Project Progress Report for Final Sprint

Project Name: Smart Parking Management System

Course: Integrative Software Engineering, 251014301

Due date: 02/02/2025

Team Details

Student Name	Student ID	Role in Team	Avatar in Kanban Board
Oren Golovchik	316594829	UI/UX Engineer Devop TEAM	
Nitzan Levi	316443852	DBA TEAM	
Shir Falach	206587495	Team Leader Product owner TEAM	
Lital Khotyakov	316238260	QA Engineer System architect TEAM	
Alexander Kaminsky	318244530	Technical Writer Scrum master TEAM	

Contents

System Requirements Document

Introduction

Purpose of System

The Smart Parking Management System is a software solution designed to optimize parking space management in organizational or commercial areas. It provides administrators with monitoring capabilities and users with an intuitive interface to find available parking. The system aims to:

- Improve user convenience.
- Enhance management efficiency.
- Reduce environmental impact.

Scope

The system will focus on the following features:

- Parking availability updates.
- User-friendly reservation system.

Out of Scope:

- Physical sensor integration.
- SMS notification capabilities.
- Advanced predictive analytics modules.

Objectives

- Accessibility: Provide information about parking availability.
- Efficiency: Enable effective management and reservation of parking spaces.
- Sustainability: Support environmentally friendly practices by reducing idle vehicle movement.

Actors and goals

Actors names, primary/support, Description, Goals:

Actors names	**Type**	**Description**	**Goals**
Driver (End User)	Primary	A user who needs to find and reserve an available parking spot in the parking lot.	- Locate available parking spots quickly. - Reserve a parking spot.
Parking Lot Manager (Operator)	Primary	A facility manager, responsible for monitoring parking usage, ensuring optimal management of the parking lot.	- Monitor parking spot usage. - View parking status. - Resolve parking issues effectively.
System Administrator (Admin)	Primary	System admin that can review users, commands and delete parking spots.	-View register users - Delete all users - Delete all parking spots

Functional Requirements

Functional requirements are defined using UML use cases to illustrate system functionalities and interactions. Below are key use cases:

Use Case: Parking Spot Reservation

Goals: Allow users to reserve parking spots in advance.

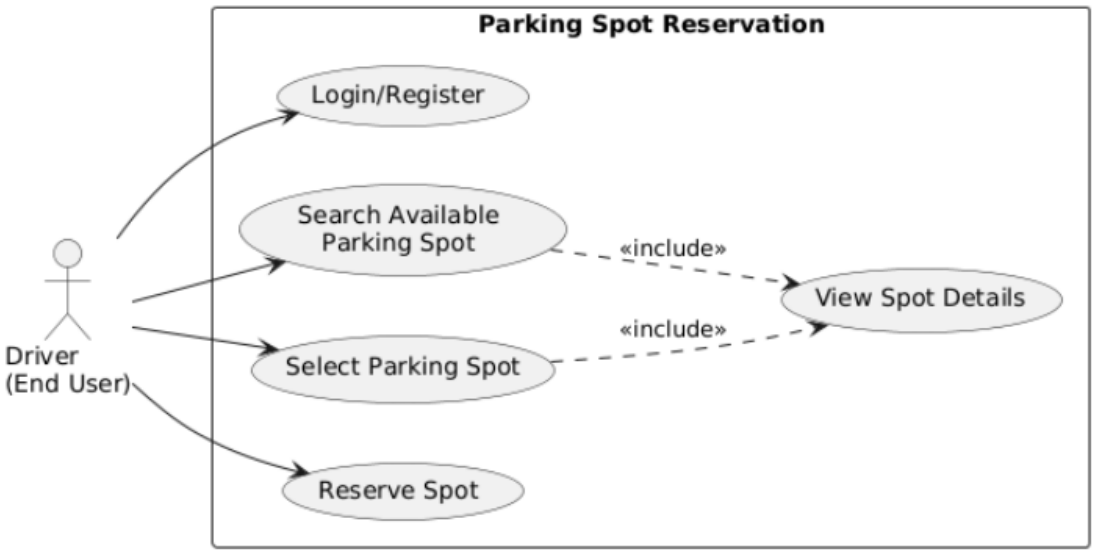
Actors: Driver (End User)

Main Flow:

1. User searches for an available parking spot.
2. If a parking spot is available, the user selects and reserves the spot.
3. The reservation is confirmed, and the parking spot status is updated to "Occupied".

Alternative Flow:

1. If no spots are available, notify the user saying: "No parking spots available".
    - Notify: show an error message saying: "No parking spots available".
2. Go back to the home page.



Use Case: Parking Space Management

Goals: Allow operators to manage parking spaces and their status.

