**PantherLot Interactive**

**Course:** CEN4010

**Section:** U01

**Date:** July 12, 2011

**Professor:** Peter Clarke

**EXECUTIVE SUMMARY**

Every year, the FIU community grows larger with the influx of new coming freshmen students that get accepted into the university. This causes distress for both current members and new comers alike in terms of finding parking spaces on campus. For this reason we are developing PantherLot Interactive, a garage parking system that will reduce the time for the members of the FIU community to find an available parking spot.

Detailed in this document, are the purpose and the scope of our system, the functional and non-functional requirements, the software architecture of the PantherLot Interactive System, the architectural patterns used, the description of all of the subsystems that are to be implemented, the detailed design of all the classes and the design patterns used.

**TABLE OF CONTENTS**

Executive Summary i

1. Introduction 1

1.1 Purpose of System 1

1.2 Functional and Non-functional Requirements 1

1.2.1 Functional Requirements 1

1.2.2 Non-functional Requirements 2

1.3 Design methodology used 3

1.4 Definitions, Acronyms, and Abbreviations 4

1.5 Overview of Document 4

2. Proposed Software Architecture 5

2.1 Overview 6

2.2 System Decomposition 7

2.3 Hardware and Software Mapping 8

2.4 Persistent Data Management 10

3. Detailed Design 11

3.1 Overview 11

3.2 Object Interaction 12

3.3 Detailed Class Design 16

4. Glossary 17

5. Appendix 19

5.1 Appendix A 19

5.2 Appendix B 20

5.3 Appendix C 32

5.4 Appendix D 33

5.5 Appendix E 34

**1. INTRODUCTION**

This chapter gives a brief introduction to PantherLot Interactive. In this section, the purpose and scope of the system is defined as well as any definitions, acronyms and abbreviations that will be used throughout the document and will conclude with an overview of the document.

**1.1 Purpose of system**

Parking has been a hassle ever since FIU started expanding every year with over 4,000 students coming into our facilities every semester. So far, in order to obtain a parking spot in the middle of the day has resulted into people hunting down for parking spots all over FIU. Sometimes people have to spend 30 minutes or more just driving around looking for someone to leave so you can take their spot. The purpose of PantherLot Interactive is to alleviate the time spent from students, faculty and staff members trying to find the best possible parking spot at any given time, with the aid of scanners and sensors letting people know even before they come in the parking lot whether or not they can be accommodated.

* 1. **Functional and Non-functional Requirements**

In this section, we describe the functional and non-functional requirements of the use cases we are going to implement.

### 1.2.1 Functional Requirements

* The system shall retrieve the user’s information from ID

**Use cases**: Scan ID (PLI002)

**Constraint:** The User must be in the FIU DB, ID must be scan able.

* The system shall check for a free spot and then assign it to the user

**Use cases**: Parking Spot assigned Display (PLI010), Display Directions (PLI011)

**Constraint:** Parking lot is full or specific type is not available.

* The system shall inform both user and security officer when parked in the wrong spot

**Use cases**: Wrong Parking User Notification (PLI014), Wrong Parking Security Notification (PLIS06).

**Constraint:** The system is unable to send notifications at the time.

* The system shall notify the Security Officer of an ID already in use

**Use cases**: Stolen ID Security Alert (PLIS05)

**Constraint:** The system is unable to send notifications at the time.

See Appendix B for functional requirements of all use cases PLI002, PLI010, PLI011, PLI014, PLIS05 and PLIS06.

### Nonfunctional Requirements

**Usability:**

On average, most of the system takes should be understood by anyone using it, so that no previous training is required.

**Reliability:**

On average, the system should not fail more than 5% within a year of use.

**Performance:**

Requests should be handled in less than 5 seconds, if no other requests exist. Response time may increase due to increase in active users.

**Supportability:**

The system should display instructions, messages and alerts so that they are precise and easy to understand whether it’s the user or the security officer that receives them.

**Implementation:**

All system menus, options and user interactions must be implemented in Java with an Interactive GUI.

See Appendix B for non-functional requirements of all use cases PLI002, PLI010, PLI011, PLI014, PLIS05 and PLIS06.

**1.3 Design Methodology**

For this project, we are using the Unified Software Development Process (USDP) to develop this software. We believe that the USDP model is a good selection because of its iterative and incremental process nature. Because of this, you can always add functionality as you go into the process. Additionally, USDP is driven by use-cases along the path of requirements, implementation, test and deployment.

