Parker - Smart Parking Assistant

(Course Project - Software Design CSE564)

Team 13:

Parvakumar Patel	(5242-846)
Parth Miyani	(9065-885)
Prayag Patel	(4791-378)
Sudhanva Rao	(6938-986)

Overview

Parker is an automatic Smart Parking Assistant (SPA) that will completely handle the working of a parking lot.

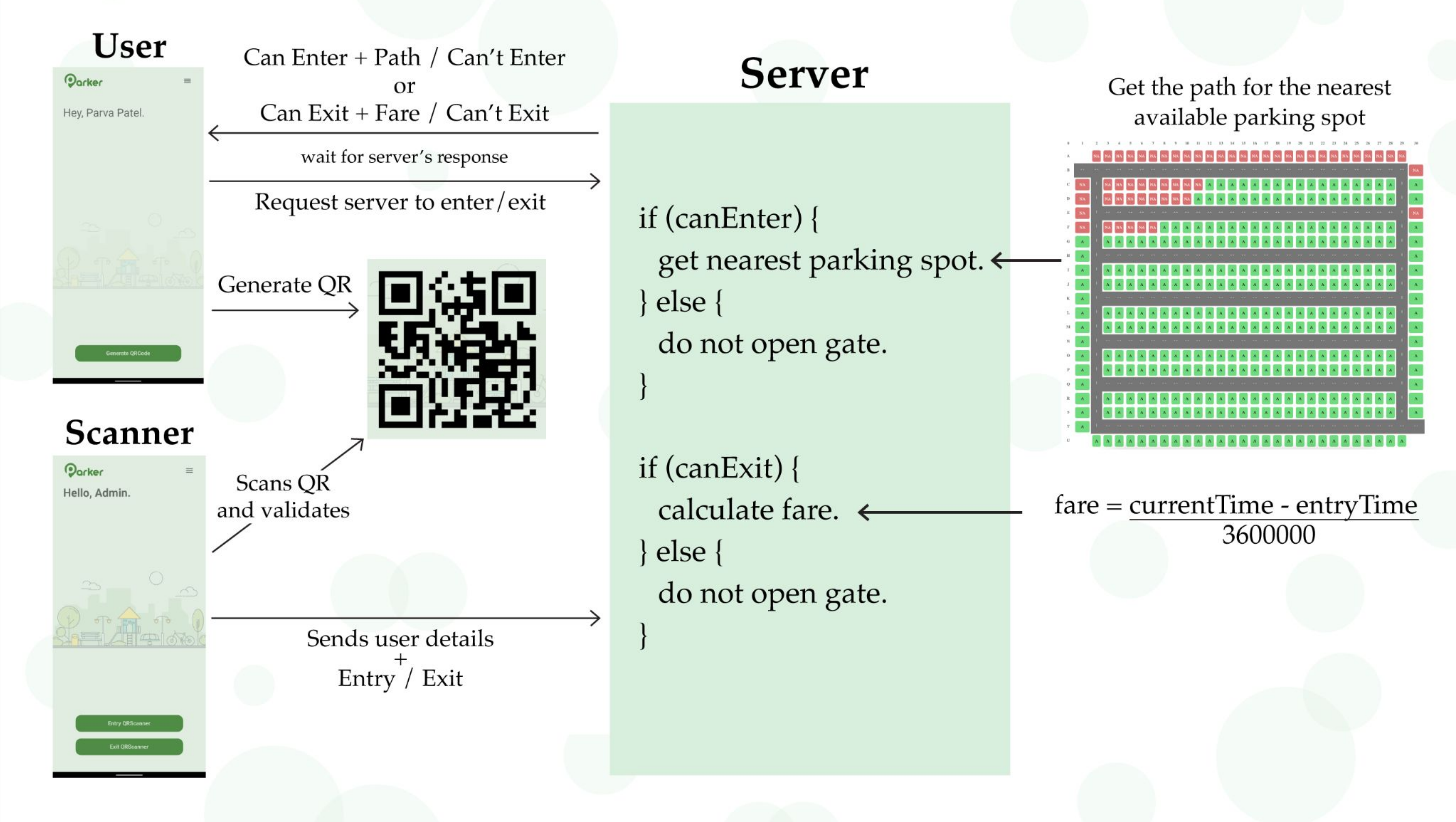
It has the following features:

- **Customized user login**
- **Handle a parking lot of 404 spots with sensors**
- **Obtain nearest available parking**
- **Guide the user to the assigned parking**
- **Add payment cards and ease payments without any human intervention**

Sensors: For the 404 parkings, Parker will be using an array of 404 proximity sensors and those sensors will detect if there is any car parked in the parking or not.

Using this information, the server fetch the nearest available parking using BFS in the grid.

Model



User

Parker

Hey, Parva Patel.

Generate QRCode

Can Enter + Path / Can't Enter
or
Can Exit + Fare / Can't Exit

wait for server's response

Request server to enter / exit

Generate QR

Scanner

Parker

Hello, Admin.

Entry QRScanner

Exit QRScanner

Scans QR
and validates

Sends user details
+
Entry / Exit

Server

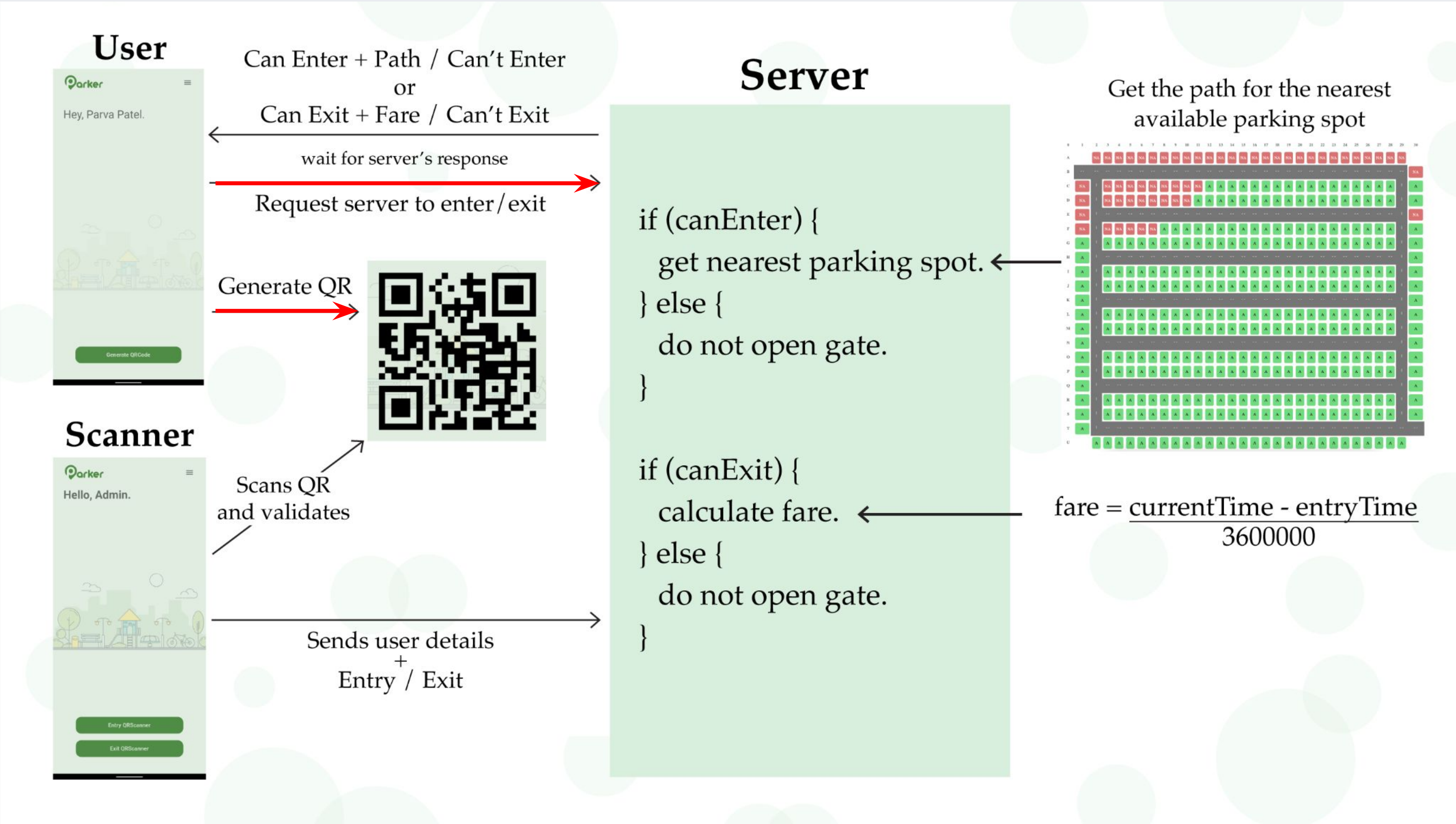
```
if (canEnter) {
  get nearest parking spot.
} else {
  do not open gate.
}

if (canExit) {
  calculate fare.
} else {
  do not open gate.
}
```

