TakPark: Parking Space Rental Application
Complete Project Overview

Project Documentation

March 6, 2025

Contents

1 Project Overview **2**

2 Technology Stack **2**
2.1 Frontend Technologies . . . 2
2.2 Backend Technologies . . . 2
2.3 Mobile Features . . . 2
2.4 Cloud & Infrastructure . . . 2
2.5 DevOps & Monitoring . . . 3
2.6 Additional Tools . . . 3

3 Docker Configuration **3**
3.1 Docker Development Environment . . . 3
3.1.1 Docker Installation . . . 3
3.1.2 Docker Compose Configuration . . . 3
3.1.3 Backend Dockerfile . . . 4
3.1.4 Frontend Dockerfile . . . 5
3.1.5 Docker Ignore Files . . . 5
3.2 Docker Production Configuration . . . 5
3.2.1 Backend Production Dockerfile . . . 5
3.2.2 Mobile App Production Build . . . 5
3.2.3 Production Docker Compose . . . 6
3.3 Docker Deployment with AWS . . . 7

4 Development Environment Setup **7**
4.1 Docker-based Setup (Recommended) . . . 7
4.2 Windows Setup (Alternative) . . . 8
4.2.1 System Preparation . . . 8
4.2.2 Package Managers and Tools . . . 8
4.2.3 Development Dependencies . . . 8
4.2.4 Android Development Setup . . . 8
4.2.5 Database Setup . . . 9
4.2.6 Project Initialization . . . 9
4.3 Arch Linux Setup (Alternative) . . . 9
4.3.1 System Preparation . . . 9
4.3.2 Node.js Environment Setup . . . 9
4.3.3 Database Setup . . . 9
4.3.4 React Native Dependencies . . . 10
4.3.5 Project Initialization . . . 10
4.3.6 Additional Development Tools . . . 10
4.3.7 Environment Configuration . . . 10

5 Mobile Sensor Features **10**
5.1 Location & Navigation . . . 10
5.2 Camera-based Features . . . 11
5.3 Motion & Orientation Sensors . . . 11
5.4 Ambient Sensors . . . 11
5.5 Proximity & Bluetooth Features . . . 11

6 Sensor Implementation Libraries **11**
6.1 Location Services . . . 11
6.2 Camera & Vision . . . 11
6.3 Motion Sensors . . . 11
6.4 Bluetooth & Proximity . . . 12

7 GitHub Setup and CI/CD Pipeline **12**
7.1 Repository Structure . . . 12
7.2 GitHub Actions Workflow . . . 12

8 Implementation Strategy **14**
8.1 Phase 1: Core Application Setup . . . 14
8.2 Phase 2: Frontend Development . . . 14
8.3 Phase 3: Backend Development . . . 14
8.4 Phase 4: Sensor Integration . . . 14
8.5 Phase 5: Deployment & Optimization . . . 14

9 Docker Development Workflow **15**
9.1 Common Development Tasks . . . 15
9.2 Docker-Specific Tips . . . 15

10 Conclusion **15**

1 Project Overview

TakPark is a mobile application designed to allow users to rent and lend parking spaces. The application leverages modern mobile development technologies, cloud infrastructure, and mobile device sensors to provide an intuitive user experience for both parking space owners and renters. Users can list their available parking spots, while renters can search, book, and pay for parking spaces through the mobile app. The platform uses location-based services to connect drivers with available parking spaces, helping to reduce congestion and optimize the use of urban parking resources.

