PantherLot Interactive

Course: CEN4010
Section: U01
Date: August 2, 2011
Professor: XXXXX X. XXXXXX

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 project planning including the assignment of the roles of team members, and all the requirements elicitation and requirement analysis of the proposed parking garage system.

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.

TABLE OF CONTENTS

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 44,000 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. Generally spending 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.

1.2 Scope of System

PantherLot Interactive 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. Each parking lot that uses this system will have a welcome screen that will show the parking spaces available and taken at the time of arrival and will serve as a guideline telling users whether or not they should bother looking for a parking spot in that place at that time and shorten their time looking for a space there. The minimal requirements for the main display screen are determined by the university. This display is further refined by each user's credentials.

1.3 Development 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.

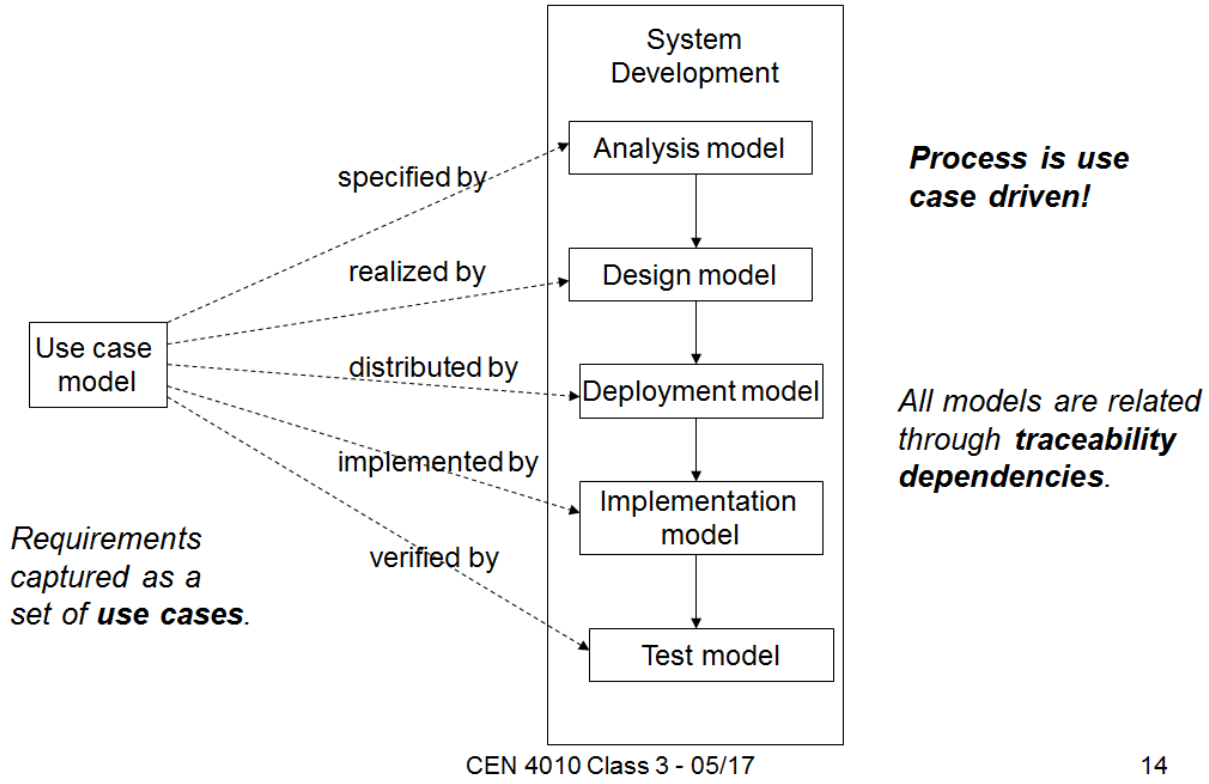

Figure 1 Unified Software Development Process

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

FIU – Florida International University.
FIU DB – FIU's Student, Faculty and Staff Database.
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 1 is the introduction, purpose, scope of the system, definition methodology, definitions, acronyms and abbreviations and the overview of the document. Chapter 2 describes the current system. Chapter 3 outlines our project plan and tasks involved in the development of PLI. Chapter 4 is the requirements of system, including the functional and nonfunctional requirements, use case diagram and requirements analysis. Chapter 5 describes the proposed system architecture, contains the overview, subsystem decomposition, hardware and software mapping and persistent data management. Chapter 6 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 7 is the testing process, which include: system tests, subsystem tests, and evaluation of tests. Chapter 8 is the glossary of the terms used in this document. Chapter 9 consists of the appendices for this document. Appendix A contains the Gantt chart of the project schedule. Appendix B contains the use cases with nonfunctional requirements. Appendix C contains the user interfaces. Appendix D contains the detailed class diagrams. Appendix E contains class interfaces for the subsystems being implemented. Appendix F contains the documented code and Appendix G contains the diary of meetings and tasks.

2. CURRENT SYSTEM

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.

In FIU today we implement a decal purchase system where the owner of the car has to be registered in classes that semester in order to be eligible to get the student parking decal. Other parking decals are available if you work for FIU or if you teach here. (Color in the parking) The main problem that we have nowadays is that since there is no real control of who comes in or out of the parking lot or who remains inside is that people start driving around inside the parking lot wasting gas for long periods of time. Sometimes even for more than 30 minutes at a time.

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.

3. PROJECT PLAN

This section 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 Project Organization

Team Leader

Manages all meetings, assist team members as needed delegate work to other teammates, ensures all tasks are completed in a timely manner, ensure group meetings go as planned, and relegates workflow.

Minute Taker

Keeps records of all meetings, what was discussed at each meeting, the time of the meetings, and what members were present. The Minute Taker will also post this information to the forum for viewing by all members.

Time keeper

The Development Tester Coordinates testing of the newly implemented parts of the system. This individual also makes sure that the different parts of the system properly function when sections are added, changed, or deleted.

System Architect

The System architect is in charge of the high level organization of the system, setting the groundwork for the system to be implemented.

System Developer

System Developer is responsible for the design and optimization of the system, such that the system's performance is up to the standard of the requirements.

Software Developer

The Software Developer is responsible for designing and implementing the high level part of the application as well as writing part of the code of the program.

3.2 Hardware and Software Requirements

Hardware needed

1. Server: Intel® Xeon® 5600 series 6 core processor, 64 Gb Ram, 2x PCIe G2 Slots Network Controller 1GbE NC382i Multifunction Ports
2. Router: with Dynamic DNS
3. Touch Display Screens
4. Scanners (30cm range)
5. Sensors for the Parking Spot (20cm to 150cm)
6. ATmega328 microcontroller

Software needed

1. Visual Studio 2010
2. Microsoft Word
3. Star UML
4. Netbeans
5. JUnit
6. Windows 2008 Server
7. VMware virtualization software
8. Arduino Uno
9. Windows 7 interface

3.3 Work Breakdown

The task necessary for the development and implementation of the PantherLot Interactive system are divided into three different milestones. The first milestone is the software requirement documentation which encompasses the tasks involving the definition of the project, and the software requirements elicitations and requirements analysis. The second milestone is the software design documentation which involves the task related to the design of the system. The last milestone consists of the actual development and testing of the PantherLot Interactive system. Here we have a table containing all the tasks each with an estimated duration and for the actual visual representation, go to Appendix A.

ID	Task Name	Duration	Start	Finish	Predecessors
1	Discuss Project Ideas, purpose and features	4 days	Thu 5/12/11	Tue 5/17/11	
2	Decide on hardware/software	3 days	Tue 5/17/11	Thu 5/19/11	1
3	Assign role/tasks	1 day	Tue 5/17/11	Tue 5/17/11	1
4	Create Schedule for entire semester	3 days	Mon 5/23/11	Wed 5/25/11	3
5	Create 30 use cases	13 days	Thu 5/26/11	Sat 6/11/11	4
6	Create 6 security use cases	13 days	Thu 5/26/11	Sat 6/11/11	4
7	Create SRD Document	10 days	Tue 5/31/11	Mon 6/13/11	4
8	Review and Finalyze SRD document	3 days	Sun 6/12/11	Wed 6/15/11	5,6,7
9	Create SRD Presentation	1 day	Mon 6/13/11	Mon 6/13/11	8
10	First Deliverable Presentation	1 day	Tue 6/14/11	Tue 6/14/11	9
11	Submit SRD Document	1 day	Wed 6/15/11	Wed 6/15/11	10
12	Design System	20 days	Thu 6/16/11	Wed 7/13/11	11
13	Create SDD Document	20 days	Thu 6/16/11	Wed 7/13/11	11
14	Review and Finalyze SDD document	2 days	Sun 7/10/11	Mon 7/11/11	
15	Create SDD presentation	1 day	Tue 7/12/11	Tue 7/12/11	14
16	SDD presentation	1 day	Tue 7/12/11	Tue 7/12/11	14
17	Submit SDD Document	1 day	Tue 7/12/11	Tue 7/12/11	14
18	Implement Sofware	15 days	Thu 7/14/11	Wed 8/3/11	17
19	Test Software	4 days	Wed 7/27/11	Mon 8/1/11	
20	Create Final Document	16 days	Thu 7/14/11	Thu 8/4/11	17
21	Create Final Presentation	1 day	Tue 8/2/11	Tue 8/2/11	19
22	Present Software	1 day	Tue 8/2/11	Tue 8/2/11	19
23	Submit Final Doc	1 day	Tue 8/2/11	Tue 8/2/11	19

4. REQUIREMENTS OF SYSTEM

One of the opinions shared by the majority of the members of the FIU community is that finding a parking space when visiting the main campus is hard. This problem worsens every year as the FIU community grows at a faster rate than the amount of new parking spaces available. Instead of just building new parking garages, our solution is to maximize the current resources of parking spaces we have by implementing a more robust parking system for the parking garages we currently have. Our system, PantherLot Interactive, will reduce the amount of time that the members of the FIU community take to find a parking space. The high level abstraction of the system is that when a person looking for parking enters a parking garage, the system will identify the user, check for available parking spots and tell the user where to park. This will remove the factor of having cars aimlessly wondering around the garage waiting and searching for a parking spot. Removing this factor, will dramatically reduce the amount of time a user has to spend looking for a parking spot. The system also maximizes the use of parking space resources by keeping track of all spots currently taken and the ones that are available. In other words, PantherLot Interactive offers optimization for the dimensions of both time and space of our parking garages.

4.1 Functional and Non-functional Requirements Abbreviations

Functional Requirements

1. The system shall let the security officer communicate with the parking user.
2. The system shall retrieve the user's information from ID (relate to use case).
3. The system shall assign a parking spot to a handicap ParkingUser.
4. The system shall assign a parking space according to the user type.
5. The system shall acknowledges that the guest user paid and increasing the time available.
6. The system shall track of the parking space time allowed for the user to be parked in a guest parking space.
7. The system shall check the parking garage availability.
8. The system shall treat the user as a guest when the ID Scan Fails.
9. The system shall check for a guess parking space available when the ID Scan Fails.
10. The system shall proceed to display the directions to park on the spot assigned to the user.
11. The system shall proceed to the main menu and be ready for the next user.
12. The system shall display "Parked" when user leaves spot.
13. The system will notify the security officer the spot has been reserved
14. The system shall inform both user and security officer when parked in the wrong spot
15. The system shall search for available guest parking spots for the parking user when there is a duplicate ID.
16. The system shall assign a guest spot if no more parking is available for faculty.
17. The system shall assign a guest spot if no more parking is available for student.
18. The system shall make a spot available again if the ParkingUser is not parked within the time limit.
19. The system shall notify the Security Officer of an ID already in use
20. The system shall updates and gathers the information of the user who is parked in the wrong place.
21. The system shall notify the security officer when guest parking is expired and ParkingUser still parked in the spot.
22. The system shall increase the number of available parking spot when user leaves.
23. The system shall save emergency request in the system and marked as handled.
24. The system shall alert the fire department and the security officer incase of emergency.
25. The system shall store system emergencies in the log.
26. The system shall save current state and notify Security Officer after power loss.