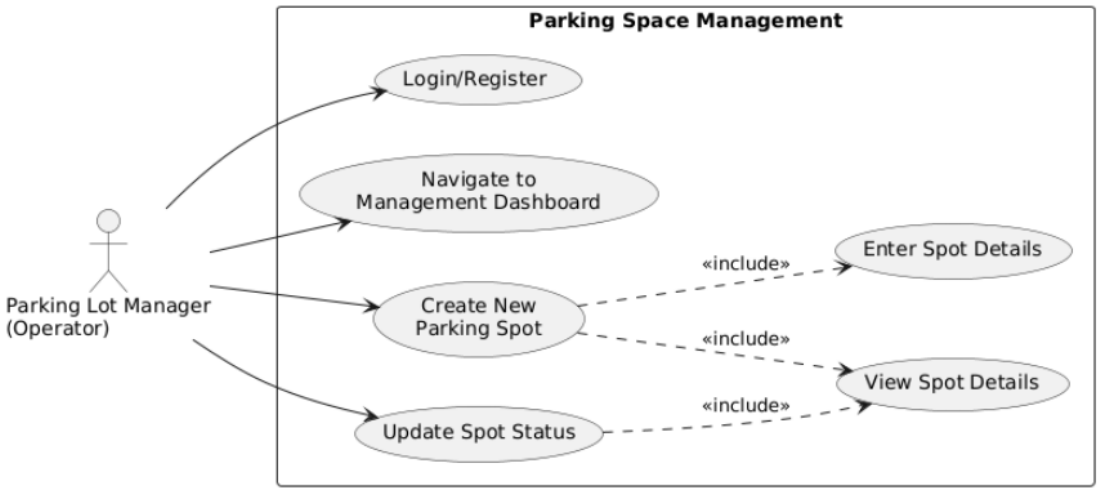
Actors: Parking Lot Manager (Operator)

Main Flow:

1. Operator navigates to object management dashboard.
2. Operator selects "Create New Object" option.
3. Operator enters required parking spot details:
   Location coordinates, Status (Available/Occupied), Type, Alias and Active status.
4. Operator submits the creation form
5. The creation is confirmed, and the parking spot details are shown.
6. Operator can update parking spot status as needed.

Alternative Flow:

1. If creation fails, notify the operator with error message
2. Return to management dashboard



Use Case: User Search and Navigation

Goals: Allow end users to search for available parking spaces.

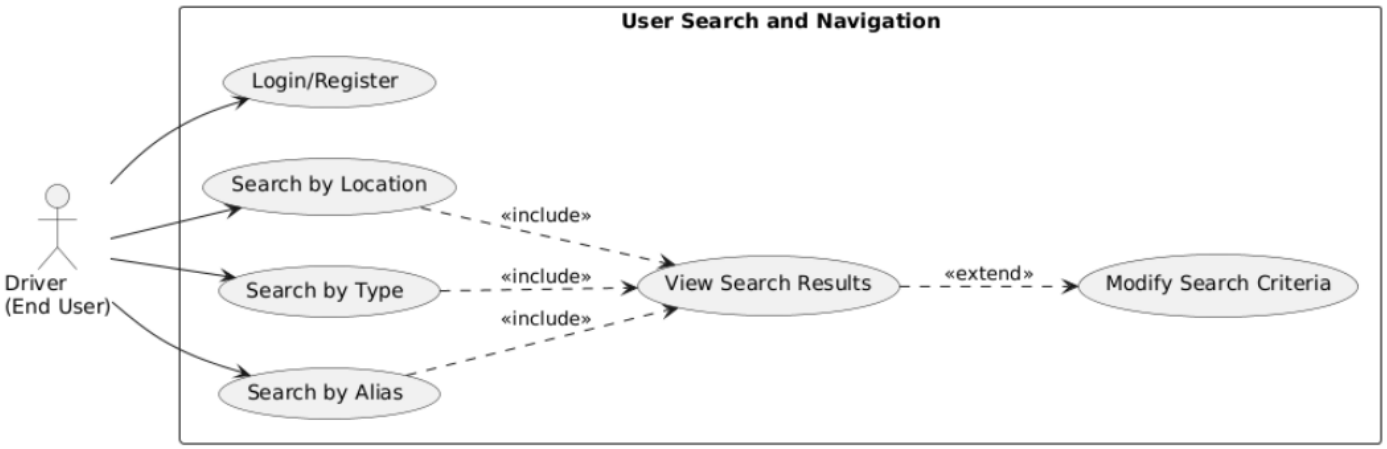
Actors: Driver (End User)

Main Flow:

1. User searches for an available parking spot using criteria:
    1. Location-based: Using latitude/longitude and distance
    2. Type-based: Searching for specific parking spot types
    3. Alias-based: Using exact alias or pattern matching
2. User can view detailed information about each available parking spot.

Alternative Flow:

1. If no spots match criteria, notify the user saying: "No parking spots available"
2. User can modify search criteria.



Non-Functional Requirements

Requirement number	Requirement Type	Requirement Description
1	S - Supportability	Log Management tool, with easy access for error detection. The log management should be retrievable and readable by developers for debugging and maintenance purposes.

Appendices

Technologies List

- Programming Language: Java.
- Framework: Spring Boot.

Technologies Used in Server Application:

- Database: PostgreSQL.
- Version Control: Bitbucket.
- REST API for parking management operations, Key modules:
    - Spring Data JPA for managing database operations.
    - Spring MVC for handling user requests and responses.