Get the path for the nearest available parking spot

$$\text{fare} = \frac{\text{currentTime - entryTime}}{3600000}$$

Model



User

Can Enter + Path / Can't Enter
or
Can Exit + Fare / Can't Exit

wait for server's response

Request server to enter / exit

Generate QR

Scanner

Scans QR
and validates

Sends user details
+
Entry / Exit

Server

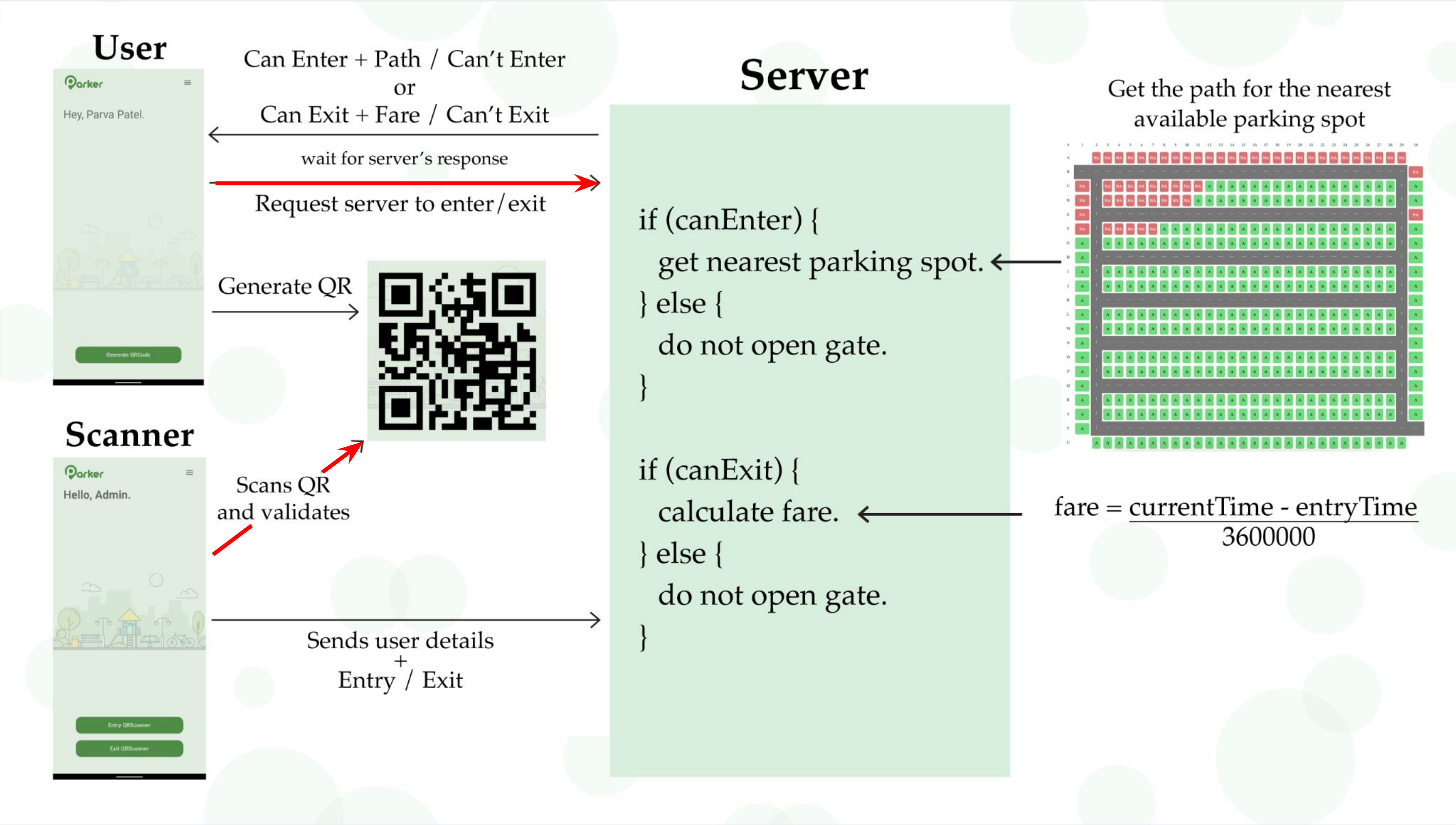
```
if (canEnter) {
  get nearest parking spot.
} else {
  do not open gate.
}

if (canExit) {
  calculate fare.
} else {
  do not open gate.
}
```

Get the path for the nearest
available parking spot

$$\text{fare} = \frac{\text{currentTime - entryTime}}{3600000}$$

Model



User

Can Enter + Path / Can’t Enter
or
Can Exit + Fare / Can’t Exit

wait for server’s response

Request server to enter / exit

Generate QR

Scanner

Scans QR
and validates

Sends user details
+
Entry / Exit

Server

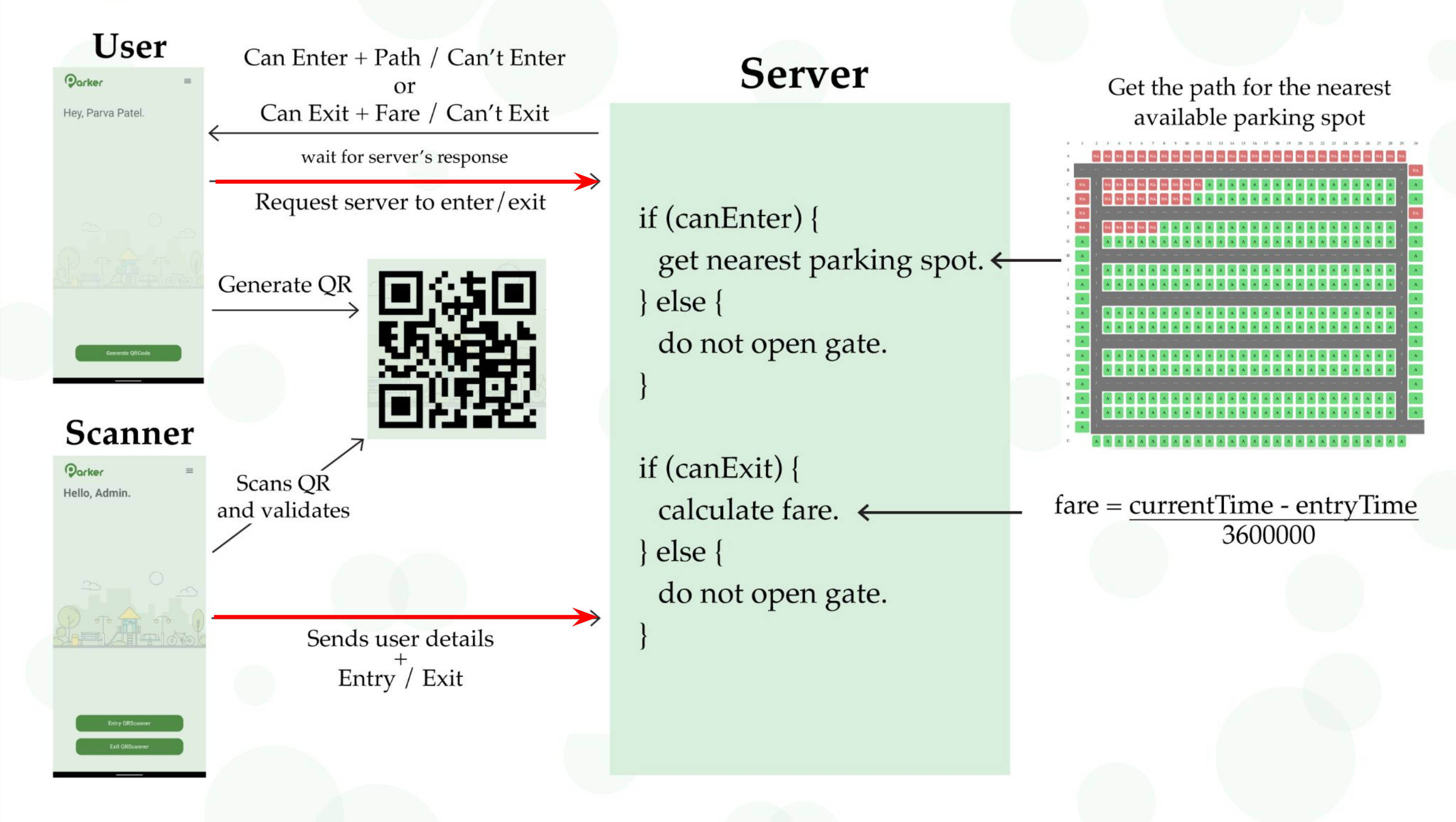
```
if (canEnter) {
    get nearest parking spot.
} else {
    do not open gate.
}

if (canExit) {
    calculate fare.
} else {
    do not open gate.
}
```

