Addis Ababa University

Department of Computer Science

Introduction to Software Engineering

Project Title: Parking Lot Management System

System Design Document

Prepared by:

1. Natnael Mesfin UGR/8654/15
2. Mikiyas Fasil UGR/9231/15
3. Yosef Solomon UGR/7358/15

Submitted to: Ayalew Belay(PhD)

Submission date: December 19, 2024

Table of Content

1. Introduction

1.1. Purpose of the System

The purpose of this system design document is to serve as a comprehensive guide that defines the architecture, components, interfaces and data of the parking lot management system. The document defines the high-level system structure, components and subsystems of the parking lot management system. It serves as a roadmap for the software development by outlining the logical and physical design without implementation. This system design supports the scalability and maintainability of the project ensuring future modifications and upgrades. It ensures the parking lot management system meets design specifications and handles boundary conditions and errors.

Additionally, the document mitigates risks by identifying dependencies and constraints, while supporting seamless integration of hardware components, and user interfaces.

Finally, the parking lot management system design document ensures compliance with legal, regulatory and industry standards, serving as a formal record for audits and long term documentation

1.2. Design Goals

The design goals for the Parking Lot Management System focus on ensuring the parking lot management system is efficient, scalable and user-friendly, while meeting the operational needs of both administrators and employees.

These goals include:

- Optimised usage of parking slots

The parking lot management system employs real-time tracking and allocation algorithms to ensure efficient utilization of available parking spaces. This optimization reduces time spent searching for parking and maximizes the parking lot's revenue potential by minimizing idle spaces. This design goal originated from the technical goals of the parking lot management system.

- Accurate bill calculation and generation

The parking lot management system calculates parking fees with precision based on factors such as entry and exit timestamps, predefined rates, and any applicable discounts. Automated ticket validation and payment integration to ensure customers are charged the correct amount. This design goal is a refinement of the non-functional requirement “accuracy” of the parking lot requirement analysis document.

- Real-time data availability

The parking lot management system ensures up to date information on parking slot availability, payment statuses, and system health by leveraging continuous synchronization between the hardware and the software components of the parking lot management system. Real-time data availability enhances decision-making for users, operators, and administrators. This design goal is a refinement of the non-functional requirement “reliability” of the parking lot requirement analysis document

- Role based access control - technical

The parking lot management system restricts data based on user roles(e.g. Admin, employee). This approach confirms data security and operational integrity by allowing each actor to perform their designated tasks.This design goal originated from the technical goals of the parking lot management system.

- Scalable architecture for future expansion

The parking lot management system is designed with a modular architecture that supports adding new features in the future. It can be expanded to accommodate additional parking slots or locations. This design goal is a refinement of the non-functional requirement “scalability” of the parking lot requirement analysis document.

- Efficient data storage and retrieval

The parking lot management system uses a robust file management system optimized for fast retrieval and efficient storage of large datasets, such as parking lots, user profiles, and payment records.

- Refined features for enhanced user interaction

The parking lot management system offers user-friendly interfaces for employees, and administrators, designed with a focus on ease of navigation and accessibility. Intuitive console dashboards provide the admin with insights (records) and control over parking operations. This design goal is a refinement of the non-functional requirement “user-friendly interface” of the parking lot requirement analysis document.

Design Dependencies

Design dependencies for a parking lot management system refer to the external and internal factors that are required or influence the successful design and implementation of the parking lot management system.

Below are some key design dependencies in our system:

- Hardware Dependencies: Barriers/Gates, Payment Consoles/Terminals, Servers/Networking Equipment
- Software Dependencies: File management, OS Compatibility, Cloud Services
- Infrastructure Dependencies: Electricity, Internet Connectivity, Data Backup and Recovery Systems
- Legal and Regulatory Dependencies: Parking Policies, Data privacy regulations, Payment Compliance Standards
- Operational Dependencies: Employee Training, Maintenance Requirements, Customer Support and Feedback System
- Environmental Dependencies: Physical Layout of Parking Lots, Climate Conditions

1.3. Definition, Acronyms, and Abbreviations

- Admin: refers to the administrator responsible for managing the parking lot, including reservation and employee management.
- Employee: refers to an individual in charge of overseeing parking lot status and managing the physical operations of the parking lot. This includes reserving a parking slot, assisting customers with locating and parking on their designated floor, and ensuring successful completion of payment
- Customer: refers to the individual who is parking a car in the parking lot.
- Parking Slot: refers to the designated space within a parking lot where the vehicles are parked.
- Reservation: refers to the process of booking a specific parking slot for a particular time.
- Parking Rate: refers to the charge applied per unit time a vehicle occupies a parking slot.
- User: admin, employee or customer using the parking lot management system.

1.4. References

B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

IEEE, "*IEEE 830-1998 - Recommended Practice for Software Requirements Specifications*," IEEE, 1998.

GeeksforGeeks, "What is Systems Design - Learn System Design," https://www.geeksforgeeks.org/what-is-system-design-learn-system-design /.

1.5. Document Overview

This document primarily emphasizes the necessary processes, along with the software and hardware requirements, for the successful implementation of the software. It describes the architecture, operational workflow, and resources needed to ensure the parking lot management system fulfills its functional and non-functional criteria.

Additionally, it details the integration of essential components such as the user interface, payment systems, and administrative dashboards. The document highlights the importance of scalability and interoperability. It also tackles critical considerations such as data security, real-time updates, and role-based access control to guarantee reliability and user satisfaction. Furthermore, it offers an in-depth examination of dependencies, risks, and fallback strategies to address potential failures, along with a roadmap for system deployment, testing, and maintenance to assure long-term operational efficiency and adaptability.

2. Current Software Architecture

2.1. Overview of the current system

The current parking lot management systems in Ethiopia mostly rely on manual operations, leading to inefficiencies in slot management, billing and data tracking. Employees manually check availability, calculate fees, and generate reports, which is error-prone and time-consuming.

Additionally, there is minimal or no integration of modern technologies, leading to an overreliance on human intervention for routine tasks. Payment methods are predominantly cash-based, creating challenges in financial reconciliation and increasing the likelihood of revenue leakage. Moreover, customer experience is often negatively affected due to long waiting times, unclear processes and the absence of a user-friendly parking lot ticketing system.

2.2. Process Flow

1. Vehicle Arrives:

   Customers enter the parking lot with their vehicle.

2. Employee Checks Slot Availability (Manual):

   An employee inspects the parking lot to identify available spaces.

3. Customer Receives Slot Information:

   The employee communicates and shows the available slot to the customer

4. Vehicle is Parked:

   The customer parks their vehicle.

5. Employee Manually Calculates Fee:

   At the end of the parking period, the customer moves their vehicle to the exit gate. After that the employee calculates the parking fee manually based on the estimated duration.

6. Customer Pays (Cash):

   The customer pays the parking fee in cash.

7. Employee Issues Receipt (Manual):

   The employee writes or prints a manual receipt for the payment

8. Vehicle Exits

   Customer exits the parking lot with their vehicle

In conclusion, the current architecture is highly fragmented, resource-intensive, and unable to scale effectively to meet the growing demand for efficient parking solutions in metropolitan areas like Addis Ababa.

3. Proposed Software Architecture

3.1. Overview

A centralized, modular system with components for parking space management, allocation, ticketing, payment processing and analysis. This architecture is built to support future scalability. The proposed system follows a modular and scalable design to establish a flexible, high performance and easier to maintain system. At the core of the parking lot management system lies a multi-layer architecture consisting of the presentation layer, business logic layer and the data storage layer.

- Presentation Layer

The presentation layer is designed using java's GUI interface that communicates with the server via an application programming interface. It is the topmost layer and acts as the interface between the user and the parking lot management system. It is responsible for displaying information and receiving input.

Responsibilities

Render data in understandable format

Handle user input and interactions

Provide feedback

Manage sessions and user state (e.g. login status)

- Business Logic Layer

The business logic layer handles the core processing and decision-making, utilizing microservices to enable scalability and deployment. This is handled by the local server using Java's object oriented principles. This layer is responsible for orchestrating the flow of data between the presentation and the data storage layer.

Responsibilities

Process requests from the presentation layer.

Apply business rules to make the parking lot management system behave as required.

Handle transactions, ensure consistency and integrity of operations.

Maintain user authentication and authorization.

- Data Storage Layer

The data storage layer utilizes java's file saving system to accommodate structured and unstructured data. It is responsible for managing the persistent data and providing access to it.

Responsibilities

Persist and retrieve data requested by the business logic layer.

Manage and optimize database transactions to ensure data consistency (ACID properties).

Provide efficient data retrieval, while having high performance during high load.

3.2. Subsystems Decomposition

The Subsystem Decomposition of the parking management system is designed to define the modular structure that organizes the system's functionalities into distinct and cohesive units. It addresses specific systems that are independent and managed separately to ensure maintainability and scalability. Each subsystem is designed to focus on a specific aspect of the parking management workflow.
The following subsystems work collaboratively to enable efficient employee management, vehicle registration, record tracking, and other essential operations. By separating the responsibilities into manageable components, the system ensures secure data handling and reliable storage solutions to provide a robust and user-friendly parking management solution.

1. User Management Subsystem:

It is responsible for the authentication and authorization of users within the system. It handles login and logout functionalities to ensure secure access.
Additionally, this subsystem may include features that allow users to manage their accounts and profiles, and enable them to efficiently interact with the system and access authorized tasks.

2. Employee Management Subsystem:

It is responsible for the efficient management of employees. It allows administrators to add and remove employees, as well as view employee information. It includes use cases for adding, removing, and displaying employee details.

3. The Vehicle Management Subsystem:

This subsystem is responsible for handling vehicle-related functions within the system. It registers vehicles during check-in, calculates payments, and generates bills during check-out. It allows for streamlined parking process through the “check in” and “check out” use case functionalities

4. Record Management Subsystem:

This subsystem facilitates comprehensive record retrieval for employees and administrators. It enables employees to monitor the status of the parking lot and provides administrators with access to exclusive daily reports, including payment details, to efficiently oversee and manage the system's operations.