2 Technology Stack

2.1 Frontend Technologies

- **Framework:** React Native with Expo
- **Language:** TypeScript
- **State Management:** Redux Toolkit
- **Navigation:** React Navigation
- **UI Components:** NativeBase
- **Form Handling:** Formik with Yup validation

2.2 Backend Technologies

- **Language:** Node.js with TypeScript
- **Framework:** NestJS
- **Database:** PostgreSQL
- **ORM:** Prisma
- **Authentication:** Firebase Authentication
- **Real-time Communication:** Socket.io

2.3 Mobile Features

- **Maps:** React Native Maps
- **Geolocation:** Expo Location
- **Payments:** Stripe SDK
- **Push Notifications:** Firebase Cloud Messaging

2.4 Cloud & Infrastructure

- **Cloud Provider:** AWS
- **Hosting:** AWS EC2 or ECS
- **Serverless:** AWS Lambda
- **Containerization:** Docker
- **Caching:** Redis
- **Container Orchestration:** Kubernetes or AWS ECS

2.5 DevOps & Monitoring

- **CI/CD:** GitHub Actions
- **Error Tracking:** Sentry
- **Performance Monitoring:** Firebase Performance Monitoring
- **Container Registry:** AWS ECR or Docker Hub

2.6 Additional Tools

- **API Documentation:** Swagger
- **Testing:** Jest, React Native Testing Library
- **Code Quality:** ESLint, Prettier

3 Docker Configuration

3.1 Docker Development Environment

3.1.1 Docker Installation

```
# Windows (using Chocolatey)
choco install docker-desktop -y

# Arch Linux
sudo pacman -S docker
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER

# macOS
brew install --cask docker
```

3.1.2 Docker Compose Configuration

```
version: '3.8'

services:
  # Backend API Service
  backend:
    build:
      context: ./takpark-backend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=development
      - DATABASE_URL=postgresql://postgres:postgres@db:5432/takpark
      - REDIS_URL=redis://redis:6379
    volumes:
      - ./takpark-backend:/usr/src/app
      - /usr/src/app/node_modules
    depends_on:
      - db
      - redis
    networks:
      - takpark-network

  # Frontend Development Server
  frontend:
    build:
      context: ./takpark-mobile
      dockerfile: Dockerfile.dev
    ports:
```

```
      - "19000:19000" # Expo main port
      - "19001:19001" # Metro bundler
      - "19002:19002" # Expo web UI
    environment:
      - EXPO_DEVTOOLS_LISTEN_ADDRESS=0.0.0.0
      - REACT_NATIVE_PACKAGER_HOSTNAME=localhost
    volumes:
      - ./takpark-mobile:/usr/src/app
      - /usr/src/app/node_modules
    networks:
      - takpark-network

  # PostgreSQL Database
  db:
    image: postgres:14
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=takpark
    volumes:
      - postgres-data:/var/lib/postgresql/data
    networks:
      - takpark-network

  # Redis Cache
  redis:
    image: redis:alpine
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
    networks:
      - takpark-network

  # Adminer Database Management Tool
  adminer:
    image: adminer
    ports:
      - "8080:8080"
    depends_on:
      - db
    networks:
      - takpark-network

networks:
  takpark-network:
    driver: bridge

volumes:
  postgres-data:
  redis-data:
```

3.1.3 Backend Dockerfile

```
FROM node:20-alpine

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "run", "start:dev"]
```

3.1.4 Frontend Dockerfile

```
FROM node:20-alpine

WORKDIR /usr/src/app

# Install expo-cli globally
RUN npm install -g expo-cli

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 19000 19001 19002

CMD ["npm", "start"]
```

3.1.5 Docker Ignore Files

```
node_modules
npm-debug.log
build
.dockerignore
.git
.gitignore
.env
```

3.2 Docker Production Configuration

3.2.1 Backend Production Dockerfile

```
FROM node:20-alpine AS builder

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm ci

COPY . .

RUN npm run build

# Production image
FROM node:20-alpine

WORKDIR /usr/src/app

COPY --from=builder /usr/src/app/dist ./dist
COPY --from=builder /usr/src/app/node_modules ./node_modules
COPY --from=builder /usr/src/app/package*.json ./

ENV NODE_ENV production

EXPOSE 3000

CMD ["node", "dist/main"]
```