Technologies Used in Client Application:

- Thymeleaf - Template engine for server-side rendering of HTML.
- Spring WebFlux (WebClient) - For making HTTP requests to your server.
- Spring MVC - For handling web requests and responses.
- HTML/CSS - For frontend user interface.
- Maven - Build tool and dependency management.
- Architecture: MVC (Model-View-Controller).
- Spring DevTools - For development convenience (auto-restart).
- Client-Server Communication: RESTful APIs.
- Layered architecture (controllers, services, models).
- Version Control: Git.

General Project Summary

The team developed a Smart Parking Management System across 5 sprints, handling REST API development, system layering, and data integrity features. The system provides administrators with monitoring capabilities and users with an intuitive interface to find available parking.

What worked well for the team and should be retained in future industry projects.

- Clear role definition and task division among team members
- Effective sub-group organization for parallel work streams
- Successful implementation of planned features across all sprints
- Regular progress updates and immediate problem-solving
- Strong technical execution, completing core functionality on schedule

How the team can improve its work on future projects.

- Earlier sprint planning and task assignment
- More frequent full-team meetings (recommended: sprint start and midpoint)
- Better coordination between sub-groups
- More structured timeline management for sub-tasks
- Earlier testing initiation

What the team enjoyed most while working on the project.

- Collaborative problem-solving
- Working in specialized sub-groups
- Technical challenge implementation (location search, Haversine formula)
- Successful feature delivery across sprints

What the team would do differently if starting the project now, based on knowledge and experience gained during the semester.

- Start with more structured project management in Trello
- Establish regular meeting schedules from the beginning
- Define clearer deadlines for sub-tasks
- Begin testing earlier in each sprint
- Plan for client development earlier in the project timeline (after the first sprint for sure)

Was teamwork conducted remotely in addition to in-person meetings?

The team operated entirely remotely through Google Meet, which proved highly effective:

- Screen sharing enabled simultaneous code review and problem-solving
- Team members could work on code concurrently while discussing
- Military service accommodation (Oren could participate regardless of location)
- Flexible scheduling increased participation and productivity
- **Tools used for remote collaboration:**
    - Google Meet: Primary collaboration platform for all meetings
    - Trello: Task management and sprint tracking
    - Bitbucket/Git: Code version control and collaboration
    - Screen sharing: Real-time code review and pair programming
- **Impact of remote work on project performance:**
    - Enhanced flexibility in meeting times and participation
    - Efficient sub-group collaboration through breakout discussions
    - Real-time problem-solving through screen sharing
    - Seamless code sharing and review process
    - Improved work-life balance for team members
    - Accommodated team members with military commitments

Sprint 5 - General summary

In Sprint 5, we held a team meeting to distribute tasks within our sub-groups, continuing our established workflow. The sprint's focus was on completing the client-side development, adding automation testing to the server, and writing test cases for the client. Additionally, we prepared the final project report by gathering all necessary information and created the final presentation for class.

What went well for the team during the sprint

- **Task Division:** Tasks were clearly defined and effectively divided among sub-groups, ensuring each member had a focused responsibility.
- **Timelines:** We established clear deadlines for our work, which helped maintain a reasonable pace and ensure progress.
- **Collaboration:** Despite challenges, the sub-groups collaborated well and worked efficiently within their respective areas.

What could be improved in teamwork

- **Full-Team Meetings:** Coordinating a mutual time for the entire team to meet proved challenging. More regular full-team discussions at the start and midpoint of each sprint could improve alignment.
- **Testing Completion:** Completing all the planned testing on time was difficult, highlighting a need for better time allocation and earlier prioritization of testing tasks.

What problems did the team encounter through this phase of work

- **Client Development Time Constraints:** Developing the client within a limited time frame was challenging, especially as it required significant effort to ensure proper functionality and alignment with the server.
- **Testing Challenges:** Writing and completing test cases for the client was difficult since the client development was still in progress. This delayed the testing process and added pressure on the timeline.
- **Presentation Challenges:** Preparing the final presentation was challenging because the client was not fully completed, making it harder to showcase a finished product to the class.