27. The system shall restart when the system becomes unresponsive.
28. The system shall be able to shut down after Security officer command.
29. The system shall inform the security officer about a ParkingUser pressing the handicap button.
30. The system shall let the user communicate with the Security officer when needed.
31. The system shall inform Security Officer if someone else tries to shut down the system.
32. The system shall inform Security Officer if someone used the emergency vehicle access
33. The system shall inform the Security Officer when someone presses the guest spot without being scanned.
34. The system shall notify the Security Officer if someone presses the handicap button.
35. The system shall notify the security officer when a duplicate ID is detected.
36. The system shall notify the security officer when a ParkingUser takes a wrong spot.

See Appendix B for functional requirements of all use cases

Nonfunctional Requirements

Usability:

- No previous training is required.
  (PLI001, PLI002, PLI003, PLI004, PLI005, PLI006, PLI006, PLI007, PLI008, PLI009, PLI010, PLI011, PLI012, PLI013, PLI014, PLI015, PLI016, PLI017, PLI018, PLI019, PLI021, PLI022, PLI023, PLI024, PLI025, PLI026, PLI027, PLI028, PLI029, PLI030, PLIS01, PLIS02, PLIS03, PLIS04, PLIS05, PLIS06)

Reliability:

- Mean time to Failure – 5% failure for every twenty four hours of operation. (PLI001, PLI002, PLI004, PLI005, PLI006, PLI007, PLI008, PLI009, PLI018, PLI019, PLI020, PLI022, PLI023, PLI024, PLI026, PLI027, PLI028, PLI029, PL030, PLIS01, PLIS03, PLIS04, PLIS06)
- Mean time to Failure - 10% failures for one year of use. (PLI003, PLI020, PLI021, PLI025)
- Mean time to Failure – 5% failures for a year of use. (PLI010, PLI011, PLI012, PLI013, PLI014)
- Mean time to Failure - 1% failures for one day of use. (PLI015, PLI016, PLI017, PLIS05)
- Mean time to Failure – 5% failures for every month of use. (PLIS02)

Performance:

- On average, the security officer must be able to handle four calls per hour. (PLI001)
- On average, information should be scanned and saved within 4 seconds. (PLI002)
- On average, request takes about 3 seconds. (PLI003, PLI004, PLI006, PLI007, PLI008, PLI009, PLI021, PLI022, PLI024, PLI025, PLI026, PLI027, PLI028, PLI029, PLI030, PLIS03, PLIS04, PLIS05)
- On average, request takes about 5 seconds. (PLI010, PLI011, PLI015, PLI016, PLI017, PLI023, PLIS01)
- On average, request takes about 8 seconds. (PLI005)
- The system has to be able to display the parking space within 5 seconds. (PLI010)
- On average, maintenance request takes about 30 minutes in a 24 hour period. (PLI012)
- The system should be able to reserve the spot within one second of confirmation. (PLI013)
- The system has to be able to generate the message within 2 seconds. (PLI014)
- On average, request takes about 15 seconds. (PLI018, PLI019)
- The system should remove the user's information and send the informational message within 5 seconds. (PLI020)
- On average, request to the security officer takes about 3 seconds. (PLIS02)
- The system has to be able to generate the message within 2 seconds of the confirmation. (PLIS06)

Supportability:

- The system should deliver a clear message to the security officer (PLI001)
- The scanning device should be compatible with all types of IDs (PLI002)
- The screen must display data clearly enough for the user to read it. (PLI003, PLI006, PLI007, PLI008, PLI009, PLI015, PLI016, PLI017, PLI018, PLI019, PL020, PLI021, PLI024, PLI025, PLI026, PLI027, PLI028, PLI029, PLI030, PLIS01, PLIS03, PLIS04, PLIS05)
- Supportability: The system should be compatible with the FIU DB. (PLI004)
- System must support all forms of payment types. (PLI005)
- The displayed image and information should be clear and easy to follow. (PLI010, PLI011)
- The displayed information should be clear and easy to understand for the user. (PLI012, PLI022, PLI023)
- The Security User interface should be clear and precise in order to reserve spot. (PLI013)
- The message displayed should be clear and precise in order for the user to be able to understand. (PLI014)
- The alarm must be compatible to interact with the software. (PLIS02)

Implementation:

- Using java to deliver the notification to the security officer. (PLI001, PLI002,
- Using java with GUI to implement. (PLI003, PLI006, PLI007, PLI008, PLI009, PLI010, PLI011, PLI012, PLI013, PLI014, PLI015, PLI016, PLI017, PLI021, PLI022, PLI024, PLI025, PLI026, PLI027, PLI028, PLI029, PLI030)
- Using Java program to assign a parking spot based on type (PLI004)
- Java software will process payments and updates user's time. (PLI005)
- Using java clients and servers with GUI to implement (PLI018, PLI019)
- Java software will be used to remove the assigned parking spot and send the informational message to the administrator. (PLI020)
- The send request form should be correctly handled by IE and Mozilla. (PLI023)

Security:

- Administrator needs to input credentials in order to shut down the system (PLIS01)
- Security officer needs to check the status of the fire alarm and the reasons it was triggered (PLIS02)
- System will check all the guess parking spaces and compared to the database. If unauthorized user is detected, the parking administrator should give a ticket to the user. (PLIS03)
- System will send notification to Security Officer when unauthorized user enters the spot. (PLIS04)
- Security officer needs to find out which car is using an unauthorized decal and allow the parking user to park on a spot. (PLIS05)
- 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. (PLIS06)

See Appendix B for non-functional requirements of all use cases

4.2 Use case diagram

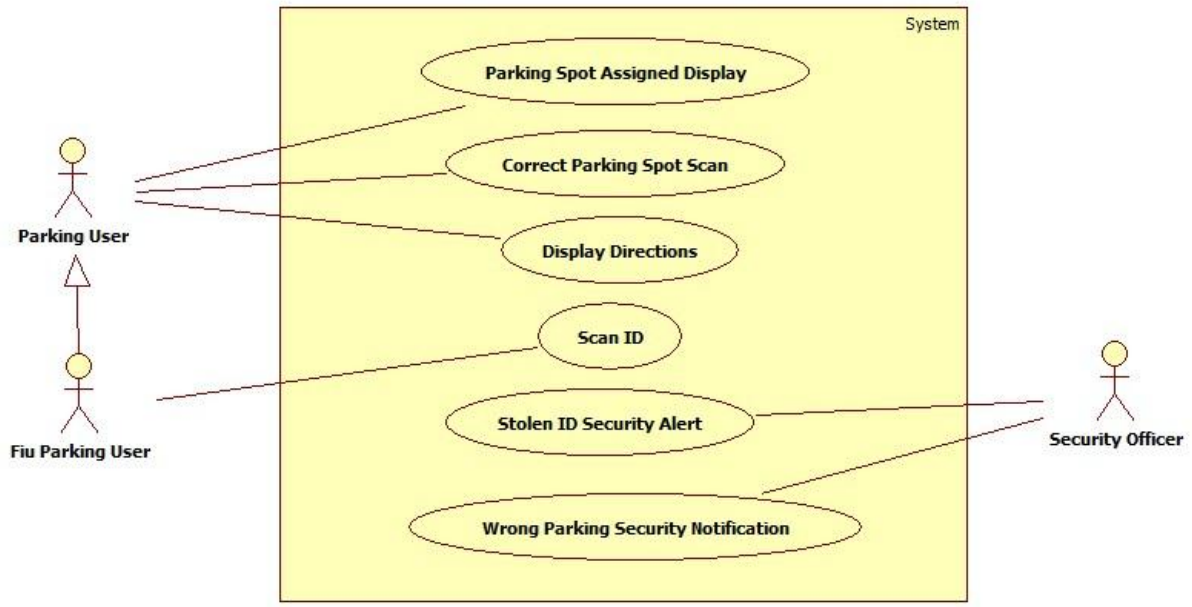

Fig 4.2 PantherLot Interactive Use Case Diagram

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

4.3 Requirements Analysis

Our system is based on the unified software development process, USDP. This type of system model is use case driven, that's why this section will be oriented to use cases. The use cases are divided into three packages, which are: Parking System, Security Officer, FIU database. All use cases diagrams found in this paper were created using a freeware called StarUML. In order to get a better idea of the systems, please refer to Appendix B for the detailed description of the use cases, and section 4.2 for the visual representation of use cases (diagrams).

For the Scan ID use case we came up with recognition software that would identify current FIU users from the student, faculty and staff database. Then we realized that the fastest way to ID someone coming into the parking lot would be to have the parking decal double as a bar code as well that would be scanned by the system upon entering the parking lot rather than an ID swiping system that is used in places like MDC were it takes a long time for everyone to scan their ID every time they come to the parking lot, especially if it is the rush hour for the morning classes.

Another functionality we came up with was the integration of the tracking of people coming in and out of the parking lot to have extra control and make more efficient the way that people can determine whether or not you can park in this parking lot as you are entering and the system letting you know within 5 seconds to go to the parking spot (Display directions) and verifying if you parked at the space provided (Correct parking spot scan).

As an extra necessary feature it would be required that there is a parking lot attendant (Security Office) that is going around tending to things like people parking in the wrong spot (Wrong parking security Notification). This is recommended and necessary in other to keep track of people parking correctly and keeping accountability and also for aiding the parking spot assignment

5. PROPOSED SOFTWARE ARCHITECTURE

In this section, 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, there are 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.

5.1 Overview

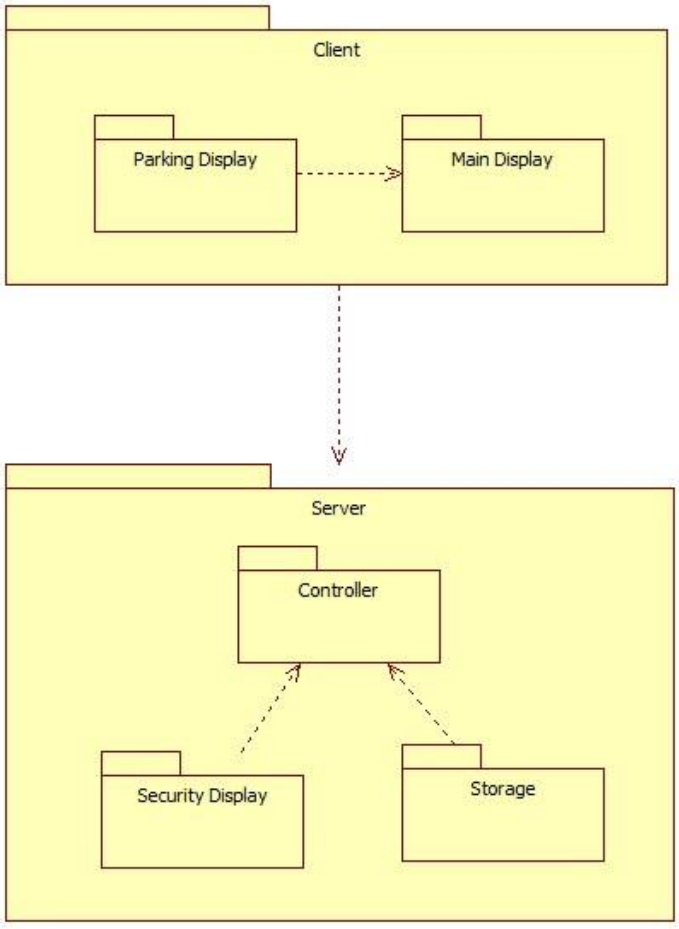

Figure 5.1 PantherLot Interactive Architecture

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.