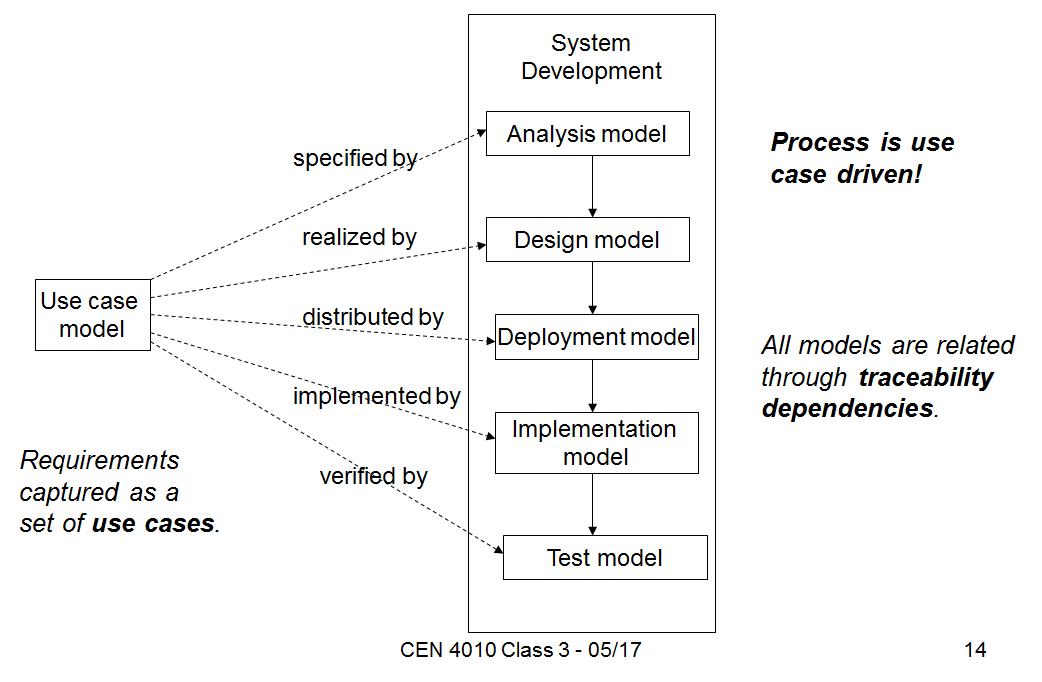


Figure 1 Unified Software Development Process

We used the use case diagrams, sequence diagrams, deployment diagrams and class diagram in the Unified Modeling Language (UML) to represent the design. Because of USDP being use case driven we can road map our process. As a result, this approach helps us to validate our test cases to guarantee results.

**1.4 Definitions, Acronyms and Abbreviations**

In this section, we briefly describe all the acronyms and abbreviations by providing what they stand for.

FIU – Florida International University.

FIU DB – FIU’s Student, Faculty and Staff Database.

GUI – Graphical User Interface

HTML – Hypertext Markup Language.

PLI – PantherLot Interactive.

UI – User Interface.

UML – Unified Modeling Language.

URL – Uniform Resource Locator.

**1.5 Overview of document**

The remainder of the document contains detailed information about the development of PLI and is separated into six chapters. Chapter 2 describes the proposed system architecture. Chapter 3 outlines our system design showing the minimal class diagram, object interaction and detailed class design, project plan and tasks involved in the development of PLI. Chapter 4 is the glossary of terms used in this document. Chapter 5 consists of the appendices for this document. Appendix A contains the use case diagram. Appendix B contains the use cases with nonfunctional requirements. Appendix C contains the detailed class diagrams. Appendix D contains class interfaces for the subsystems being implemented and Appendix E contains the diary of meetings and tasks.

**2. PROPOSED SOFTWARE ARCHITECTURE**

In this chapter, the proposed software architecture for PantherLot Interactive is shown and explained. First, we give an overview showing the package diagram, and an overview of each subsystem. Below we will explain the system architecture and design pattern chosen for our system. Also, we will list the Subsystem Decomposition and describe the major subsystems and their related use cases. The Hardware and Software Mappings are also described and we end the section with the Persistent Data Management, which identifies the security requirements for the system and the data that needs to be stored.

As explained above, our system will allow any user who wants to make any use of the facilities to quickly and easily find a parking space with no setbacks. Nowadays, we can may similar systems in use for controlling parking garages. One of the current systems in use that is similar to ours is the Car Parking Management System created by Work Space Manager. This system was designed to be implemented on workplace stations only.

There are some similarities to our system, the first being that both systems will have total control of every single parking space in the parking garage, and have a graphical representation of the parking garage. Also, both system use scanners to identified users coming into the parking garage. The car park access control of our system relies on the information obtained from user’s decal while the CPMS use user’s car plate. The CPMS include a convenient feature that allows the system to release a parking space as soon as a guest checks out from the desk. However, our system will have sensors in each individual parking space making our system very accountable in time.

In addition, the car parking management system has the option for some users, employees, to reserve a parking space for quests. On the contrary, our system will be based on a queue mechanism making it fair for the users.

**2.1 Overview**

In this section, we give an overview of the architectural patterns that will be used in implementing the system. Here we also have divided the system into its corresponding subsystems and the classes contained within each subsystem.

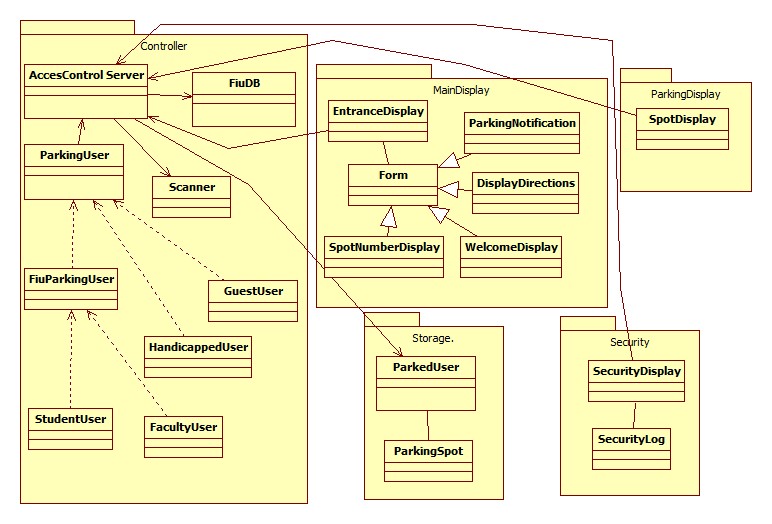
****

Figure Error! No text of specified style in document..1 PantherLot Interactive Architecture

Description: This is the package diagram for ParkingLot Interactive system. Also shows the Client Server and Repository Architecture.

The subsystems of the system are Parking Display Subsystem, Main Display Subsystem, Security Subsystem, Controller Subsystem and Storage Subsystem.

