# Task Tracker

# Task Tracker for the Team

| Sr No | Activity | Status | Assignee |
| --- | --- | --- | --- |
| 1 | Project Overview & Documentation | Completed | Bhavin |
| 2 | Models to be used in the Project | In Progress | Bhavin |
| 3 | Dataset Identification or Generate Synthetic Data for Simulating Smart Parking Scenario | Not Started | Bhavin |
| 4 | Dataset Identification for Parking Spot Occupancy/availability Check from Images | Not Started | |
| 5 | Dataset Identification for Car License Plate Reading from Images | Not Started | |
| 6 | Model for Parking Spot Occupancy | Not Started | |
| 7 | Model for Car License Plate | Not Started | |
| 8 | Data Cleansing for IoT Data | Not Started | |
| 9 | Near-term occupancy levels and projected revenue are forecasted using time-series models. | Not Started | |
| 10 | Operational issues are identified using **anomaly-detection models** on parking data. | Not Started | |

# Project Overview

# Optimizing Smart Parking: Data-Driven Operational Insights

## Problem Statement

Parking facilities suffer from a critical lack of real-time data, leaving essential questions unanswered: What is the current vehicle count, which zones are utilized, how long do patrons stay, and what is the projected revenue?

Current, unreliable methods—like manual checks, flawed entry/exit logs, and unauditable billing—cause driver frustration and congestion due to unknown occupancy. They also create operational blind spots for management, delaying detection of issues like overcrowding or unauthorized exits, and result in inaccurate, delayed reporting. Furthermore, the lack of data prevents easy performance benchmarking across different periods (weekdays vs. weekends, event days, peak hours).

## Proposed Solution

This simulated Smart Parking system uses IoT-style entry/exit events (with timestamps and license plates) to track vehicles. It processes these events into parking sessions to monitor real-time occupancy, duration, and movement. A pricing engine calculates fees based on configurable rules (hourly, free time, peak, daily max). The processed data yields analytics on space utilization, peak hours, average stay, revenue, and anomalies. This fully simulated, scalable model mirrors a real-world smart parking setup.

## Dataset Description

This dataset contains simulated IoT data from a smart parking facility, including time-stamped entry and exit events from parking gates. Records detail facility ID, gate ID, event type, license plate, and event time, used to derive parking sessions for calculating duration, occupancy, and charges. Designed to mimic real-world conditions like peak traffic and data inconsistencies, this synthetic data is structured like actual IoT parking data, suitable for testing analytics, dashboards, and billing logic.

## List of Tables

### Parking Events Table (`PARKING_EVENTS`)

This table stores raw IoT-style data generated at the parking facility's entry and exit gates. Each record represents a vehicle entering or leaving the facility, along with the event timestamp, gate information, and license plate details. Since the data is captured directly from sensors, it may include errors or missing values, reflecting real-world conditions.

### Parking Sessions Table (`PARKING_SESSIONS`)

The parking sessions table is derived from the raw events and represents a complete parking activity for each vehicle. It combines entry and exit events to calculate parking duration, charges, and identify abnormal situations such as missing exits or duplicate entries. This table is primarily used for occupancy analysis and billing.

### Pricing Rules Table (`PRICING_RULES`)

This table defines the pricing structure applied to parking sessions. It includes hourly rates, free parking limits, peak-hour multipliers, and maximum daily charges. Separating pricing rules from event data allows flexible updates to tariffs without affecting historical records.

### Parking Facility Table (`PARKING_FACILITY`)

The facility table stores static information about the parking location, including total capacity and location details. It provides context for utilization and occupancy analytics and supports analysis across multiple facilities.

# Model Architecture

# Model Architecture Overview – Smart Parking Analytics

This project uses a layered machine learning approach, starting with computer vision models for real-time vehicle and parking analysis and extending to predictive and anomaly-detection models for operational insights. Each model is purpose-built to solve a specific part of the parking lifecycle, from physical vehicle detection to downstream analytics.

At the edge, **CNN-based computer vision models** are used to analyze parking area camera feeds and determine parking spot occupancy. Depending on camera configuration, this is implemented either through direct classification of predefined parking spots (occupied vs. empty) or by detecting vehicles within the frame and associating them with parking zones. These models provide continuous, high-confidence occupancy signals at the individual spot and facility level.

For vehicle identification, a **YOLO-based object detection model** is used to detect license plates from entry, exit, or parking-area camera images. YOLO is selected for its low-latency performance and strong accuracy in real-time detection scenarios. The detected license plate regions are passed to an **OCR model** (such as a CRNN or lightweight transformer-based recognizer) to extract the plate text. This enables consistent vehicle identification without reliance on manual input or faulty gate logs.

Once parking events and sessions are derived, **time-series forecasting models** are introduced to predict near-term occupancy levels and revenue. Models such as Temporal Convolutional Networks (1D CNNs) or LSTM/GRU-based architectures are trained on historical occupancy, entry/exit rates, and temporal features to anticipate peak periods and capacity constraints.

Finally, **anomaly-detection models** are applied to parking sessions and event data to identify operational issues. Techniques such as Isolation Forests or autoencoders flag abnormal patterns, including missing exit events, unusually long parking durations, duplicate entries, or sensor inconsistencies. These models help improve data quality, billing accuracy, and operational reliability.

Together, this model stack reflects a realistic, scalable smart parking architecture, combining real-time perception with predictive and diagnostic intelligence.