Design Patterns

Singleton Pattern: Our system uses the Singleton design pattern. This design pattern is more commonly found in the Subsystem linking to the repository; the repository is where our functions reside to execute what's needed. Our system needs to gather the information from the user's ID and see which function it needs from the repository. The use cases that comply with this pattern are: Scanning ID (PLI-002), Parking Spot Assign Display (PLI-010), and Display Directions (PLI-011) to name a few. Refer to Appendix C for the detailed class diagram.

Command Pattern: Our system also uses the Command design pattern. This design pattern is found in our Client/Server architecture since the client creates Concrete Commands which encapsulate a service to be applied to a Receiver. We have the requester and provider. The invoker actually executes or undoes the command.

5.2 Subsystem Decomposition

Parking Display Subsystem: This subsystem takes care of the general user. 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), and 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.

5.3 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: distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IRED (infrared emitting diode) and signal processing circuit. This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor 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).

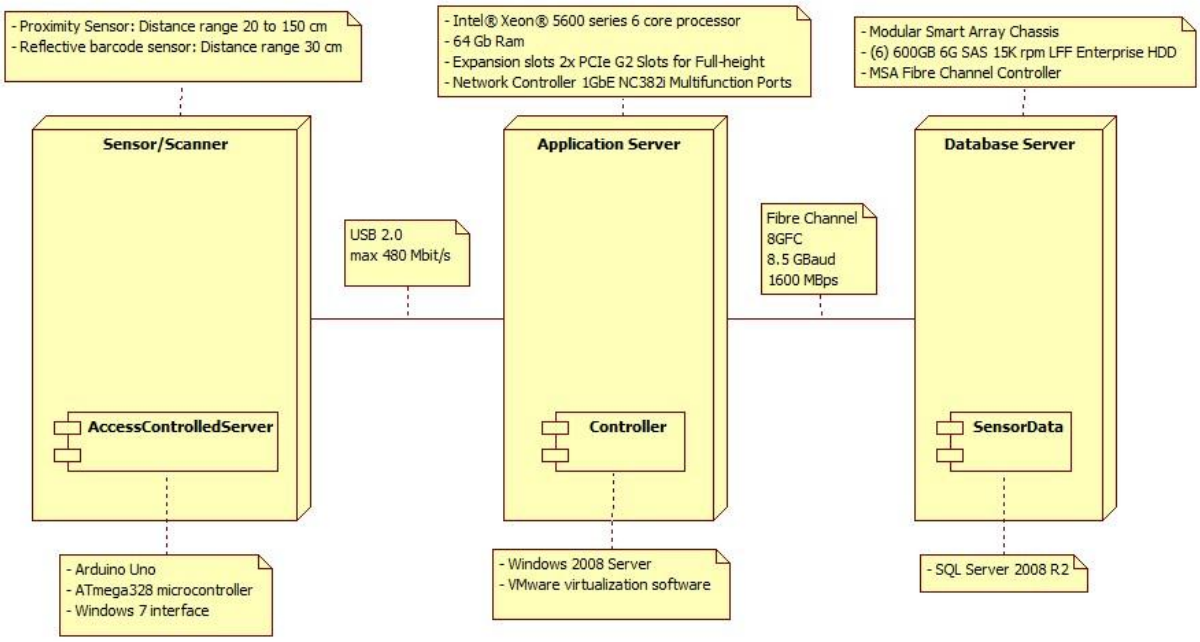

Figure Error! No text of specified style in document.**.2 PantherLot Interactive Deployment Diagram**

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

5.4 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.

Field	Type	Null	Comment
spotNumber	**int**	**no**	**the spot number in the garage**
spotType	**varchar(10)**	**no**	**the type of user that parks on this spot**
floor	**int**	**no**	**the floor where this spot is located**
parkingNumber	**varchar(5)**	**no**	**the number labeled on the parking spot**
direction	**varchar(8)**	**no**	**the cardinal direction of the parking spot**

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.

6. DETAILED DESIGN

This section 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.

6.1 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.

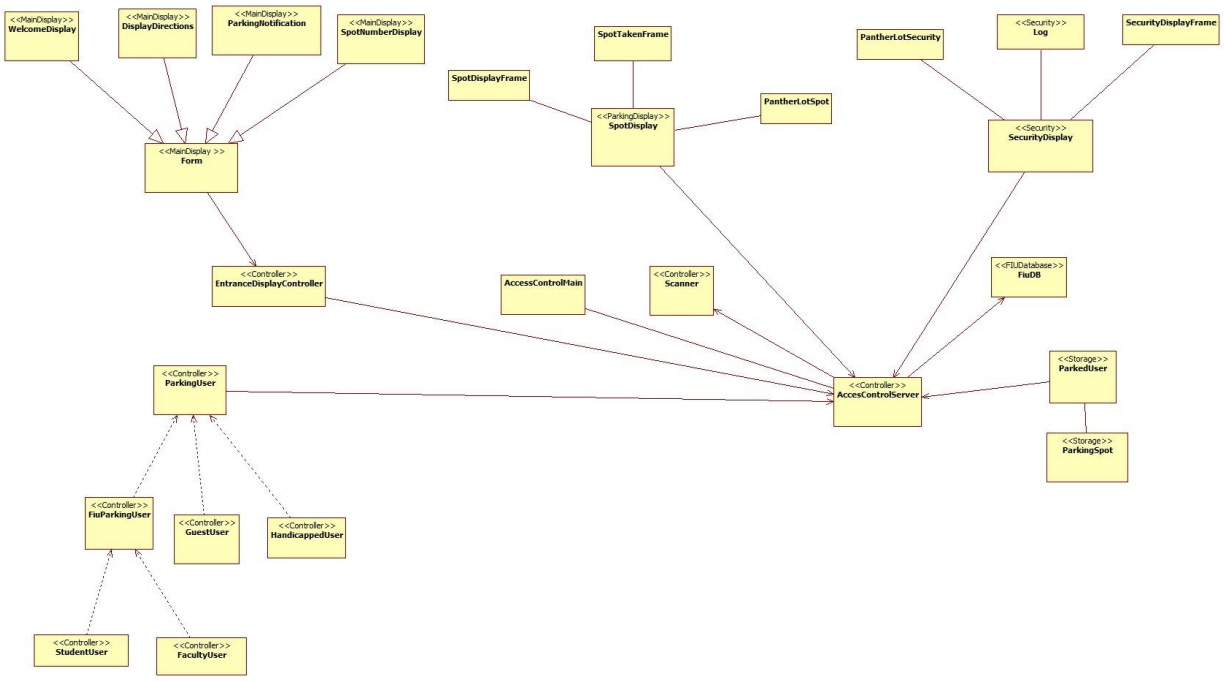

Fig 6.1. PLI Minimal Class Diagram

6.2 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.

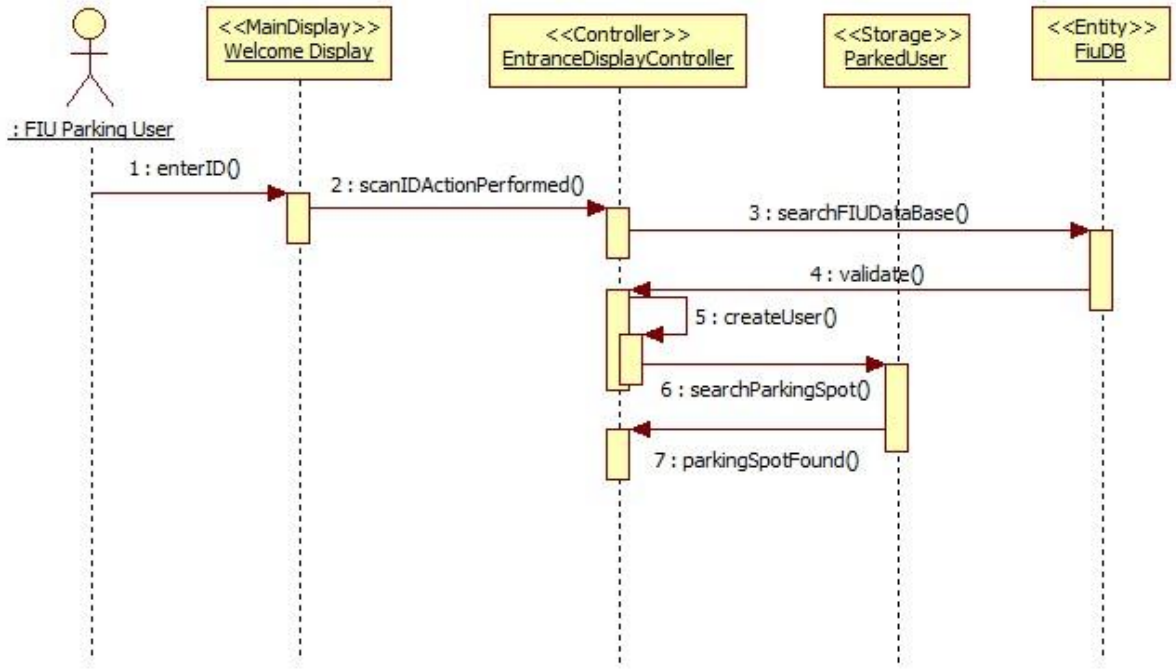

Fig 6.2.1 Scan ID Sequence Diagram

Description: This sequence diagram represents how a FIU parking user's ID (student, faculty or handicapped) is scanned and validate by the PantherLot Interactive. This sequence diagrams begins when the FIU parking user drives up to the entrance and let the scanner reads the ID. This information is passed to the EntranceDisplayController where it gets validate. After the ID number is validated with the FiuDB, the system creates a new user and finds an appropriate available parking spot for that type of user.