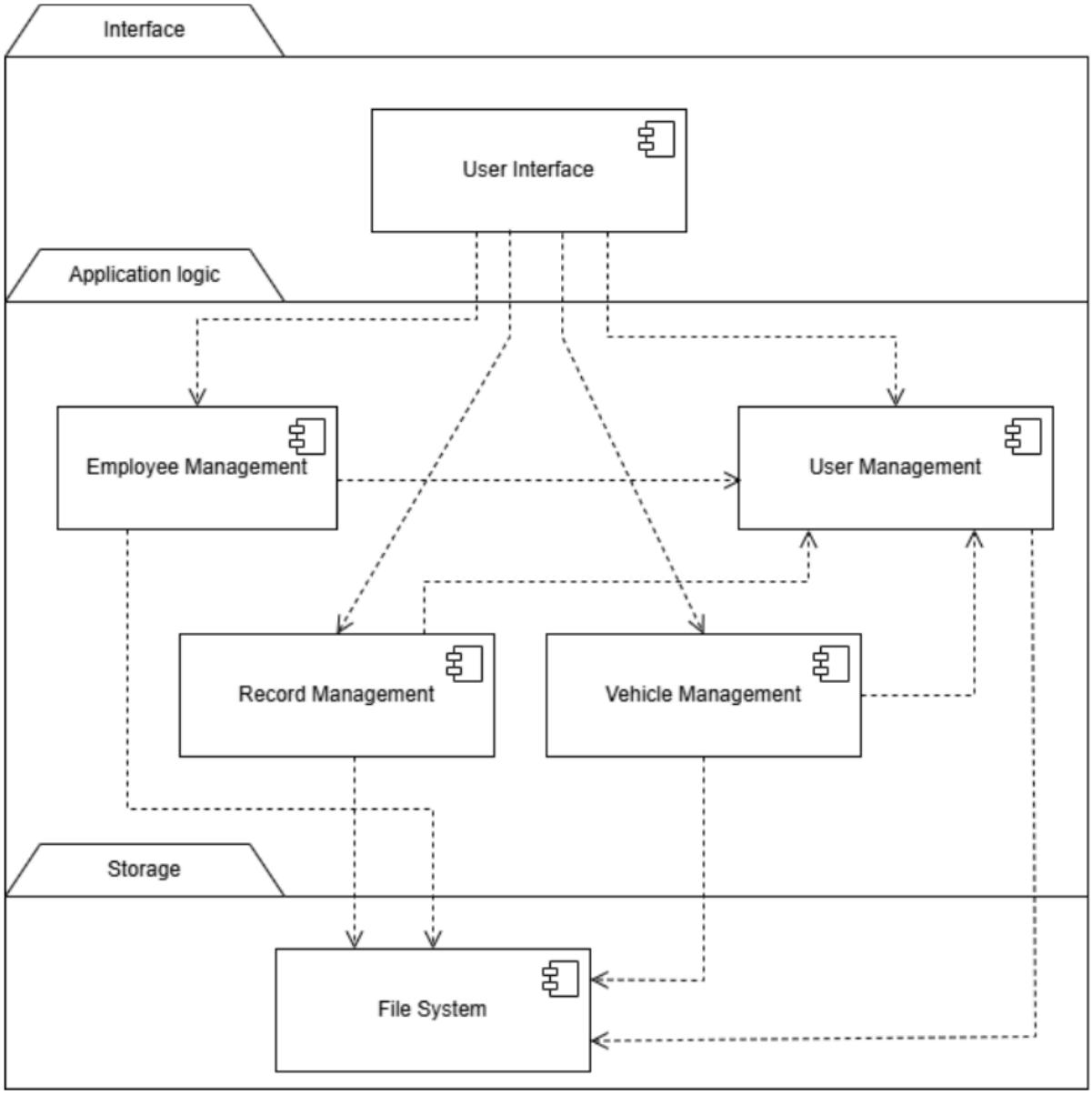
5. User Interface:

This subsystem is responsible for providing a graphical interface through which users interact with the system. It allows admins and employees to access and navigate the system's features through components such as buttons, forms, and dashboards to facilitate interaction with the underlying functionalities.

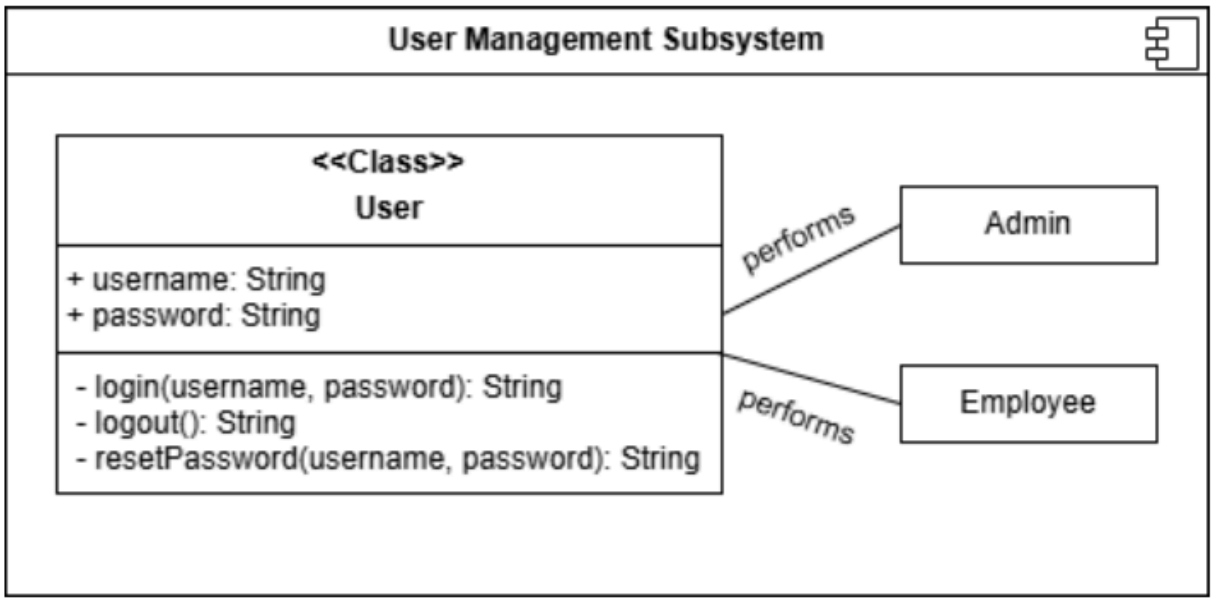
6. File System:

This subsystem serves as a repository for storing essential information related to vehicles and employees. It organizes and manages data to enable easy access to the system’s operations.

Subsystem Decomposition Diagram

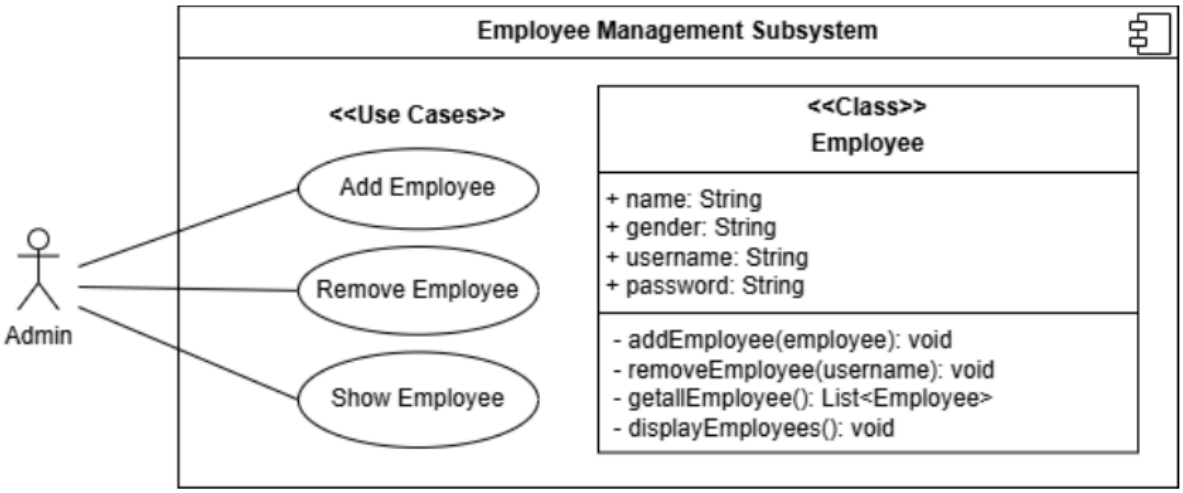


1. User Management Subsystem



- Classes
    - User: Is a class that serves as a template for defining specific user classes. It contains attributes “username” and “password”. It is a template for defining “Admin” and “Employee” classes.
    - Admin: Represents an administrator of the system who is assigned administrative roles in the Parking Lot Management Software.
    - Employee: Represents an Employee who is assigned employee roles in the Parking Lot Management Software.
- Responsibilities of the User Management Subsystem
    - Authentication: The user management subsystem is responsible for efficiently verifying a user's identity to ensure that only authorized individuals can access the system.
    - Authorization: The user management subsystem is responsible for assigning appropriate access rights, distinguishing between admins and employees, and granting them access to the relevant system features.
    - Password Management: The user management subsystem handles secure password management operations. Its operations are updating and validating passwords.

