SMART CAR PARKING SYSTEM

1. Introduction

The Smart Car Parking System is a mobile-based parking management solution built using **Flutter (Frontend)**, **Node.js (Backend)**, **MySQL (storage)**, **Firebase (real-time sync)**, **Twilio (SMS)**, and **M-Pesa Daraja Ratiba Sandbox** for automated payments.
The system allows operators to assign parking slots, reserve them, communicate with drivers via SMS, and automate payment upon exit.

This final documentation presents the project structure, user interface, system architecture, workflows, and the newly added features such as **slot reservation**, **Ngrok tunnels**, and **Firebase emulator hosting**.

2. Problem Statement

Traditional parking systems rely heavily on manual processes which are time-consuming, prone to errors, and do not scale well. Parking operators must track vehicles manually, handle confirmations, and collect payments physically — resulting in long wait times and operational inefficiencies.

3. Objectives

Main Objective

To develop a fully automated mobile-based Smart Parking System that simplifies slot assignment, reservation, notifications, and payment.

Specific Objectives

- Automate slot selection and assignment.
- Implement slot reservation and confirmation.
- Provide real-time slot status display using Firebase.
- Send SMS confirmations using Twilio.
- Integrate automated M-Pesa payment prompts via Daraja Ratiba Sandbox.
- Use Ngrok to expose local webhook endpoints for SMS and M-Pesa callbacks.
- Manage parking duration and calculate charges automatically.
- Store historical data using MySQL.

4. System Requirements

Software Requirements

- Flutter SDK
- Node.js & NPM
- MySQL Server / XAMPP
- Firebase CLI & Emulator Suite
- Ngrok
- Twilio SMS API
- M-Pesa Ratiba Sandbox API
- VS Code / Android Studio