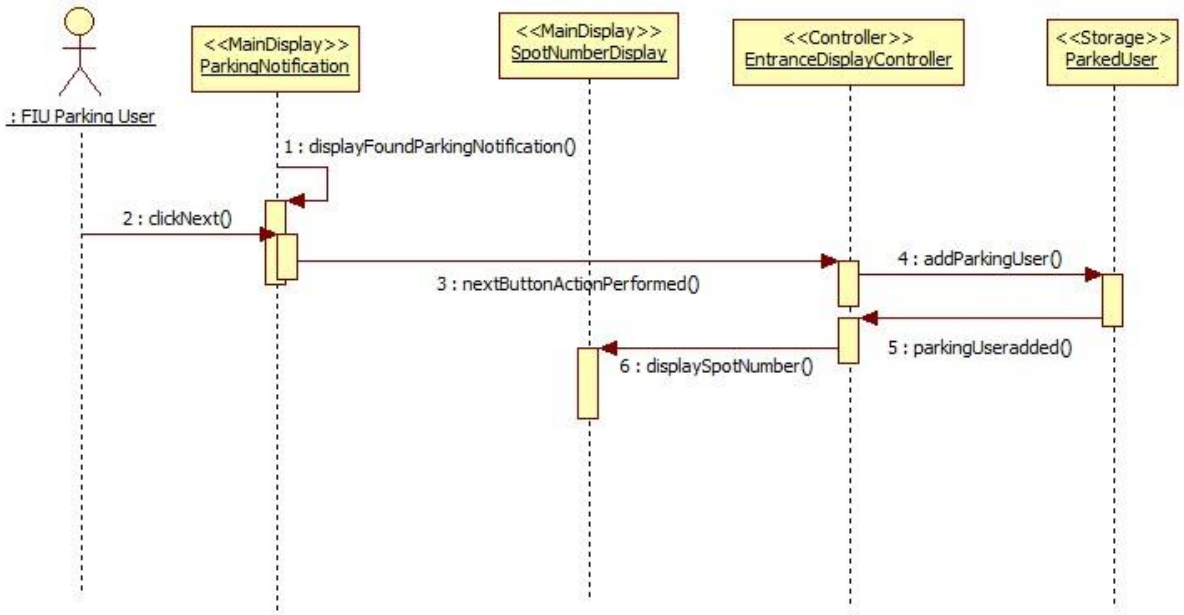

Fig 6.2.2 Parking Spot Assigned Display Sequence Diagram

Description: This sequence diagram represents how the system relates a specific parking spot of the parking garage with the FIU parking user information. The sequence diagram begins when the system found a parking available on the parking garage and prompt the user to click “next”. Once the user press the “next” button, the information is passed to the EntranceDisplayController. This controller adds this fiu parking user to the parking garage database. Finally, the system displays the spot number to the user on the main display.

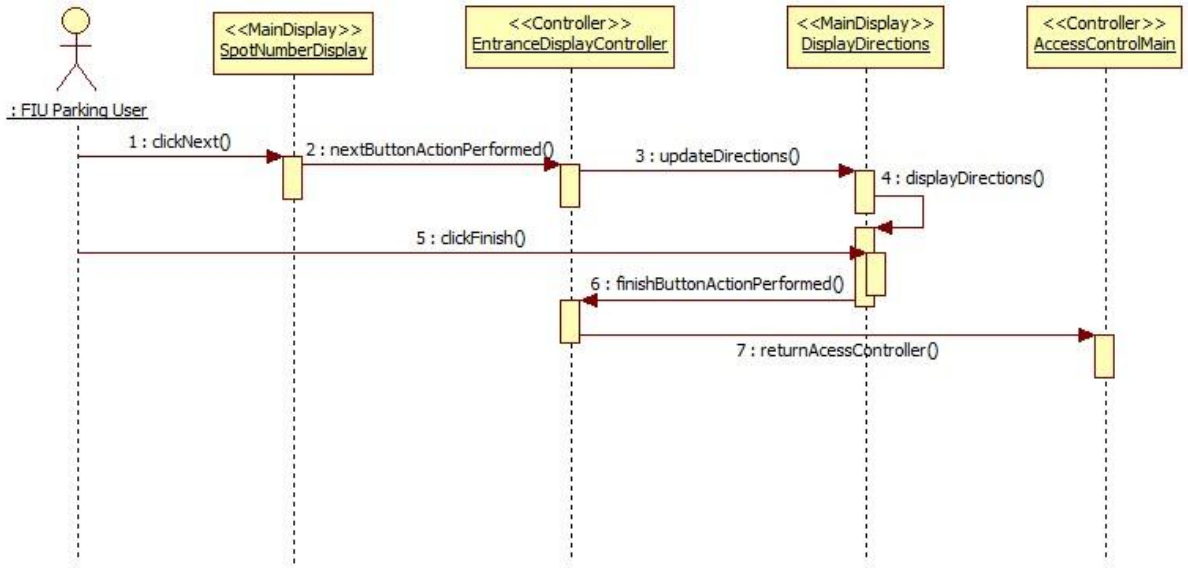

Fig 6.2.3 Display Directions Sequence Diagram

Description: This sequence diagrams represents how the system display the directions to the fiu parking user on the main display. The user already knows what spot number the system assigned to him/her. Therefore, the user presses the "next" button to get the directions. Once the user interacts with the system, the EntranceDisplayController is acknowledged and send the appropriate directions to the main display by invoking updateDirections() and displayDirections() respectively. Moreover, the user is now ready to check the directions on the main screen and click on "finish" whenever this information is acquired.

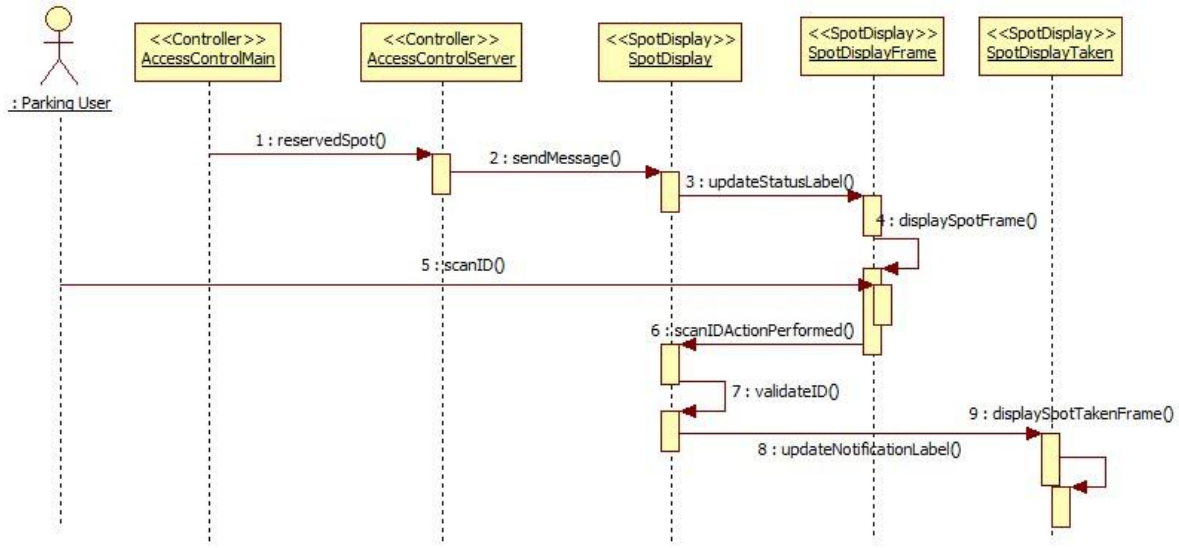

Fig 6.2.4 Correct Parking Spot Scan Sequence Diagram

Description: This sequence diagram represents how the system notifies to the parking user that he/she was parked in the wrong parking spot. The sequence diagram begins when a parking spot is reserved in the parking garage. A user parked into an assigned spot and the system scans user's ID to verify if the user is parked correctly. When the system checks the parking spot validity, it displays that the user is parked correctly or not on the Spot Display.

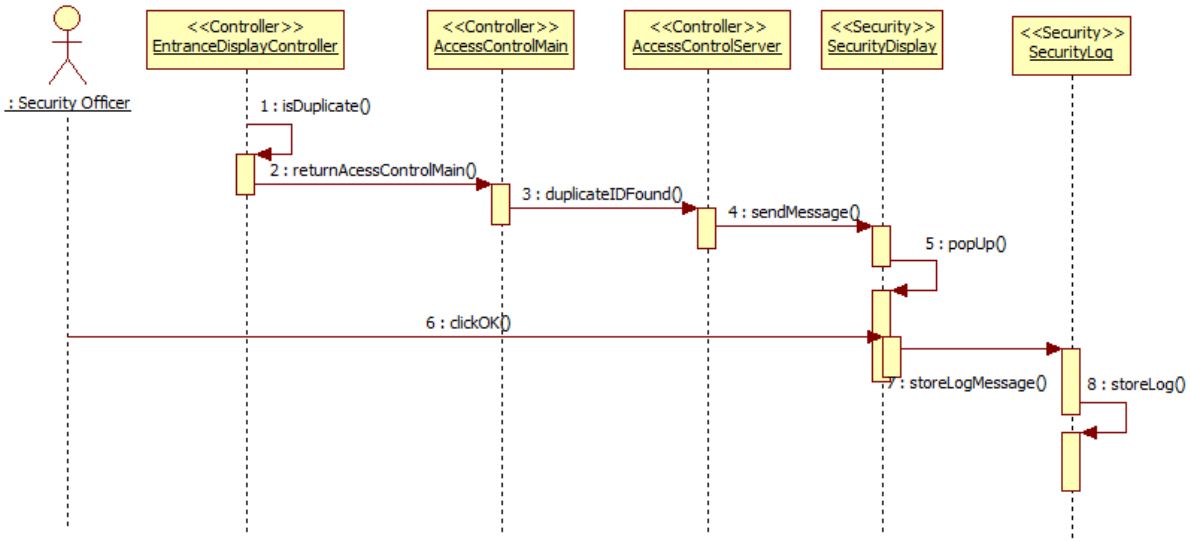

Fig 6.2.5 Stolen ID Security Alert Sequence Diagram

Description: This sequence diagram represents how the system notifies to the security officer that a user is attempting to enter the parking garage with an ID already used on the system. This sequence diagram begins when the system found that a duplicate ID is used to enter the parking garage. The AccessControlMain and AccessControlServer manage to send the message describing the issue (the respective functions call duplicateIDFound() and sendMessage() are invoked) to the security display located at the security office. Once this information is displayed on the security display, popUp() is called, the security officer proceed by selecting "Ok", and the log message is stored in the SecurityLog file using the storeLog() function call.

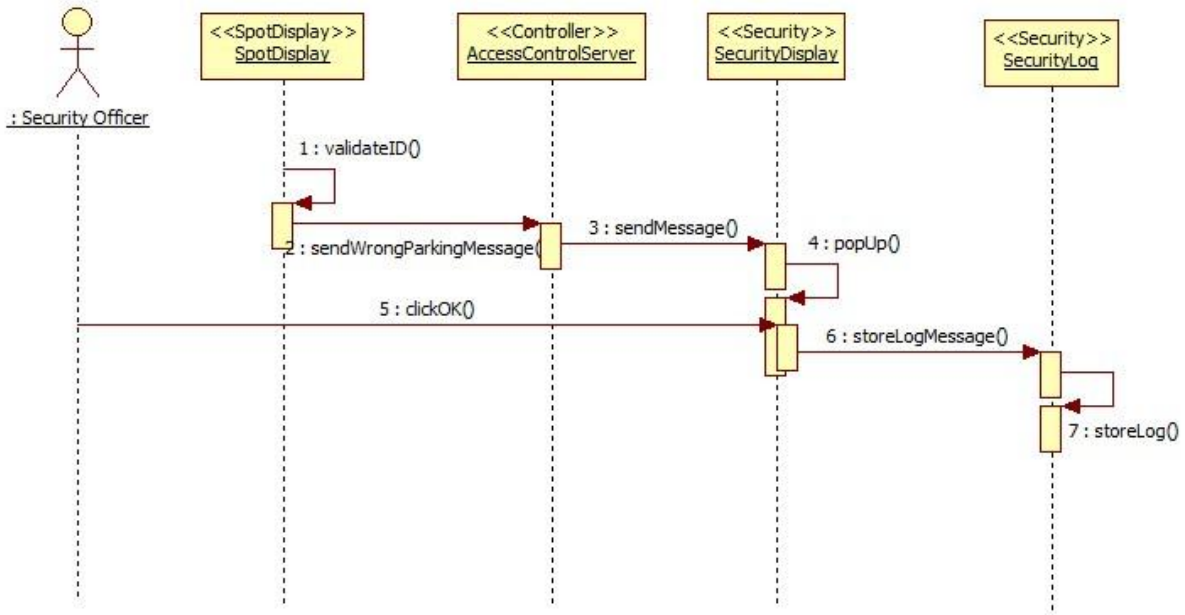

Fig 6.2.6 Wrong Parking Security Notification Sequence Diagram

Description: This sequence diagram represents how the system notifies to the security officer that a user parked in a parking spot that was not specifically assigned to that user. This sequence diagram begins when the system scan the user's ID at the parking spot once the user is parked. The AccessControlServer manage to send a message call describing the issue using the sendMessange() to the security display located at the security office. Once this information is displayed on the security display, popUp() is called, the security officer proceed by selecting "Ok", and the log message is stored in the SecurityLog file using the storeLog() function called.