2. Employee Management Subsystem

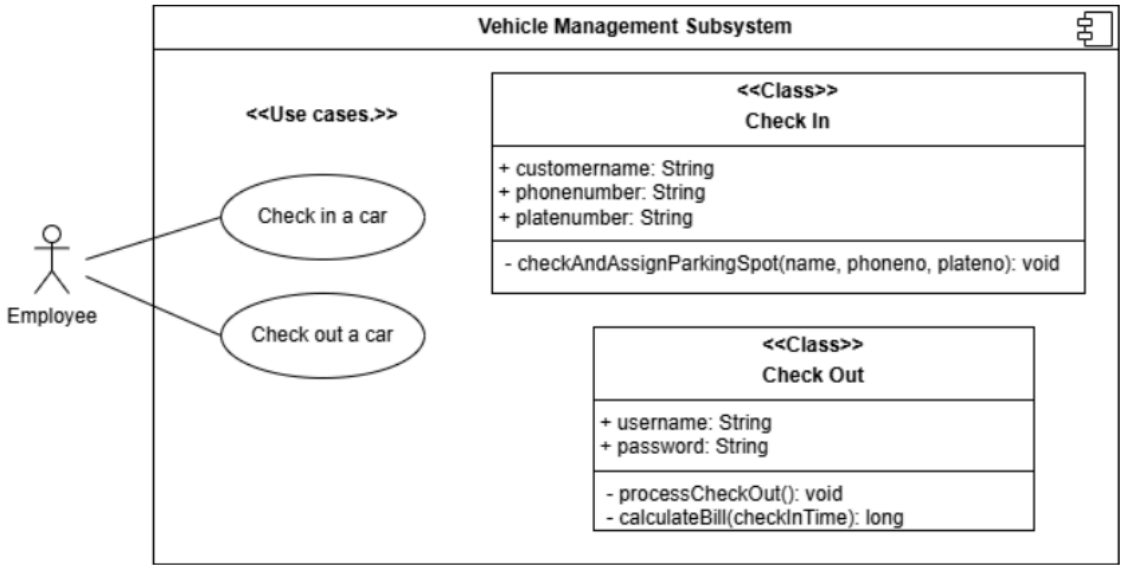


- Classes
    - Employee: The Employee class serves as the foundational class for managing employee details. It includes functionalities for adding, removing, and showing employee information. It uses attributes such as name, gender, username, and password to register the essential details of an employee to enable the system to effectively manage their information.
- Responsibilities of the Employee Management Subsystem
    - Employee Management: The Employee class is responsible for managing employee details, including adding, removing, and displaying employee information.
    - Employee Registration: The Employee class uses attributes such as name, gender, username, and password to register the necessary details of an employee to properly keep records and enhance system interaction.
    - Update Employee Records: Allow modification of employee details when necessary for accurate and up-to-date information.

- Employee Display: The Employee class is responsible for showing the relevant details of an employee, such as their name, gender, username, and other associated attributes, to authorized administrators within the system. This assists easy viewing of employee data as needed for management purposes.

- Use Cases
  - Add Employee: It registers a new employee to the filesystem based on the attributes using the "addEmployee" function.
  - Remove Employee: It removes an employee from the filesystem based on the details provided. It uses the "removeEmployee" function.
  - Show Employee: It shows the employee list that are currently working.

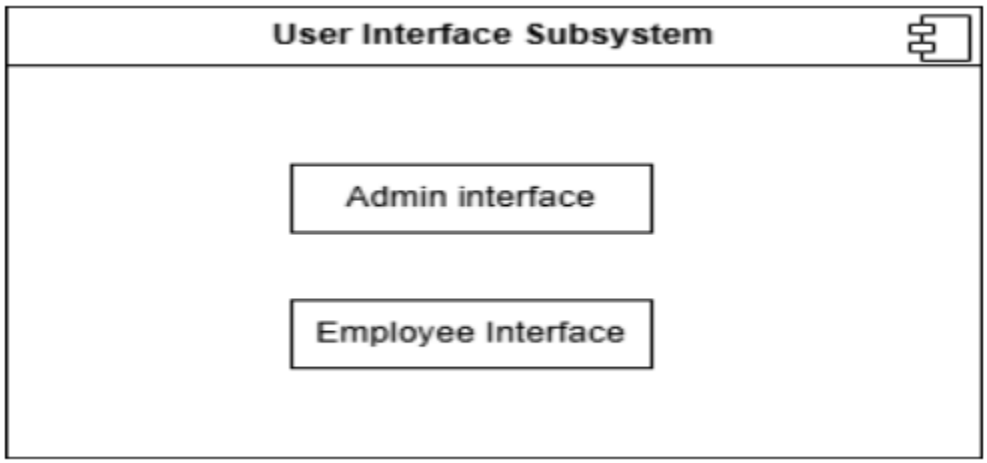
3. Vehicle Management Subsystem



- Classes
    - Check In: It serves as the core component for monitoring the check-in process. It verifies available parking spots, assigns slots to vehicles, and registers customer details. Using attributes such as customer name, phone number, and plate number, it ensures accurate and clear registration of customer information during the check-in process.
    - Check Out: The Check-out class manages the check-out process by calculating the parking fee, generating the bill, and freeing up the parking spot. It uses customers' plate number to retrieve the relevant information and ensure an accurate bill is generated to leave the parking spot available for the next vehicle.

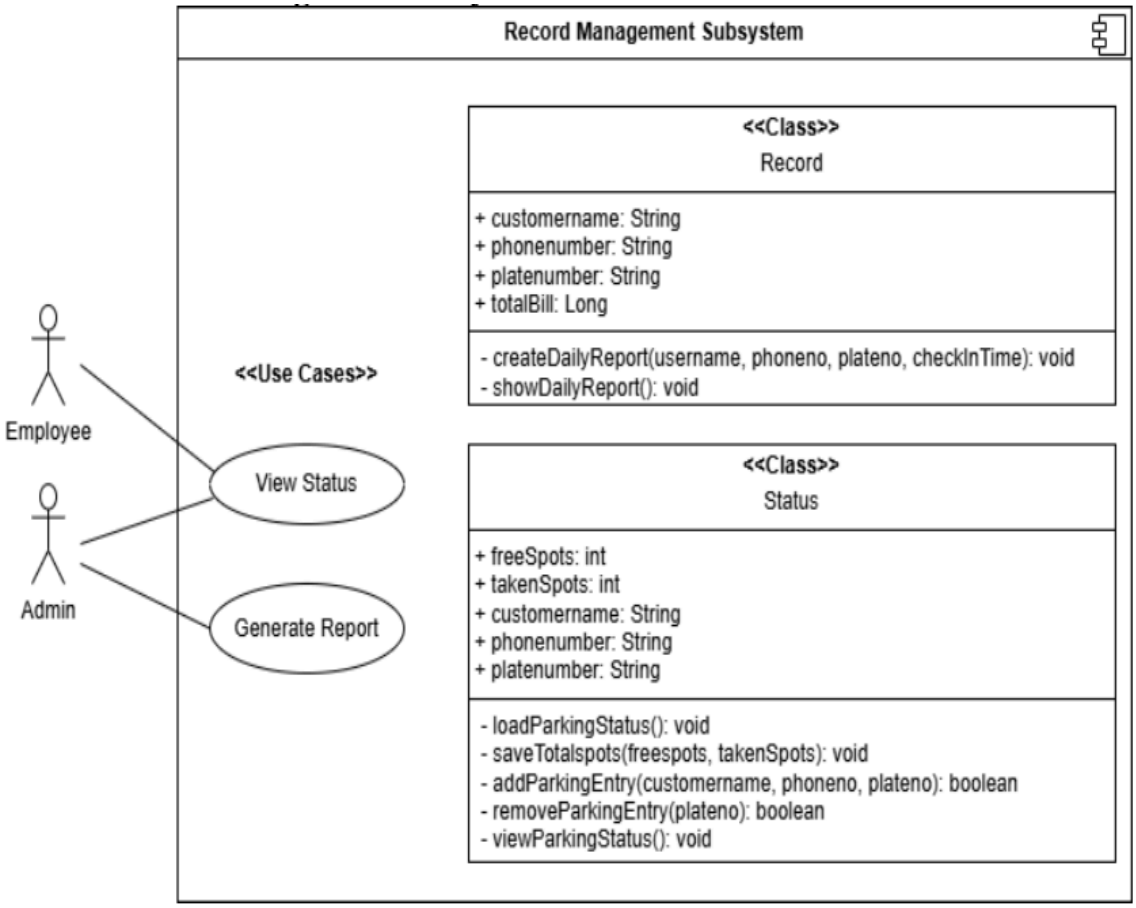
- Responsibilities of the Vehicle Management Subsystem
    - Manage Check-in Process: Monitor available parking spots, assign parking slots to oncoming vehicles, and register customer details (name, phone number, plate number).
    - Track Parking Duration: Begin tracking the parking duration for each vehicle upon check-in.
    - Calculate Parking Fee: Calculate the parking fee based on the vehicle's duration of stay.
    - Generate Bill and Complete Checkout: Generate the final bill for the customer and free the assigned parking spot upon checkout.
    - Update Parking Status: Maintain an updated status of parking spot availability to optimize space usage.
- Use Cases
    - Check in a car: This use case enables employees to register a car and assign a parking slot based on real-time availability.
    - Check out a car: This use case allows employees to calculate and generate bill, and free a parking slot.

4. User Interface Subsystem



- Classes
    - Admin Interface: It serves as the user interface designed specifically for administrators. It provides access to critical system management functionalities, allowing admins to oversee and manage administrative operations.
    - Employee Interface: It is designed for employees. It offers them a streamlined interface to perform their daily tasks related to vehicle check-in, check-out, and managing customer interactions.
- Responsibilities of User Interface Subsystem
    - User Interaction: Provide a graphical interface through which users (admins and employees) can interact with the system. This facilitates smooth navigation between different functionalities.
    - Data Presentation: Display relevant information clearly, such as vehicle records, employee details, parking status, and billing information to enable the user easily interpret and act upon it.
    - Task Execution: Enable users (admins and employees) to execute key tasks, such as adding or removing employees, checking in or checking out vehicles, and generating reports, in a responsive environment.
    - User Access Control: Restrict access to certain features based on user roles (admin or employee). This ensures that sensitive functions are protected from unauthorized access.