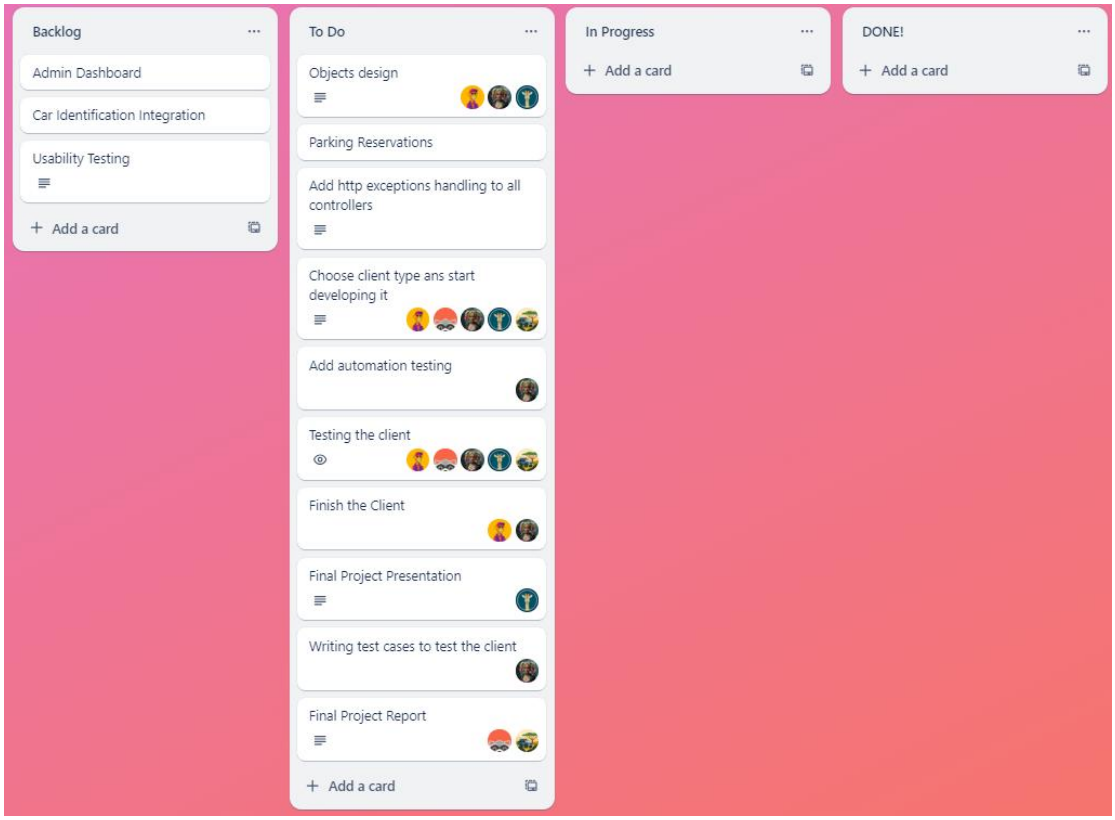
Why did we not complete all the planned work

- We did.