6.3 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: Thi 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:** *displayConnections.size() > 0*

Context: *AccessControlServer*::*mapConnections()* **Pre:** *displayConnections.isEmpty()*

Context: *AccessControlServer*:: *mapConnections ()* **Post:** *!displayConnections.isEmpty()*

Context: *AccessControlServer*::*startServer ()* **Pre:** *!serverSocket.isBound()*

Context: *AccessControlServer*:: *startServer ()* **Post:** *serverSocket.isBound()*

Context: *AccessControlServer*::*sendMessage(msg, pout)* **Pre:** *msg != null, pout != null*

Context: *AccessControlServer*:: *sendMessage (msg, pout)* **Post:** *pout != null*

Context: *AccessControlServer*:: *sendStatus ()* **Pre:** *garage.getStatus().size() > 0, securityOut != null*

Context: *AccessControlServer*:: *sendStatus ()* **Post:** *securityOut != null*

Context: *AccessControlServer*::*reserveSpot(spot, id)* **Pre:** *spot != null,!displayConnections.isEmpty(), displayConnections.contains(spot.getParkingSpot()), id != null*

Context: *AccessControlServer*::*reserveSpot(spot, id)* **Post:** *spot != null,!displayConnections.isEmpty(), displayConnections.contains(spot.getParkingSpot())*

Context: *AccessControlServer*:: *wrongUserDetected (msg)* **Pre:** *msg != null, securityOut != null*

Context: *AccessControlServer*:: *wrongUserDetected (msg)* **Post:** *securityOut != null*

Context: *AccessControlServer*:: *duplicateIdFound (msg1, msg2)* **Pre:** *msg1 != null, msg2 != null, securityOut != null*

Context: *AccessControlServer*:: *duplicateIdFound (msg1, msg2)* **Post:** *securityOut != null*

Context: *AccessControlServer*:: *addDisplay (key, out, spot)* **Pre:** *key! = null, out != null, spot != null, !displayConnections.containsKey(key)*

Context: *AccessControlServer*:: *addDisplay (key, out, spot)* **Post:** *displayConnections.get(key) = out, spot != null, displayConnections.size() + 1*

Context: *AccessControlServer*:: *removeDisplay (key, spot)* **Pre:** *displayConnections.contains(key), spot != null*

Context: *AccessControlServer*:: removeDisplay *(key, spot)* **Post:** *displayConnections.get(key) == null,spot.getPrintWriter() == null*

Context: *AccessControlServer*:: *isConnectionAvailable (key)* **Pre:** *key != null, displayConnections != null*

Context: *AccessControlServer*:: isConnectionAvailable *(key)* **Post:** *displayConnections != null*

7. TESTING PROCESS

This section describes the testing performed to verify and validate that some of the functions of that have been integrated into the proposed system. Section 7.1 provides several system tests. Section 7.2 provides several tests for the outlined subsystem. Section 7.3 provides the results of the system tests.

7.1 System Tests

The following test plan was created to verify the exactness of the following use cases: Login, Initiate Chat Window and Send Message. Appendix B within this report holds the complete functional and non-functional information used to produce the test program.

8. 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.

File – Anything a Student can upload to their portfolio.

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.

Milestone – End-point of a software process activity.

Model* – An abstract representation of a system that enables us to answer questions about the system.

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.

System Intercom - is a stand-alone voice communications system for use within a building or small collection of buildings, functioning independently of the public telephone network.

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.

9. APPENDIX

9.1 Appendix A – Gantt Chart

ID	Task Mode	Task Name	Duration	Start	Finish
1		Discuss Project Ideas, purpose and features	4 days	Thu 5/12/11	Tue 5/17/11
2		Decide on hardware/software	3 days	Tue 5/17/11	Thu 5/19/11
3		Assign role/tasks	1 day	Tue 5/17/11	Tue 5/17/11
4		Create Schedule for entire semester	3 days	Mon 5/23/11	Wed 5/25/11
5		Create 30 use cases	13 days	Thu 5/26/11	Sat 6/11/11
6		Create 6 security use cases	13 days	Thu 5/26/11	Sat 6/11/11
7		Create SRD Document	10 days	Tue 5/31/11	Mon 6/13/11
8		Review and Finalyze SRD document	3 days	Sun 6/12/11	Wed 6/15/11
9		Create SRD Presentatio	1 day	Mon 6/13/11	Mon 6/13/11
10		First Deliverable Presentation	1 day	Tue 6/14/11	Tue 6/14/11
11		Submit SRD Document	1 day	Wed 6/15/11	Wed 6/15/11

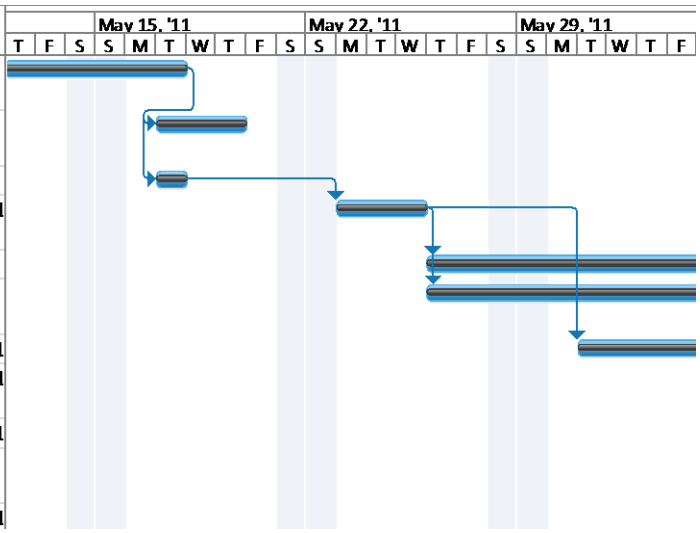

Figure A.1: Gantt Chart Part 1

ID	Task Mode	Task Name	Duration	Start	Finish
5		Create 30 use cases	13 days	Thu 5/26/11	Sat 6/11/11
6		Create 6 security use cases	13 days	Thu 5/26/11	Sat 6/11/11
7		Create SRD Document	10 days	Tue 5/31/11	Mon 6/13/11
8		Review and Finalyze SRD document	3 days	Sun 6/12/11	Wed 6/15/11
9		Create SRD Presentatio	1 day	Mon 6/13/11	Mon 6/13/11
10		First Deliverable Presentation	1 day	Tue 6/14/11	Tue 6/14/11
11		Submit SRD Document	1 day	Wed 6/15/11	Wed 6/15/11
12		Design System	20 days	Thu 6/16/11	Wed 7/13/11
13		Create SDD Document	20 days	Thu 6/16/11	Wed 7/13/11
14		Review and Finalyze SDD document	2 days	Sun 7/10/11	Mon 7/11/11
15		Create SDD presentatio	1 day	Tue 7/12/11	Tue 7/12/11
16		SDD presentation	1 day	Tue 7/12/11	Tue 7/12/11
17		Submit SDD Document	1 day	Tue 7/12/11	Tue 7/12/11
18		Implement Sofware	15 days	Thu 7/14/11	Wed 8/3/11
19		Test Software	4 days	Wed 7/27/11	Mon 8/1/11
20		Create Final Document	16 days	Thu 7/14/11	Thu 8/4/11
21		Create Final Presentation	1 day	Tue 8/2/11	Tue 8/2/11
22		Present Software	1 day	Tue 8/2/11	Tue 8/2/11
23		Submit Final Doc	1 day	Tue 8/2/11	Tue 8/2/11

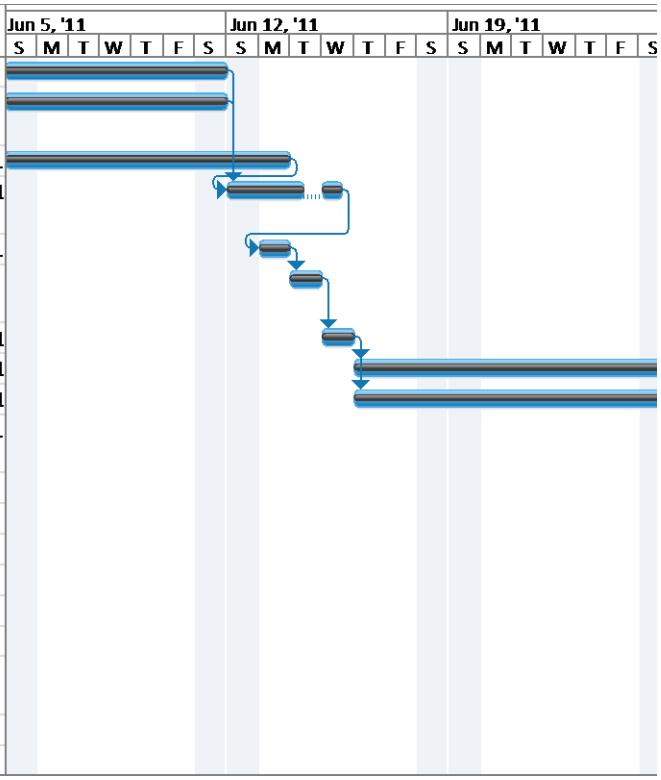

Figure A.2: Gantt Chart Part 2

ID	Task Mode	Task Name	Duration	Start	Finish
5	✓	Create 30 use cases	13 days	Thu 5/26/11	Sat 6/11/11
6	✓	Create 6 security use cases	13 days	Thu 5/26/11	Sat 6/11/11
7	✓	Create SRD Document	10 days	Tue 5/31/11	Mon 6/13/11
8	✓	Review and Finalyze SRD document	3 days	Sun 6/12/11	Wed 6/15/11
9	✓	Create SRD Presentatio	1 day	Mon 6/13/11	Mon 6/13/11
10	✓	First Deliverable Presentation	1 day	Tue 6/14/11	Tue 6/14/11
11	✓	Submit SRD Document	1 day	Wed 6/15/11	Wed 6/15/11
12	✓	Design System	20 days	Thu 6/16/11	Wed 7/13/11
13	✓	Create SDD Document	20 days	Thu 6/16/11	Wed 7/13/11
14	✓	Review and Finalyze SDD document	2 days	Sun 7/10/11	Mon 7/11/11
15	✓	Create SDD presentatio	1 day	Tue 7/12/11	Tue 7/12/11
16	✓	SDD presentation	1 day	Tue 7/12/11	Tue 7/12/11
17	✓	Submit SDD Document	1 day	Tue 7/12/11	Tue 7/12/11
18	✓	Implement Sofware	15 days	Thu 7/14/11	Wed 8/3/11
19	✓	Test Software	4 days	Wed 7/27/11	Mon 8/1/11
20	✓	Create Final Document	16 days	Thu 7/14/11	Thu 8/4/11
21	✓	Create Final Presentation	1 day	Tue 8/2/11	Tue 8/2/11
22	✓	Present Software	1 day	Tue 8/2/11	Tue 8/2/11
23	✓	Submit Final Doc	1 day	Tue 8/2/11	Tue 8/2/11

