Position: System Analyst

Appliance: Smart Vacuum Cleaner in HACS

1. Introduction

In our HACS project, my position was Requirements Engineer. My main contribution was to gather requirements, analyze user needs, and document the functional and non-functional requirements for the Smart Vacuum Cleaner module.

The vacuum cleaner was an important appliance because it is used daily in smart homes. It saves time and effort and becomes more useful when controlled remotely. My responsibility was to understand what users want from the vacuum cleaner and then describe these needs in clear and simple requirements.

2. Role of Requirements Engineer (Theory)

In software engineering, the Requirements Engineer is responsible for requirements elicitation, analysis, documentation, and validation. According to theory, requirements are the foundation of the software project. If requirements are not correct, then the whole project may fail even if the design and coding are perfect.

The duties of a Requirements Engineer include:

Talking with stakeholders or users to understand their needs.

Writing functional requirements (what the system must do).

Writing non-functional requirements (how the system must behave).

Preparing use cases that show how the system will be used.

Validating requirements to make sure they are complete and realistic.

In our project, I acted like the bridge between the users and the developers. I converted user needs into clear requirements so that the designer, developer, and tester could work without confusion.

3. Purpose of My Contribution

The purpose of my work was:

To collect realistic requirements for the vacuum cleaner.

To make sure requirements were clear, simple, and complete.

To avoid adding unnecessary features that would make the system complex.

To prepare proper use cases and scenarios.

4. Scope of My Work

My scope included:

Gathering requirements from the perspective of home users.

Listing and writing all functional requirements for the vacuum cleaner.

Adding non-functional requirements for performance, reliability, and usability.

Preparing use case descriptions.

Helping the tester by explaining expected outcomes.

I did not work directly on coding or system design, but my contribution guided the rest of the team by providing a clear requirements document.

5. Requirement Elicitation Process (Theory)

According to theory, there are many ways to gather requirements:

1. Interviews – asking questions to users.

2. Observation – watching how users interact with current systems.

3. Brainstorming – discussing with the team.

4. Questionnaires – writing questions for users.

In our project, since we did not have real users, I imagined myself as a homeowner and thought about what features would be most useful. I also discussed with my teammates and finalized the requirements.

6. Functional Requirements of Vacuum Cleaner

After analyzing, I wrote the following functional requirements:

1. The system must allow the user to turn ON and OFF the vacuum cleaner remotely.

2. The system must support an Auto Cleaning Mode to clean the entire house.

3. The system must support a Spot Cleaning Mode for cleaning a selected small area.

4. The system must allow the user to schedule cleaning at a given time.

5. The system must allow the user to pause and resume cleaning.

6. The system must allow the user to cancel cleaning anytime.

7. The system must provide alerts for low battery and dust bin full.

These requirements describe what the system should do, not how it should be implemented.

7. Non-Functional Requirements

I also wrote non-functional requirements because they define how well the system must work. For the vacuum cleaner, I decided:

Performance: The system should respond to commands within 2 seconds.

Reliability: The vacuum cleaner must complete cleaning without failure.

Security: Only logged-in users should control the appliance.

Usability: Interface must be easy to understand, even for first-time users.

Maintainability: Future updates like adding new cleaning modes should be possible.

8. Use Case Descriptions

As part of requirements engineering, I wrote use cases to describe how the user interacts with the vacuum cleaner.

Use Case 1: Remote ON/OFF

Actor: User

Pre-condition: User is logged in

Steps: User presses ON → Vacuum starts cleaning → User presses OFF → Vacuum stops

Post-condition: Vacuum cleaner turns ON/OFF successfully

Use Case 2: Schedule Cleaning

Actor: User

Pre-condition: Valid login, system is connected

Steps: User selects “Schedule” → Enters time (e.g., 7 pm) → System saves schedule → Vacuum starts automatically at set time

Post-condition: Cleaning starts at scheduled time

Use Case 3: Cancel Cleaning

Actor: User

Steps: User presses Cancel → Vacuum stops cleaning immediately

Post-condition: Cleaning process ends and system goes idle

9. Example Scenarios Considered

While writing requirements, I imagined real-life scenarios:

Scenario 1: A working professional wants to start the vacuum from office.

Scenario 2: A family schedules the vacuum at 7 pm every evening.

Scenario 3: The vacuum is running, but the user cancels because guests arrive.

Scenario 4: Battery becomes low during cleaning → system shows alert.

Scenario 5: Dust bin is full → notification appears on mobile app.

By considering these scenarios, I ensured that requirements covered real user needs.

10. Challenges Faced

The main challenge I faced was deciding the right scope of requirements. If too many features were added, the project could become too large. If too few were added, it would not look useful. I solved this by focusing on essential and realistic features only.

Another challenge was writing requirements in clear and simple words so that the developer and tester could understand them easily.

11. Outcome of My Contribution

Because of my work:

We had a clear requirement list.

Functional and non-functional requirements were ready before design started.

Use cases were available for the designer, developer, and tester.

The project stayed within scope and did not add unnecessary features.

12. Learning and Reflection

From theory, I learned that requirements engineering is the most important phase of software development. A small mistake in requirements can cause failure later.

From practice, I learned how to think like a user and write requirements in simple form. I also learned the importance of use cases and scenarios in explaining requirements.

13. Conclusion

As the Requirements Engineer, my contribution was to gather and document the requirements of the Smart Vacuum Cleaner in HACS. I prepared functional and non-functional requirements, use cases, and scenarios. My work gave the team a strong foundation to design, develop, and test the system.

This project taught me that requirement engineering is not just a theoretical concept but a practical process that guides the whole project.