5. Record Management Subsystem



- Classes
    - Record Class: This class is responsible for generating and displaying daily reports that are accessible only to administrators. These reports include critical information, such as payment and billing details to offer insights into the financial activities of the parking lot. The data within the reports is derived from transactions occurring throughout the day.
    - Status Class: It serves as a tool for both administrators and employees to view the current status of the parking lot. It provides real-time information about the availability of parking spots and the status of vehicles. By using customer details like name, phone number, and plate number, the Status Class ensures that both admins and employees can track and manage the parking lot's occupancy efficiently.

- Responsibilities of Record Management Subsystem
    - Data Management: The subsystem ensures accurate and up-to-date transaction and billing information for daily reporting. It organizes and stores essential data related to parking lot activities.
    - Real-Time Data Representation: The subsystem generates daily reports summarizing activities, payments, and vehicle details to provide a financial overview. Additionally, it offers real-time updates on parking space availability, allowing users to monitor parking lot occupancy and track vehicle movements in the lot.
    - User Interaction: The subsystem facilitates user interaction through intuitive interfaces for both admins and employees, enabling efficient management of operations. The user interfaces provide clear visual elements and actionable features, ensuring a smooth experience for monitoring and managing parking lot activities.
    - Access Control: The Record subsystem ensures that only authorized users (admins) can access sensitive financial data and generate reports. It provides secure access based on user roles. The Status subsystem provides access to both admins and employees, as it offers a view of the parking lot's status to support both operational monitoring and daily activities.
- Use Cases
    - Generate Report: This use case allows admins to view and monitor daily reports including financial transactions.
    - View Status: This use case enables both administrators and employees of the parking lot to oversee the real-time status of the parking lot.

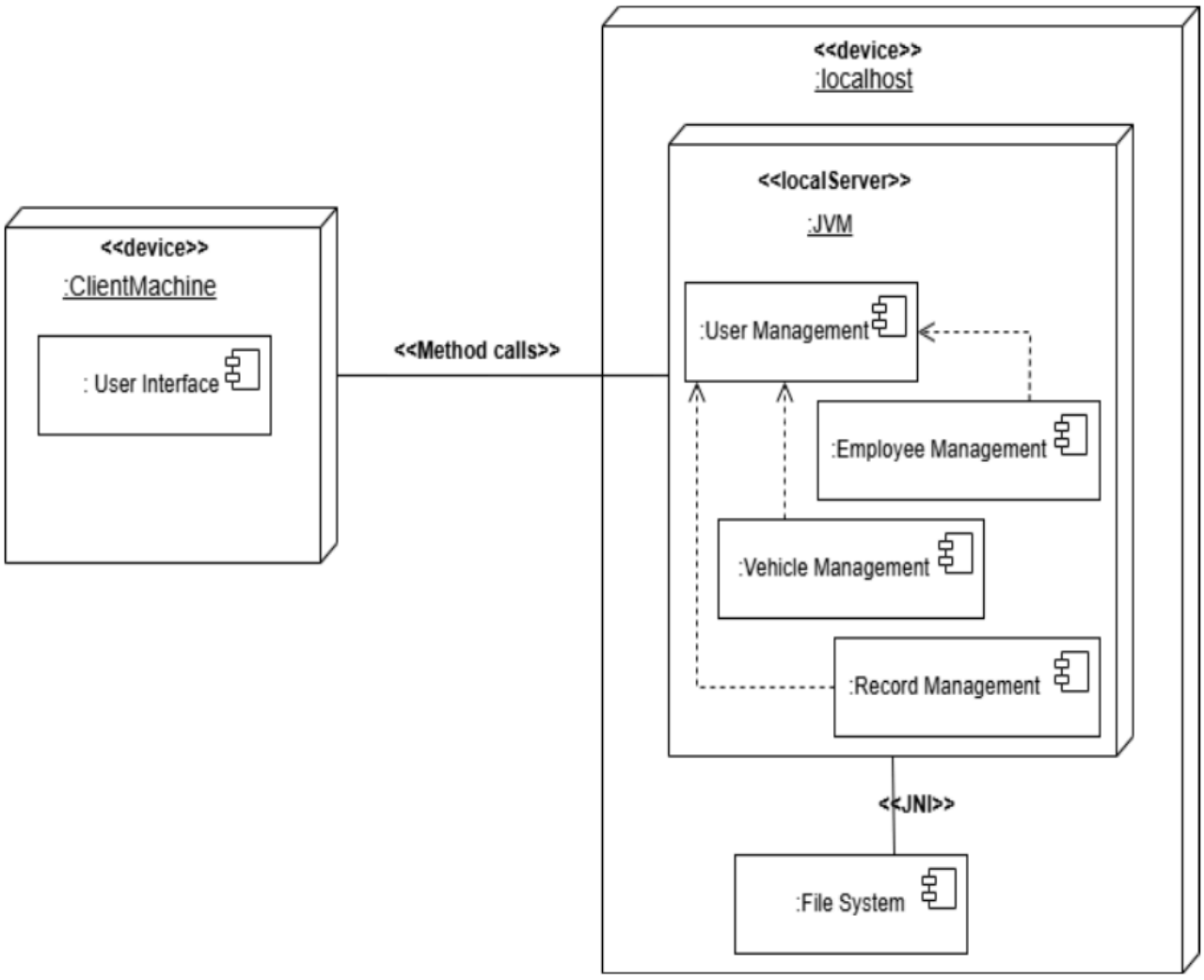
6. File System

- Responsibilities of File Management Subsystem

The File System subsystem is responsible for storing, organizing, and managing critical data related to the parking management system. It ensures that all records, including employee details, vehicle information, transaction history, and billing data, are securely stored in a structured format.

This subsystem facilitates efficient retrieval and updating of data as needed. It supports key system operations such as employee management, vehicle check-in/check-out, and report generation. By maintaining a reliable and accessible storage solution, the File System subsystem contributes to the overall stability and functionality of the parking management system.

3.3. Hardware / Software Mapping



Description of mapping subsystems to devices and processors

1. Client Machine

- Type: Physical hardware device.
- Subsystem: User Interface

    - The user interface component resides on the client machine. It helps users to interact with the application. It provides interfaces such as input fields and buttons to communicate with the backend system.
- Communication:
    - The client machine communicates with the localhost (server) via method calls, which represent direct Java API invocations or function calls in a standalone setup.

2. Local Host

- Type: Logical representation of the local server environment hosted on the same machine as the client.
- Subsystems Hosted on the locally on JVM:
    - JVM: The Java Virtual Machine acts as the runtime environment for the software subsystems by translating Java code into native machine instructions.
    - User Management: Handles authentication (e.g., login, logout) and user-related functionalities like profile management.
    - Employee Management: Manages employee details, including adding, removing, and retrieving employee information.
    - Vehicle Management: Handles vehicle operations like check-in and check-out processes.
    - Record Management: Provides access to parking lot records and daily reports for administrators and employees.
- Communication
    - Subsystems interact with one another through internal method calls within the JVM as shown by arrows in the diagram. For example:
        - The Vehicle Management subsystem may request user authentication from the User Management subsystem.

3. File System

- Type: External hardware/software resource managed by the operating system.
- Purpose:
    - Stores essential data related to vehicles, employees, and operational records.

- Communication:
    - The JVM interacts with the File System using JNI (Java Native Interface). JNI acts as a bridge between the Java application and native system calls for file operations, such as creating, reading, and writing files.

3.4. Persistent Data Management

3.4.1. Identifying Persistent Objects

1. User

The User object is created and accessed by the User Management subsystem. It should be persistent to ensure that user accounts, roles, and authentication details remain available across different sessions.

Attributes: username, password

2. Employee

The Employee object is created and accessed by the Employee Management subsystem.
The persistence of the employee object is necessary to enable administrators to easily manage employee records and ensure the system tracks employee activities effectively.

Attributes: name, gender, username, password

3. Vehicle

The Vehicle object is created and accessed by the Vehicle Management subsystem. It should be persistent to track vehicle check-in and check-out details for billing and data storage purposes.

Attributes: customername, phonenumber, platenumber

4. Payment

The Payment object is created and accessed by the Vehicle Management subsystem. The persistence of the payment object is essential to ensure that payment details are available for auditing, reporting, and generating bills.

Attributes: amount, checkouttime

5. Daily Report

The Daily Report object is created and accessed by the Record Management subsystem. It should be persistent to provide administrators with summarized daily parking statistics, including vehicle count and revenue.

Attributes: totalBill, freeSpots, takenSpots

Selecting a Storage Strategy

In our storage management strategy, we have chosen to utilize a file system approach. This decision is driven by the need for efficient data organization, ease of access, and the ability to manage both small and large files. A file system ensures that data is stored in a structured way, making it easy to locate, update, and maintain.

Fundamental Strategies for Implementing the File System in Vehicle Management

- Hierarchical Data Organization: The file system provides a structured, hierarchical organization for the vehicle-related and employee-related data. Data for each vehicle can be stored within dedicated directories, with subdirectories for specific categories such as registration details and transaction history. This logical organization ensures that data is easily accessible and manageable.
- Efficient Data Retrieval: The file system allows for the quick retrieval of vehicle data based on filenames or associated metadata such as license plate numbers, check-in times, and payment statuses. This enables quick access to critical information when required, such as during the billing process or checkout procedures.

3.5. Access Control and Security

Admin: Full Access to manage employees, generate reports and view parking statuses.

Employee: Limited Access to manage parking status, facilitate check in and check out and manage their profiles.

The following table illustrates the global access table for the parking lot management system:

Object Actors	Office	On-Site
Admin	addEmployee() removeEmployee() showEmployee() viewParkingStatus() generateReport()	
Employee	viewParkingStatus() changePassword()	checkInACar() checkOutACar()

3.6. Global Software Control

The flow control when it comes to the parking lot management system is threads based. This aligns with the need to handle multiple simultaneous user requests. The parking lot management system uses the Java file system for managing operations and storing data. The Java File System enables direct interaction with resources while supporting a thread-per-request model to process each request. Each request, such as booking a parking slot, processing a payment, or validating a ticket, is handled by a separate thread, using parallel processing.