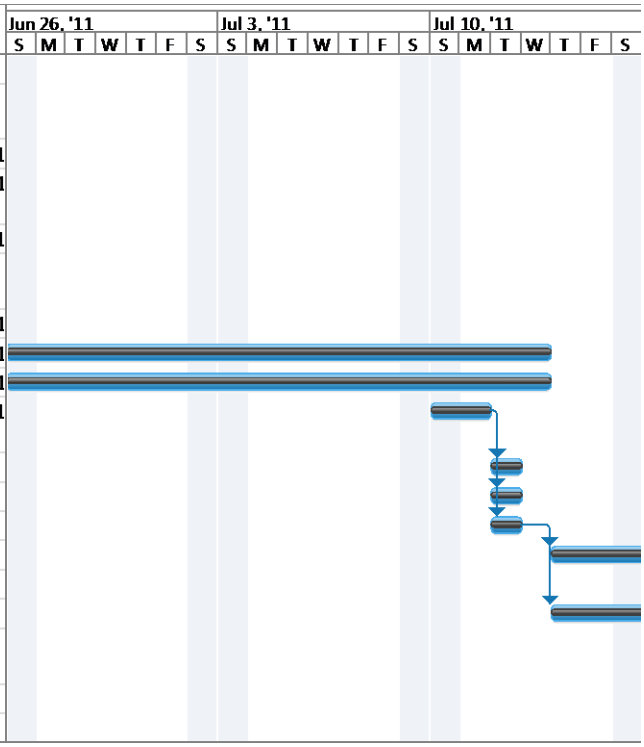

Figure A.3: Gantt Chart Part 3

ID	Task Mode	Task Name	Duration	Start	Finish
5	✓	Create 30 use cases	13 days	Thu 5/26/11	Sat 6/11/11
6	✓	Create 6 security use cases	13 days	Thu 5/26/11	Sat 6/11/11
7	✓	Create SRD Document	10 days	Tue 5/31/11	Mon 6/13/11
8	✓	Review and Finalyze SRD document	3 days	Sun 6/12/11	Wed 6/15/11
9	✓	Create SRD Presentatio	1 day	Mon 6/13/11	Mon 6/13/11
10	✓	First Deliverable Presentation	1 day	Tue 6/14/11	Tue 6/14/11
11	✓	Submit SRD Document	1 day	Wed 6/15/11	Wed 6/15/11
12	✓	Design System	20 days	Thu 6/16/11	Wed 7/13/11
13	✓	Create SDD Document	20 days	Thu 6/16/11	Wed 7/13/11
14	✓	Review and Finalyze SDD document	2 days	Sun 7/10/11	Mon 7/11/11
15	✓	Create SDD presentatio	1 day	Tue 7/12/11	Tue 7/12/11
16	✓	SDD presentation	1 day	Tue 7/12/11	Tue 7/12/11
17	✓	Submit SDD Document	1 day	Tue 7/12/11	Tue 7/12/11
18	✓	Implement Sofware	15 days	Thu 7/14/11	Wed 8/3/11
19	✓	Test Software	4 days	Wed 7/27/11	Mon 8/1/11
20	✓	Create Final Document	16 days	Thu 7/14/11	Thu 8/4/11
21	✓	Create Final Presentation	1 day	Tue 8/2/11	Tue 8/2/11
22	✓	Present Software	1 day	Tue 8/2/11	Tue 8/2/11
23	✓	Submit Final Doc	1 day	Tue 8/2/11	Tue 8/2/11

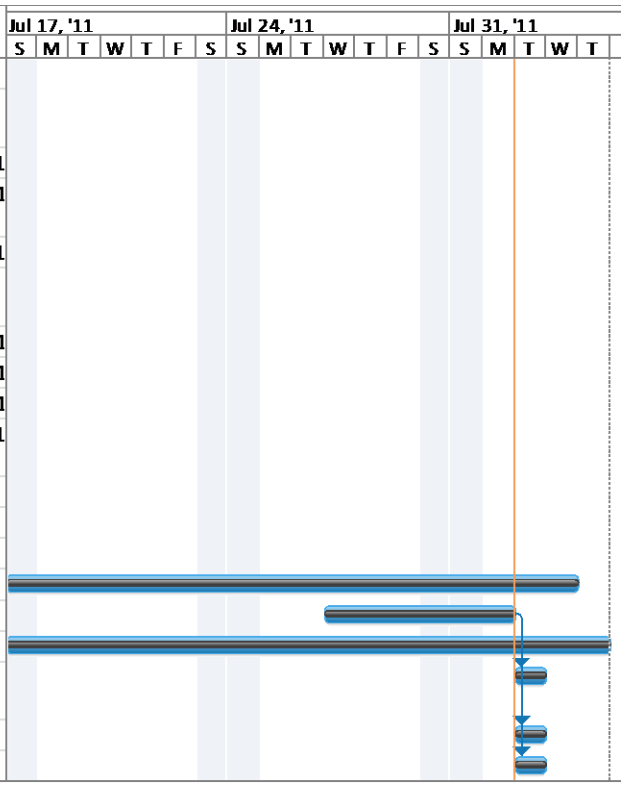

Figure A.4: Gantt Chart Part 3

9.2 Appendix B – Use Cases with non-functional

Use Case ID: PLI001 – Security call to user.

Use Case Level: High-level

Details:

- **Actor:** Parking User, Security Officer.
- **Pre-conditions:**
    1. Security Officer wants to communicate with the Parking User.
    2. Intercom system must be on and functional.
- **Description:**
    1. Use case begins when the Security Officer needs to contact a Parking User.
    2. The Security Officer then presses the speak button.
    3. Use case ends when the Parking User acknowledges the call.
- **Post-conditions:**
    1. The security officer is communicating with the user to help solve the problem.

Alternative Courses of Action:

Parking User calls the Security Officer first.

Exceptions:

Either Security Officer or Parking User are unable to communicate using the Intercom system.

Related Use Cases:

User call for help (PLI030)

Decision Support

Frequency: On average four calls will be made to the Security Officer per hour.

Criticality: High. Allows a Security Officer to communicate with a Parking User as needed.

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

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failure for every twenty four hours of operation.
- Performance: On average, the security officer must be able to handle 4 calls per hour.
- Supportability: The system should deliver a clear message to the security officer.
- Implementation: Using java to deliver the notification to the security officer.

Modification History

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 06/11/2011

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 to search for its 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: PLI003 – Handicap user

Use Case Level: High-level

Details:

- **Actor:** Handicapped user.
- **Pre-conditions:**
  1. The system is not able to identify the type of user.
- **Description:**
  1. Use case begins when the system treats the user as a guest.
  2. The system shall provide the user the option to be treated as a handicapped guest.
  3. The user has to press the handicapped button displayed on the screen.
  4. The system shall treat the user as a handicapped user and search for available parking spots.
  5. The user must park in the assigned parking spot.
  6. Use case ends when a notification is sent to the administrator check in the parking spot if the user has a handicapped parking permit.
- **Post-conditions:**
  1. The number of notifications for the administrator increases by one.
  2. The number of available handicapped parking spots decreases by one.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) the user can decide to not press this option.

Exceptions:

1. The user leaves the garage without replying to the screen display.

Related Uses Case:

None.

Decision Support

Frequency: On average, over 10 requests per year are made.

Criticality: High. In case of emergency system needs to be able to alert

Risk: High.

Constraints:

- Usability: No previous Training Time. Implemented by system.
- Reliability: Mean time to Failure – 10% failures for one year of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display data clearly enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/25/2011

Use Case ID: PLI004 – Search matching ID.

Use Case Level: High-level

Details:

- **Actor:** FIU DB
- **Pre-conditions:**
  1. User's ID has successfully been scanned.
  2. The information was correctly retrieved from the user's ID.
- **Description:**
  1. Use case begins when the system scans the ID
  2. Sends the Information to the FIU DB to retrieve the information needed.
  3. The FIU DB matches the ID to the file in the database and sends it to the system
  4. Use case ends when the system finds a compatible type of Parking User.
- **Post-conditions:**
  1. The system will assign a parking space according to the user type
  2. If no decal or not found it will assign a guest spot if available.

Alternative Courses of Action

1. Parking User can cancel the action and leave

Exceptions:

1. The searching method runs in an infinite loop.
2. The information on the Type of parking spot is not retrieved.

Related Use Cases:

None

Decision Support

Frequency: 2000 searches will be executed daily.

Criticality: High. The system needs to match the correct information to the Parking User.

Risk: Low. If the system encounters an infinite search loop and doesn't return anything.

Constraints:

- Usability: No previous training time.
- Reliability: Mean time to Failure – 5% failures for every twenty four hours of operation.
- Performance: On average, the search needed should take about 3 seconds.
- Supportability: The system should be compatible with the FIU DB.
- Implementation: Using Java program to assign a parking spot based on type.

Modification History

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/11/2011

Use Case ID: PLI005 – Add more time to guest parking.

Use Case Level: Low-level

Details:

- **Actor:** Parking User.
- **Pre-conditions:**
  1. The user has been continuously parked on the guest spot assigned by the system.
  2. The user had previously paid for the spot when the system assigned it.
- **Description:**
  1. Use case begins when user pays to extend his time for the use of the parking spot.
  2. The user decides how much extra time is needed and pays accordingly.
  3. Use case ends when the user has paid already for the extra time.
- **Post-conditions:**
  1. The system acknowledges the user paid and increasing the time available.
  2. System begins countdown from the new parking time.

Alternative Courses of Action

None.

Exceptions:

Unable to pay to increase the time for parking spot.

Related Use Cases:

Guest Charging Mechanism (PLI006)

Decision Support

Frequency: On average, 45 users will pay to add more time per day.

Criticality: High. The system needs to receive payment to increase the parking spot access.

Risk: High. Machine is not able to accept payment at this time and User has to leave.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure- 5% failures for every twenty four hours of operation.
- Performance: On average, request takes about 8 seconds.
- Supportability: System must support all forms of payment types.
- Implementation: Java software will process payments and updates user's time.

Modification History

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/11/2011

Use Case ID: PLI006 – Guest Charging Mechanism

Use Case Level: High-level

Details:

- **Actor:** System, User.
- **Pre-conditions:**
  1. System assigned a guest parking space to a user.
- **Description:**
  1. Use case begins when the guest arrives to the parking space.
  2. The system detects that the user successfully parked in the right parking space.
  3. Use case ends when the user has paid for parking and the system acknowledges it.
- **Post-conditions:**
  1. The system keeps track of the parking space time allowed for the user to be parked in a guess parking space.

Alternative Courses of Action:

None.

Exceptions:

1. System will not take payment at the time the Parking User tries to pay.

Related Use Cases:

Add more time to guest parking (PLI005).

Decision Support:

Frequency: On average, 150 guest users will park in FIU.

Criticality: High. The systems should acknowledge a user has paid and keep track of time.

Risk: Medium. Implementing this use case requires knowledge of Java with GUI.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: Screen must display data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/05/2011

Use Case ID: PLI007 – No Parking Spot Available.

Use Case Level: Low-level.

