SOFTWARE ENGINEERING PROJECT

KU Parking

BY

Nanthawat Duang-ead
Setthapon Thadisakun

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KASETSART UNIVERSITY

Academic Year 2026

KU Parking

BY

Nanthawat Duang-ead
Setthapon Thadisakun

This Project Submitted in Partial Fulfillment of the Requirement for Bachelor Degree of Engineering (Software Engineering)
Department of Computer Engineering, Faculty of Engineering KASERTSART UNIVERSITY
Academic Year 2026

Approved By:

Advisor . . . . . **Date** . . . . . .
(Associate Professor Chaiporn Jaikaeo)

Head of Department . . . . . **Date** . . . . . .
(Associate Professor Punpiti Piamsa-nga)

Abstract

Finding an optimal parking spot that is vacant within walking distance of the destination can be time-consuming in high-traffic areas with multiple parking areas such as university campuses or government complexes. These common inconveniences affect drivers entering the area, especially visitors or non-residents, and often lead to wasted time, unnecessary fuel consumption, and aimless searching between parking areas. The motivation for this project is to establish a system that can indicate to the user the availability of multiple parking areas while reducing the potential cost of deploying the system, which the traditional parking management might exceed the budget. KU Parking addresses this by utilizing existing infrastructure—surveillance cameras already installed in parking areas for security purposes—integrated with an image processing model to detect vacant parking spaces. Users can check for availability in all parking areas near the destination before arriving. The project is designed to deploy on the cloud to enhance installability. The effectiveness of the project can be evaluated based on the model's accuracy and user feedback. The goal is to provide accurate data that minimizes inconvenience and reduces the time required to find a parking spot.

Acknowledgement

Put your acknowledgement paragraph here.

Nanthawat Duang-ead
Setthapon Thadisakun

Table of Contents

List of Tables

Page

List of Figures

Page

Chapter 1

Introduction

1.1 Background

In large campuses or complexes with multiple departments or offices within the same area, parking often becomes a common inconvenience for commuters. Many individuals waste time driving to a parking lot close to their destination, only to find their preferred lot full, forcing them to drive elsewhere. This not only leads to frustration but also results in unnecessary fuel consumption and traffic congestion.

Studies indicate that cruising for parking can contribute to between 8 and 74 percent of overall traffic congestion and may take anywhere from 3.5 to 14 minutes to secure a spot [1]. This issue is even more frustrating for non-resident visitors unfamiliar with the area, such as those navigating Kasetsart University, where finding available parking can be particularly challenging without prior knowledge of the area layout. The complexity is further compounded by the road system, which consists of multiple one-way streets, making it difficult for drivers to efficiently navigate between parking areas and forcing them into longer detours when the preferred parking is not available.

Many parking facilities address this challenge by installing parking indicators that display the number of available spots or highlight specific vacant spaces. These systems help users determine parking availability before entering the lot, reducing unnecessary driving time and congestion. However, they typically rely on sensors installed in individual parking spots to detect occupancy. While effective, such sensor-based systems require significant installation and maintenance costs. As a result, many authorities choose not to implement these systems due to budget constraints, leaving drivers to search for parking without any indicator.

1.2 Problem Statement

A problem statement refers to a clear central issue, challenge, or question that the project aims to address or explore. It is a declaration that highlights the specific problem the author intends to examine, discuss, or solve throughout the course of the project.

1.3 Solution Overview

A software solution overview provides a high-level and concise description of a software product or system. It serves as an introduction to the software, offering a glimpse into its key features, functionalities, and the problems it aims to address. This overview is often presented in documentation, marketing materials, or other communication channels to give stakeholders, potential users, or decision-makers a quick understanding of what the software does and why it is valuable.

1.3.1 Features

1. Feature Name: Short Description of Feature
2. Feature Name: Short Description of Feature

1.4 Target User

The target user in a software project refers to the specific group or demographic of individuals for whom the software is designed and developed. Identifying the target user is a crucial step in the software development process as it helps the development team tailor the software to meet the needs, preferences, and requirements of that particular user group. Understanding the characteristics, behaviors, and expectations of the target users is essential for creating a user-friendly and effective software solution.

Here are some key aspects related to defining the target user in a software project:

Demographics: This includes factors such as age, gender, occupation, education level, and other demographic characteristics. Different age groups or professional backgrounds may have distinct preferences and requirements when it comes to software usability.

Skill Level: Consideration of the users' technical proficiency and familiarity with similar software or technology. The level of technical expertise can influence the complexity of the user interface, the need for tutorials or documentation, and other user support features.