All Kanban Boards

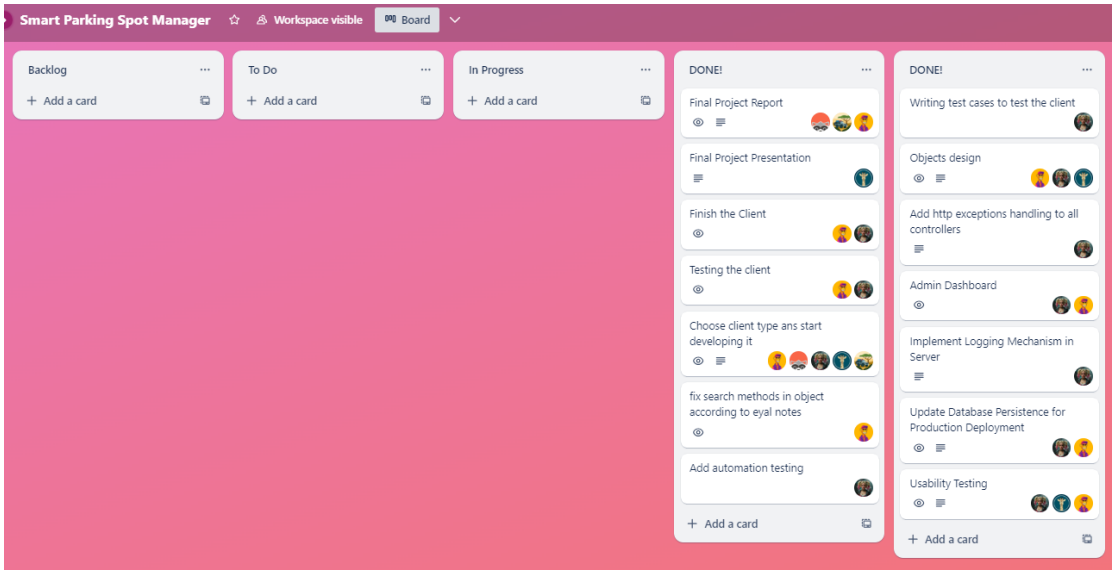
Kanban Board - Sprint 5 Initiation

Screenshot 21/01/2025



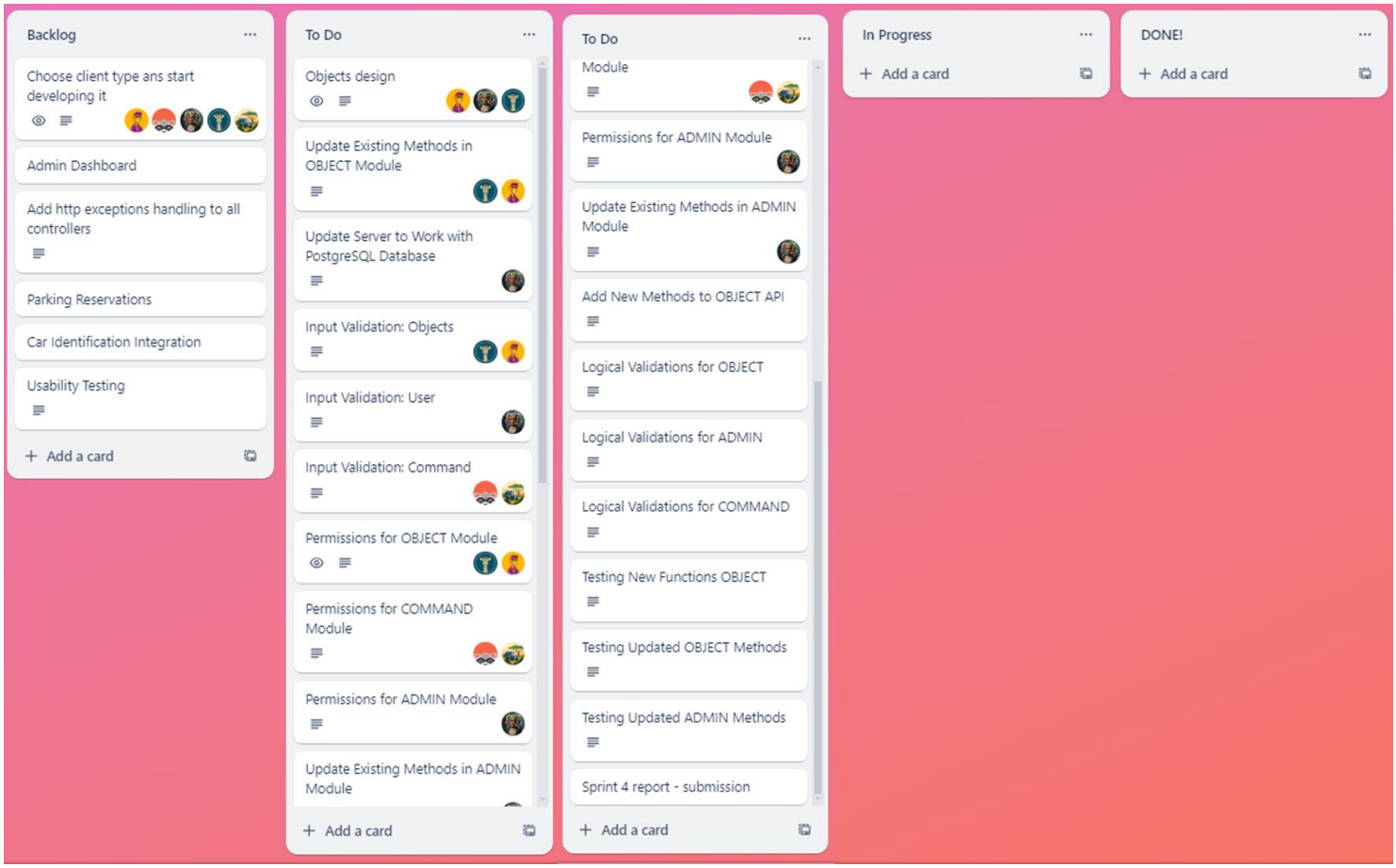
Kanban Board - Sprint 5 Final Progress