3.2.2 Mobile App Production Build

```
FROM node:20-alpine AS builder

WORKDIR /usr/src/app

# Install expo-cli globally
```

```
RUN npm install -g expo-cli eas-cli

COPY package*.json ./

RUN npm ci

COPY . .

# Build for specific platforms
RUN npx eas-cli build --platform android --non-interactive --no-wait

# This is a build step only - actual build happens on EAS servers
```

3.2.3 Production Docker Compose

```
version: '3.8'

services:
  # Backend API Service - Production
  backend:
    build:
      context: ./takpark-backend
      dockerfile: Dockerfile.prod
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - DATABASE_URL=postgresql://postgres:${DB_PASSWORD}@db:5432/takpark
      - REDIS_URL=redis://redis:6379
    depends_on:
      - db
      - redis
    networks:
      - takpark-network
    deploy:
      replicas: 2
      update_config:
        parallelism: 1
        delay: 10s
      restart_policy:
        condition: on-failure

  # PostgreSQL Database
  db:
    image: postgres:14
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=${DB_PASSWORD}
      - POSTGRES_DB=takpark
    volumes:
      - postgres-data:/var/lib/postgresql/data
    networks:
      - takpark-network
    deploy:
      placement:
        constraints: [node.role == manager]

  # Redis Cache
  redis:
    image: redis:alpine
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
    networks:
      - takpark-network
    deploy:
      placement:
        constraints: [node.role == manager]

```

```
  # Nginx for API Gateway
  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/ssl:/etc/nginx/ssl
    depends_on:
      - backend
    networks:
      - takpark-network
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]

networks:
  takpark-network:
    driver: overlay

volumes:
  postgres-data:
  redis-data:
```

3.3 Docker Deployment with AWS

```
# Configure AWS CLI
aws configure

# Create ECR Repository for backend
aws ecr create-repository --repository-name takpark-backend

# Build and push backend image
docker build -t takpark-backend:latest -f Dockerfile.prod .
docker tag takpark-backend:latest ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/
    takpark-backend:latest
aws ecr get-login-password | docker login --username AWS --password-stdin ${AWS_ACCOUNT_ID}.
    dkr.ecr.${AWS_REGION}.amazonaws.com
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/takpark-backend:latest

# Create ECS cluster
aws ecs create-cluster --cluster-name takpark-cluster

# Deploy services using AWS CloudFormation
aws cloudformation deploy \
  --template-file ecs-deployment.yml \
  --stack-name takpark-stack \
  --parameter-overrides \
    ClusterName=takpark-cluster \
    ImageURI=${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/takpark-backend:latest
```

4 Development Environment Setup

4.1 Docker-based Setup (Recommended)

```
# Clone the repository
git clone https://github.com/yourusername/takpark.git
cd takpark

# Start the Docker development environment
docker-compose up -d

# Access the services
# Backend: http://localhost:3000
# Database Admin: http://localhost:8080
# Expo: http://localhost:19002
```

```
# Run commands inside containers
docker-compose exec backend npm run migration:run
docker-compose exec frontend npm install some-package

# View logs
docker-compose logs -f backend
```

4.2 Windows Setup (Alternative)

4.2.1 System Preparation

```
# Install Node.js and npm from the official website
# https://nodejs.org/en/download/

# Install Git from the official website
# https://git-scm.com/download/win

# Install VS Code from the official website
# https://code.visualstudio.com/download
```

4.2.2 Package Managers and Tools

```
# Install Chocolatey (Open PowerShell as Administrator)
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::
    SecurityProtocol -bor 3072;
iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
    install.ps1'))

# Install additional tools using Chocolatey
choco install -y yarn
choco install -y python
choco install -y postman
choco install -y docker-desktop
```

4.2.3 Development Dependencies

```
# Install global npm packages
npm install -g expo-cli
npm install -g typescript
npm install -g @nestjs/cli
npm install -g prisma

# Install Java JDK for Android development
choco install -y openjdk11
```