Details:

- **Actor:** System, User.
- **Pre-conditions:**
    1. System has successfully scanned the user ID.
- **Description:**
    1. Use case begins when the system recognizes there are not parking spaces available in the parking garage.
    2. The system shall notify the user to go the closest parking garage that has at least 5 parking spaces available.
    3. Use case ends when the user click “ok” on the screen.
- **Post-conditions:**
    1. System goes back to home screen.

Alternative Courses of Action:

None.

Exceptions:

1. The system checks for parking garage availability and determine it is full while it has some parking spaces available.
2. The system sends the user to go to another parking garage that has no parking space available.

Related Use Cases:

None

Decision Support:

Frequency: On average 100 users will face non-parking space availability.

Criticality: High. Allow the user to quickly find a parking space without wasting a lot of time.

Risk: Medium. Implementing this use case requires knowledge of java with GUI.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/05/2011

Use Case ID: PLI008 – ID Scan Failed.

Use Case Level: High-level

Details:

- **Actor:** System, User.
- **Pre-conditions:**
  1. System attempts to scan the ID.
- **Description:**
  1. Use case begins when the system recognizes the user does not have a valid ID.
  2. The systems shall ask the user if she/he is a guest.
  3. System will look to see if guest parking spots are available.
  4. The use case ends when the system prompts the User whether or not he/she wants to use the guest parking.
- **Post-conditions**:
  1. System treats the User as a guest.

Alternative Courses of Action:

1. In step 4 from description, user presses cancel when prompted to use a guest spot, and goes back to home screen and waits for the next user.
2. If no guest parking is available system will send user to another parking garage.
3. In step 2 from description, the user selects "no" when asked if user is a guest.

Exceptions:

1. User presses handicap button.

Related Use Cases:

Scanning ID (PLI002)

Decision Support

Frequency: On average, over 50 requests per day are made by the user.

Criticality: High. The system has to look up the User and check user type.

Risk: Medium. Implementing this use case requires knowledge of java with GUI.

Constraints:

- Usability: No previous Training Time
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLI009 – ID failed or no guest parking available.

Use Case Level: High-level

Details:

- **Actor:** System, User.
- **Pre-conditions:**
    1. System has failed scanned ID, and check for a guess parking space available.
- **Description:**
    1. Use case begins when the system finds there are not guess parking space available.
    2. System shall check for a parking garage that has guess parking spaces available.
    3. System finds a parking garage with guess parking spaces available.
    4. System notifies the user to go to the specific parking garage.
- **Post-conditions:**
    1. System goes back to home screen.
- **Alternative Courses of Action:**
    1. In step D.3 (step 3 of Description section) the system didn't find a parking garage with a guess parking space available, and notifies the user to come back later.

Exceptions:

1. The parking garage has guess parking space available.
2. The system sends the user to a parking garage that doesn't have guess parking space available.

Related Use Cases:

Scanning ID Failed (PLI008)

Decision Support:

Frequency: On average 50 users without proper ID will be facing no guess parking space availability.

Criticality: High. The system should find and display the different parking spaces options.

Risk: Medium. Implementing this use case requires knowledge of java with GUI.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/05/2011

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.
    2. The user can see in the garage display which spot has been assigned for him.
    3. 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:**
    1. 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: PLI012–Correct Parking Spot Scan

Use Case level: Medium-Level

Details:

- **Actor:** Parking User.
- **Pre-conditions:**
    1. User has arrived to the parking spot.
    2. The system has knowledge of which ID has been assigned to that spot.
- **Description:**
    1. When the user parks at a parking spot
    2. The system will scan the id
    3. The system will confirm the user has parked on the right spot by displaying “Parked”.
    4. Use case ends when the system has confirmed user parked on the right spot..
- **Post-conditions:**
    1. The System will display “Parked” until user leaves spot.

Alternative Courses of Action

1. If user parked on the wrong spot it will notify by displaying “wrong spot” and the number of the spot assigned to that user.
2. If user parked on the wrong spot it will notify the security officer.

Exceptions:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average 5 cars will park at a spot per day.

Criticality: High. It will confirm the user has parked at the right spot.

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

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for one year of use.
- Performance: On average, maintenance request takes about 30 minutes in a 24 hour period for
- Supportability: The displayed information should be clear and easy to understand for the user.
- 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: PLI013–Security Parking Spot Override

Use Case Level: High-Level.

Details:

- **Actor:** Security Officer.
- **Pre-conditions:**
    1. Parking Spot is not available for use.
- **Description:**
    1. The Security officer will access the parking spot database.
    2. The Security officer selects the spot
    3. Use case ends when security reserves the spot with the OK button.
- **Post-conditions:**
    1. The system will save the log with the action taken by security.
    2. The system will notify the security officer the spot has been reserved.

Alternative Courses of Action

None.

Exceptions:

None.

Related Use Cases: None

Decision Support

Frequency: On average 5 parking spots will be override by security by month

Criticality: High. It will reserve a spot that cannot be used.

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

Constraints:

- Usability: Security Officer should be able to understand the information on the message generated by the system.
- Reliability: Mean time to Failure – 5% failures for a year of use.
- Performance: The system should be able to reserve the spot within one second of confirmation.
- Supportability: The Security User interface should be clear and precise in order to reserve spot.
- Implementation: Using java 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.
    3. The system will Display "Wrong Spot" and the number of the correct parking spot assigned to that user.
    4. 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: PLI015 - Duplicate ID User Alert.

Use Case Level: High-level

Details:

- **Actor:** Parking user.
- **Pre-conditions:**
    1. System has successfully scanned the parking user ID.
- **Description:**
    1. Use case begins when the system has found another car currently parked with the same ID as the current parking user.
    2. The system shall display the information on screen that there is another car with the same ID currently using the system.
    3. The parking user shall acknowledge the existence of the duplicate ID.
    4. The parking user can then opt to be treated as a guest for the moment.
    5. Use case ends when the parking starts treating the system user as a guest user.
- **Post-conditions:**
    1. The system will treat the parking user as a guest user.
    2. The system will search for available guest parking spots for the parking user.

Alternative Courses of Action:

1. In step D.3 (step 3 of Description section) the user has the option to cancel the request and leave the garage.
2. In step D.4 the user has the option to cancel the request and leave the garage.

Exceptions:

The user decides to leave without replying to the information on the display screen.

Related Uses Case:

Search matching ID (PLI004).

Decision Support

Frequency: On average, 6 parking duplicate users will park at FIU at the same time per day.

Criticality: Medium. It allows the parking user to park even though someone already using that ID.

Risk: Medium.

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.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/25/2011

Use Case ID: PLI016 - No Faculty Spot Available.

Use Case Level: High-level

Details:

- **Actor:** Parking user.
- **Pre-conditions:**
    1. System has successfully scanned the user ID.
    2. The system has identified the user's ID as faculty type.
- **Description:**
    1. Use case begins when the system is not able to find an available faculty parking space for the faculty user.
    2. The system shall display information on screen that there are currently no faculty spots available in the garage.
    3. The faculty user shall accept the information in the screen
    4. The faculty user is then granted the option to be treated as a student for the moment.
    5. Use case ends when the system starts treating the faculty user as a student user.
- **Post-conditions:**
    1. The system will treat the user as a student user.
    2. The system will search for available student parking spots.

Alternative Courses of Action:

1. In step D.3 (step 3 of Description section) the user has the option to cancel the request and leave the garage.
2. In step D.4 the user has the option to cancel the request and leave the garage.

Exceptions:

1. The user decides to leave without replying to the information on the display screen.

Related Uses Case:

No Parking Available (PLI007).

Decision Support

Frequency: On average, 10 faculty users will not find available faculty parking spots per day.

Criticality: High. It allows the faculty users to park in other parking spots when there are no faculty spots available.

Risk: Medium.

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.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/30/2011

Use Case ID: PLI017 - No Student Spot Available.

Use Case Level: High-level

Details:

- **Actor:** Student user.
- **Pre-conditions:**
    1. System has successfully scanned the user ID.
    2. The system has identified the user's ID as student type.
- **Description:**
    1. Use case begins when the system is not able to find an available student parking space for the student user.
    2. The system shall display information on screen that there are currently no student spots available in the garage.
    3. The student user shall accept the information in the screen
    4. The student user is then granted the option to be treated as a guest for the moment.
    5. Use case ends when the system starts treating the student user as a guest user.
- **Post-conditions:**
    1. The system will treat the student user as a guest user.
    2. The system will search for available guest parking spots.

Alternative Courses of Action:

1. In step D.3 (step 3 of Description section) the student user has the option to cancel the request and leave the garage.
2. In step D.4 the student user has the option to cancel the request and leave the garage.

Exceptions:

1. The user decides to leave without replying to the information on the display screen.

Related Uses Case:

No Parking Available (PLI007).

Decision Support

Frequency: On average, 200 student users will not find available students parking spots per day.

Criticality: Medium. It allows the student users to park in other parking spots when there are no student spots available.

Risk: Medium.

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.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/30/2011

Use Case ID: PLI018 – Not Parked within Time Limit.

Use Case Level: High-level

Details:

- **Actor:** Parking user.
- **Pre-conditions:**
    1. System has successfully reserved a parking space for a user.
    2. System has successfully given the user directions to the parking spot.
- **Description:**
    1. Use case begins when system user accepts the directions to go to the assigned parking spot.
    2. The system shall start a 20 minute counter.
    3. 20 minutes must pass by without the system having successfully acknowledged that the user parked on the assigned parking spot.
    4. Use case ends when the system removes the user's ID from the reserved parking spot.
- **Post-conditions:**
    1. The number of available parking spaces increases by one.

Alternative Courses of Action:

In step D.2 (step 2 of Description section) the user can decide to not press this option.

Exceptions:

The user decides to leave without replying to the information on the display screen.

Related Uses Case:

Scan ID (PLI002), Display directions (PLI011).

Decision Support

Frequency: On average, 25 parking users will not park after being assigned a parking space per day.

Criticality: High. It allows the system to clear any parking that will not be used in order to allow new users to be able to park in that spot.

Risk: Medium.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for one day of use.
- Performance: On average, request takes about 15 seconds.
- Supportability: The screen must display data clearly enough for the user to read it.
- Implementation: Using java clients and servers with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/30/2011

Use Case ID: PLI019 – Handicap Button.

Use Case Level: High-level

Details:

- **Actor:** Handicapped user.
  - **Pre-conditions:**
    1. The system is not able to identify the type of user.
  - **Description:**
    1. Use case begins when the system treats the user as a guest.
    2. The system shall provide the user the option to be treated as a handicapped guest.
    3. The user has to press the handicapped button displayed on the screen.
    4. The system shall treat the user as a handicapped user and search for available parking spots.
    5. The user must park in the assigned parking spot.
    6. Use case ends when a notification is sent to the administrator check in the parking spot if the user has a handicapped parking permit.
  - **Post-conditions:**
    1. The number of notifications for the administrator increases by one.
    2. The number of available handicapped parking spots decreases by one.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) the user can decide to not press this option.

Exceptions:

1. The user decides to leave without replying to the information on the display screen.

Related Uses Case:

None.

Decision Support

Frequency: On average, there are 10 guest handicapped users using the system per day.

Criticality: High. Allows the guest handicapped user to be assigned a proper parking spot.

Risk: High.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for one day of use.
- Performance: On average, request takes about 15 seconds.
- Supportability: The screen must display data clearly enough for the user to read it.
- Implementation: Using java clients and servers with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/30/2011

Use Case ID: PLI020 – Wrong spot ID release.