Screenshot 31/02/2025



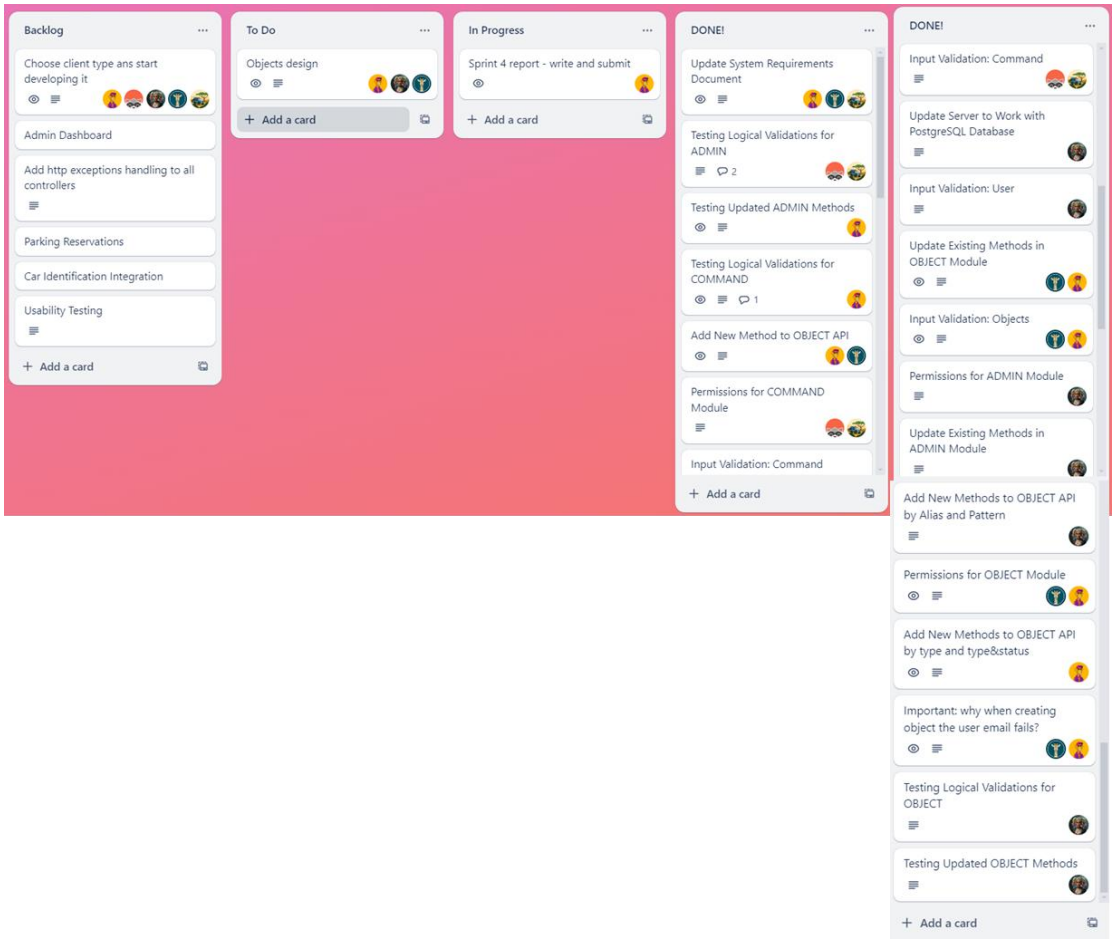
Kanban Board - Sprint 4 Initiation

Screenshot 07/01/2025



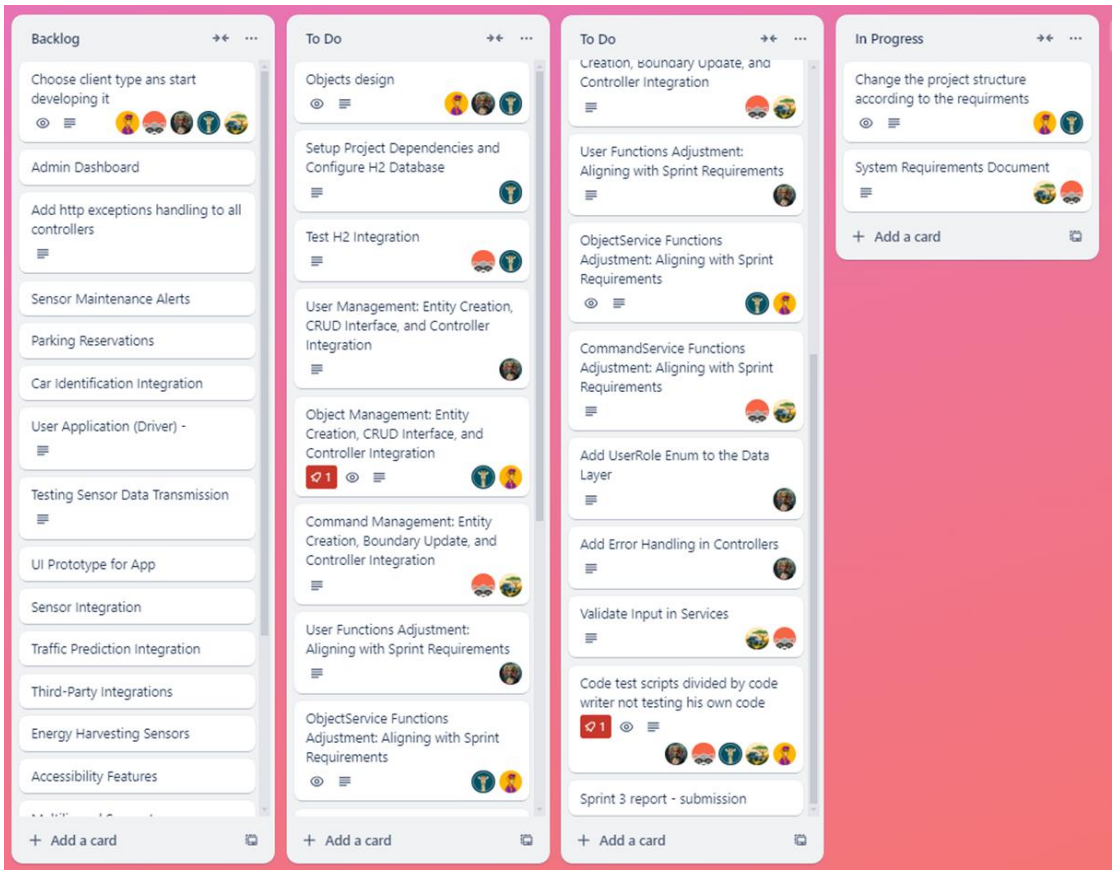
Kanban Board - Sprint 4 Final Progress

Screenshot 19/01/2025