Hardware Requirements

- Android phone or Tablet
- Laptop or PC with 8GB RAM+
- Stable internet

5. Project Structure

flutter_app(Flutter Frontend)/

lib/

services/

api_service.dart

firebase_service.dart

screens/

grid_screen.dart

entry_form.dart

reserve_sheet.dart

models/

parking_slot.dart

services/firebase_service.dart

main.dart

pubspec.yaml

backend(Node.js)/

routes/

slots.js

reservations.js

mpesa.js

twilio.js

sql/

schema.sql

DB_TYPE=mysql.sh

server.js

db.js

cron.js

parking.db

Safaricom APIs.postman_collection.json

setup-db.js

firebase-service-account.json

.env

setup-ngrok.ps1

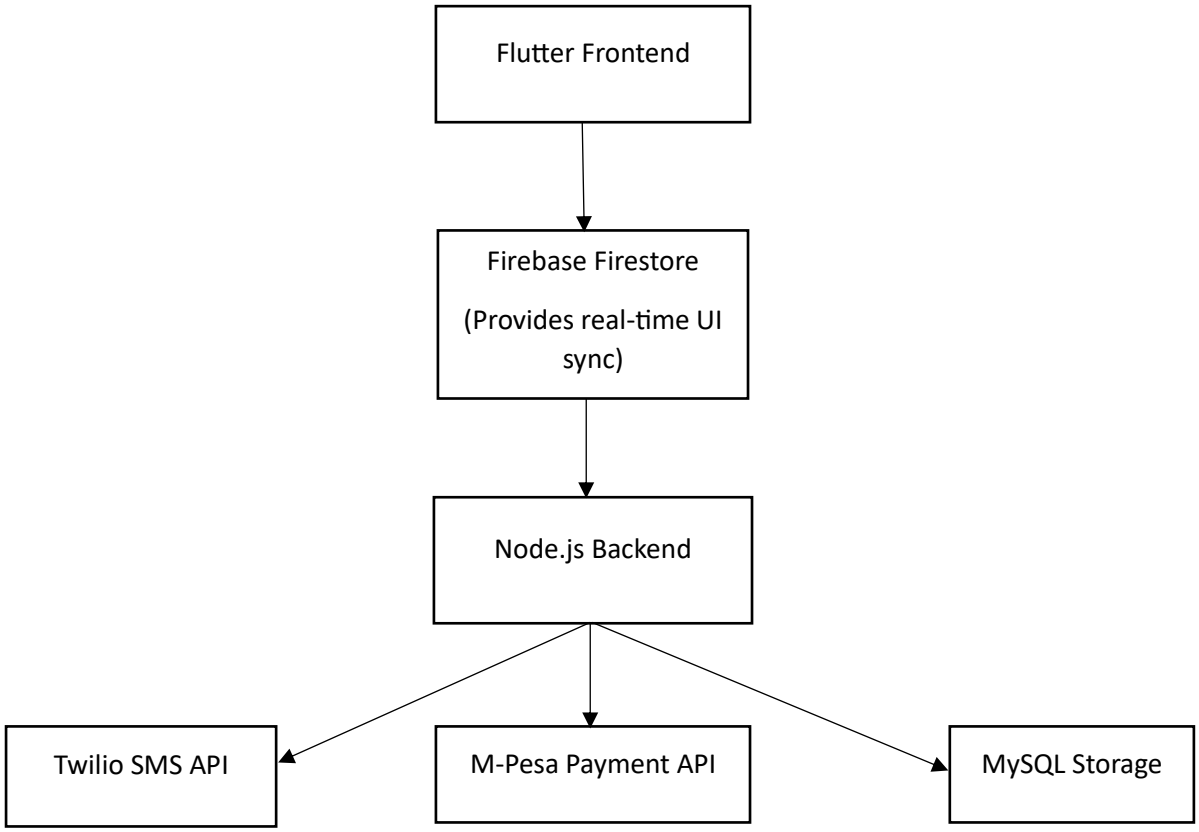
setup-ngrok.sh

Firebase Emulator Config

firebase.json > ...

```
{
    "emulators": {
        "firestore": {
            "host": "localhost",
            "port": 8080
        },
        "ui": {
            "enabled": true,
            "host": "localhost",
            "port": 4000
        }
    }
}
```