4.2.4 Android Development Setup

```
# Install Android Studio from the official website
# https://developer.android.com/studio

# After installation, use Android Studio to install:
# - Android SDK
# - Android SDK Platform-Tools
# - Android Virtual Device

# Set environment variables (PowerShell)
[Environment]::SetEnvironmentVariable("ANDROID_HOME", "C:\Users\YOUR_USERNAME\AppData\Local\
    Android\Sdk", "User")
[Environment]::SetEnvironmentVariable("Path", $env:Path + ";%ANDROID_HOME%\platform-tools", "
    User")
```

4.2.5 Database Setup

```
# Install PostgreSQL
choco install -y postgresql

# Start PostgreSQL service
net start postgresql

# Create database (in Command Prompt)
createdb -U postgres takpark
```

4.2.6 Project Initialization

```
# Create project directory
mkdir C:\Projects\TakPark
cd C:\Projects\TakPark

# Initialize React Native project
expo init takpark-mobile
cd takpark-mobile
npm install

# Initialize backend
cd ..
nest new takpark-backend
cd takpark-backend
npm install @prisma/client
npx prisma init
```

4.3 Arch Linux Setup (Alternative)

4.3.1 System Preparation

```
# System update
sudo pacman -Syu

# Install essential development tools
sudo pacman -S base-devel git curl wget
sudo pacman -S nodejs npm
sudo pacman -S yarn
sudo pacman -S postgresql
sudo systemctl enable postgresql
sudo systemctl start postgresql
sudo pacman -S redis
sudo systemctl enable redis
sudo systemctl start redis
```

4.3.2 Node.js Environment Setup

```
# Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
source ~/.bashrc

# Install LTS Node.js
nvm install --lts
nvm use --lts

# Install global packages
npm install -g expo-cli typescript @nestjs/cli prisma
```

4.3.3 Database Setup

```
# Initialize PostgreSQL
sudo -u postgres initdb --locale en_US.UTF-8 -E UTF8 -D /var/lib/postgres/data

```

```
# Create database user
sudo -u postgres createuser --interactive

# Create database
sudo -u postgres createdb takpark
```

4.3.4 React Native Dependencies

```
# Android tools
sudo pacman -S android-tools android-udev
sudo pacman -S jdk17-openjdk

# Android Studio (optional)
yay -S android-studio
```

4.3.5 Project Initialization

```
# Create project directories
mkdir -p ~/Projects/TakPark
cd ~/Projects/TakPark

# Initialize React Native project
expo init takpark-mobile
cd takpark-mobile
npm install

# Initialize backend
nest new takpark-backend
cd takpark-backend
npm install @prisma/client
npx prisma init
```

4.3.6 Additional Development Tools

```
# VS Code
sudo pacman -S code

# Docker setup
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER
```