Since the parking lot management system operates on a threads-based architecture, concurrency needs to be managed explicitly. We define specific strategies and rules to handle simultaneous accesses to shared data, such as synchronized updates to parking slot statues, secure payment processing, and real-time tracking of system resources.

We define the following strategy and rules for dealing with concurrent accesses to shared data:

- Boundary objects should be as stateless as possible to avoid issues related to event concurrency. Any state required for processing events should be encapsulated within the event payload or passed explicitly via parameters. Boundary objects should validate events data upon receipt to ensure consistency and prevent invalid states from propagating through the system.
- Control objects should act as even processors and should utilize asynchronous event queues. Ensure that services or DAOs invoked by control objects are unchanged, so repeated or delayed events do not cause inconsistencies. Each control object should process one event at a time, relying on the event-driven framework to handle parallel processing.
- Entity objects should not provide direct access to their fields. They should instead rely on event models to update state. When state changes are necessary, emit domain events for consistency and inform other components of the change. Use event versioning or timestamps to detect and resolve conflicts in case multiple events attempt to update the same state simultaneously.
- Methods for accessing state in entity objects should be designed for non-blocking, asynchronous execution. Utilize event-based mechanisms like optimistic concurrency control, where state changes are validated only when persisted.

For this parking lot management system, a centralized flow is ideal:

Centralized components: Handle parking availability, payment processing, and reporting

Major events and actions in the parking lot management system:

- Vehicle Entry

  Check parking availability

  Generate and assign parking tickets

- Slot Management

  Monitor slot status

  Update availability in real-time

- Payment Processing

  Calculate parking fees based on a predetermined parking rate and time spent during parking

- Vehicle Exit

  Successful exit of vehicle upon ticket validation and successful payment.

Mapping Responsibilities across sub-systems

- User management
  Manages user information (like vehicle license plate, phone number)
- Employee management
  Facilitate the on - site processes needed to create a seamless parking experience
- Vehicle management
  Manages the checking in and the checking out of the vehicle
- User interface
  Means for the user to interact with the parking lot management system.
- Admin Console

Oversee and configure the overall operation of the parking lot management system and the employees

Access analytics and reports for continuous improvement

Vehicle Entry Flow

1. Employee detects a vehicle at the entry gate.
2. Central server checks the availability of slots.
3. If space is available:

   Generate parking tickets.

   Vehicle is ushered in to its designated parking slot
4. If space is unavailable:

   Employees should inform the user of the unavailability of slots.

Vehicle Exit Flow

1. Customer scans the parking ticket at the console provided by the employee at the exit gate.
2. The console sends the parking information to the central server where parking status is verified and parking time is calculated.
3. Customer is informed of the parking fee to be paid calculated based on the predetermined parking rate and the time spent calculated from the parking ticket.
4. Employees facilitate the payment process by asking the customer their most convenient payment method.
5. If payment is successful:

   Receipts are issued to the customer.

   Vehicle is let out of the parking space.
6. If payment is unsuccessful:

   Inform the customer of the failed payment and prompt the customer to try again.

3.7. Boundary Conditions

- Use Case Configuration

  Identify and manage the lifecycle of persistent objects that are created during system initialization. Archive or delete the objects as part of maintenance.

  System Configuration

  Allow administrators to set up parameters such as parking fees, operating hours, and maximum capacity.

- System Startup

  Ensure all subsystems initialized successfully.

  Load real-time parking space availability data into a database or file system.

  Initial State

  The parking lot management system begins with no pre-loaded data, this means all records are added after the parking lot management system is launched.

- System Shutdown

  Successfully save the current state of the parking lot management system.

  Notify users and administrators of shutdown.

  Final State

  Data is archived for historical analysis, and the server is shut down.

- Exception Handling

  Hardware Failures

  Provide rollback options to recover, and also provide fallback options until the parking lot management system is fixed.

  Network/ Server Failures

  Notify the admin or the employee of the failure. We expect the actor involved to react to the failure and restore connection, at the cost of losing data that was filled in the parking registration form.

- Error Behavior

  Ensure all connected hardware functions properly and are properly synchronized with the software system.

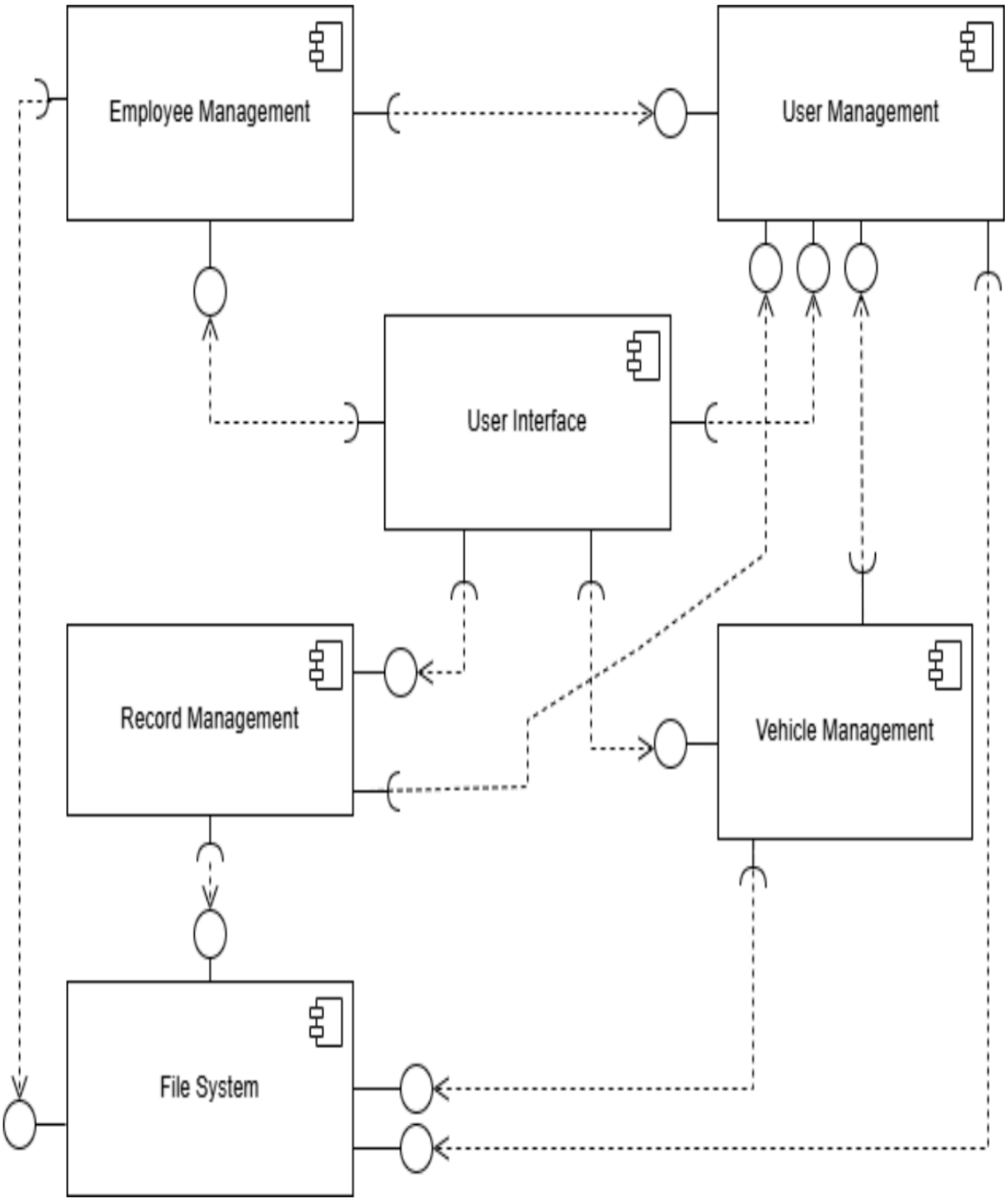
  Log errors for troubleshooting and continuous improvement.