**Client Server and Repository Architecture:** Our project uses the client server and repository architecture. These architectures are made up of different operations: The repository architecture access’s and modifies the data from the central repository which contains a single data structure; the subsystems interact through this central data structure. In the Client/Server architecture the request for service is done via a remote procedure call mechanism or common object broker; control flow in the clients and the servers is independent except for synchronization to manage requests or to receive results. The client is the requester and the server the provider. PantherLot Interactive is essentially a system that uses its subsystems to access the methods residing in the Repository. The database interface allows us to find the customer’s information successfully in order to understand what type of access is requested or to acknowledge if customer is a student or faculty member. In our system, the users will interact with the system through our main display. They will get their ID scanned and the functions in the repository will handle the remaining of the interaction until the user receives the confirmation or the parking spot data. The data storage layer, or data layer, contains the methods for accessing the data for the client in Client/Server architecture; the server possesses all the information regarding faculty member and students. The repository contains the logic for the entire system. This layer takes data from the subsystem main display or sensor and passes it to the repository for function acknowledgment.

**2.2 Subsystem Decomposition**

In this section, go into detail of the specific description of each subsystem and how each one of them is related to the use cases that are to be implemented.

**Parking Display Subsystem:** This subsystem is responsible for handling the parking user needs. It is in charge of letting a user know if the system has available parking spots. It will also handle if a user parks incorrectly and sends a notification. The use cases associated with this subsystem are: Wrong Parking User Notification (PLI-014) from Appendix B.

**Main Display Subsystem:** This subsystem contains the functionality for displaying the available parking spots and scanning the user’s ID; here the users will know where to find the information regarding the assigned parking spots, as well as directions to it. The use cases associated with this subsystem are: Scanning ID (PLI-002), Parking Spot Assign Display (PLI-010), Display Directions (PLI-011) from Appendix B.

**Security Display Subsystem:** This subsystem involves displaying the messages to the security officer, triggered by the user when they request help from the security officer. In this display the security officer must confirm the message and read the user’s concern or query. The use cases associated with this subsystem are: Stolen ID Security Alert (PLI-S05), Wrong Parking Security Notification (PLI-S06) from Appendix B.

**Controller Subsystem:** This subsystem takes care of searching for a specific match based on the users ID against the server in order to find the correct parking spot. Once it retrieves the information a parking spot TYPE can be assigned (student or faculty). The use cases associated with this subsystem are: Stolen ID Security Alert (PLI-S05), Wrong Parking Security Notification (PLI-S06), Scanning ID (PLI-002), Parking Spot Assign Display (PLI-010), Display Directions (PLI-011), Wrong Parking User Notification (PLI-014) from Appendix B.

**Storage Subsystem:** This subsystem takes care or holds the information pertaining as to how many parking spots are available or used, also knows what TYPE of parking spots are available or unavailable distinguishes between student, faculty, guess and others. The use cases associated with this subsystem are: Parking Spot Assign Display (PLI-010), Display Directions (PLI-011 from Appendix B.

**2.3 Hardware and Software Mapping**

In this section, we provide information on how our system will link the hardware and software together in order to get information from the user and process it.

**Sensor/Scanner:** Device that is constantly scanning the parking spots in the garage system, this information than is stored in our storage database for retrieval in our application server.

**Application Server:** The application server is where the controller and almost all of our methods are physically stored. The entire PantherLot Interactive system is here.

**Database Server:** The database server is where all the data is stored the information in this server pertains to the parking spot information (available/unavailable parking).



Figure Error! No text of specified style in document..2 PantherLot Interactive Deployment Diagram

Description: This is the deployment diagram for the PantherLot Interactive system. It shows our Repository and Client/Server architecture.

**2.4 Persistent Data Management**

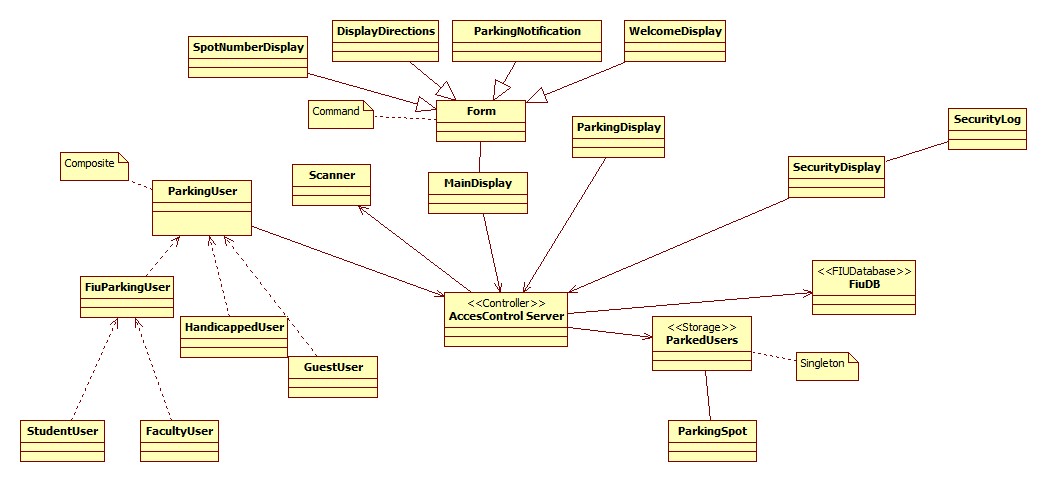
The persistent data stored within PantherLot Interactive consists of general parking spot information, student parking information, and faculty parking information. Confidential data, such as student information and faculty member’s information are not stored within PantherLot interactive. Access to our system is not password restricted since our system database just hold information about parking status, the information pertaining to students and faculty members are retrieved from the school’s database which only allows us to see the information pertaining to parking garage access data. Therefore, security measures pertaining to our data are not needed.

The database consists of information relating to: faculty parking spots, guest parking spots, student parking spots, and their availability. Each of these items is aggregated into their respective subsystems. The subsystems within PantherLot Interactive are: Parking Display Subsystem, Main Display Subsystem, Security Subsystem, Controller Subsystem and Storage Subsystem. Information related to parking spot availability is contained in the storage subsystem. User parking spot information display is contained in the Parking Display subsystem. The Controller subsystem contains the scanning device to get the user’s ID information. The Security subsystem contains information related to the user’s requests for assistance or any type of erroneous issues in the PantherLot system. The Main Display subsystem contains the information pertaining to the assigned parking spot, like directions and the parking spot number.