4.3.7 Environment Configuration

```
# Add to ~/.bashrc or ~/.zshrc
export ANDROID_HOME=$HOME/Android/Sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

5 Mobile Sensor Features

5.1 Location & Navigation

- **GPS-based Parking Spot Locating:** Use geolocation to show nearby available parking spots
- **Turn-by-turn Navigation:** Guide users to their reserved parking spot
- **Geofencing:** Automatically detect arrival/departure from parking spots
- **Proximity Alerts:** Notify users when they're near their reserved spot

5.2 Camera-based Features

- **License Plate Recognition:** Scan and verify vehicles using the device camera
- **QR Code Scanning:** Quick check-in/check-out by scanning codes at parking locations
- **Augmented Reality Guidance:** Visual overlay showing directions to parking spots
- **Parking Spot Photo Verification:** Allow owners to upload photos of their spots

5.3 Motion & Orientation Sensors

- **Automatic Parking Detection:** Use accelerometer to detect when a car has parked
- **Vehicle Movement Alerts:** Notify owners if their vehicle moves unexpectedly
- **Parking Assistance:** Use sensors to help guide parking in tight spaces
- **Shake-to-Report:** Quick issue reporting by shaking the device

5.4 Ambient Sensors

- **Weather-based Pricing:** Adjust rates based on local weather conditions
- **Light Detection:** Automatically brighten app UI in low-light conditions
- **Temperature Monitoring:** Alert users about extreme temperatures that might affect vehicles

5.5 Proximity & Bluetooth Features

- **Bluetooth Beacon Integration:** Precise indoor parking navigation
- **Contactless Access:** Use NFC or Bluetooth to open parking gates/barriers
- **Proximity-based Check-in:** Automatic check-in when arriving at parking location
- **Car Finding Feature:** Help users locate their parked car using Bluetooth signal strength

6 Sensor Implementation Libraries

6.1 Location Services

- `react-native-geolocation-service`: Precise GPS tracking
- `react-native-maps`: Map visualization
- `@react-native-community/geolocation`: Basic location features

6.2 Camera & Vision

- `react-native-vision-camera`: High-performance camera access
- `react-native-camera`: QR/barcode scanning
- `react-native-text-recognition`: License plate recognition

6.3 Motion Sensors

- `react-native-sensors`: Accelerometer, gyroscope access
- `expo-sensors`: If using Expo

6.4 Bluetooth & Proximity

- `react-native-ble-plx`: Bluetooth Low Energy communication
- `react-native-nfc-manager`: NFC capabilities

7 GitHub Setup and CI/CD Pipeline

7.1 Repository Structure

```
takpark/
 .github/
     workflows/
         backend-ci.yml
         mobile-ci.yml
 takpark-backend/
     src/
     test/
     Dockerfile
     Dockerfile.prod
     package.json
 takpark-mobile/
     src/
     Dockerfile.dev
     app.json
     package.json
 docker-compose.yml
 docker-compose.prod.yml
 nginx/
     nginx.conf
 README.md
```

7.2 GitHub Actions Workflow

```yaml
name: Backend CI

on:
  push:
    branches: [ main ]
    paths:
      - 'takpark-backend/**'
  pull_request:
    branches: [ main ]
    paths:
      - 'takpark-backend/**'

jobs:
  test:
    runs-on: ubuntu-latest

    services:
      postgres:
        image: postgres:14
        env:
          POSTGRES_USER: postgres
          POSTGRES_PASSWORD: postgres
          POSTGRES_DB: takpark_test
        ports:
          - 5432:5432
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5

    steps:
    - uses: actions/checkout@v3

    - name: Set up Node.js
```

```
      uses: actions/setup-node@v3
      with:
        node-version: '20'
        cache: 'npm'
        cache-dependency-path: './takpark-backend/package-lock.json'

    - name: Install dependencies
      run: |
        cd takpark-backend
        npm ci

    - name: Run tests
      run: |
        cd takpark-backend
        npm run test
      env:
        DATABASE_URL: postgresql://postgres:postgres@localhost:5432/takpark_test

    - name: Build Docker image
      if: github.event_name != 'pull_request'
      run: |
        cd takpark-backend
        docker build -t takpark-backend:${{ github.sha }} -f Dockerfile.prod .

    - name: Push to ECR
      if: github.event_name != 'pull_request'
      uses: aws-actions/amazon-ecr-login@v1
      with:
        registry: ${{ secrets.AWS_ACCOUNT_ID }}.dkr.ecr.${{ secrets.AWS_REGION }}.amazonaws.
    com

    - name: Tag and push image
      if: github.event_name != 'pull_request'
      run: |
        docker tag takpark-backend:${{ github.sha }} ${{ secrets.AWS_ACCOUNT_ID }}.dkr.ecr.${{
     secrets.AWS_REGION }}.amazonaws.com/takpark-backend:latest
        docker push ${{ secrets.AWS_ACCOUNT_ID }}.dkr.ecr.${{ secrets.AWS_REGION }}.amazonaws.
    com/takpark-backend:latest
```