Get the path for the nearest available parking spot

$$\text{fare} = \frac{\text{currentTime - entryTime}}{3600000}$$

Model



User

Can Enter + Path / Can't Enter
or
Can Exit + Fare / Can't Exit

wait for server's response

Request server to enter / exit

Generate QR

Scanner

Scans QR
and validates

Sends user details
+
Entry / Exit

Server

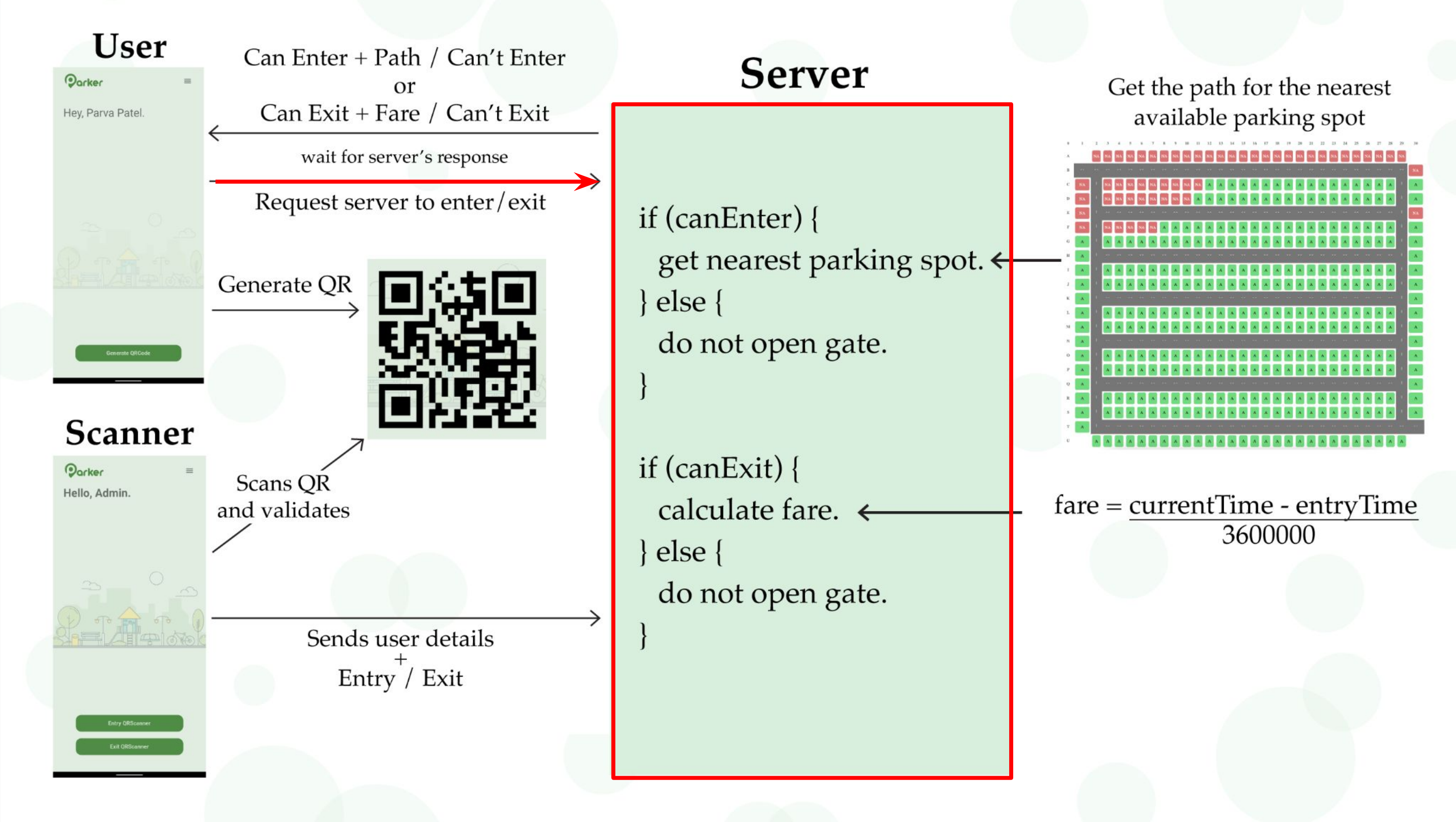
```
if (canEnter) {
  get nearest parking spot.
} else {
  do not open gate.
}

if (canExit) {
  calculate fare.
} else {
  do not open gate.
}
```

Get the path for the nearest available parking spot

$$\text{fare} = \frac{\text{currentTime - entryTime}}{3600000}$$

Model



User

Can Enter + Path / Can't Enter
or
Can Exit + Fare / Can't Exit

wait for server's response

Request server to enter / exit

Generate QR

Scanner

Scans QR
and validates

Sends user details
+
Entry / Exit

Server

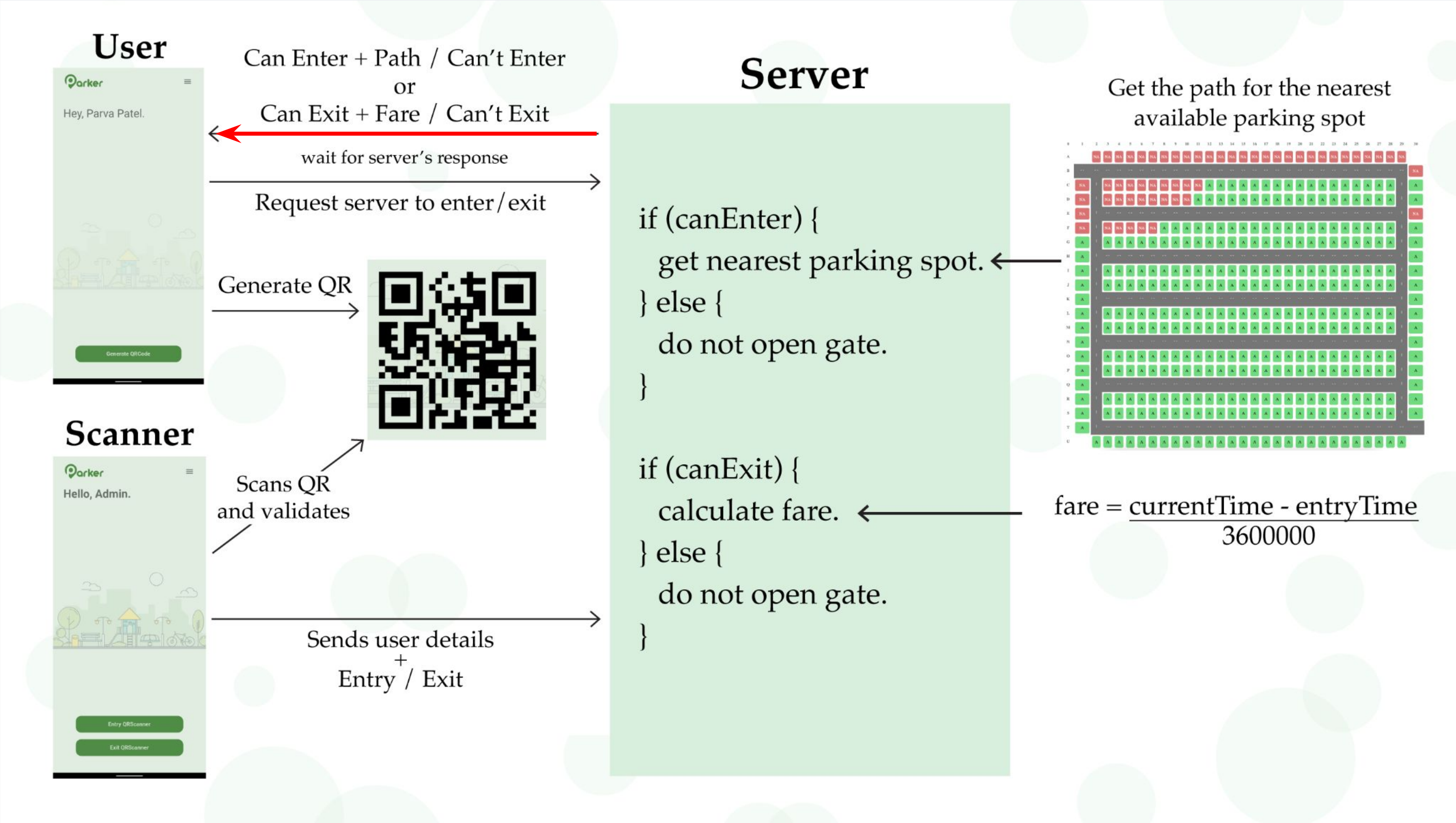
```
if (canEnter) {
  get nearest parking spot.
} else {
  do not open gate.
}

if (canExit) {
  calculate fare.
} else {
  do not open gate.
}
```

Get the path for the nearest
available parking spot

$$fare = \frac{currentTime - entryTime}{3600000}$$

Model



Can Enter + Path / Can't Enter
or
Can Exit + Fare / Can't Exit

wait for server's response

Request server to enter / exit

Generate QR

Scans QR and validates

Sends user details + Entry / Exit

Server

```
if (canEnter) {
  get nearest parking spot.
} else {
  do not open gate.
}

if (canExit) {
  calculate fare.
} else {
  do not open gate.
}
```

Get the path for the nearest available parking spot

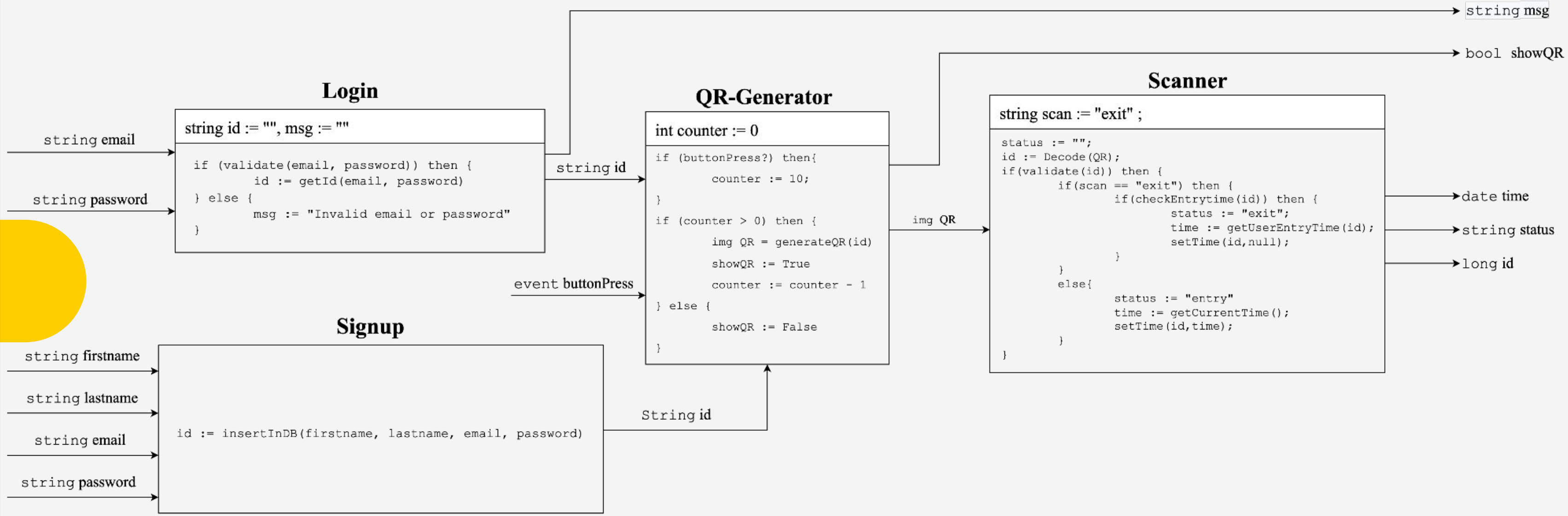
$$\text{fare} = \frac{\text{currentTime - entryTime}}{3600000}$$

SRC Specifications

There are three SRCs.