Industry or Domain: For software solutions designed for specific industries or domains, understanding the unique challenges, workflows, and terminology within that industry is crucial. Tailoring the software to meet industry-specific needs is often necessary.

1.5 Benefit

Describe potential benefits of your solution.

1.6 Terminology

Terminology refers to the specific language, jargon, or specialized vocabulary used to describe concepts, ideas, or subjects within a particular field or domain. The use of terminology is often essential for clarity and precision, especially in books that cover technical, scientific, academic, or specialized topics.

Chapter 2

Literature Review and Related Work

In this chapter, describe other solutions/research that address the same topic as your project. If you are working on a software project, create a list of alternative solutions and analyze them in the competitor analysis section. If you are working on a research project, describe your related work research in the literature review section.

2.1 Competitor Analysis

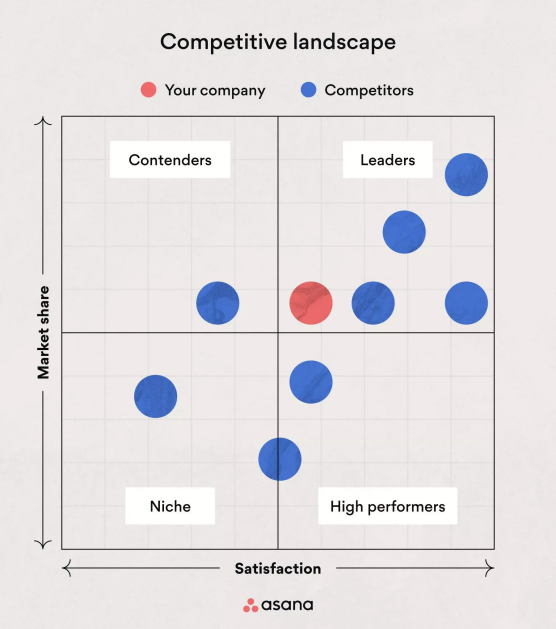

Figure 2.1: Competitive Landscape by Asana

Refer to an article "How to create a competitive analysis (with examples)" by Asana. You can use the Competitor Landscape (left image) or Competitor Analysis Framework (right image) for your project.

2.2 Literature Review

Add a literature review section if it fits with your project.

Chapter 3

Requirement Analysis

3.1 Stakeholder Analysis

<TIP: List your stakeholders for your project here/>

Stakeholders are individuals, groups, or entities that have an interest, concern, or stake in a particular project, decision, organization, or system. These are individuals or groups who can affect or be affected by the outcomes of your project.

3.2 User Stories

<TIP: Write user stories for each of your stakeholders here./>

User stories are a technique used in agile software development to capture and describe functional requirements from an end user's perspective. They are a way of expressing software features or functionality in a simple, non-technical language that can be easily understood by both developers and stakeholders.

3.3 Use Case Diagram

<TIP: Write a use case diagram for your project here. Refer to an article "What is a use case diagram?" by Lucidchart for help./>

3.4 Use Case Model

A use case is a detailed description of how a system interacts with an external entity (such as a user or another system) to accomplish a

specific goal. Use cases provide a high-level view of the functionality of a system and help in capturing and documenting its requirements from the perspective of end users.

<TIP: Write use cases for your project here. Make sure to use the appropriate type of use case for each scenario (brief, casual, and fully-dressed use case)./>

3.5 User Interface Design

<TIP: Put the initial design of your application here. You can showcase a detailed design of a specific page or a sitemap of your application. See an example below./>

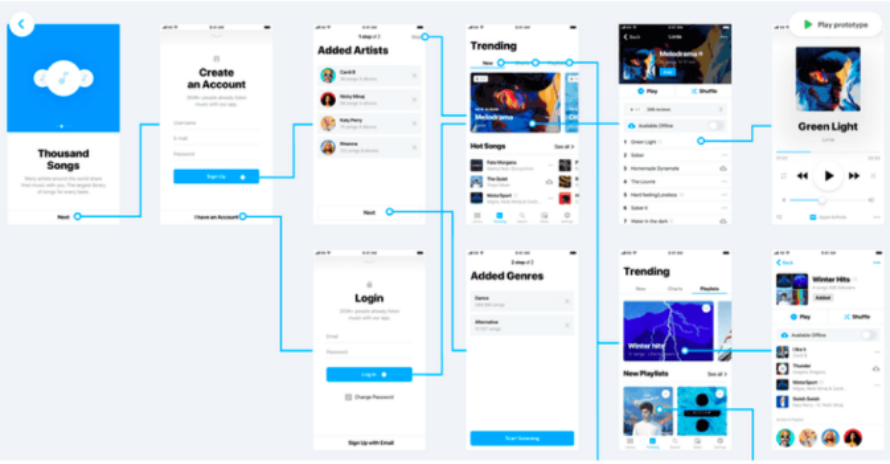

Figure 3.1: User Interface Design

Chapter 4

Software Architecture Design

<TIP: Describe how you design your application using Unified Modelling Language (UML). There should be at least two diagrams that describe the software architecture. You may add additional or remove unnecessary diagrams. However, there needs to be a coherency between them at the end./>

