DynamicPricingSystem

System Design Description

Abstract
This System Design Description (SysDD) details the implementation of the DynamicPricingSystem which computes parking prices using occupancy, time, weather and event data. The system follows the Arrowhead service oriented model and exposes pricing and logging services inside the local cloud.

Contents

1 Overview

This document describes the DynamicPricingSystem. The system ingests AggregatedOccupancy, Time, Weather and EventSchedule services, applies AI based decision logic and outputs the DynamicPricing and PricingLog services.

In Section 2, we describe implementation details of the system

2 Implementation

This implementation is based on the SysD document DynamicPricing_sysd.

2.1 Implementation language and tools

- Language: Python 3.14
- The IDE will be Visual Studio Code.
- Required libraries: PyTorch for model training and inference. Numpy for numerics
- Stateful or stateless: Request handling is stateless while model parameters andd configurations are stateful in memory.

2.2 Functional properties implementation

- Resources: CPU for inference, GPU during training, network access to internal services.
- Data handled and stored:
  - Database: SQL
- Result provided as:
  - DynamicPricing response with current price and validity window.
  - PricingLog records with timestamp, inputs and outcome for the History Database

The reinforcement learning approach described by Apte et al. [1] provides the basis for the DynamicPricingSystem. In their work, a Q-Learning agent interacts with a simulated retail environment to learn optimal prices that maximize revenue under changing market conditions. This concept is adapted here for dynamic parking fee control.

The environment in this system uses occupancy, time, weather and event data to estimate demand. The state space represents the current occupancy and context conditions, while the action space consists of price adjustments. The reward function combines revenue and occupancy balance to prevent both underpricing and overpricing.

Training is performed offline using historical data from the History Database, allowing the agent to learn optimal pricing policies safely. The trained model weights are stored and used during real-time inference, ensuring fast and stable pricing decisions within the local cloud.

2.3 Non functional properties implementation

Typical non-functional properties implemented are e.g.

2.3.1 Security

As provided by the Arrowhead Framework.

2.3.2 Internal monitoring

Key metrics include request latency, training duration, inference duration, model version and error rates.

2.3.3 Configuration

- Accepted configuration: target occupancy, price bounds, update interval, feature toggles, model path.
- Configuration formats: YAML file or database table

For each of the non-functional operations the following details should be documented

3 Services

The implementation services is based on the following SD and IDD documents:

- SD: dynamicPrice_SD
- SD: priceLogging_SD
- IDD: dynamicPriceHTTP_IDD
- IDD: priceLoggingHTTP_IDD

Table 1: References to documentation for services produced and consumed.

Services produced	SysD ref	SD ref	IDD ref
dynamicPrice	DynamicPricing_sysd	dynamicPrice_sd	dynamicPriceHTTP_idd
priceLogging	DynamicPricing_sysd	priceLogging_sd	priceLoggingHTTP_idd
Services consumed	**SysD ref**	**SD ref**	**IDD ref**
occupancyData	OccupancySystem_sysd	occupancyData_sd	occupancyDataHTTP_idd
currentTime	TimeSystem_sysd	currentTime_sd	currentTimeHTTP_idd
localWeatherData	WeatherSystem_sysd	localWeatherData_sd	localWeatherDataHTTP_idd
nearbyEventsData	EventSystem_sysd	nearbyEventsData_sd	nearbyEventsDataHTTP_idd
trainingData	HistoricalDatabase_sysd	trainingData_sd	trainingDataHTTP_idd

4 References

[1] M. Apte, K. Kale, P. Datar, and P. Deshmukh, “Dynamic retail pricing via q-learning – a reinforcement learning framework for enhanced revenue management,” 2024. [Online]. Available: https://arxiv.org/abs/2411.18261

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2025-10-14	1.0.0	Initial release	Mattias Öhman