**3. DETAILED DESIGN**

This chapter discusses the schedules of the development plan. Here the roles of each team member are decided and the tasks each member is required to do. The hardware and software requirements necessary to implement and maintain the system are also stated. Also, the three milestones and deliverables produced for each phase of the project are defined.

**3.1 Overview**

In this section, we will present the different design patterns implemented in our system. We will use the singleton pattern to access data from out data bases. The composite pattern will be present when using the inheritance in the parking user at FIU where hierarchy will be present. In addition, the command pattern is used when displaying the different forms. 

**Fig 3.1.1 Minimal Class Diagram**

Description: This is the minimal class diagram for the PantherLot Interactive System containing the design patterns to be implemented.

As shown by the figure above, our system will implement the Command, Singleton and Composite design patterns.

**Singleton Pattern:** Our system uses the Singleton design pattern by creating a class called a static class called ParkedUsers which contains all of the parking spots including the ones that have user parked in them. There is only one global instance of the class so that all client displays can access the information stored in it.

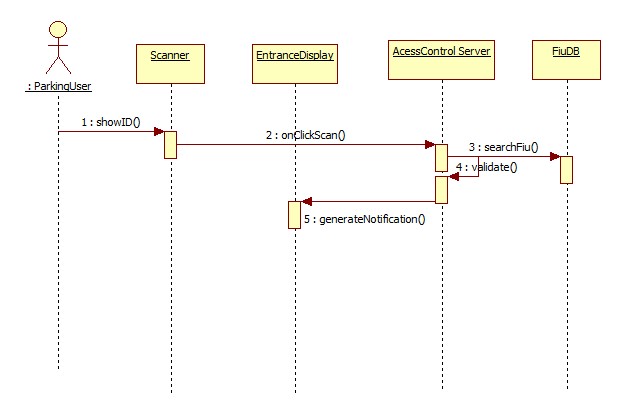
**Command Pattern:** Our system also uses the Command design pattern; we are implementing this pattern as a way to generalize all of the different display screens for the MainDisplay by using a common interface called Form. The classes that implement this interface are SpotNumberDisplay, DisplayDirections, ParkingNotification and WelcomeDisplay.

**Composite Pattern:** The PantherLot Interactive system uses the composite pattern as well; we are implementing this pattern as a way to establish a parking user hierarchy with ParkingUser as the super class. The reason for doing this is so that if there are parking garage with new different type of users, they can be added to the system without having to modify any of the code from the original implementation. The leaf nodes we are implementing in our hierarchy are StudentUser, FacultyUser, HandicappedUser and GuestUser. We also have one branch node, FiuParkingUser, which serves as a generalization for student and faculty parking users.

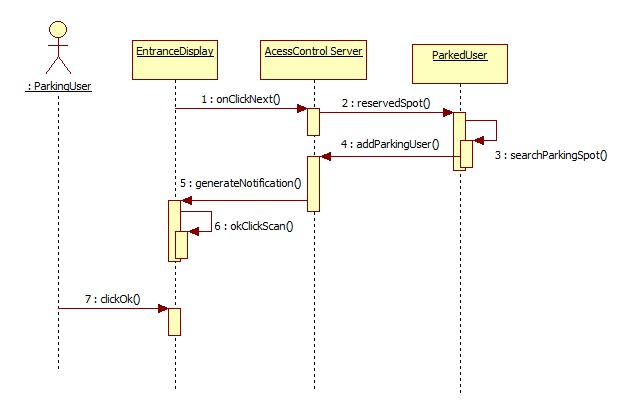
Some of our implementation classes are not related to any of the patterns so here is a brief description of the ones that have not been mentioned yet. The Scanner class is a simulation of the hardware scanner that retrieves the ID of the user from the parking decal. The FiuDB class is a simulation class that contains information of the FIU members. The ParkingDisplay class is the GUI for the display screen located in front of each parking spot. The SecurityDisplay is the GUI for the computer used by the security officer. The SecurityLog class is responsible for storing all security notifications in a log file. Lastly, we have the AccessControlServer which is where all communication happens between the display clients. This is also were the logic of the program is resides.

**3.2 Object Interaction**

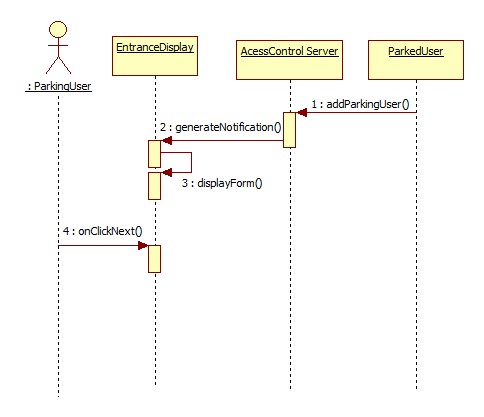
In the section below, we will introduce the different interactions in our system using sequence diagrams. These will be the use cases that we will be implemented in our system.