4. Subsystem Services

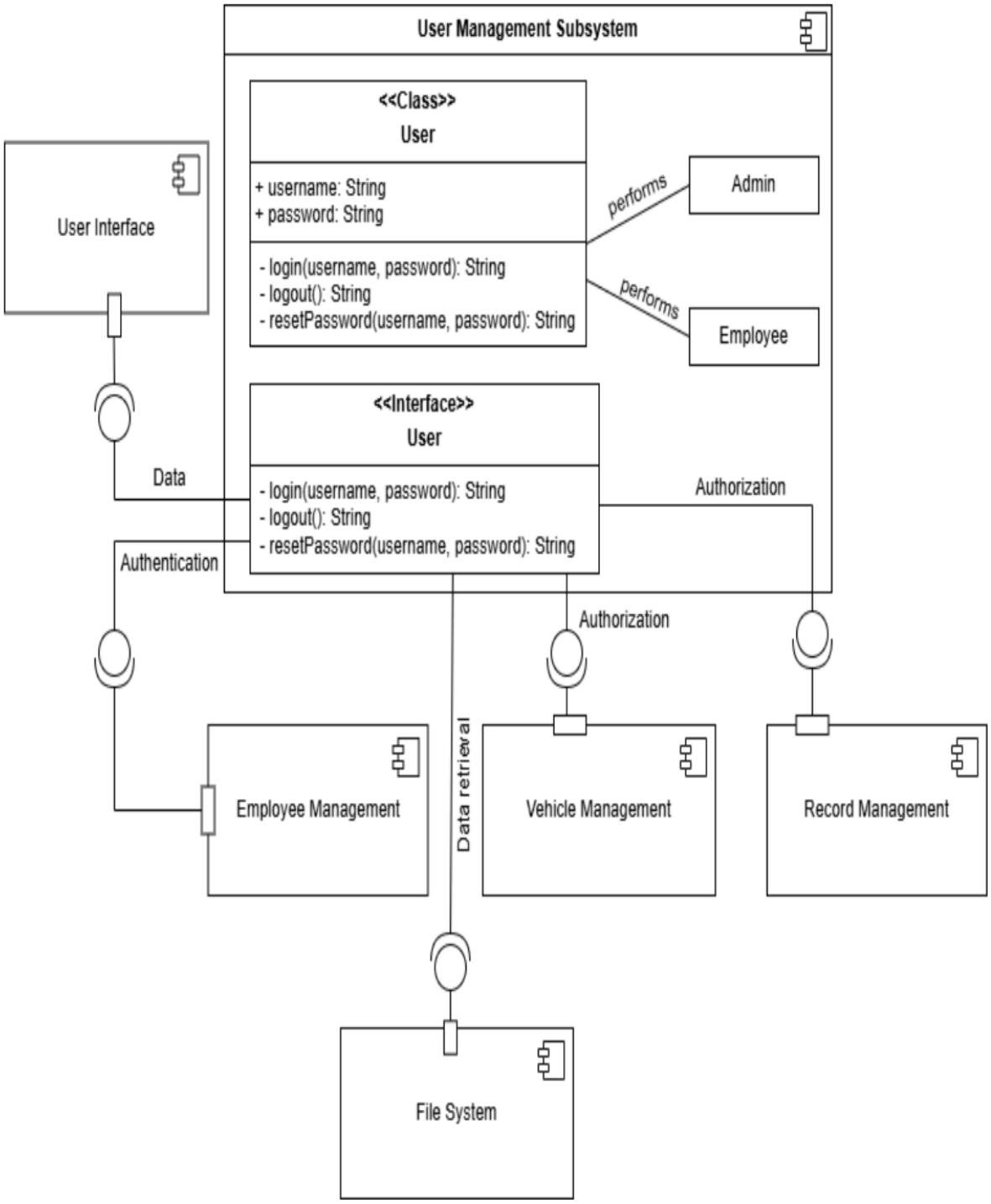
Subsystem Services and Architecture

The subsystem architecture defines the structural design of each subsystem. It details its interfaces, services, and how it interacts with other subsystems. For each identified subsystem, the architecture outlines the key services it provides, such as user authentication, data storage, or vehicle management, along with the specific responsibilities it handles. It also defines the subsystem's interfaces. With the providing and requiring interfaces clearly described, it shows how the subsystem communicates with other subsystems to ensure that essential data and services are shared efficiently.

Subsystem Architecture



1. User Management Subsystems Services and Architecture



- **User Management Subsystem Services**

The user management subsystem provides authentication and authorization services for Employee Management, Record Management and Vehicle Management subsystems. It also provides data access service for the user interface subsystem to enable the user interface render the proper information. The user management subsystem, also, requires data retrieval service from the file subsystem.

- **Coupling with other subsystems**

The user management subsystem has interface-based coupling with other subsystems. These subsystems are User Interface, Employee Management, Vehicle Management, File System and Record Management subsystems.

- Interfaces

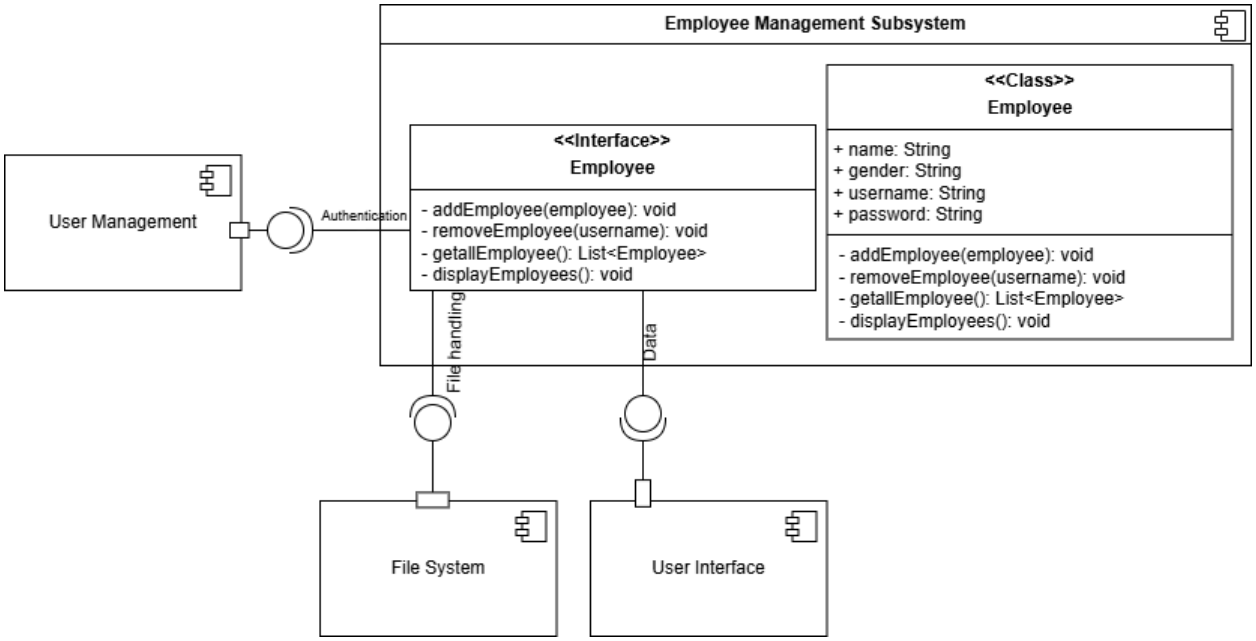
Providing Interfaces:

The user management subsystem provides authentication services to Employee Management, Record Management and Vehicle Management subsystems through the *login()* and *logout()* interfaces.

Requiring Interfaces:

The user management subsystem requires services from the file system to successfully authenticate users. They are required through file reading operation interfaces.

2. **Employee Management Subsystem Services and Architecture**



- **Employee Management Subsystem Services**

The employee management subsystem provides administrators with effective services to manage employees of the parking lot. It enables streamlined registration and monitoring of employee data. This subsystem provides data access services to the user interface subsystem to enable the user interface subsystem to efficiently display information. It also requires *file handling* and *authentication* services from file system and user management subsystems respectively.

- **Coupling with other subsystems**

The employee management subsystem has interface-based coupling with other subsystems. These subsystems are User Interface, User Management and File system subsystems.

- **Interfaces**

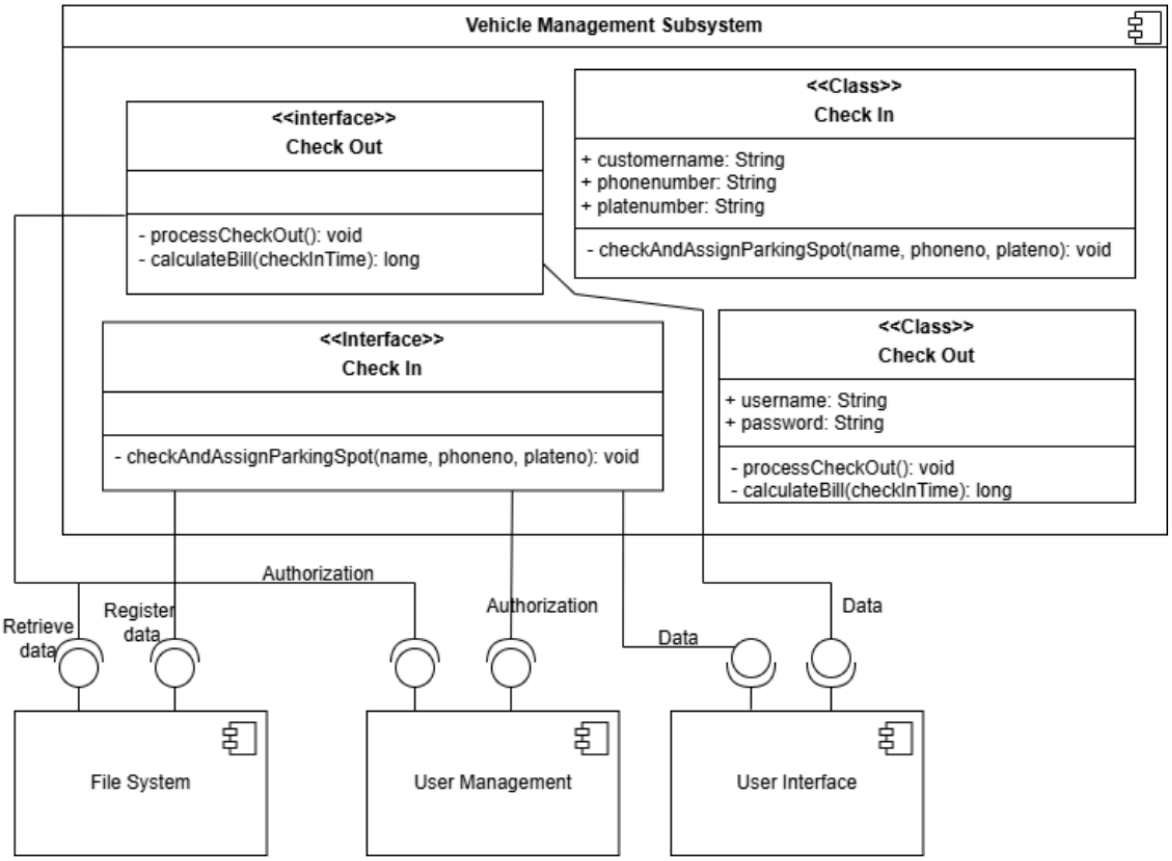
Providing Interfaces:

The employee management subsystem provides services to the user interface subsystem. It allows the user interface subsystem to render the data properly through the *displayEmployee()* interface.