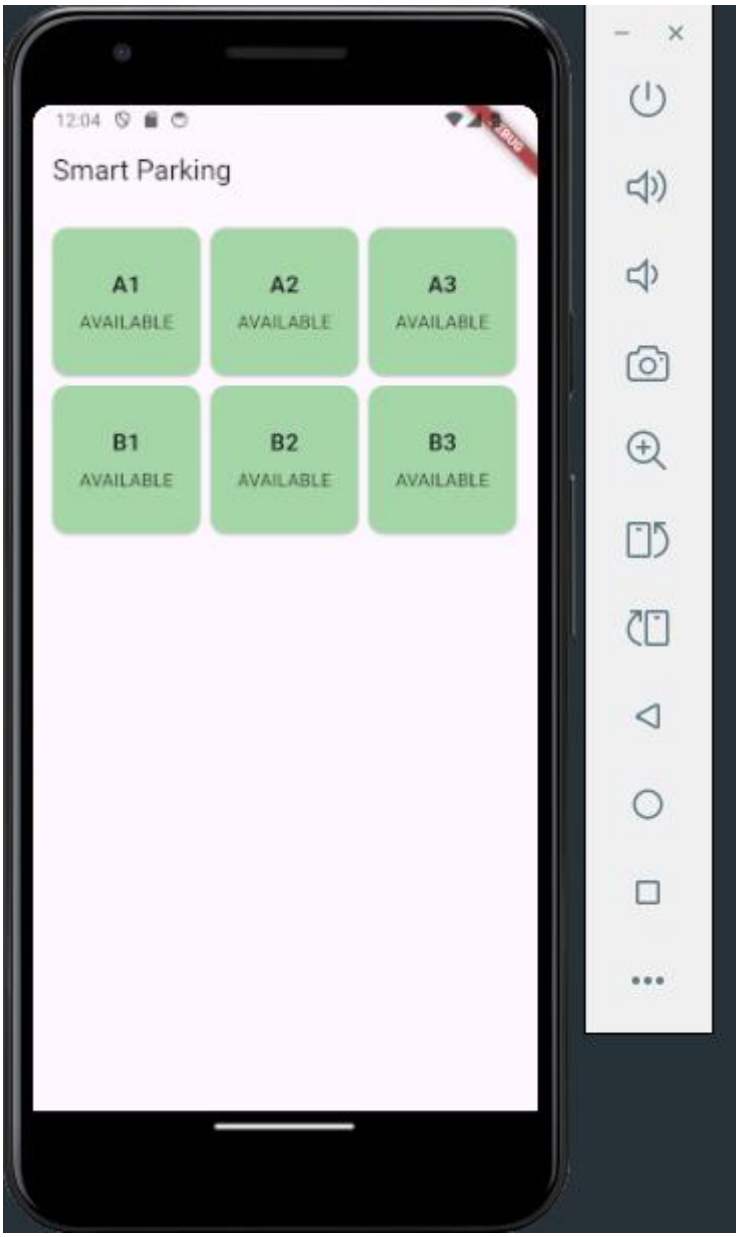
6. System Architecture



7. User Interface Design (UI Screens)

7.1 Slot Grid Screen

- Green = Available
- Yellow = Reserved
- Red = Occupied



7.2 Slot Assignment Screen

- Enter vehicle plate
- Enter phone number
- Confirm selection

- Triggers SMS

7.3 Reservation Screen

- Reserve slot
- Cancel reservation
- Confirm reservation → becomes occupied

❖ ***For the above reference 9. Reservation features***

8. System Workflow

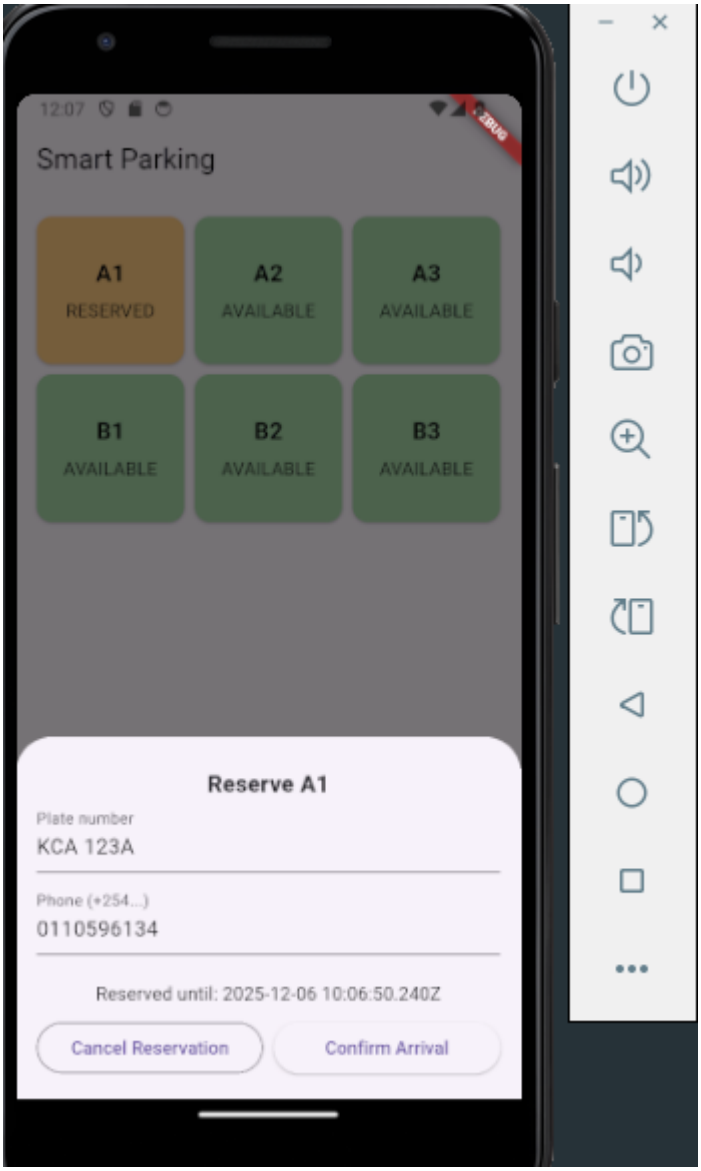
Parking Flow

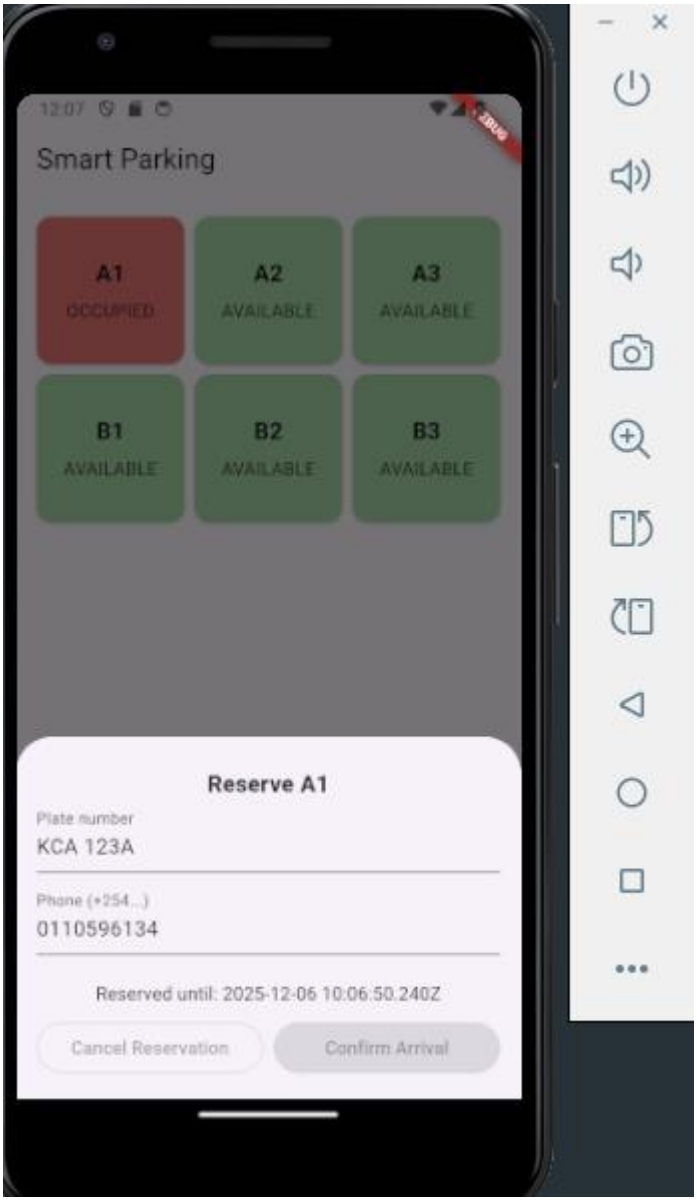
1. Driver arrives
2. Operator selects available slot
3. Operator enters details
4. System sends SMS via Twilio
5. Parking time begins
6. Driver leaves
7. System auto-calculates fee
8. M-Pesa STK push sent
9. Upon payment, slot becomes free

9. Reservation Feature

Reservation Workflow

1. Operator taps a green slot
2. Selects **Reserve Slot**
3. Slot turns **yellow** when reserved
4. Operator may **confirm** or **cancel**
5. Confirm → slot turns **red** (assigned)
6. Cancel → slot returns to **green**
7. System logs reservation to MySQL + Firestore
8. Firestore pushes real-time UI updates
9. Reservation auto-expires if not confirmed


12:07
DEBUG
Smart Parking
A1
RESERVED
A2
AVAILABLE
A3
AVAILABLE
B1
AVAILABLE
B2
AVAILABLE
B3
AVAILABLE
Reserve A1
Plate number
KCA 123A
Phone (+254...)
0110596134
Reserved until: 2025-12-06 10:06:50.240Z
Cancel Reservation
Confirm Arrival



10. SMS Notification Feature (Twilio Integration)

When SMS is sent:

- Reservation confirmation
- Slot assignment confirmation
- Parking duration summary
- Payment request

Twilio Webhook Setup

You must verify using:

https://YOUR_NGROK_URL/twilio/sms

Twilio Dashboard → Messaging → Webhooks → set the above URL.