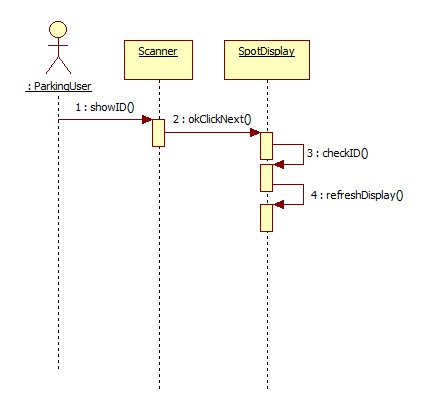
**Fig 3.2.1 Scan ID Sequence Diagram**

****

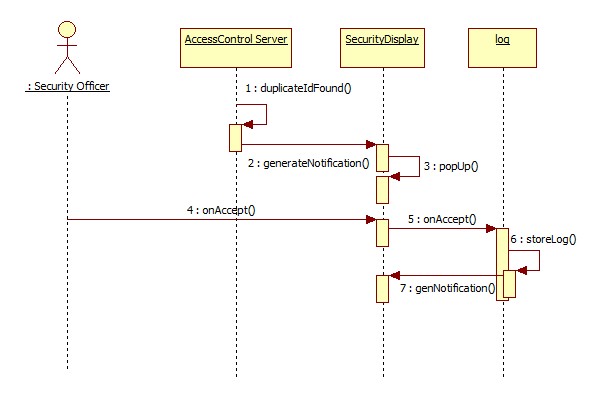
**Fig 3.2.2 Parking Spot Assigned Display Sequence Diagram**

****

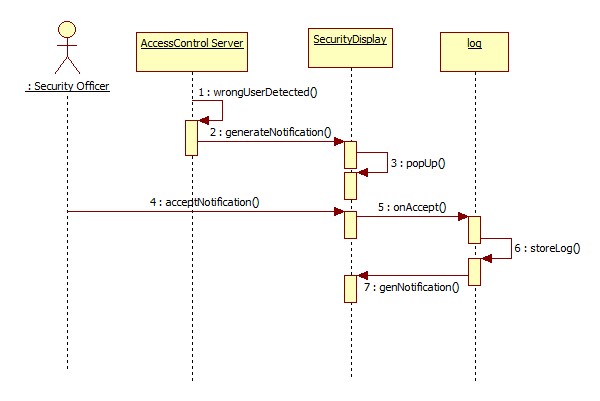
**Fig 3.2.3 Display Directions Sequence Diagram**

****

**Fig 3.2.4 Wrong Parking User Notification Sequence Diagram**



**Fig 3.2.5 Stolen ID Security Alert Sequence Diagram**



**Fig 3.2.6 Wrong Parking Security Notification Sequence Diagram**

**3.3 Detailed Class Design**

In this section, we will explain the purpose of each class; for a more throughout description and documentation of the classes, see Appendix D. We will also use the object constraint language to formally define the constraints for the main logic controller server class.

**Controller Package Classes:** For a detailed class diagram of the controller subsystem, see Appendix C figure 2.

AccessControlServer: This is the server class that controls all data access and flow. It is also responsible of creating all the parking user objects once it searches them in the FIU Database.

FacultyUser: This is an entity class to create a faculty user object who is a parking user.

FiuParkingUser: This is an abstract class that defines the parking users that are members of the FIU community.

GuestUser: This is an entity class to create a guest user object who is a parking user.

HandicappedUser: This is an entity class to create a handicapped user object who is a parking user.

ParkingUser: abstract superclass of the parking user’s heirarchy, all parking entities object are of this type.

Scanner: Simulation of the scanner hardware that scans the parking user ID from the decal.

StudentUser: This is an entity class to create a student user object who is a parking user.

**MainDisplay Package Classes:** For a detailed class diagram of the controller subsystem, see Appendix C figure 3.

DisplayDirections: GUI that displays the directions to the parking spot to the parking user.

EntranceDisplay: main class that runs the GUI for the display at the entrance of the garage

Form: Interface used to implement the command pattern by generalizing all the different types of display.

ParkingNotification: GUI that displays if the system was able to find a parking spot for the parking user.

SpotNumberDisplay: GUI that displays the parking spot number assigned to the parking user.

WelcomeDisplay: GUI that displayed when the parking user first approaches the display screen.

**ParkingDisplay Package Classes:** For a detailed class diagram of the controller subsystem, see Appendix C figure 4.

SpotDisplay: main class that runs the GUI for the parking spot screen.

**SecurityDisplay Package Classes:** For a detailed class diagram of the controller subsystem, see Appendix C figure 5.

SecurityDisplay: main class that runs the GUI for the security officer’s computer.

SecurityLog: This is the class that stores all notifications in a log file.

**Storage Package Classes:** For a detailed class diagram of the controller subsystem, see Appendix C figure 6.

ParkedUsers: This class is used to assign and remove user from parking spots. It also stores all the parking spots of the garage according to their type.

ParkingSpot: This is where the object containing all the information of each spot is created.

**Access Control Server Class OCL:**

**Context:** *AccessControlServer* **Inv:** garageSpots.size() > 0

**Context:** *AccessControlServer*::*reserveSpot()* **Pre:** *user.toString() == parkingspot.getType()*

**Context:** *AccessControlServer*::*WrongUserDetected()*

**Pre:** *user.getID() != parkingSpot.getUser()*.*getID()*

**4. GLOSSARY**

**Actor** – Anyone that can use, or misuse, the system.

**Class Diagram**\* – A model representing the different classes within a software system and how they interact with each other.

**Database** – A collection of related data.

**Database Management System** – A software package to facilitate the creation and maintenance of a computerized database.

**Handicap –** Handicap is one of the user groups that interact to the system a type. A Handicap is a user that was issued a handicap tag by an authorized doctor.

**Java** –is a programming language and it derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities.

**Java GUI** –Graphical User Interfaces in Java. A Java GUI application uses the standard Java components GUI component set, Swing, and is deployed to the desktop.

**FIU DB** – Database Management System of FIU.

**Guest** – Guest is one of the user groups that interact with the system. Alternatively, a guest is a type of user that does not have a FIU decal. This type of user is usually considered as a visitor.

**PantherLot Interactive –** Java based system that will allow student, faculty or staff member to look for a parking space as they enter the Universities’ busy parking lots making it easy and dynamic to find a space within a couple minutes with options according to their credentials, which is hosted by their University.