1) **AppComponent:** This will handle the
   - *login/signup authentication*,
   - *QR code generation* and
   - *communicate will the server* with appropriate API calls
2) **Server:** This will
   - *handle the API calls* and will be a communication link between the client and scanner.
   - *handle the parking* with the help of sensors,
   - *find the nearest available parking spot using BFS* and
   - *calculate the fare* during exit
3) **ParkingSpot:** This will *handle individual parking spots* using proximity sensors.

App Component



Login

```
string id := "", msg := ""
```

```
if (validate(email, password)) then {
        id := getId(email, password)
} else {
        msg := "Invalid email or password"
}
```

Signup

```
id := insertInDB(firstname, lastname, email, password)
```

QR-Generator

```
int counter := 0
```

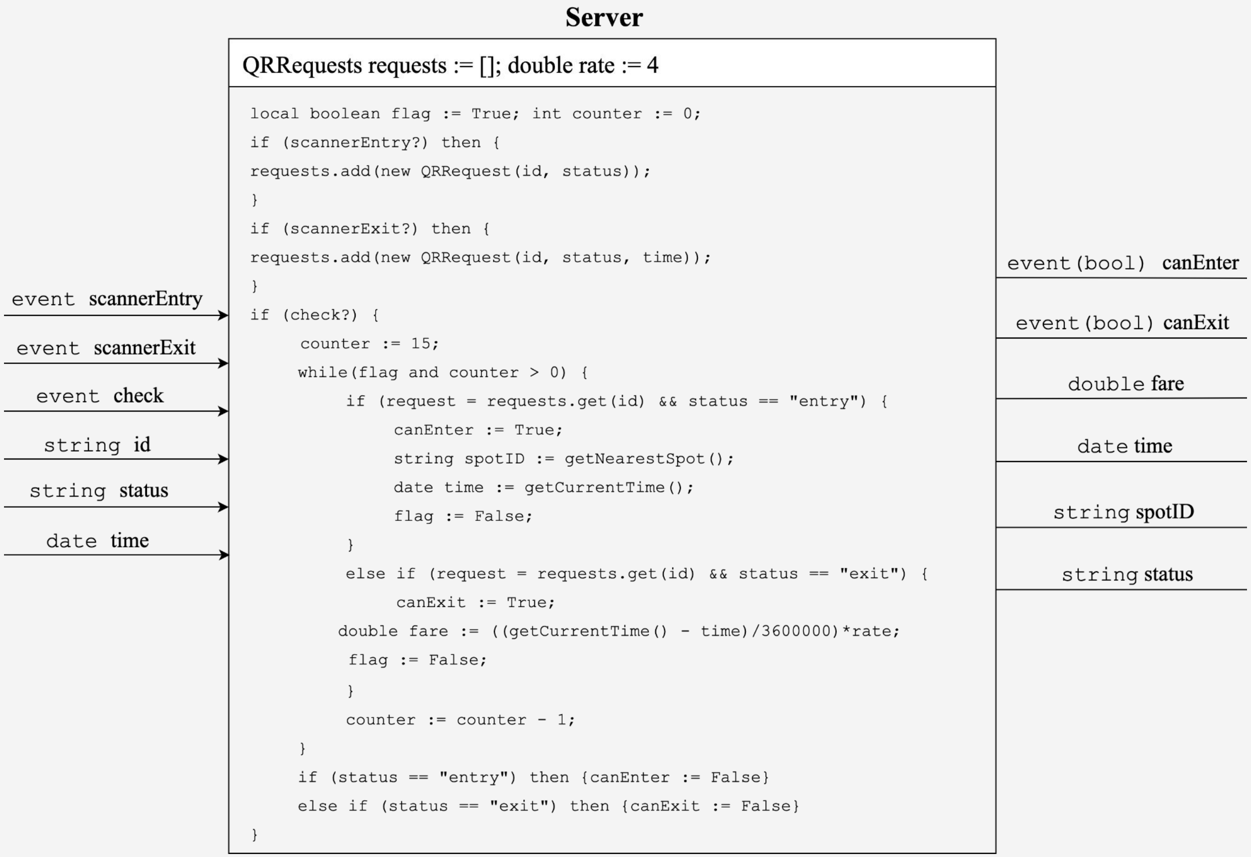
```
if (buttonPress?) then{
        counter := 10;
}
if (counter > 0) then {
        img QR = generateQR(id)
        showQR := True
        counter := counter - 1
} else {
        showQR := False
}
```

Scanner

```
string scan := "exit" ;
```

```
status := "";
id := Decode(QR);
if(validate(id)) then {
        if(scan == "exit") then {
                if(checkEntrytime(id)) then {
                        status := "exit";
                        time := getUserEntryTime(id);
                        setTime(id,null);
                }
        }
        else{
                status := "entry"
                time := getCurrentTime();
                setTime(id,time);
        }
}
```

This component will handle the application making connection with the FirebaseDB, validating outcomes and sending API requests to the server.



Server

QRRequests requests := []; double rate := 4

```
local boolean flag := True; int counter := 0;
if (scannerEntry?) then {
requests.add(new QRRequest(id, status));
}
if (scannerExit?) then {
requests.add(new QRRequest(id, status, time));
}
if (check?) {
     counter := 15;
     while(flag and counter > 0) {
          if (request = requests.get(id) && status == "entry") {
               canEnter := True;
               string spotID := getNearestSpot();
               date time := getCurrentTime();
               flag := False;
          }
          else if (request = requests.get(id) && status == "exit") {
               canExit := True;
         double fare := ((getCurrentTime() - time)/3600000)*rate;
          flag := False;
          }
          counter := counter - 1;
     }
     if (status == "entry") then {canEnter := False}
     else if (status == "exit") then {canExit := False}
}
```

event scannerEntry

event scannerExit

event check

string id

string status

date time

event(bool) canEnter

event(bool) canExit

double fare

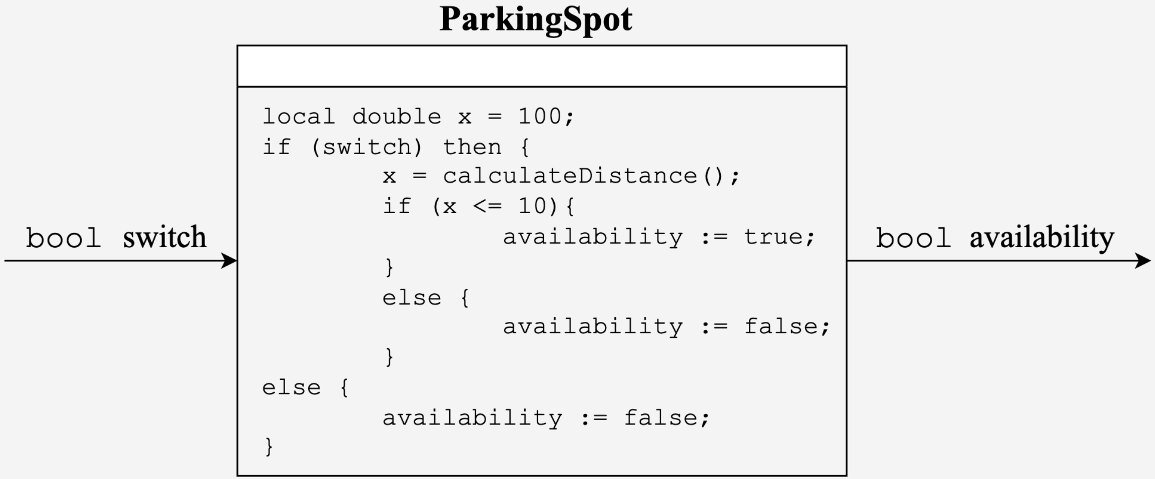
date time

string spotID

string status

Sensors

In real-life implementation, we'll have to use proximity sensors in every parking to capture the availability of the spot. Given below will be the react of the parking spot's SRC.