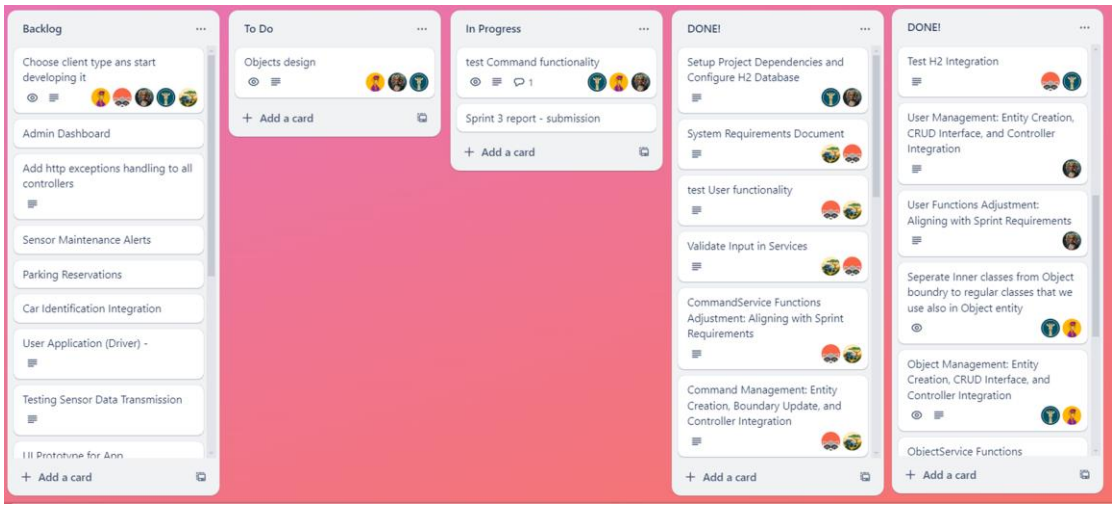


Kanban Board - Sprint 3 Initiation

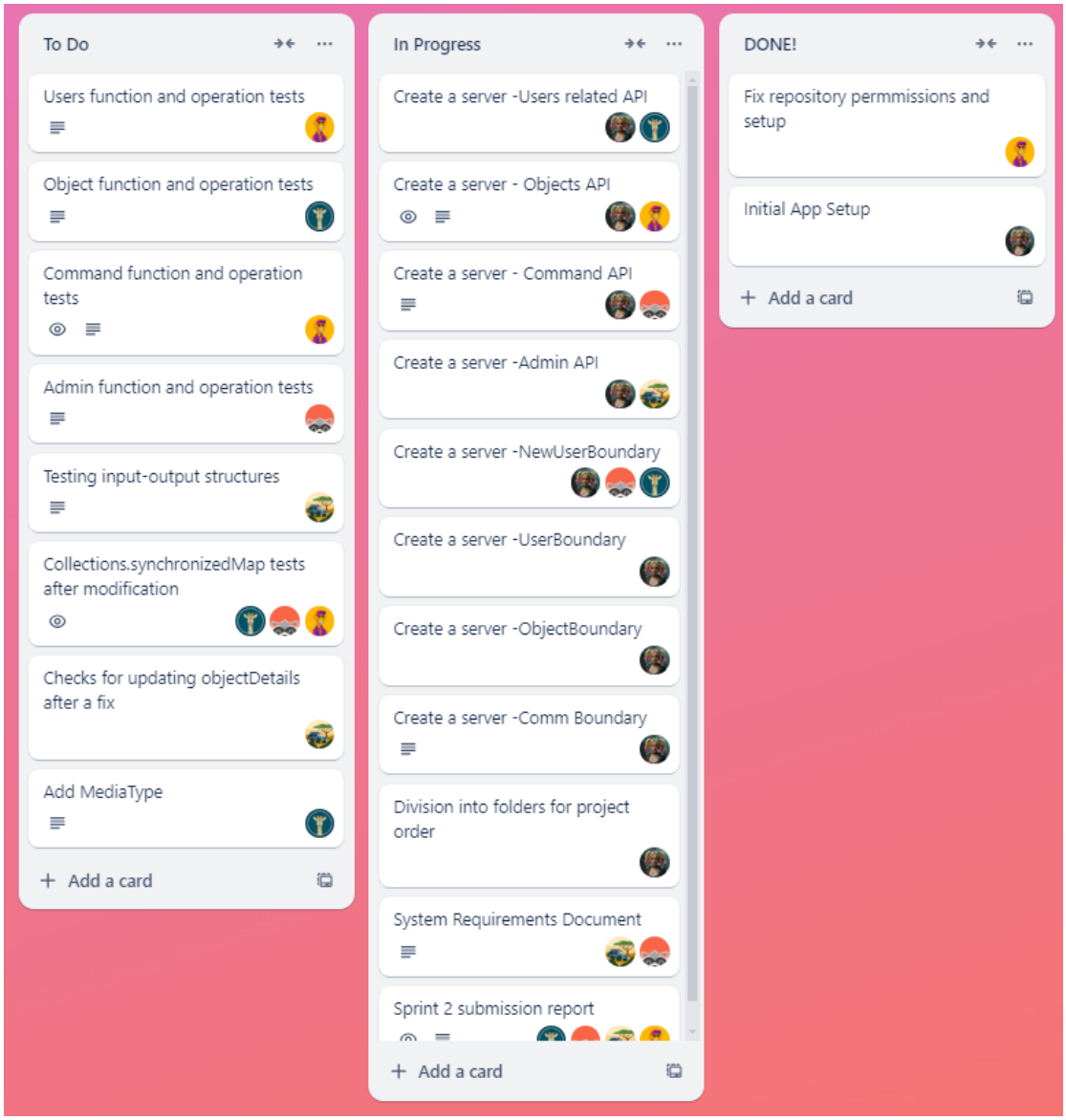
Screenshot 18/12/2024



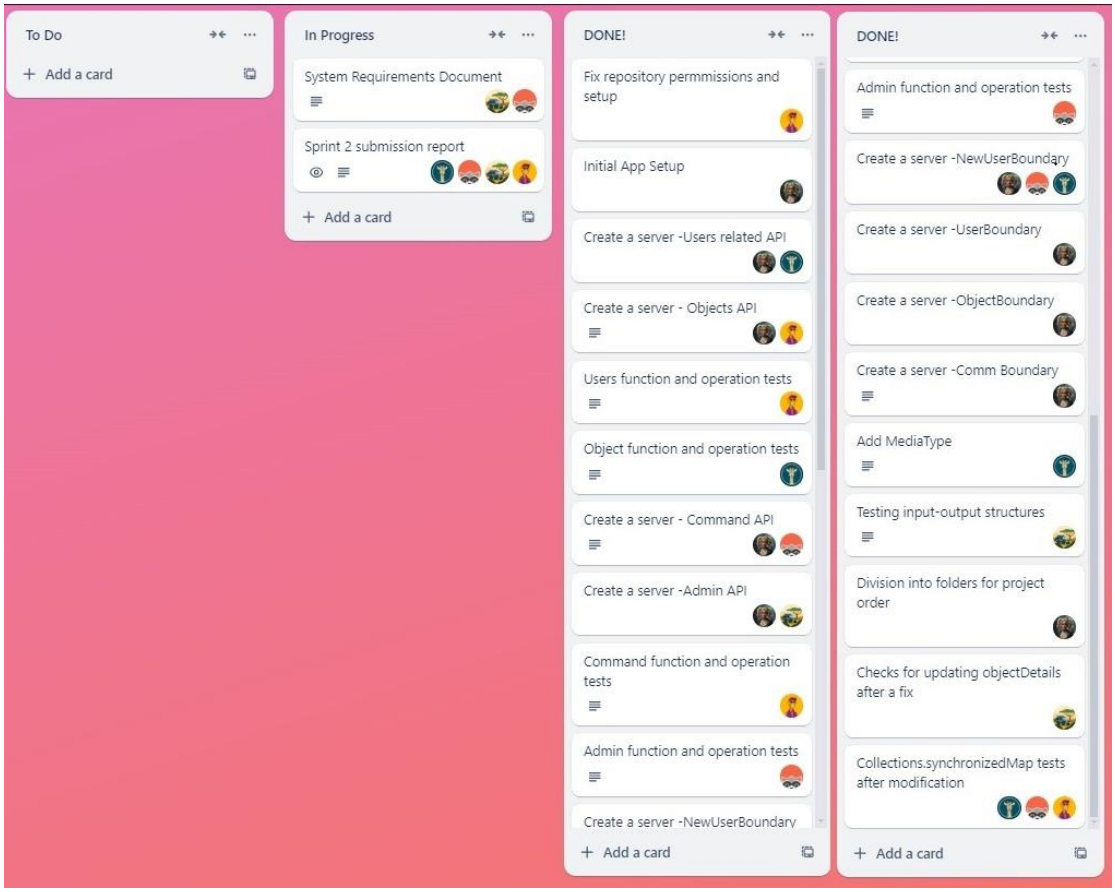
Kanban Board - Sprint 3 Final Progress