**PLI –** PantherLot Interactive

**Parking Garage** – A building design to bring parking spaces to people.

**Parking User** – see User.

**Post Condition**\* – A status that must be true after an operation is invoked.

**Precondition**\* – A status that must be true before an operation is invoked.

**Scanner –** a hardware device that is programmed to scan the information from FIU decals.

**Security Officer –** person in charge of the administration of the ParkingLot Interactive system.

**Sequence Diagram**\* – A model representing the different objects and/or subsystems of a software project and how they relate to each other during different operations for a given use case.

**Server** – A computer that provides client stations with access to files as a shared resource to a computer network.

**Student** – Student is one of the user groups that interact with the system. They are the type of user that possesses a student parking decal.

**System** – Underlying reality.

**UI** – see User Interface

**UML** – See Unified Modeling Language

**Unauthorized User** – A user that uses the system in unintended ways, usually this user doesn’t follow the systems instructions and park wherever he desired.

**Unified Modeling Language**\* – A standard set of notations for representing models

**User** – A user is any person, Faculty, Staff, Student or Guest that might be handicapped that interacts with the system. Alternatively, a user may or not have a decal depending on user type.

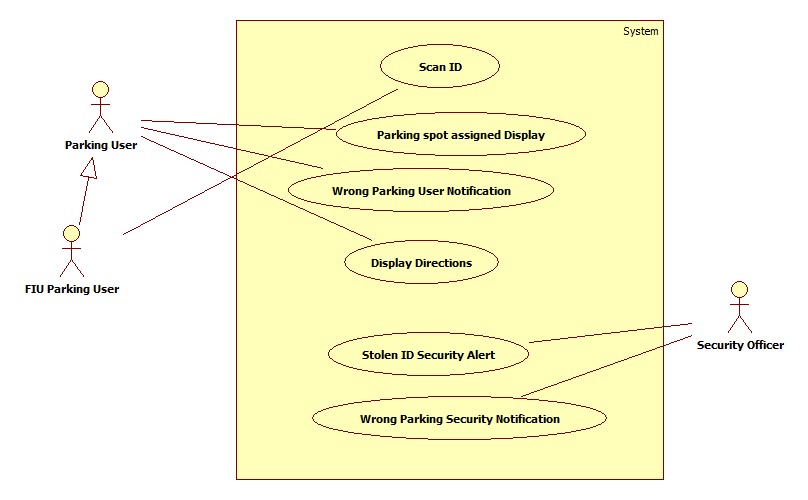
**Use Case**\* - A general sequence of events that defines all possible actions between one or many actors and the system for a given piece of functionality.

**User Interface** – the way through which a user interacts with the computer system.

**User’s ID –** user identification that is obtained from the user’s car decal and this information is hosted by FIU database.

**5. APPENDIX**

**5.1 Appendix A – Use case diagram**



**Fig A.1 PantherLot Interactive Use Case Diagram**

Description: This diagram shows the overall interaction of the different user types with the PantherLot Interactive System.

**5.2 Appendix B – Use Cases**

**Use Case ID:** PLI002 – Scan ID.

**Use Case Level:** High-level

**Details:**

* **Actor:** User.
* **Pre-conditions:**
  1. Scanning device is ready for the user’s id to be scanned.
  2. The last reading the device performed has been cleared.
* **Description:**

1. Use case begins when the user drives by and system scans the ID.
2. The system gets the users information from the ID.
3. Use case ends when the system gets the ID information and saves it.

* **Post-conditions:**
  1. System retrieves the information from the Parking User’s ID it prepares search for type.
  2. The search system begins to find out what type of parking is required for the user.
* **Alternative Courses of Action**
  1. If there is ID error he can park as a guest at a rate

**Exceptions:**

1. The scanning mechanism is damage.
2. The ID suffers from normal wear and tear making impossible to scan.

**Related Use Cases:**

Guest charging mechanism (PLI006),

---------------------------------------------------------------------------------------------------------------------

**Decision Support**

**Frequency:** On average 2000 scans will be produced daily.

**Criticality:** High. The system needs the information of the user’s ID to assign parking.

**Risk:** Medium.  If the scanning system goes down the user’s information could not be collected.

---------------------------------------------------------------------------------------------------------------------

**Constraints:**

* Usability: No previous Training Time.
* Reliability: Mean time to Failure – 5% failure for every twenty four hours of operation.
* Performance: On average, information should be scanned and saved within 4 seconds.
* Supportability: The scanning device be compatible with all types of IDs.
* Implementation: Using java to deliver the notification to the security officer.

---------------------------------------------------------------------------------------------------------------------

**Modification History**

**Owner:** Team 5

**Initiation date:** 05/14/2011

**Date last modified:** 05/22/2011

**Use Case ID:** PLI010– Parking Spot Assigned Display.

**Use Case Level:** High-Level

**Details:**

* **Actor:** Parking User
* **Pre-conditions:** 
  1. User understands or is acquainted with the system in order to proceed.
  2. The system has recognized the user type and has reserved his spot.
* **Description:**

1. Once the system has identifies there is a user and has identified his type.

* 1. The user can see in the garage display which spot has been assigned for him.
  2. Use case ends when the user presses the next button.
* **Post-conditions:** 
  + 1. The System will proceed to display the directions to park on the spot assigned to the user.

**Alternative Courses of Action**

* + - 1. Once the user has been informed of the parking space assigned he may use a Cancel button and go to a different garage.

**Exceptions:**

* + - * 1. The System informs parking garage is full, therefore will not assign any parking space.

**Related Use Cases:**

None

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average 70 cars will arrive to the parking garage per hour.

**Criticality:** High. It will display the parking a space assigned for each user that comes into the garage.

**Risk: Medium.**Displaying the information in the screen could be lost by power outages.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability**:** No previous training required
* Reliability: Mean time to Failure – 5% failures for a year of use.
* Performance**:** The system has to be able to display the parking space within 5 seconds.
* Supportability:The displayed image and information should be clear and easy to follow.
* Implementation:Using java with GUI to implement.