ParkingSpot

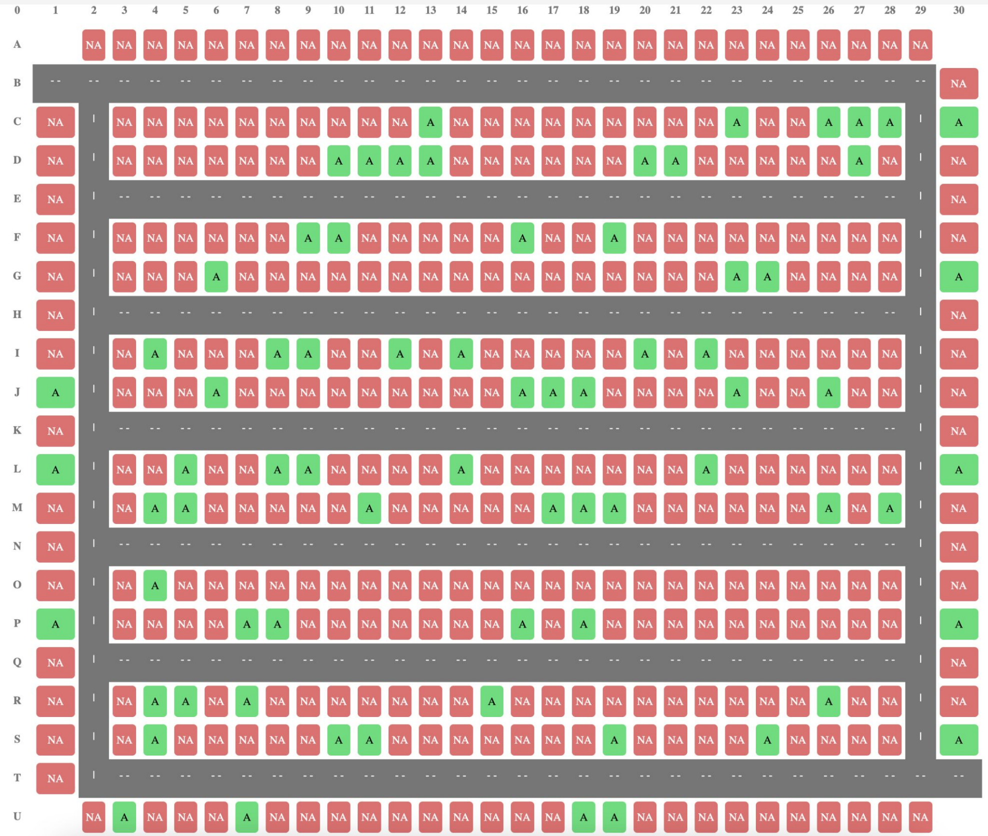
bool **switch**

```
local double x = 100;
if (switch) then {
        x = calculateDistance();
        if (x <= 10){
                availability := true;
        }
        else {
                availability := false;
        }
else {
        availability := false;
}
```

bool **availability**

For our project, we have created a simulation of the whole parking wherein we can update the status of any parking spot by clicking in the UI.

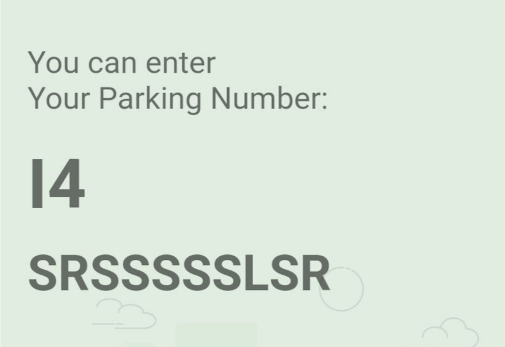
Sensors: The simulation



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A		NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
B	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	NA
C	NA	|	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	A	A	A	|	A
D	NA	|	NA	NA	NA	NA	NA	NA	NA	A	A	A	A	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	|	NA
E	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
F	NA	|	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
G	NA	|	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	|	A
H	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
I	NA	|	NA	A	NA	NA	NA	A	A	NA	NA	A	NA	A	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	|	NA
J	A	|	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	A	NA	NA	A	NA	NA	|	NA
K	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
L	A	|	NA	NA	A	NA	NA	A	A	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	|	A
M	NA	|	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	NA	NA	A	NA	A	|	NA
N	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
O	NA	|	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
P	A	|	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	A
Q	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
R	NA	|	NA	A	A	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	|	NA
S	NA	|	NA	A	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	A	NA	NA	NA	NA	|	A
T	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
U		NA	A	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	

A	Available Parking
NA	Not-Available Parking

Output when user can enter:

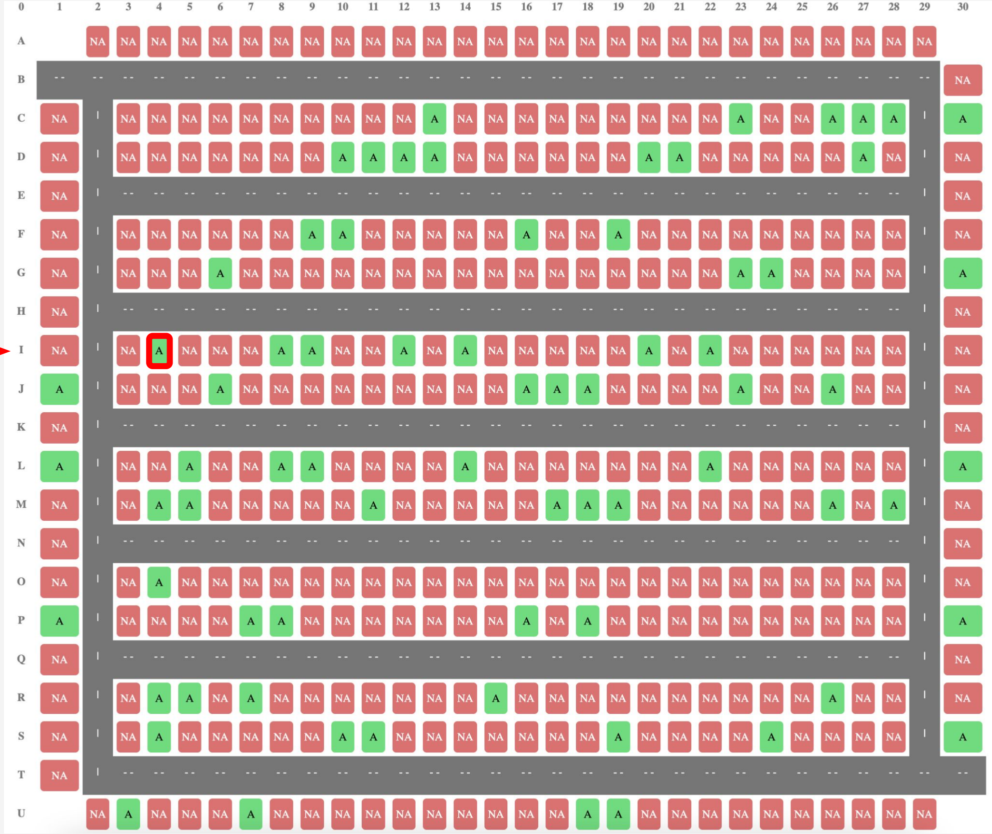


You can enter
Your Parking Number:

I4

SRSSSSSLSR

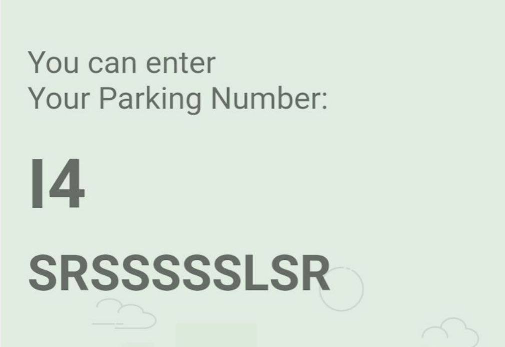
Sensors: The simulation



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A		NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
B	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	NA
C	NA	|	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	A	A	A	|	A
D	NA	|	NA	NA	NA	NA	NA	NA	NA	A	A	A	A	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	|	NA
E	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
F	NA	|	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
G	NA	|	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	|	A
H	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
I	NA	|	NA	A	NA	NA	NA	A	A	NA	NA	A	NA	A	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	|	NA
J	A	|	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	A	NA	NA	A	NA	NA	|	NA
K	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
L	A	|	NA	NA	A	NA	NA	A	A	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	|	A
M	NA	|	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	NA	NA	A	NA	A	|	NA
N	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
O	NA	|	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
P	A	|	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	A
Q	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
R	NA	|	NA	A	A	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	|	NA
S	NA	|	NA	A	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	A	NA	NA	NA	NA	|	A
T	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
U		NA	A	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	

A	Available Parking
NA	Not-Available Parking

Output when user can enter:

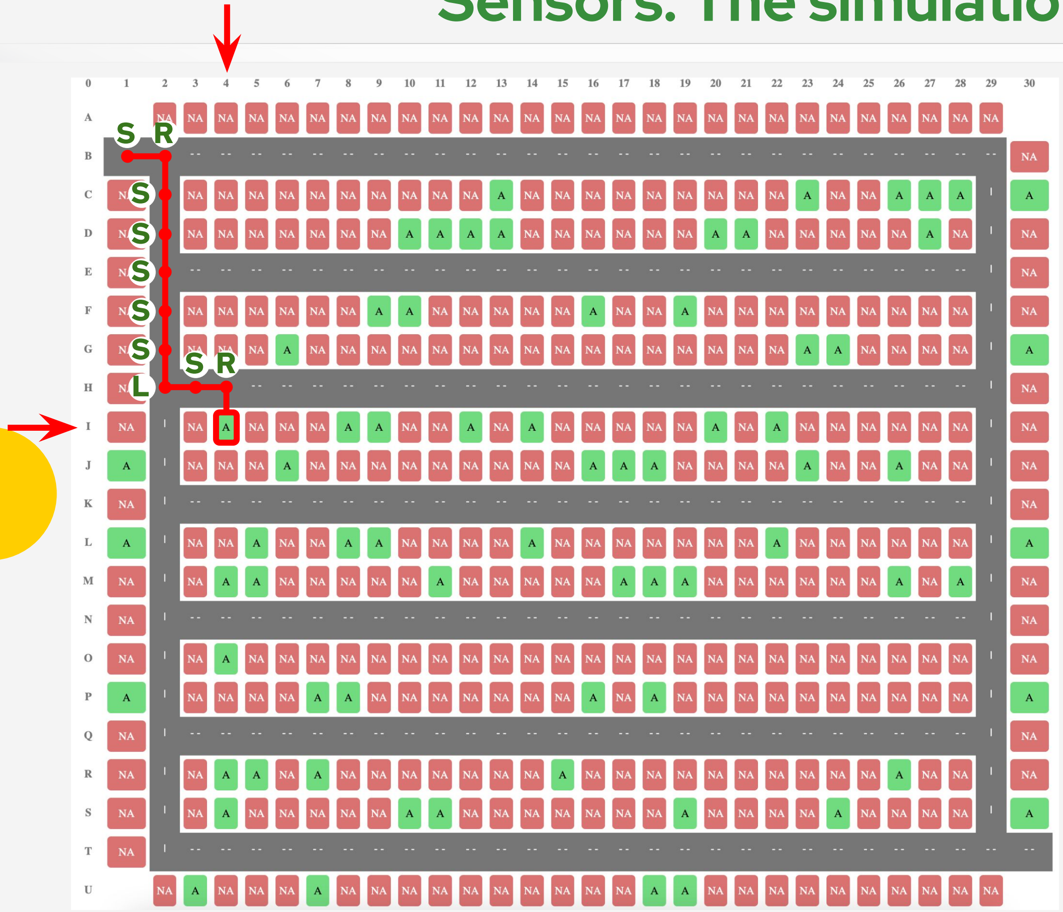


You can enter
Your Parking Number:

I4

SRSSSSSLSR

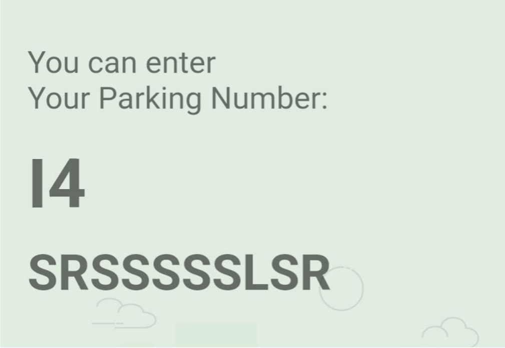
Sensors: The simulation



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A		NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
B			--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	NA
C	NA		NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	A	A	A	|	A
D	NA		NA	NA	NA	NA	NA	NA	NA	A	A	A	A	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	|	NA
E	NA		--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
F	NA		NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
G	NA		NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	|	A
H	NA				--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
I	NA	|	NA	A	NA	NA	NA	A	A	NA	NA	A	NA	A	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	|	NA
J	A	|	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	A	NA	NA	A	NA	NA	|	NA
K	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
L	A	|	NA	NA	A	NA	NA	A	A	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	|	A
M	NA	|	NA	A	A	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	A	A	A	NA	NA	NA	NA	NA	NA	A	NA	A	|	NA
N	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
O	NA	|	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	NA
P	A	|	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	|	A
Q	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	|	NA
R	NA	|	NA	A	A	NA	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	|	NA
S	NA	|	NA	A	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	A	NA	NA	NA	NA	A	NA	NA	NA	NA	|	A
T	NA	|	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
U		NA	A	NA	NA	NA	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	A	A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	

A	Available Parking
NA	Not-Available Parking

Output when user can enter:

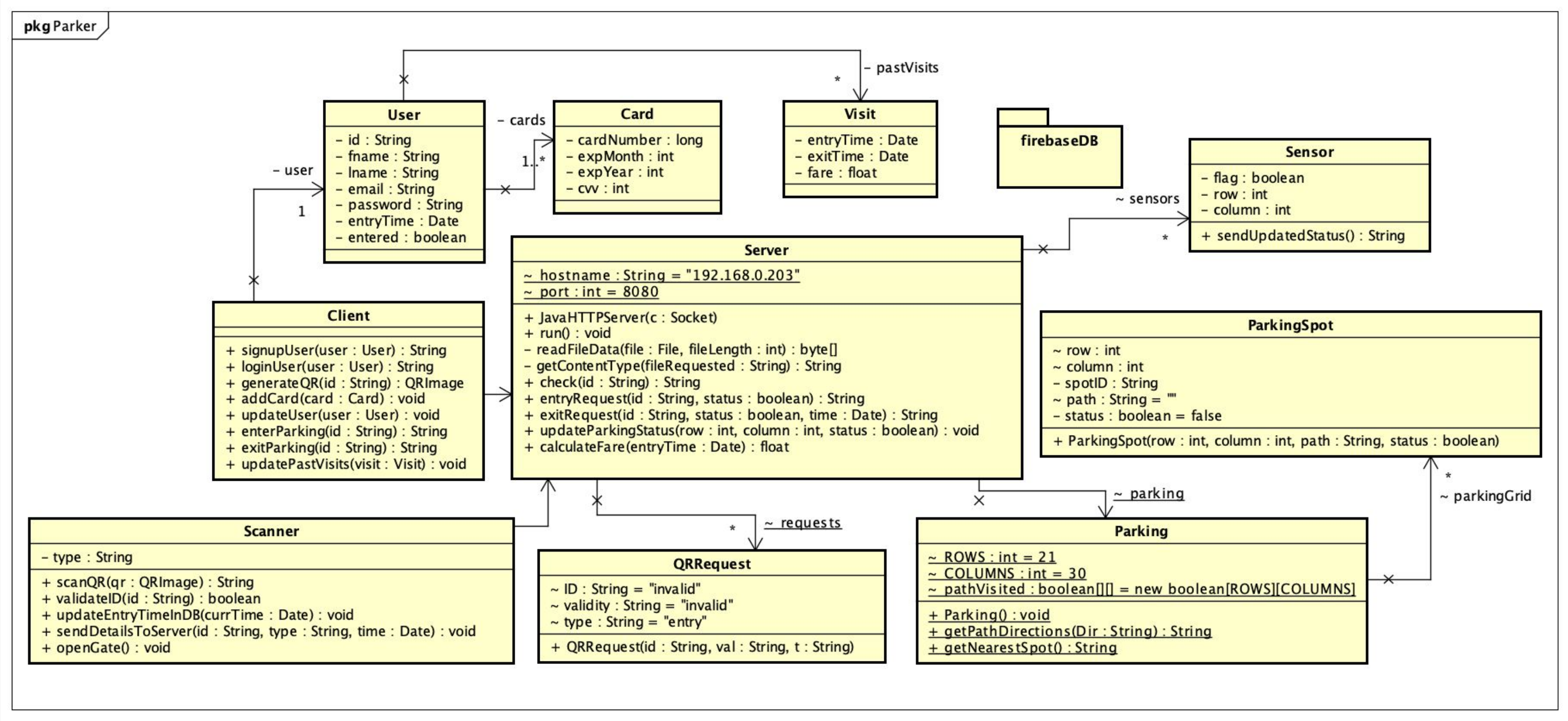


You can enter
Your Parking Number:

I4

SRSSSSSLSR

UML Class Diagrams



```
User
- id : String
- fname : String
- lname : String
- email : String
- password : String
- entryTime : Date
- entered : boolean
```

```
Card
- cardNumber : long
- expMonth : int
- expYear : int
- cvv : int
```

```
Visit
- entryTime : Date
- exitTime : Date
- fare : float
```

```
Sensor
- flag : boolean
- row : int
- column : int
+ sendUpdatedStatus() : String
```

```
Server
~ hostname : String = "192.168.0.203"
~ port : int = 8080
+ JavaHTTPServer(c : Socket)
+ run() : void
- readFileData(file : File, fileLength : int) : byte[]
- getContentType(fileRequested : String) : String
+ check(id : String) : String
+ entryRequest(id : String, status : boolean) : String
+ exitRequest(id : String, status : boolean, time : Date) : String
+ updateParkingStatus(row : int, column : int, status : boolean) : void
+ calculateFare(entryTime : Date) : float
```

```
Client
+ signupUser(user : User) : String
+ loginUser(user : User) : String
+ generateQR(id : String) : QRImage
+ addCard(card : Card) : void
+ updateUser(user : User) : void
+ enterParking(id : String) : String
+ exitParking(id : String) : String
+ updatePastVisits(visit : Visit) : void
```

