available, user can extend the parking time.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>Renter wants to extend the parking.</td>
</tr>
<tr>
<td>Precondition:</td>
<td>Renter wants to extend the parking.</td>
</tr>
<tr>
<td>Postcondition:</td>
<td>Parking session has ended.</td>
</tr>
</tbody>
</table>
| Main flow:        | 1. Renter requests to extend the parking.  
|                    | 2. Parking dApp checks availability of parking spot   
|                    | If not available:  
|                    | 3. Parking dApp ends parking session.  
|                    | 4. Renter receives notification about session end.  
|                    | else Alternative 1 |
| Alternative flow: | Alternative 1: |
| Priority:         | Must |
| Assumptions:      | Renter should have access to Parking dApp.  
|                    | Renter control by himself the parking session length (no automatic check about expiration of the parking session) |
Use case ID: name: UC#3.1: Extend Parking Session

Date created: 15.09.2021

Actors: Renter

Description: Renter has finished using the parking spot. If the parking spot is available, user can extend the parking time.

Trigger: Renter wants to extend the parking.

Precondition: Renter wants to extend the parking.

Postcondition: Parking session has ended.

Main flow:
1. Renter requests to extend the parking
2. Parking dApp checks availability of parking spot
   If not available:
   3. Parking dApp ends parking session.
   4. Renter receives notification about session end.
   else Alternative 1

Alternative flow:

Alternative 1:

Priority: Must

Assumptions:
- Renter should have access to Parking dApp.
- Renter control by himself the parking session length (no automatic check about expiration of the parking session)

a3. Parking dApp extends parking session.

a4. Parking dApp sends notification to Renter.

a5. Renter receives notification about extension.
<table>
<thead>
<tr>
<th>Use case ID: name:</th>
<th>UC#4: Payment handled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date created:</td>
<td>16.09.2021</td>
</tr>
<tr>
<td>Actors:</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
</tr>
<tr>
<td>Trigger:</td>
<td></td>
</tr>
<tr>
<td>Precondition:</td>
<td></td>
</tr>
<tr>
<td>Postcondition:</td>
<td></td>
</tr>
<tr>
<td>Main flow:</td>
<td></td>
</tr>
<tr>
<td>Alternative flow:</td>
<td></td>
</tr>
<tr>
<td>Priority:</td>
<td></td>
</tr>
<tr>
<td>Assumptions:</td>
<td></td>
</tr>
<tr>
<td>Use case ID: name:</td>
<td>UC#4: Payment handled</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Date created:</td>
<td>16.09.2021</td>
</tr>
<tr>
<td>Actors:</td>
<td>Owner, Renter</td>
</tr>
<tr>
<td>Description:</td>
<td>Once the renter has finished using the parking spot he pays to the owner.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>Renter has finished the parking and want to leave, but first he has to pay for parking.</td>
</tr>
<tr>
<td>Precondition:</td>
<td>Parking session has ended.</td>
</tr>
<tr>
<td>Postcondition:</td>
<td>Payment handled.</td>
</tr>
</tbody>
</table>
| Main flow:       | 1. Parking dApp calculates the total cost.  
|                  | 2. Parking dApp processes payment from Renter. 
|                  | 3. Renter pays for parking (spot). 
|                  | 4. Parking dApp processes payment to Owner. 
|                  | 5. Owner receives payment. |
| Alternative flow:| none                   |
| Priority:        | Must                   |
| Assumptions:     | Owner has banking account. |
Solution oriented requirements - Data model
Solution oriented requirements - Data model
Solution oriented requirements - State model

- **Idle**
  - Owner.SaveParkingSpot()

- **Reserved**
  - Renter.StartParkingSession()

- **Unpaid**
  - Renter.EndParkingSession()

- **Paid**
  - User.HandlePayment()
Solution oriented requirements - State model

Creates the `ParkingSpot` object

- `ID`, `Name`, `Location`, `CostPerMinute`

Instantiate `ParkingSpotLocation` object

- `X`, `Y` and `GeoHash`

Instantiate `CurrentAmount` object

- `Amount` and `CurrencyRate`

Assigns `ParkingTime` object

- `ID` and `ParkingSpot`
Solution oriented requirements - State model

Updates **ParkingTime** object

- ParkingStart
Solution oriented requirements - State model

- **Idle**
  - Renter.StartParkingSession()

- **Reserved**
  - Renter.EndParkingSession()

- **Unpaid**
  - User.HandlePayment()

- **Paid**

Updates **ParkingTime** object

- ParkingEnd
Solution oriented requirements - State model

Updates **ParkingTime** object

- **Cost** – is estimates using **ParkingStart**, **ParkingEnd** and **CostPerMinute**

Update **User** objects

- **Renter.Balance** – deduced by the **ParkingTime.Cost**
- **Owner.Balance** – increased by the **ParkingTime.Cost**
Save parking slot

Solution oriented requirements - Behavior model
Solution oriented requirements - Behavior model

Start parking session

1: StartParkingSession

2: FindParkingSpotToRent

3: ParkingSpotFound

4: ParkingSessionStarted
Solution oriented requirements - Behavior model

End parking session
Solution oriented requirements - Behavior model

Handle payment
Non-functional requirements: Access control model
Access control model
Access control model

AccReq#1: Only **Owner** should be able to **insert** ParkingSpot
AccReq#2: Only **Renter** should be able to **update** ParkingTime.ParkingStart
AccReq#3: Only **Renter** should be able to **update** ParkingTime.ParkingEnd
AccReq#4: Only **User** should be able to **update** Balance.Amount
Lecture goals

● Do you need a Blockchain?
  ○ Coin management system
  ○ Medical data preservation system
  ○ Sharing a parking spot

● Parking dApp scenario
  ○ Stakeholders analysis and goals
  ○ Use cases
  ○ Solution-oriented requirements
    ■ Data model
    ■ State model
    ■ Behavior model
  ○ Non-functional requirements
    ■ Access control model