**---------------------------------------------------------------------------------------------------------------------**

**Modification History**

**Owner:** Team 5

**Initiation date:** 05/14/2011

**Date last modified:** 05/14/2011

**Use Case ID:** PLI011 – Display Directions

**Use Case Level:** High-level

**Details:**

* **Actor:** Parking User.
* **Pre-conditions:**
  1. User understands or is acquainted with the system in order to proceed.
  2. The system has assigned and displayed the parking space.
* **Description:**

1. The use case begins when the System has displayed the parking space assigned.
2. The user can see in the garage display to get to the parking space assigned.
3. Use case ends when the user has pressed the next button.

* **Post-conditions:**

The System will proceed to the main menu and be ready for the next user..

**Alternative Courses of Action**

Once the user has been given directions to the parking space assigned he may use a Cancel Button and go to a different garage.

**Exceptions:**

None.

**Related Use Cases:**

None.

------------------------------------------------------------------------------------------------------------

**Decision Support**

**Frequency:** On average, 70 cars will arrive to the parking garage per hour.

**Criticality:** High. It will display directions to get to the parking a space assigned to the users.

**Risk:** High.

------------------------------------------------------------------------------------------------------------

**Constraints:**

* + - Usability: No previous Training Time.
    - Reliability: Mean time to Failure – 5% failures for one year of use.
* Performance: On average, request takes about 5 seconds.
* Supportability: The screen must display data clearly enough for the user to follow it.
* Implementation: Using java with GUI to implement.

------------------------------------------------------------------------------------------------------------

**Modification History**

**Owner:** Team 5

**Initiation date:** 05/14/2011

**Date last modified:** 05/14/2011

**Use Case ID:** PLI014–Wrong parking user notification.

**Use Case level:** High-level

**Details:**

* **Actor:** System User.
* **Pre-conditions:** 
  1. The user is acquainted with the system.
  2. The system has confirmed user parked on the wrong spot.
* **Description:**

1. The system will notify user that has parked on the wrong spot by displaying a message.

2. When system has confirmed the user has parked on the wrong spot.

* 1. The system will Display “Wrong Spot” and the number of the correct parking spot assigned to that user.
  2. Use case ends when the message has been displayed.
* **Post-conditions:** 
  + 1. The System will keep message displayed until actions is taken by the user or administrator.

**Alternative Courses of Action**

None

**Exceptions:**

None.

**Related Use Cases:**

None

---------------------------------------------------------------------------------------------------------------------

**Decision Support**

**Frequency:** On average 10 users will park at a wrong spot per hour.

**Criticality:** Medium. It will notify the user when a user has parked at a wrong spot

**Risk:** Medium. Notification will not be generated because of power outage.

---------------------------------------------------------------------------------------------------------------------

**Constraints:**

* Usability: No previous time required
* Reliability:Mean time to Failure – 5% failures for a year of use.
* Performance:The system has to be able to generate the message within 2 seconds
* Supportability: The message displayed should be clear and precise in order for the user to be able to understand.
* Implementation: Using java GUI to implement.

---------------------------------------------------------------------------------------------------------------------

**Modification History**

**Owner:** Team 5

**Initiation date:** 05/14/2011

**Date last modified:** 05/14/2011

**Use Case ID:** PLIS05 – Stolen ID Security Alert.

**Use Case Level:** High-level

**Details:**

* **Actors:** security officer, parking user.
* **Pre-conditions:**

1. An unauthorized user is already parked with the same ID as the parking user.

* **Description:**

1. Use case begins when the system notified the parking user that there is another car parked with the same ID.
2. The system displays an option to notify the security officer about a stolen ID.
3. The parking user selects on the screen to notify security.
4. The system then generates and sends a notification to the security officer.
5. Use case ends when the security officer receives and acknowledges the notification.

* **Post-conditions:**

1. The number of notifications for the security has increased by one.
2. The system stores the event on the log.

**Alternative Courses of Action:**

None.

**Exceptions:**

The security officer does not receive the notification.

**Related Uses Case:**

Duplicate ID User Alert (PLI015).

------------------------------------------------------------------------------------------------------------

**Decision Support**

**Frequency:** On average, 2 abuses of fire alarm per month are made by the user.

**Criticality:** Medium. Allows the security to get notified and find the car with the stolen ID.

**Risk:** High.

------------------------------------------------------------------------------------------------------------

**Constraints:**

* + - Usability: No previous Training Time.
    - Reliability: Mean time to Failure – 1% failures for one day of use.
* Performance: On average, request takes about 5 seconds.
* Supportability: The screen must display data clearly enough for the user to read it.
* Security: Security officer needs to find out which car is using an unauthorized decal and allow the parking user to park on a spot.

------------------------------------------------------------------------------------------------------------

**Modification History:**

**Owner:** Team 5

**Initiation date:** 05/15/2011

**Date last modified:** 06/11/2011

**Use Case ID:** PLIS06–Wrong Parking Security Notification.

**Use Case Level:** High-Level.

**Details:**

* **Actor:** Security Officer.
* **Pre-conditions:** 
  1. The system has acknowledged user parked on the wrong spot.
* **Description:**
  1. Use case starts when system has confirmed the used has parked on the wrong spot.
  2. The system will generate a message with the user ID, date, time, and parking space where user parked.
  3. The System will notify user has parked on the Wrong spot
  4. The system will send notification to Security Officer.
  5. Security receives the message and press de accept button.
  6. Use case ends when security either reserves the spot or frees the spot.
* **Post-conditions:** 
  1. The system will save the log with the action taken by security.

**Alternative Courses of Action**

None

**Exceptions:**

None.

**Related Use Cases:**

None.