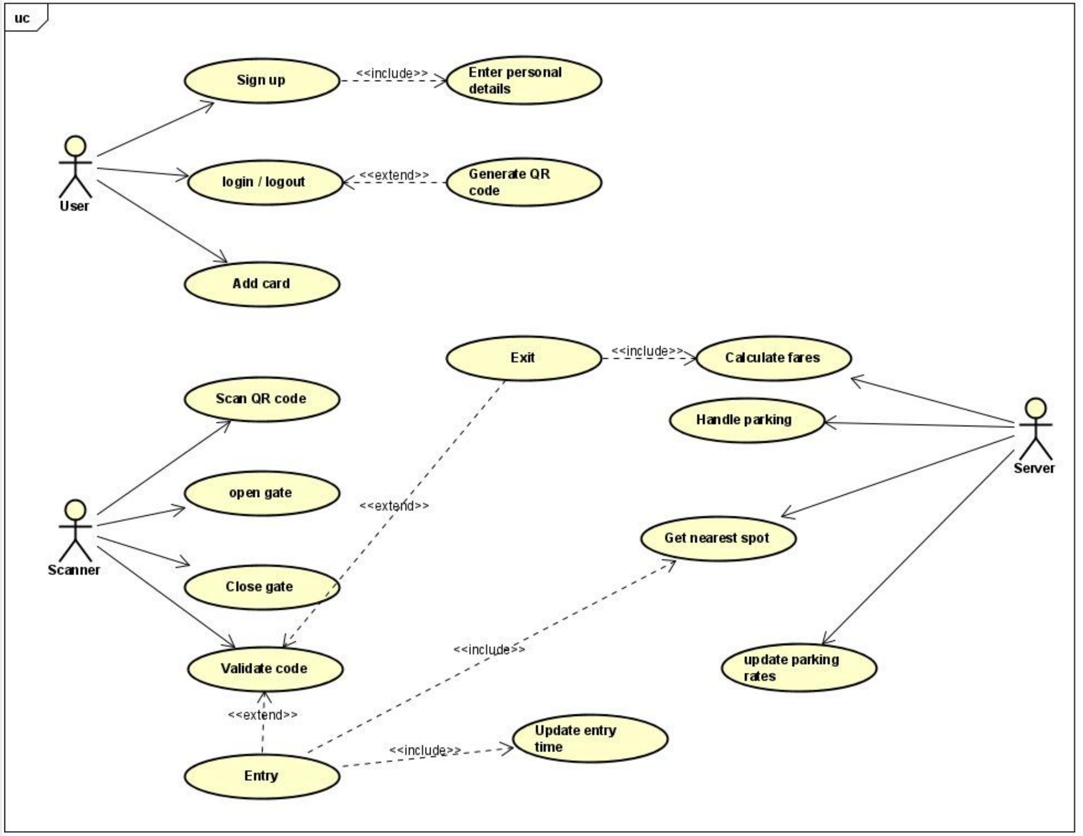
```
ParkingSpot
~ row : int
~ column : int
- spotID : String
~ path : String = ""
- status : boolean = false
+ ParkingSpot(row : int, column : int, path : String, status : boolean)
```

```
Scanner
- type : String
+ scanQR(qr : QRImage) : String
+ validateID(id : String) : boolean
+ updateEntryTimeInDB(currTime : Date) : void
+ sendDetailsToServer(id : String, type : String, time : Date) : void
+ openGate() : void
```

```
QRRequest
~ ID : String = "invalid"
~ validity : String = "invalid"
~ type : String = "entry"
+ QRRequest(id : String, val : String, t : String)
```

```
Parking
~ ROWS : int = 21
~ COLUMNS : int = 30
~ pathVisited : boolean[][] = new boolean[ROWS][COLUMNS]
+ Parking() : void
+ getPathDirections(Dir : String) : String
+ getNearestSpot() : String
```

Use case diagram

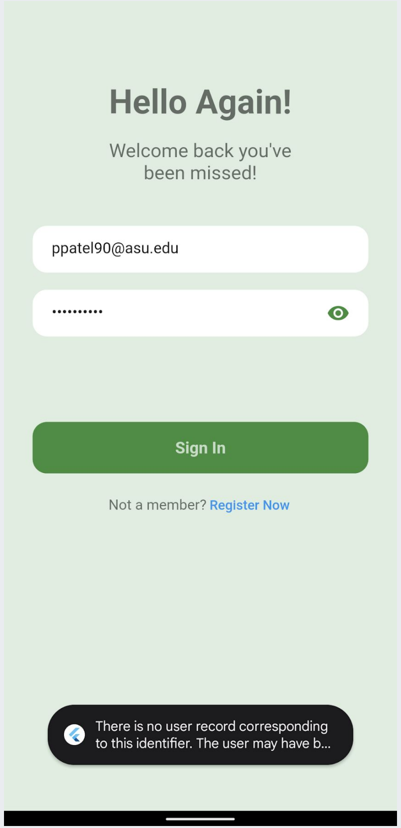


Experiments and Results

Here are the experiments that we ran on our project.

1. User is not signed up. (The is no entity corresponding to that email in the DB)

The user will be prompted by showing that *"There is no user record corresponding to the email."*



Hello Again!

Welcome back you've been missed!

ppatel90@asu.edu

••••••••••

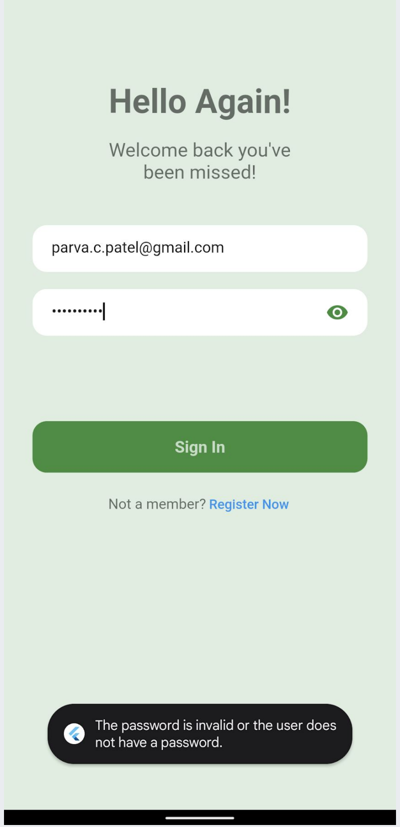
There is no user record corresponding to this identifier. The user may have b...

Experiments and Results

Here are the experiments that we ran on our project.

2. User enters wrong password.

The user will be prompted by showing that
"The password is invalid."



Hello Again!

Welcome back you've been missed!

parva.c.patel@gmail.com

••••••••••

The password is invalid or the user does not have a password.

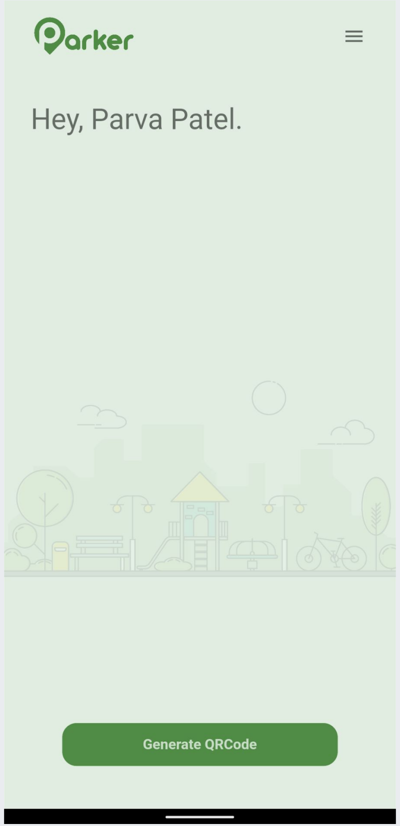
Experiments and Results

Here are the experiments that we ran on our project.

3. User successfully signs up or successfully logs in.

The app will be redirected to the dashboard where the user can:

a. Generate a QR code to enter/exit
b. See the past visits and transactions
c. Log out of the account

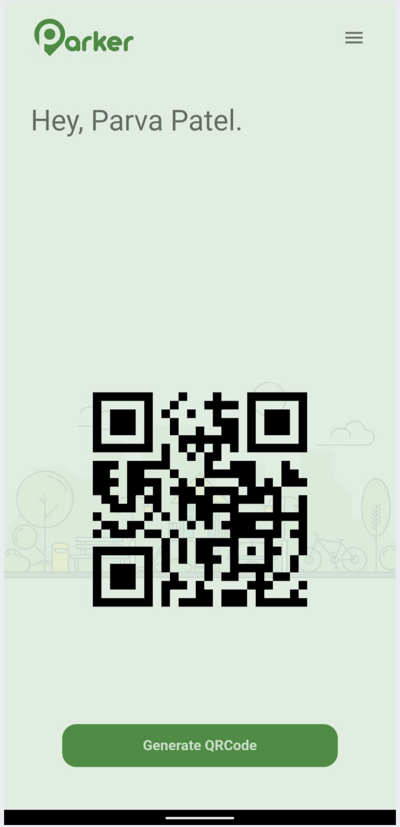


Experiments and Results

Here are the experiments that we ran on our project.

4. User generates QR code.

A QR code will pop-up in the screen. The scanner will scan this QR code and validate the user.



Experiments and Results

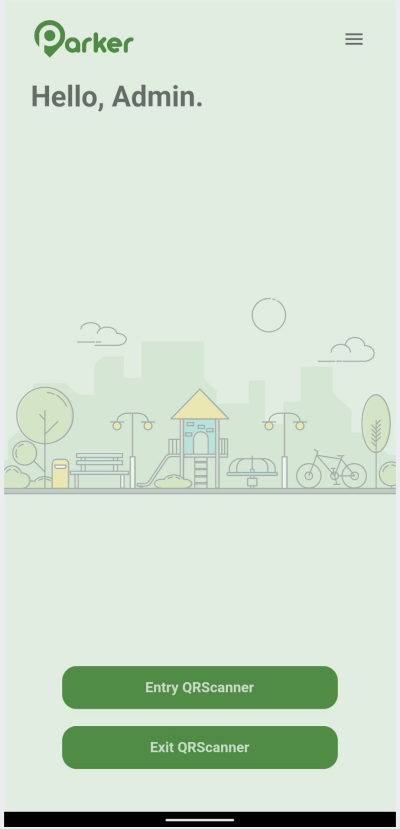
Here are the experiments that we ran on our project.

5. Admin/scanner logs in.

We have assigned a particular admin email. Logging in to that account will open the scanner.

2 types of scanners:

a. Entry scanner
b. Exit scanner



Experiments and Results

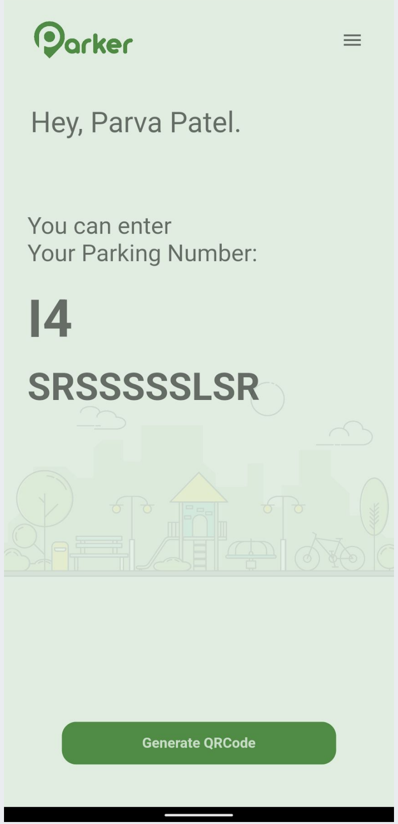
Here are the experiments that we ran on our project.