Requiring Interfaces:

The employee management subsystem requires services from the user management subsystem to successfully authenticate admins who manage employees. These services are provided through the *login()* and *logout()* interfaces of the user management subsystem. It also requires services from the file system to register and manage employee information. These services are required through the *addEmployee() and removeEmployee()* interfaces.

3. **Vehicle Management Subsystem Services and Architecture**



- **Vehicle Management Subsystem Services**

The vehicle management subsystem provides employees with effective services to manage vehicles in the parking lot. This subsystem provides data access services to the user interface subsystem to enable the user interface subsystem to efficiently display information. It also requires *data handling* and *authorization* services from file system and user management subsystems respectively.

- **Coupling with other subsystems**

The vehicle management subsystem has interface-based coupling with other subsystems. These subsystems are User Interface, User Management and File system subsystems.

- **Interfaces**

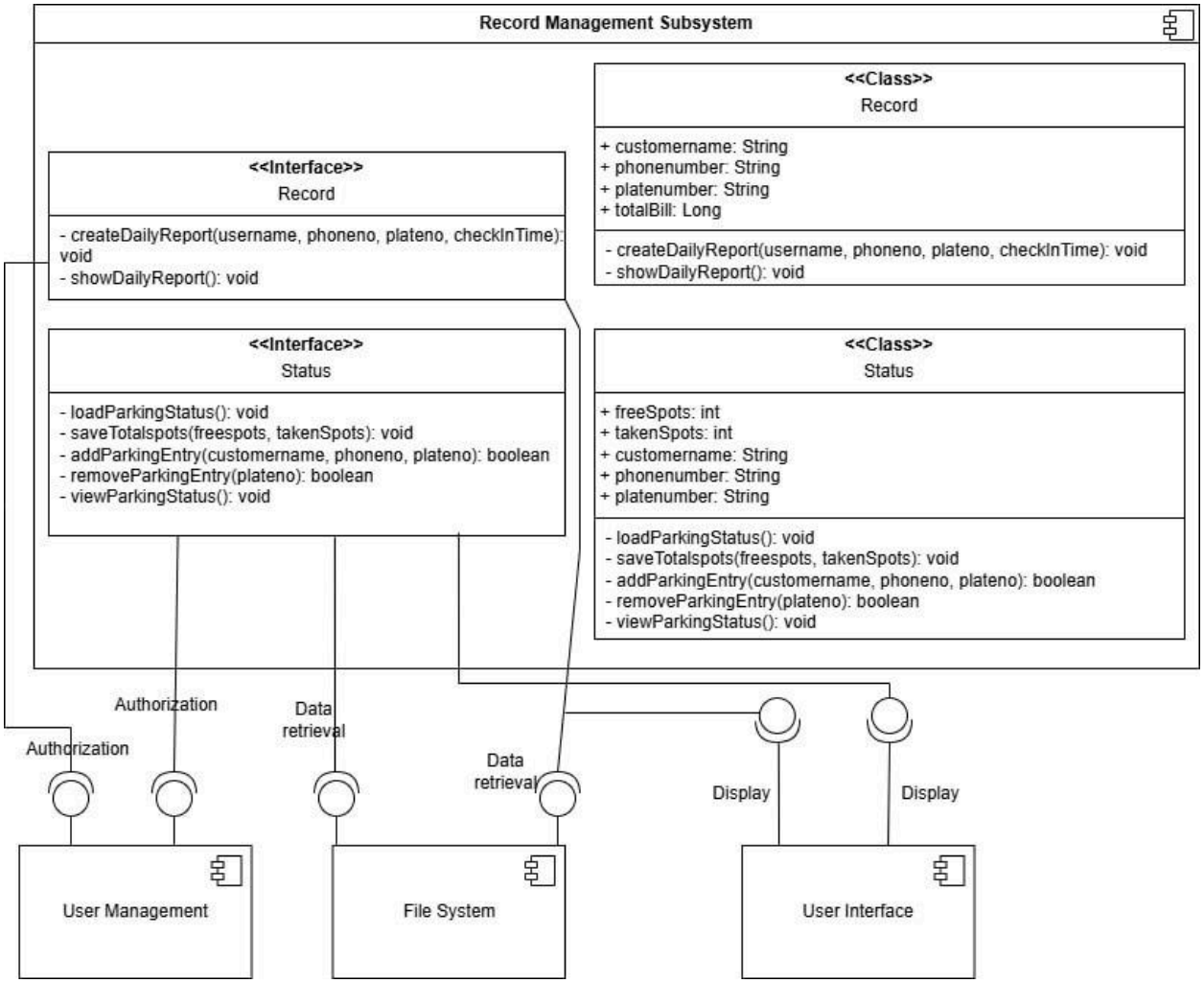
Providing Interfaces:

The vehicle management subsystem provides services to the user interface subsystem. It allows the user interface subsystem to render the required data and GUI properly.

Requiring Interfaces:

The vehicle management subsystem requires services from the file system to register, retrieve and manage vehicle information successfully. These services are required through the *checkAndAssignParkingSpot()* and *calculateBill()* interfaces. It also requires services from the user management subsystem to authenticate employees performing vehicle management operations. These services are provided through the *login()* and *logout()* interfaces of the user management subsystem.

4. **Record Management Subsystem Services and Architecture**



- **Record Management Subsystem Services**

The record management subsystem provides employees and administrators with summarized data to oversee the parking process. This subsystem provides data access service to the user interface subsystem to enable the user interface subsystem to efficiently display information. It also requires *data retrieval* and *authorization* services from file system and user management subsystems respectively.

- **Coupling with other subsystems**

The record management subsystem has interface-based coupling with other subsystems. These subsystems are User Interface, User Management and File system subsystems.

- **Interfaces**

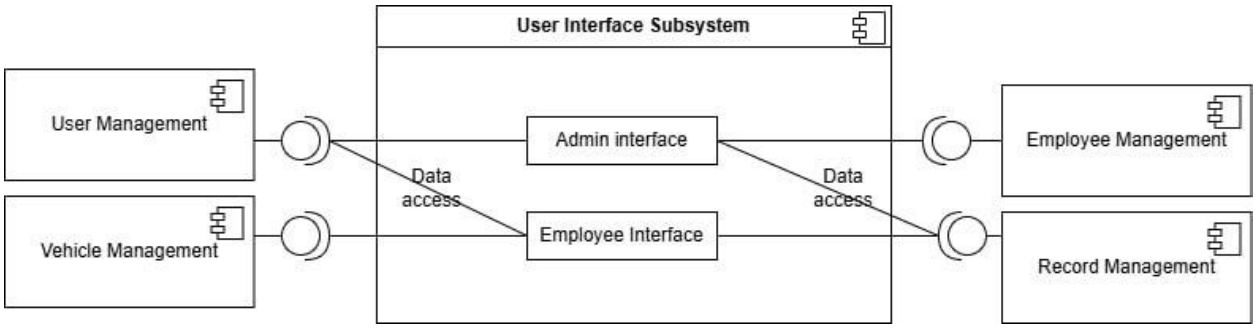
Providing Interfaces:

The record management subsystem provides services to the user interface subsystem. It allows the user interface subsystem to render the required data and GUI properly.

Requiring Interfaces:

The record management subsystem requires services from the file system to retrieve and display vehicle information successfully. These services are required through the *loadParkingstatus(), showParkingstatus(), showDailyReport() and createDailyReport()* interfaces. It also requires services from the user management subsystem to authenticate employees performing vehicle management operations. These services are provided through the *login()* and *logout()* interfaces of the user management subsystem.

5. **User Interface Subsystem Services and Architecture**



- **User Interface Subsystem Services**

The user interface subsystem provides employees and administrators with interactive layout and interfaces to efficiently interact with the system and perform operations. The services offered by this subsystem are directed to the user of the system as it enables effective rendering. While this

subsystem does not provide service to other subsystems, it requires *data access* service from all other subsystems except the file system.

- **Coupling with other subsystems**

The user interface subsystem is coupled with other subsystems. These subsystems are Employee Management, User Management, Record Management and Vehicle Management subsystems.

- **Interfaces**

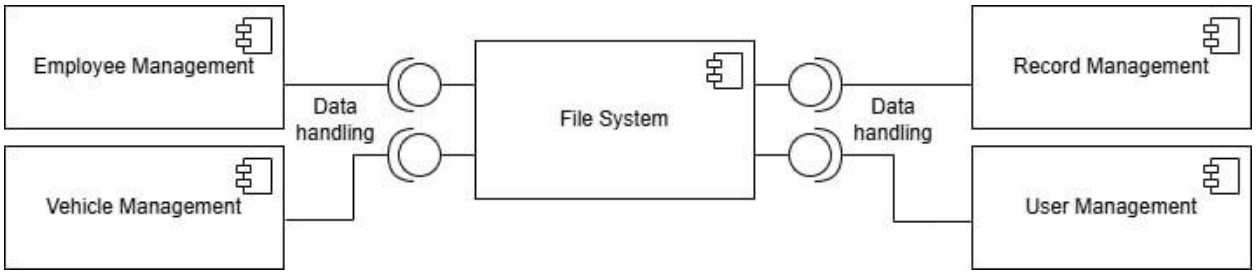
Providing Interfaces:

The user interface subsystem does not provide services to other subsystems. It renders the required data and GUI properly.

Requiring Interfaces:

The user interface subsystem requires services from the employee management subsystem, vehicle management subsystem, record management subsystem and user management subsystem. It uses event driven communications and method calls to render data to the GUI.

6. **File System Services and Architecture**



- **File Subsystem Services**

The file subsystem provides employees and administrators with up-to-date data to monitor the parking lot and make efficient decisions. This subsystem provides file registration, saving, update, deletion and retrieval. It helps to efficiently organize information in the parking lot management system.

