Certainly! Documenting a project for submission typically involves creating a comprehensive report that includes the following components:

1. Title Page
   - Title of the project
   - Your name
   - Date

2. Table of Contents:
   - List of sections and subsections with page numbers.

3. Executive Summary:
   - A brief overview of the project, its goals, and key findings.

4. Introduction:
   - Background information about the project.
   - Objectives and goals.

5. Methodology:
   - Explain how you conducted the project.
   - Mention tools, technologies, and methodologies used.

6. System Design:
   - Detailed description of the system architecture.
   - Diagrams, flowcharts, or other visual aids.

7. Implementation:
   - Describe how you built the Smart Parking system.
   - Code snippets, algorithms, or technical details.

8. Results:
   - Present the outcomes and data collected.
   - Use graphs, charts, and tables if applicable.

9. Discussion:
   - Analyze the results.
   - Discuss challenges faced and solutions found.

1. Recommendations:
   - Suggest improvements or future work.

12. References:
   - List all sources you used, including books, articles, and online resources.

13. Appendices:
   - Include any supplementary materials such as raw data, additional code, or documentation.

14. Acknowledgments:
   - Mention anyone who contributed to the project.

15. Submission Information:
   - Include any specific requirements for project submission, such as file formats or delivery methods.

Ensure that your documentation is clear, well-organized, and follows any specific guidelines provided for submission. It's important to present your project in a structured and professional manner to convey your work effectively.

```
class ParkingSpace:
    def __init__(self, name, occupied=False):
        self.name = name
        self.occupied = occupied

class SmartParkingSystem:
    def __init__(self, total_spaces):
            self.parking_spaces = [ParkingSpace(f"Space {i + 1}") for i in range(total_spaces)]

    def check_availability(self):
        available_spaces = [space for space in self.parking_spaces if not space.occupied]
        return available_spaces

    def park_car(self):
        available_spaces = self.check_availability()
        if available_spaces:
            space = available_spaces[0]
            space.occupied = True
            return f"Parked in {space.name}"
        else:
            return "No available parking spaces"

    def leave_space(self, space_name):
        for space in self.parking_spaces:
            if space.name == space_name:
                space.occupied = False
                return f"Space {space.name} is now vacant"
        return "Space not found"

if __name__ == "__main__":
    smart_parking = SmartParkingSystem(5)

    print(smart_parking.park_car())
    print(smart_parking.park_car())
    print(smart_parking.leave_space("Space 2"))
    print(smart_parking.park_car()
```

Conclusion:

A Smart Parking project leverages technology to enhance the management and utilization of parking spaces, making the experience more convenient for users and more efficient for operators. By incorporating sensors, mobile apps, payment systems, data analytics, and sustainable features, it aims to address the challenges associated with urban parking. This technology-driven approach can lead to improved traffic flow, reduced congestion, and a more environmentally friendly parking ecosystem.