---------------------------------------------------------------------------------------------------------------------

**Decision Support**

**Frequency:** On average 10 users will park at a wrong spot per hour.

**Criticality:** High. It will notify the administrator when a user has parked at a wrong spot

**Risk:** Low. Notification will not be generated because of power outage.

---------------------------------------------------------------------------------------------------------------------

**Constraints:**

* Usability: No previous training required.
* Reliability:Mean time to Failure – 5% failures for a year of use.
* Performance:The system has to be able to generate the message within 2 seconds of the confirmation.
* Supportability: The message generated should be precise and easy to understand so that the administrator can take action.
* Security: The parking user has parked on a spot that was not assigned to him. Security Officer receives a notification with data related to the user and the spot where he parked. He saves the Notification and can proceed to take action.

---------------------------------------------------------------------------------------------------------------------

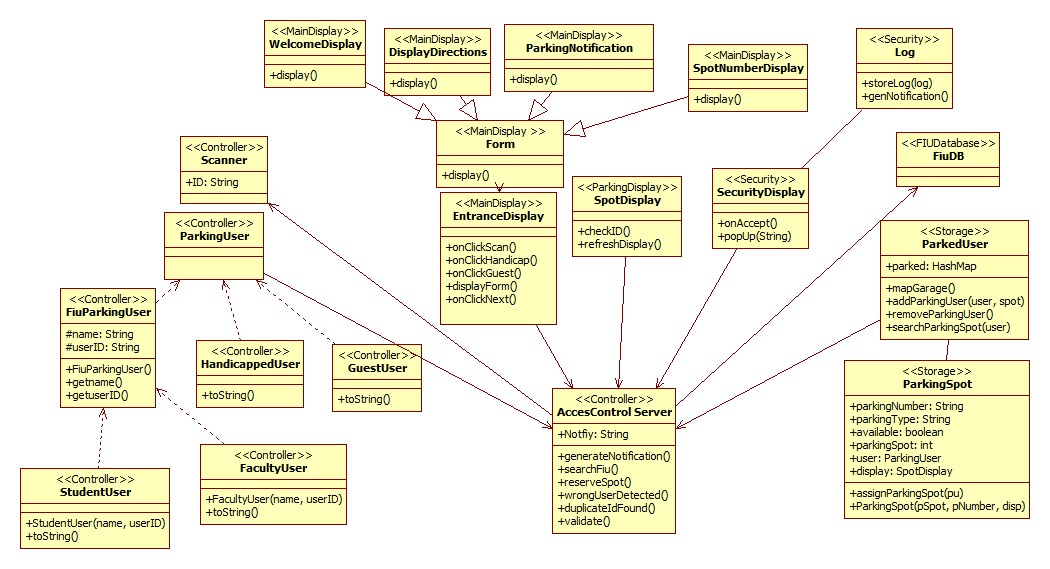
**Modification History**

**Owner:** Team 5

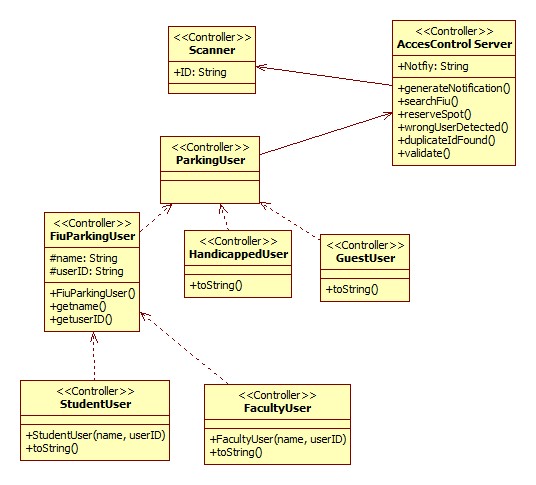
**Initiation date:** 05/14/2011

**Date last modified:** 05/14/2011

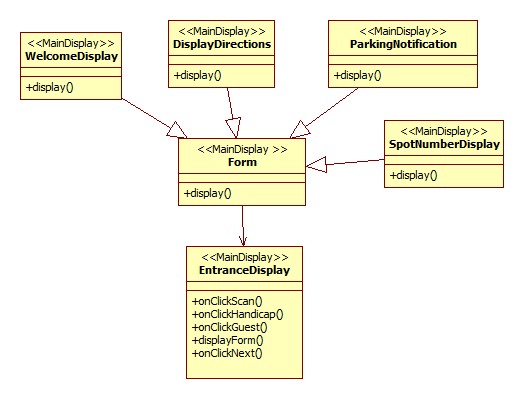
**5.3 Appendix C – Detailed Class Diagrams**

****

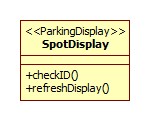
**Fig C.1 PantherLot Interactive Complete Detailed Class Diagram**

****

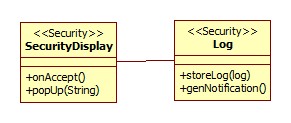
**Fig C.2 PantherLot Interactive Controller Subsystem Detailed Class Diagram**

****

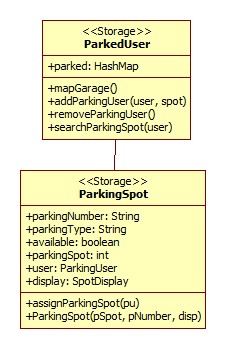
**Fig C.3 PantherLot Interactive MainDisplay Subsystem Detailed Class Diagram**

****

**Fig C.4 PantherLot Interactive ParkingDisplay Subsystem Detailed Class Diagram**

****

**Fig C.5 PantherLot Interactive SecurityDisplay Subsystem Detailed Class Diagram**



**Fig C.6 PantherLot Interactive Storage Subsystem Detailed Class Diagram**

**5.4 Appendix D – Class Interfaces**

**Link to PantherLot Interactive Javadoc:** [**PantherLot Interactive**](javadoc/index.html)

**5.5 Appendix E – Diary of meeting and tasks**