11. Payment System: M-Pesa Ratiba Sandbox

Payment Flow

1. Driver leaves
2. Backend calls M-Pesa /stkpush API
3. Ngrok exposes callback URL
4. M-Pesa responds to callback with payment result
5. Backend validates payment
6. Firestore updates slot → green

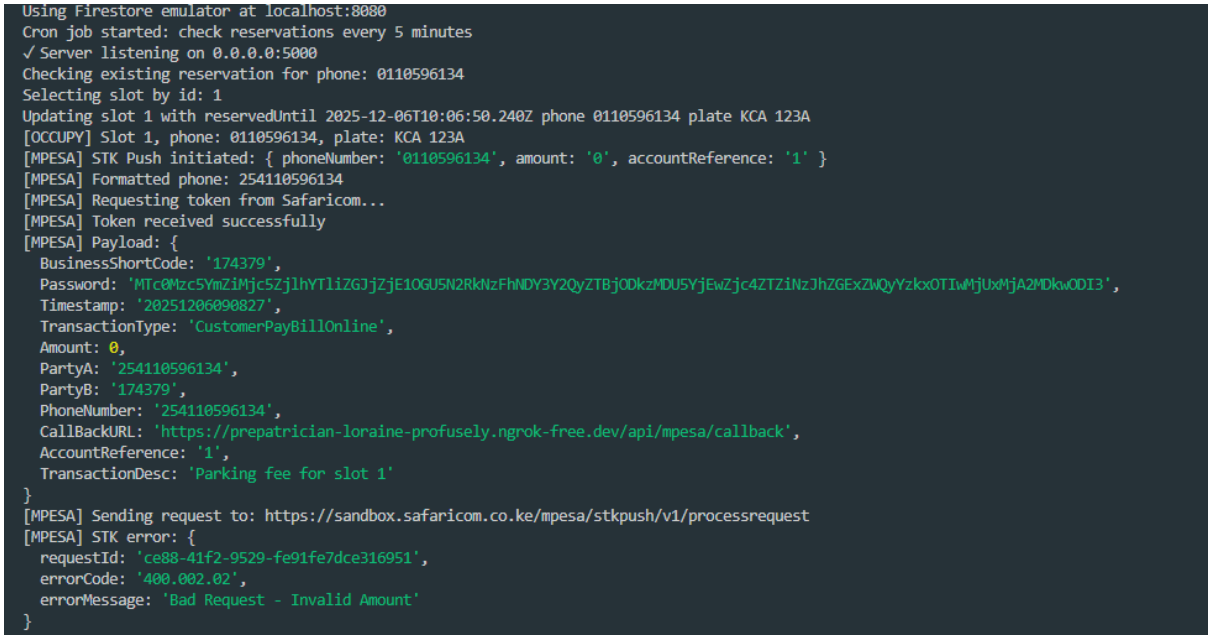
Callback URL

https:// NGROK_URL/mpesa/callback

Important

Daraja Sandbox must whitelist Ngrok URLs manually in test settings.

- ***Invalid payment message; is sent when the amount paid is less than 1 ksh.***



❖ ***Successful payment transaction.***

12. Ngrok Integration

Ngrok is used to expose your local backend to the internet for:

- Twilio SMS delivery
- M-Pesa payment callback

How to set Ngrok

ngrok http 5000

You will get a URL like:

https://3f39-102-34-11-210.ngrok-free.app

Add this to:

- Twilio webhook
- M-Pesa callback URL

13. Database Design

MySQL Tables

1. **slots** – stores slot states
2. **reservations** – stores temporary reservations
3. **transactions** – parking history
4. **payments** – payment confirmations

❖ ***Payments table***

Properties DATA Log ER Monitor

```
SELECT * FROM payments LIMIT 100
```

Search Results | Export | Cost: 215ms | 1 | Total 2

id INTEGER	slot_ INTEGER	phone TEXT	amount REAL	mpesa_receipt TEXT	status TEXT	created_at TEXT
10	1	0110596134	1.0	(NULL)	pending	2025-12-06 09:08:43
11	2	0110596314	1.0	(NULL)	pending	2025-12-06 09:11:40