- **Coupling with other subsystems**

The file subsystem has interface-based coupling with other subsystems. These subsystems are Employee Management, User Management, Record Management and Vehicle Management subsystems.

- **Interfaces**

Providing Interfaces:

The file subsystem provides robust file management services to other subsystems. It enables other subsystems to effectively manage their data during various operations.

Requiring Interfaces:

The file subsystem does not require services from other subsystems.

5. Object Design

5.1. Components

Components and Classes

Entity Objects

- ➔ Customer
  - ◆ Attributes:
    - username: String
    - password: String
  - ◆ Methods:
    - login(username, password):String
    - logout(): String
    - resetPassword(username, password): String
- ➔ Employee
  - ◆ Attributes:
    - name: String
    - gender: String
    - username: String
    - password: String

- Methods:
  - addEmployee(employee): void
  - removeEmployee(username): void
  - getAllEmployee(): List<Employee>
  - displayEmployee(): void

Boundary Objects

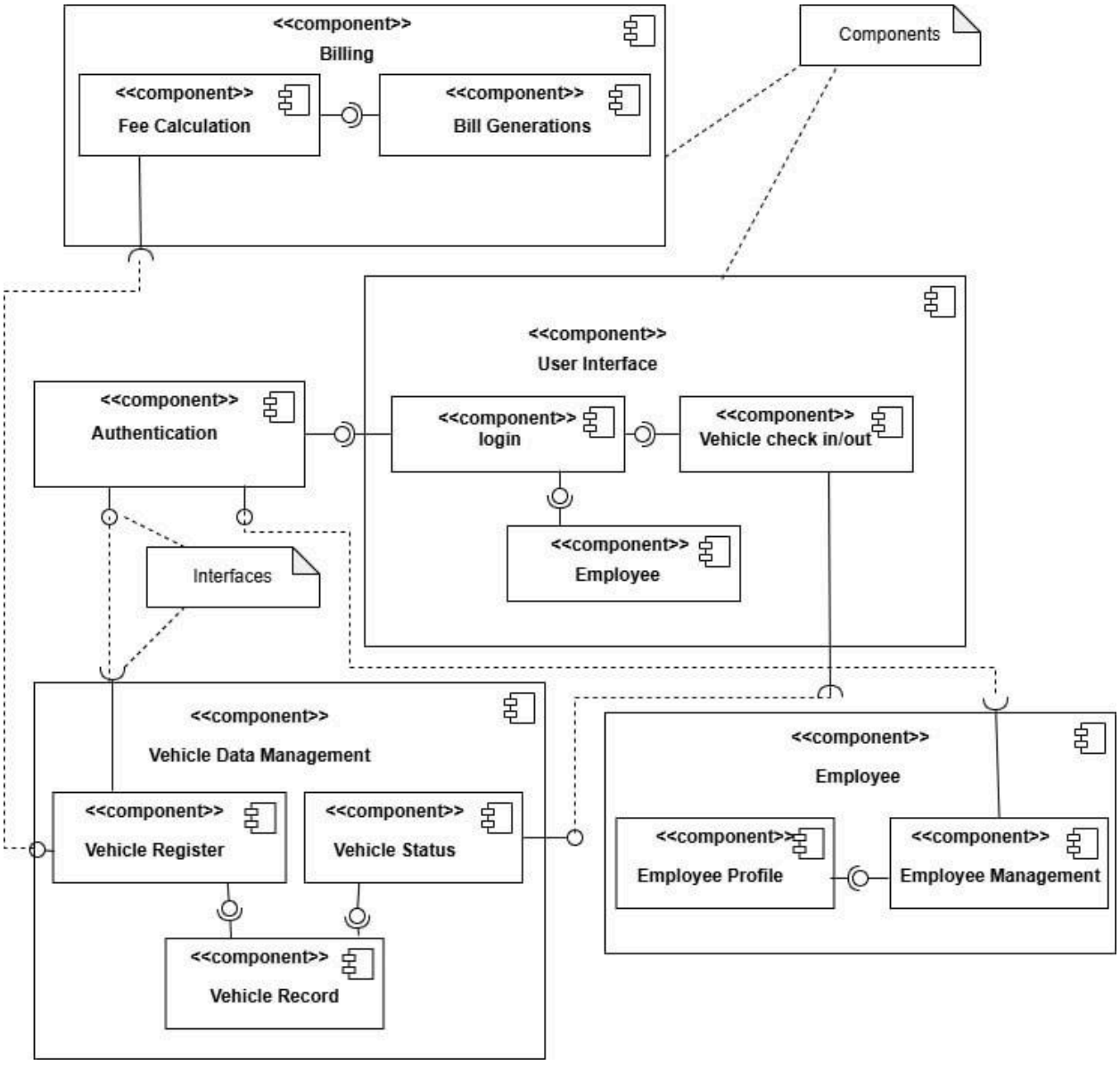
- Admin Interface
  - Interface for controlling the overall parking lot management system
- Employee Interface
  - Interface for performing their daily tasks related to vehicle check-in, check-out, and managing customer interactions

Classes

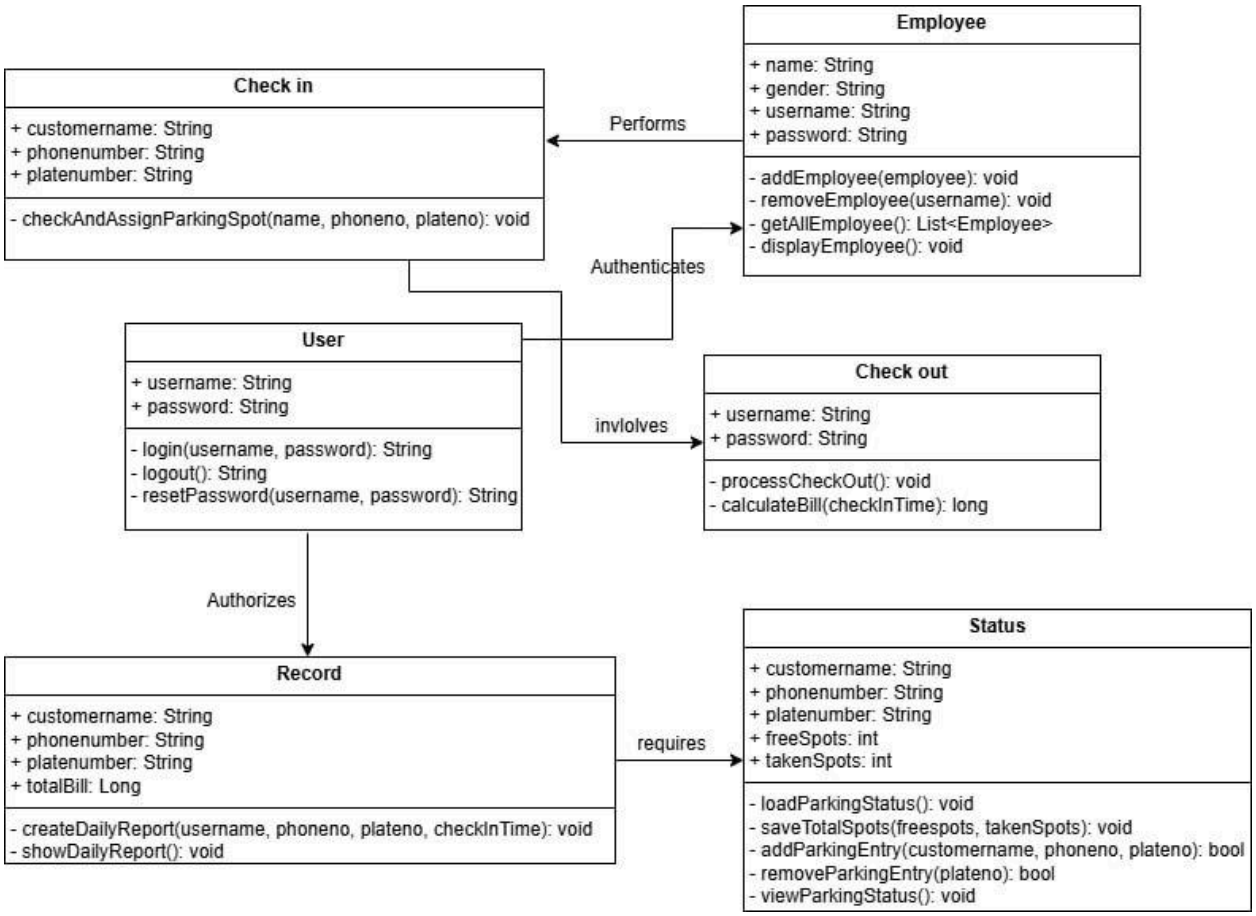
- Check In
  - Attributes
    - customerName: String
    - phoneNumber: String
    - plateNumber: String
  - Methods
    - checkAndAssignParkingSpot(name, phoneNumber, plateNumber): void
- Check Out
  - Attributes
    - userName: String
    - password: String
  - Methods
    - processCheckOut(): void
    - calculateBill(checkInTime): long

Component Diagram

The components used by the parking lot management system are shown in the diagram below:



5.2. Class diagram



6. Glossary

Subsystem: A unit or device that is part of a larger system.

Component: Is an identifiable part of a larger program

Parking Slot: Individual spaces within the parking area allocated for vehicles.

Parking lot: An area/space where cars are parked.

Fee Structure: The pricing model used to calculate charges for parking based on factors like duration or type of vehicle.

Availability Tracking: Real-time monitoring of vacant and occupied parking slots.

Persistent data: denotes information that is infrequently accessed and not likely to be modified.

Customer Registration: A feature allowing users to register their vehicles and details for parking, especially for repeat or subscription-based customers.

Payment System: A mechanism for collecting parking fees, supporting cash or digital transactions.

Access Control: The system managing vehicle entry and exit, often integrated with barriers or gates.

Reports and Analytics: Data summaries or dashboards offering insights into parking usage patterns, revenue, and overall efficiency.