To create a smart parking allotment app code, we can use the following steps:

1. Create a database to store the following information:

- Parking lot information (name, address, number of spaces, etc.)
- Parking space information (location, availability, etc.)
- User information (name, vehicle information, etc.)

2. Create a backend server to handle the following tasks:

- Receive parking lot and parking space information from sensors in real time.
- Update the database with the latest parking information.
- Process user requests to find and reserve parking spaces.

3. Create a mobile app for users to interact with the system:

- Allow users to search for parking lots and parking spaces near them.
- Show users the availability of parking spaces in real time.
- Allow users to reserve parking spaces in advance.

Here is a sample code for the backend server:

Python

```
import json
import sqlite3

class ParkingServer:
    def __init__(self):
        self.db = sqlite3.connect('parking.db')
        self.create_tables()

    def create_tables(self):
        self.db.execute('''CREATE TABLE IF NOT EXISTS parking_lots (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            address TEXT NOT NULL,
            num_spaces INTEGER NOT NULL
        )''')

        self.db.execute('''CREATE TABLE IF NOT EXISTS parking_spaces (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            parking_lot_id INTEGER NOT NULL,
```

```
            location TEXT NOT NULL,
            availability INTEGER NOT NULL
        )''')

        self.db.execute('''CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            vehicle_info TEXT NOT NULL
        )''')

    def get_parking_lots(self):
        cursor = self.db.execute('SELECT * FROM parking_lots')
        parking_lots = cursor.fetchall()

        return parking_lots

    def get_parking_spaces(self, parking_lot_id):
        cursor = self.db.execute('SELECT * FROM parking_spaces WHERE
parking_lot_id = ?', (parking_lot_id,))
        parking_spaces = cursor.fetchall()

        return parking_spaces

    def reserve_parking_space(self, user_id, parking_space_id):
        cursor = self.db.execute('UPDATE parking_spaces SET
availability = 0 WHERE id = ?', (parking_space_id,))
        self.db.commit()

        cursor = self.db.execute('INSERT INTO reservations (user_id,
parking_space_id) VALUES (?, ?)', (user_id, parking_space_id))
        self.db.commit()

    def release_parking_space(self, user_id, parking_space_id):
        cursor = self.db.execute('UPDATE parking_spaces SET
availability = 1 WHERE id = ?', (parking_space_id,))
        self.db.commit()

        cursor = self.db.execute('DELETE FROM reservations WHERE
user_id = ? AND parking_space_id = ?', (user_id, parking_space_id))
        self.db.commit()

if __name__ == '__main__':
    server = ParkingServer()

    # Get all parking lots
    parking_lots = server.get_parking_lots()

    # Get all parking spaces in a specific parking lot
```

```
    parking_spaces = server.get_parking_spaces(parking_lots[0]['id'])

    # Reserve a parking space
    server.reserve_parking_space(1, parking_spaces[0]['id'])

    # Release a parking space
    server.release_parking_space(1, parking_spaces[0]['id'])
```

This is just a sample code, and you may need to modify it to meet your specific needs. For example, you may want to add additional features such as payment processing or real-time navigation.