❖ ***Slots table***

Properties DATA Log ER Monitor

```
SELECT * FROM slots LIMIT 100
```

Search Results | Export | Cost: 9ms | 1 | Total 6

id INTEGER	name TEXT	status TEXT	reserved_by TEXT	reserved_until TEXT	start_time TEXT	end_time TEXT
1	A1	available	0110596134	2025-12-06T10:06:50.240Z	2025-12-06T09:07:24.028Z	2025-12-06T09:08:43.938Z
2	A2	available	+254110596134	2025-11-18T10:31:35.328Z	2025-12-06T09:10:18.570Z	2025-12-06T09:11:40.680Z
3	A3	available	(NULL)	(NULL)	2025-11-24T06:13:45.777Z	2025-11-24T17:13:16.341Z
4	B1	available	(NULL)	(NULL)	2025-11-18T14:17:54.241Z	2025-11-24T17:13:24.836Z
5	B2	available	(NULL)	(NULL)	2025-11-29T10:19:42.308Z	2025-11-29T10:20:46.764Z
6	B3	available	(NULL)	(NULL)	2025-11-19T09:41:51.759Z	2025-11-24T06:14:23.027Z

plate TEXT	phone TEXT	created_at TEXT	updated_at TEXT
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52
(NULL)	(NULL)	2025-11-11 19:59:52	2025-11-11 19:59:52

Firestore Structure

- Firestore handles **real-time UI sync**.

slots/

slot_1/

status: "reserved"

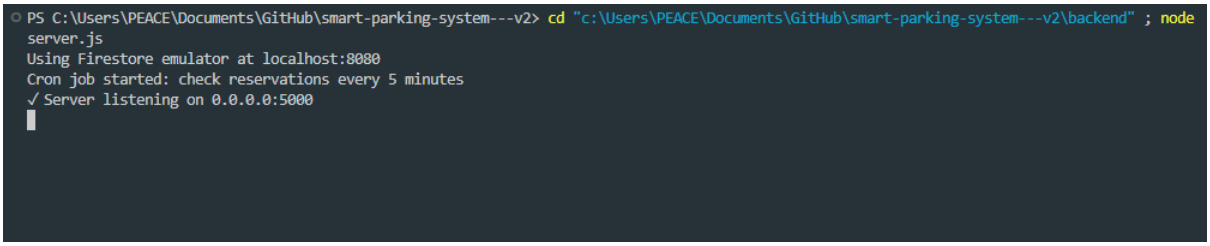
reserved_by: "KCA 123A"

14. Backend Implementation (Node.js)

Server Components

- **Cron.js** for routes
- **Twilio SDK** for SMS
- **Axios** for M-Pesa requests
- **Firebase Admin SDK** for Firestore sync

❖ ***Demonstration:***



Main Routes

POST /reserve

POST /confirm-reservation

POST /cancel-reservation

POST /assign

POST /mpesa/stkpush

POST /mpesa/callback

POST /twilio/sms

GET /slots

15. Firebase Hosting & Emulator Setup

Emulator Running

firebase emulators:start

Firebase.json

Why Emulator?

- Free to run
- Perfect for local development
- Works seamlessly with Node.js backend

16. Testing and Validation

Tests Performed

- UI functionality
- Reservation expiry
- SMS delivery
- Payment callback
- Database synchronization

Tools Used

- Postman
- Flutter Integration Test
- Firebase Emulator Suite
- MySQL Workbench

Troubleshooting

Error	Cause	Fix
`Invalid CallBackURL`	Localhost not public	Use ngrok
`System is busy`	Safaricom sandbox overloaded	Retry after 1-2 minutes
`Invalid credentials`	Wrong consumer key/secret	Verify .env values
`Invalid phone number`	Format issue	Use format: `+254...`
`Invalid amount`	Amount < 1 KES	Ensure amount ≥ 1

17. Security Considerations

- .env protects API keys
- Firebase service account is not committed
- HTTPS via Ngrok
- Validation before processing M-Pesa callback

18. Limitations & Future Improvements

Current Limitations

- Only reservation by operator (no mobile driver app)
- Payment depends on sandbox
- Requires stable internet

Future Improvements

- Optical Number Plate Recognition (ANPR)
- Full driver mobile application
- QR-code parking entry
- Expand to multi-story parking

19. Conclusion

The project successfully demonstrates a fully working **Smart Car Parking System** integrating **real-time sync, SMS automation, reservation logic, and mobile money payments**. By combining Flutter,

Firebase, MySQL, Ngrok, Twilio, and M-Pesa Ratiba Sandbox, the system provides a modern, automated, and scalable parking management solution suitable for real-world deployment.