4.1 Domain Model

<TIP: Describe the business concept of your project. Showcase a domain model that captures the said concept./>

4.2 Design Class Diagram

<TIP: Showcase a design class diagram for your project and explain how it works here. You can group classes into packages or layers to communicate your design better./>

4.3 Sequence Diagram

<TIP: Sequence diagrams describe how the software runs at runtime. You do not have to create a sequence diagram for every scenario. However, there should be one for all the main ones./>

<ChatGPT: Creating a sequence diagram for every use case is not strictly necessary, but it can be a valuable tool in certain situations. Sequence diagrams are particularly useful for illustrating the interactions

between different components or objects in a system over time, showcasing the flow of messages or actions between them./>

4.4 Algorithm

<TIP: Optional, If you are working on a research project that proposes a new algorithm, you can describe your algorithm here. It can be in the form of pseudocode or any diagram that you deem appropriate./>

Chapter 5

Software Development

5.1 Software Development Methodology

<TIP: Describe your software development methodology in this section. />

5.2 Technology Stack

<TIP: Describe your technology stack here. See the following example from ThaiProgrammer.org />

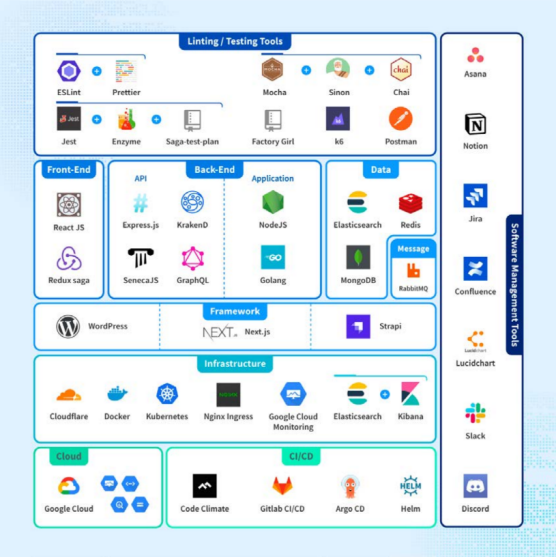

Figure 5.1: Example technology stack

5.3 Coding Standards

<TIP: Describe your coding standard for this project here. />

5.4 Progress Tracking Report

<TIP: Show that you have been working on this project overtime. It can be in the form of a burndown chart or a contribution graph from GitHub./>

Chapter 6

Deliverable

6.1 Software Solution

<TIP: Share a link to your Github repository. Showcase screenshots of the application and briefly describe each page here. />

6.2 Test Report

<TIP: Describe how you test your project. Place a test report here. If you use continuousintegration and deployment (CI/CD) tools, describe your CI/CD method here. />

Chapter 7

Conclusion and Discussion

<TIP: Discuss your work here. For example, you can discuss software patterns that you use in this project, software libraries, difficulties encountered during development, or any other topic. />

Reference

Bibliography

[1] D. Shoup, “Cruising for parking,” *Transport Policy*, vol. 13, no. 6, 2006.

Appendix A

Appendix A: Example

<TIP: Put additional or supplementary information/data/figures in appendices. />

Appendix B

Appendix B: About LaTeX

LaTeX (stylized as LaTeX) is a software system for typesetting documents. LaTeX markup describes the content and layout of the document, as opposed to the formatted text found in WYSIWYG word processors like Google Docs, LibreOffice Writer, and Microsoft Word. The writer uses markup tagging conventions to define the general structure of a document, to stylize text throughout a document (such as bold and italics), and to add citations and cross-references.

LaTeX is widely used in academia for the communication and publication of scientific documents and technical note-taking in many fields, owing partially to its support for complex mathematical notation. It also has a prominent role in the preparation and publication of books and articles that contain complex multilingual materials, such as Arabic and Greek.

Overleaf has also provided a 30-minute guide on how you can get started on using LaTeX. [**?**]