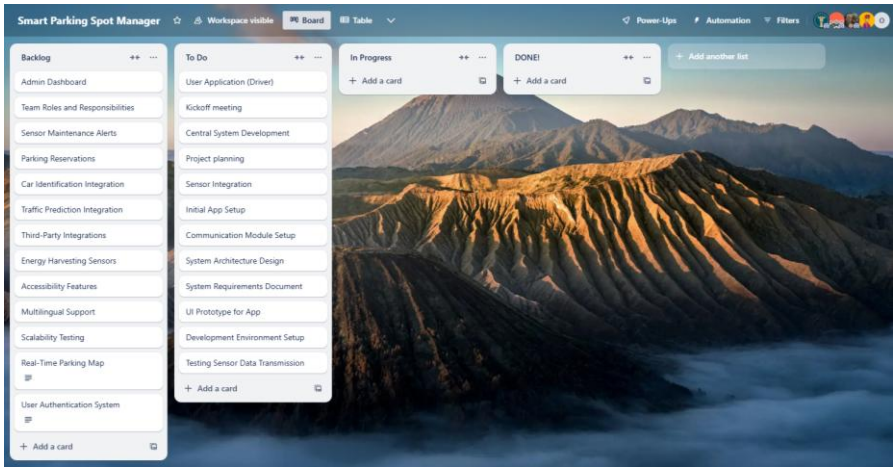
Kanban Board - Sprint 2 Initiation



Kanban Board - Sprint 2 Final Progress



Kanban Board - Sprint 1 Initiation



Kanban Board - Sprint 1 Final Progress