```
name: Mobile CI

on:
  push:
    branches: [ main ]
    paths:
      - 'takpark-mobile/**'
  pull_request:
    branches: [ main ]
    paths:
      - 'takpark-mobile/**'

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '20'
        cache: 'npm'
        cache-dependency-path: './takpark-mobile/package-lock.json'

    - name: Install dependencies
      run: |
        cd takpark-mobile
        npm ci

    - name: Run tests
      run: |
        cd takpark-mobile
```

```
      npm run test

  - name: Build Expo project
    if: github.event_name != 'pull_request'
    run: |
      cd takpark-mobile
      npx expo prebuild

  - name: Set up EAS
    if: github.event_name != 'pull_request'
    run: |
      npm install -g eas-cli
      cd takpark-mobile
      eas build --platform android --non-interactive --auto-submit
    env:
      EXPO_TOKEN: ${{ secrets.EXPO_TOKEN }}
```

8 Implementation Strategy

8.1 Phase 1: Core Application Setup

- Set up project repositories and Docker configuration
- Configure CI/CD pipelines with GitHub Actions
- Implement basic authentication with Firebase
- Create database schema with Prisma

8.2 Phase 2: Frontend Development

- Implement UI components with NativeBase
- Set up navigation flow with React Navigation
- Integrate Redux Toolkit for state management
- Create map integration with parking spot visualization

8.3 Phase 3: Backend Development

- Develop RESTful API endpoints
- Implement business logic for parking space rentals
- Configure payment processing with Stripe
- Set up real-time updates with Socket.io

8.4 Phase 4: Sensor Integration

- Implement geolocation and geofencing features
- Add camera-based QR code scanning
- Develop motion sensor integrations
- Optimize battery usage for continuous monitoring

8.5 Phase 5: Deployment & Optimization

- Deploy containers to AWS ECS
- Set up auto-scaling and load balancing
- Implement performance monitoring and alerting
- Configure database backups and disaster recovery

9 Docker Development Workflow

9.1 Common Development Tasks

```
# Start all services
docker-compose up -d

# Stop all services
docker-compose down

# View logs
docker-compose logs -f

# Rebuild containers after dependency changes
docker-compose up -d --build

# Run database migrations
docker-compose exec backend npm run migration:run

# Access PostgreSQL database
docker-compose exec db psql -U postgres -d takpark

# Install new packages in backend
docker-compose exec backend npm install package-name

# Install new packages in frontend
docker-compose exec frontend npm install package-name

# Run tests
docker-compose exec backend npm test
docker-compose exec frontend npm test
```

9.2 Docker-Specific Tips

- **Volume Mounts**: The Docker setup uses volume mounts to ensure code changes are immediately reflected in the running containers without rebuilding.
- **Hot Reloading**: Both NestJS and Expo have built-in hot reloading that works with the Docker setup.
- **Container Networking**: All containers are on the same network, allowing them to communicate by service name (e.g., backend can reach db via "db:5432").
- **Data Persistence**: Database data is stored in Docker volumes to persist between container restarts.
- **Performance**: For better performance on macOS and Windows, consider using Docker Desktop's resource allocation settings to provide more CPU and memory.

10 Conclusion

The TakPark application combines modern web and mobile technologies with advanced sensor capabilities to create a comprehensive parking space rental platform. By following the implementation strategy outlined in this document, the development team can efficiently build a scalable, performant application that provides value to both parking space owners and renters.

The Docker-based development environment ensures consistent development experiences across all platforms, streamlining the onboarding process and eliminating environment-specific issues. The containerized approach also aligns with modern DevOps practices, facilitating CI/CD integration and cloud deployment.

By leveraging mobile sensor capabilities such as geolocation, camera, motion sensors, and Bluetooth, TakPark can provide a seamless user experience that goes beyond simple booking functionalities, creating a truly modern and innovative parking solution.