6. User can enter.

If the QR code is valid, the user can enter.

The server will calculate the nearest parking spot and the path to that spot and revert back with the path.

https://drive.google.com/file/d/1PL0IC4N_Ci-khYk1UHLmFhGDVBAiWvnP/view?usp=sharing



Hey, Parva Patel.

You can enter
Your Parking Number:

I4

SRSSSSSLSR

Experiments and Results

Here are the experiments that we ran on our project.

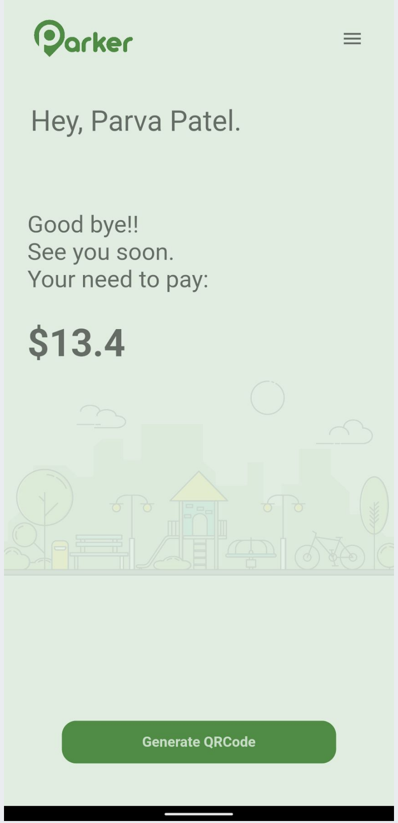
7. User can exit.

If the QR code is valid, there is an entry time registered in the DB, the user can exit.

The server will calculate the fare by the amount to time, the vehicle stayed in the parking. (Base fare = $4, Charge per hour = $4)

*fare = minimum of ($4, $4 * {(currentTime - entryTime) in hours})*

https://drive.google.com/file/d/1j0BePAuifebg0No2WYghCpOHZRGHQ_Rw/view?usp=sharing



Hey, Parva Patel.

Good bye!!
See you soon.
Your need to pay:

$13.4

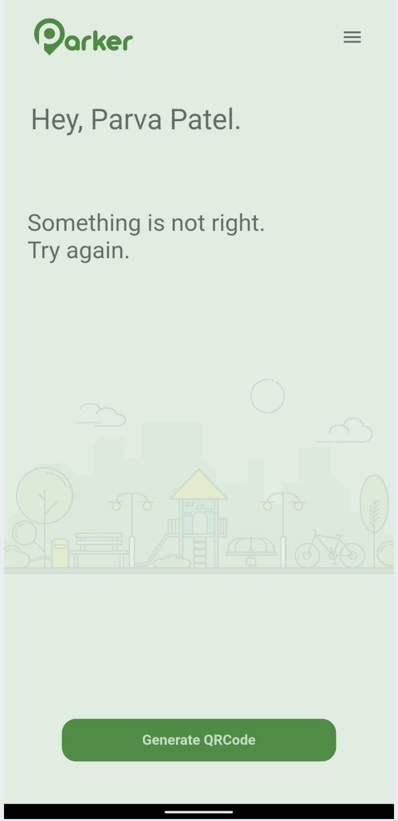
Experiments and Results

Here are the experiments that we ran on our project.

8. User cannot enter/exit.

There can be 2 reasons for user not being able to enter:

a. The QR code is invalid.
b. The scanner did not scan the QR. The QR code will be available to scan for 10 seconds.



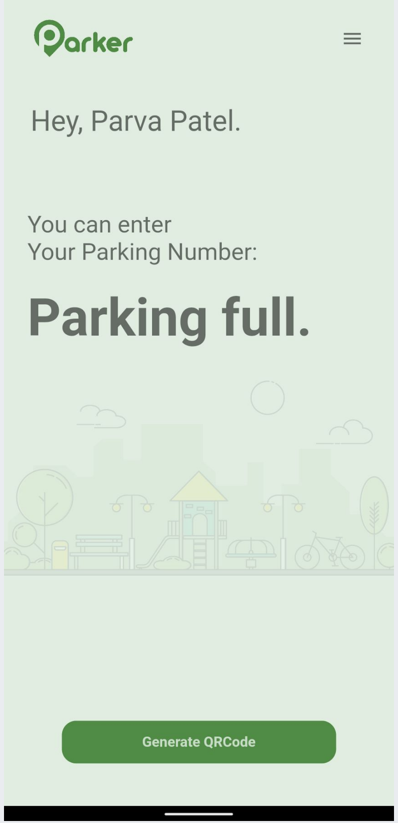
Hey, Parva Patel.

Something is not right.
Try again.

Experiments and Results

Here are the experiments that we ran on our project.

9. Parking is full.



Hey, Parva Patel.

You can enter
Your Parking Number:

Parking full.

Conclusion

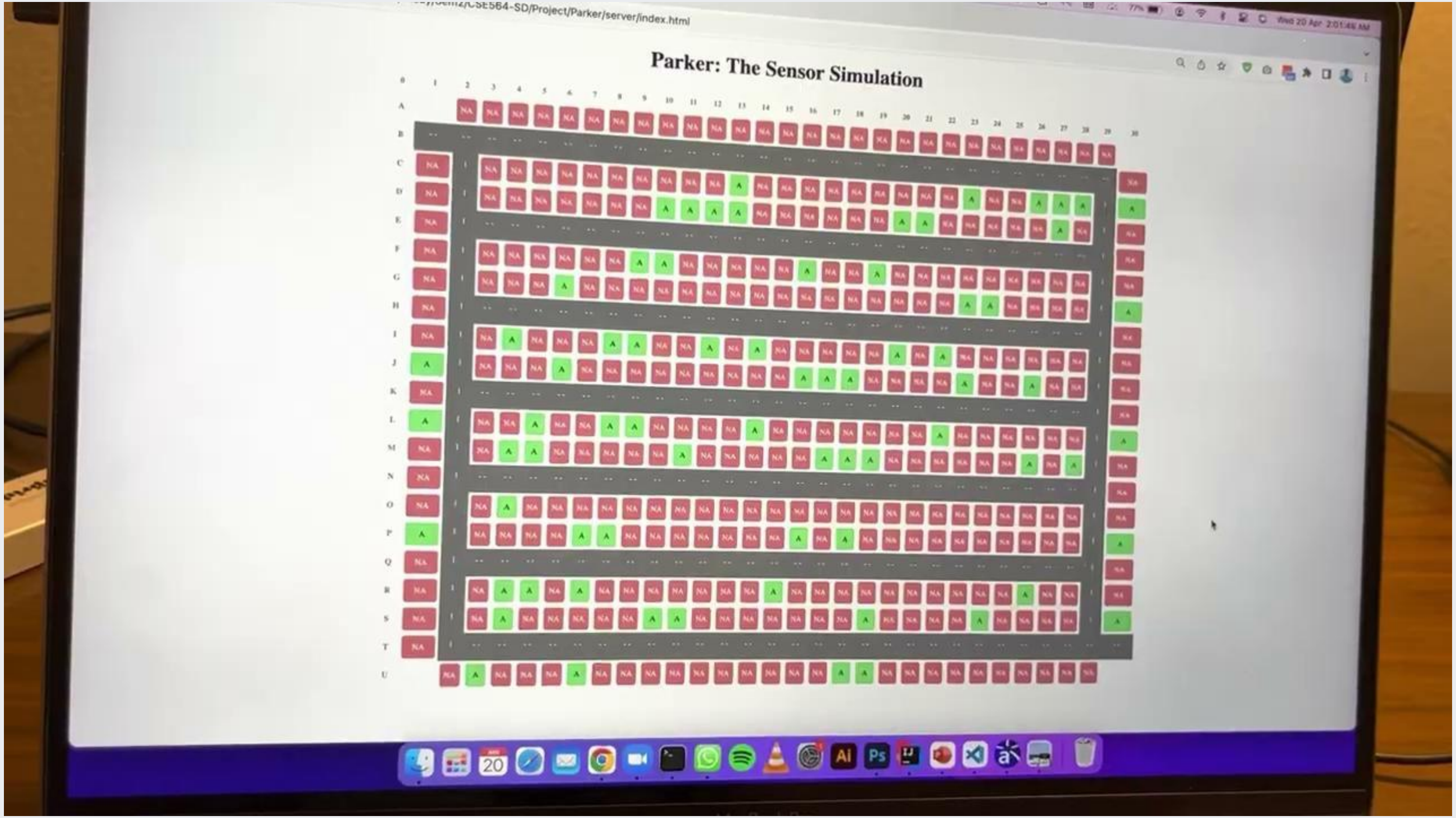
Cyber Component: The user interface is based on a smart phone application, and **Parker** will include services such as login, signup, QR code creation, online payment, parking space and car guiding.

Physical Component: Barcode scanner (Camera), Actuator and Proximity Sensors.

Tools and languages: Java, Flutter (Dart) and Firebase.

A distributed middleware based on a hierarchical **multi-agent framework** is used in SPA to enhance the resilience of CPSs over heterogeneous network. The CPS is analysed taking into account the three layers namely physical, network and application layer.

https://drive.google.com/file/d/1uTkEF732RiZGHEqUcJna2em0vkB9Px-W/view?usp=sharing



…/Sem2/CSE564-SD/Project/Parker/server/index.html

Parker: The Sensor Simulation

Thank you

Any questions?