This document provides a checklist format for designing various systems commonly encountered in software development.

**I. System Overview**

* **System Name:** Briefly describe the system's purpose and functionality.
* **High-Level Design:** Outline the overall architecture of the system, including its components and interactions.
* **Scalability:** How will the system handle increased load and data volume?
* **Availability:** What measures are in place to ensure system uptime and redundancy?
* **Security:** How will the system protect user data and prevent unauthorized access?

**II. Specific System Checklists**

**1. Parking Lot System**

* **Low-Level Design:**
  + Data structures for representing vehicles and parking spaces.
  + Algorithms for assigning and releasing parking spaces.
  + Error handling for invalid operations (e.g., exceeding capacity)

Reference: [Parking lot:- low-level design. Parking-Lot OO design | by Abhijeet Gulve | Medium](https://medium.com/@abhigulve06/parking-lot-low-level-design-in-java-2be46101daec)

**2. Notification System**

* **System Design:**
  + Types of notifications (e.g., email, SMS, push notifications)
  + Delivery mechanisms for different notification channels
  + Scalability to handle high volumes of notifications

Reference: [Design Notification Services | System Design - GeeksforGeeks](https://www.geeksforgeeks.org/design-notification-services-system-design/)

**3. Cache System**

* **Complete Tutorial on LRU Cache with Implementations:**
  + Caching strategy (e.g., Least Recently Used (LRU))
  + Data eviction policy when cache capacity is reached
  + Cache invalidation techniques to ensure data consistency

Reference: [Complete Tutorial on LRU Cache with Implementations - GeeksforGeeks](https://www.geeksforgeeks.org/lru-cache-implementation/)

**4. URL Shortener System Design**

* **URL Shortener (bit.ly, TinyURL, etc):**
  + Generation of unique and short URLs
  + Mapping of short URLs to original long URLs
  + Redirection mechanism for resolving shortened URLs

Reference: <https://www.geeksforgeeks.org/problems/design-a-tiny-url-or-url-shortener2031/1>

**III. Additional Considerations**

* **API Design:** How will the system expose functionality through APIs for integration with other systems?
* **Monitoring and Logging:** How will the system be monitored for performance and errors?
* **Testing Strategy:** What are the approaches for testing the system's functionality and performance?

This format provides a structured approach to consider key aspects during system design. Remember to adapt and expand upon these checklists based on the specific system under development.