Use Case Level: Medium-level

Details:

- **Actor:** System.
- **Pre-conditions:**
    1. The system has assigned a parking spot to the user.
    2. The user is heading towards the parking spot.
- **Description:**
    1. Use case begins when the system checks for parking accuracy and finds the wrong user on the parking spot assigned to another user.
    2. The user waits for the system to remove the assigned parking spot in order to be able to relocate to a different spot.
    3. System sends security officer informational message with the ID information of the incorrectly parked user.
    4. Use case ends when the system removes the previous user from the designated parking spot and sends message to the security officer.
- **Post-conditions:**
    - The system updates and gathers the information of the user who is parked in the wrong place.
    - Sends the administrator a message with the information of the user whom is parked incorrectly.

Alternative Courses of Action

1. In step 1 of the Description section, we can have the user whom parking spot was taken request to have a different spot immediately assigned and have the system issue an email confirmation to the security officer with the ID information of the wrongfully parked user.

Exceptions:

1. The sensors are not working correctly therefore the system never realizes that the incorrect user is parked on another user's parking spot.
2. The system is not able to redirect the user to another parking spot due to errors in software or unavailable parking spots.

Related Use Cases: System sends user information to security officer.

Decision Support

Frequency: On average 5 removals will be conducted by the system per hour.

Criticality: High. The system needs to remove the current user thus enabling another parking spot to be assigned.

Risk: High. If the system is not able to remove the user's information, it would be impossible to assign another parking spot.

Constraints:

- Usability: No previous training time.
- Reliability: Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
- Performance: The system should remove the user's information and send the informational message within 5 seconds.
- Supportability: The removal of the user information from the assigned parking spot must be handled by the program (Java).
- Implementation: Java software will be use to remove the assigned parking spot and send the informational message to the administrator.

Modification History

Owner: Team 5

Initiation date: 05/22/2011

Date last modified: 06/11/2011

Use Case ID: PLI021 - Guest parking time expired.

Use Case Level: High-level

Details:

- **Actor:** Parking User.
- **Pre-conditions:**
    1. The guest has parked and paid for certain amount of time.
- **Description:**
    1. Use case begins when the time expired for the parking.
    2. The system will notify the security officer.
    3. Use case ends when the security officer accept the notification.
- **Post-conditions:**
    - The information sent to the log and save.

Alternative Courses of Action:

1. The guest adds more time and the system will notify security officer.

Exceptions:

None.

Related Uses Case:

Add more time to guest spot (PLI005).

Decision Support

Frequency: On average, over 10 requests per day are made.

Criticality: High. The guest must pay for the parking spot.

Risk: High.

Constraints:

- Usability: No previous Training Time. Implemented by system.
- Reliability: Mean time to Failure – 10% failures for one year of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display data clearly enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/25/2011

Use Case ID: PLIS22 –User leaves parking spot.

Use Case Level: High-level

Details:

- **Actor:** Parking User
- **Pre-conditions:**
    1. User has left to the parking spot.
    2. The system has knowledge that the parking spot is empty now.
- **Description:**
    1. Use case begins when the user leaves the parking spot.
    2. The system will confirm the user has left the spot by displaying "available".
    3. Use case ends when the system has confirmed user left parking spot.
- **Post-conditions:**
    1. The system should increase the number of available parking spot.

Alternative Courses of Action:

None

Exceptions:

None.

Related Uses Case:

None.

Decision Support

Frequency: On average 20 cars will leave the a spot per hour.

Criticality: High. It will confirm the user has left the right spot.

Risk: Low.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The displayed information should be clear and easy to understand for the user.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLI023 – Emergency Alarm notification.

Use Case Level: High-level

Details:

- **Actor:** System
- **Pre-conditions:**
    1. Emergency alarm goes off and sends signal to the system
- **Description:**
    1. Use case begins when System goes into Emergency Response State
    2. The system detects the event and alarm was activated.
    3. After the emergency alarm goes off The system then sends a message to the Security Officer.
    4. Use case ends when the Security Officer sends signal to the system.
- **Post-conditions:**
    1. The request has been saved in the system and marked handled.

Alternative Courses of Action:

None

Exceptions:

1. The help link on the CS web page is not active.
2. After the user enters the required data the send button is not active.

Related Uses Case:

None.

Decision Support

Frequency: On average 10 requests are made daily by CS user.

Criticality: High. Allows the CS user to report any problems encountered with either computer software or hardware in the CS department.

Risk: Medium. Implementing this use case employs standard web-based technology.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every twenty four hours of operation.
- Performance:Request should be sent and saved within 5 seconds.
- Supportability: The displayed information should be clear and easy to understand for the user.
- Implementation: The send request form should be correctly handled by IE and Mozilla.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLI024 – Emergency vehicle access.

Use Case Level: High-level

Details:

- **Actor:** Emergency Personal
- **Pre-conditions:**
    1. There is a fire and the fire alarm is activated.
- **Description:**
    1. Use case begins the fire alarm was activated.
    2. Security officer acknowledges the system notification and call for emergency.
    3. Use case ends when emergency vehicle arrived.
- **Post-conditions:**
    1. The fire department and the security officer are alerted.

Alternative Courses of Action:

None.

Exceptions:

None

Related Uses Case:

Emergency Response System (PLI025).

Decision Support

Frequency: On average, 3 request per day.

Criticality: High. In case of emergency system needs to be able to alert.

Risk: High.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLI025 – Emergency Response System

Use Case Level: High-level

Details:

- **Actor:** System.
- **Pre-conditions:**
    1. There is a fire and the fire alarm is activated.
- **Description:**
    1. Use case begins when System goes into Emergency Response State
    2. The system detects that the fire alarm was activated.
    3. The system then sends a message to the Local Authorities
    4. The system then sends a message to the Administrator
    5. Use case ends when Administrator acknowledges the system notification.
- **Post-conditions:**
    1. The number of requests stored in the system has increased by one.
    2. The request has been saved in the system and marked not handled.

Alternative Courses of Action:

None.

Exceptions:

1. Unable to send message to local authorities.
2. Unable to send message to Administrator

Related Uses Case:

None.

Decision Support

Frequency: On average, over 10 requests per year are made.

Criticality: High. In case of emergency system needs to be able to alert

Risk: High.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 10% failures for one year of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display data clearly enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/14/2011

Date last modified: 05/25/2011

Use Case ID: PLI026 – Power Failure.

Use Case Level: High-level

Details:

- **Actor:** System, Security Officer.
- **Pre-conditions:**
    1. There is a battery backup installed
- **Description:**
    1. Use case begins when System goes into Emergency State
    2. The system shall save all current settings
    3. The system shall notify the Administrator about the power loss.
    4. Use case ends when the Administrator acknowledges the system message.
- **Post-conditions:**
    1. System will need to be restarted after power loss
    2. Requests done after power loss will be rejected

Alternative Courses of Action:

None.

Exceptions:

1. Power goes back on in less than a minute

Related Uses Case:

System Restart (PLI027).

Decision Support

Frequency: On average, over 10 requests per year are made

Criticality: High. System needs to be able to save data before battery runs out

Risk: Medium.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/25/2011

Use Case ID: PLI027 – System Restart.

Use Case Level: High-level

Details:

- **Actor:** Security Officer
- **Pre-conditions:**
    1. System is not responding
    2. System Error.
- **Description:**
    1. Use case begins when System becomes unresponsive
    2. Administrator will be required to authenticate him/herself as the administrator.
    3. Use case ends when the system saves current state and system restarts.
- **Post-conditions:**
    1. System will restart

Alternative Courses of Action:

1. System restart is not working and system will be restarted with a cool restart.

Exceptions:

None

Related Uses Case:

System Shutdown (PLI028)

Decision Support

Frequency: On average, over 5 requests per day are made by the user.

Criticality: High. Only authorized personnel should be able to restart the system.

Risk: Low.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/28/2011

Use Case ID: PLI028 – System Shutdown.

Use Case Level: High-level

Details:

- **Actor:** Security Officer.
- **Pre-conditions:**
    1. System is not responding
    2. System Error.
- **Description:**
    1. Use case begins when System becomes unresponsive
    2. Administrator will be required to authenticate him/herself as the administrator.
    3. Use case ends when the system saves current state and system shuts down.
- **Post-conditions:**
    1. System will shutdown

Alternative Courses of Action:

1. Administrator tries to restart the system

Exceptions:

None.

Related Uses Case:

System Restart (PLI027)

Decision Support

Frequency: On average, over 50 requests per day are made by the user.

Criticality: High. The system has to look up the User and check user type.

Risk: Low.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLI029 – Security handicap check

Use Case Level: High-level

Details:

- **Actor:** System.
- **Pre-conditions:**
    1. . There is a user at the entrance that needs parking
- **Description:**
    1. Use case begins when the user pressed the handicap option at the entrance and is granted access.
    2. A message is sent to the Administrator to go check if user parked in the handicap space and if user has government issued handicap tag
    3. Use case ends when the user is authenticated as handicap.
- **Post-conditions:**
    1. User parked the car

Alternative Courses of Action:

1. User wrong fully access the parking lot and is not handicap
2. User parked in another parking spot
3. User parked in handicap spot, is not handicap and is issued a ticket.

Exceptions:

None.

Related Uses Case:

Wrong parking security notification (PLI013), Wrong parking user notification (PLI0114).

Decision Support

Frequency: On average, over 50 requests per day are made by the user.

Criticality: High. The system has to look up the User and check user type.

Risk: Low.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/28/2011

Use Case ID: PLI030 – User Call for help.

Use Case Level: High-level

Details:

- **Actor:** System, User, Administrator
- **Pre-conditions:**
    1. System attempts to scan the ID.
- **Description:**
    1. Use case begins when the user presses the "Call for help" bottom on the screen.
    2. System will try to connect the user with the administrator.
    3. The use case ends when the user gets in contact with the Administrator.
- **Post-conditions**:
    1. System will go back to the home screen.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) the system cannot connect the user with the administrator, and notifies the user.

Exceptions:

- System doesn't try to connect the user with the administrator.

Related Use Cases: None.

Decision Support

Frequency: On average, over 50 requests per day are made by the user.

Criticality: High. The system has to connect the user with the administrator every time it is needed

Risk: Medium. System intercom will be implemented for this use case.

Constraints:

- Usability: No previous e Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Implementation: Using java with GUI to implement.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/05/2011

Use Case ID: PLIS01 – Unauthorized System Shutdown.

Use Case Level: High-level

Details:

- **Actor:** Unauthorized user
- **Pre-conditions:**

1. System is not responding
2. System Error.

- **Description:**

1. Use case begins when System goes into Emergency State
2. ParkingLotUser tries to click many buttons on the screen.
3. User manages to get to the system shutdown screen.
4. User is prompt for autorotation code.
5. Use case ends when User cannot proceed.

- **Post-conditions:**

1. There is a security code active to authenticate

Alternative Courses of Action:

1. Parking lot attendant comes over while User tries everything to shut down system.

Exceptions:

None.

Related Uses Case:

None.

Decision Support

Frequency: On average, over 10 requests per day are made by the user.

Criticality: High. The system has to look up the User and check user type.

Risk: High.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 5 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.

2. Security. Administrator needs to input credentials in order to shut down the system

---

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/07/2011

Use Case ID: PLIS02 – Emergency Vehicle access abuse.

Use Case Level: High-level

Details:

- **Actor:** Unauthorized user
- **Pre-conditions:**
  1. System is working correctly.
  2. The alarm is not triggered.
- **Description:**
  1. Use case begins when system goes into Emergency State
  2. User triggered or pulled the alarm without any cause.
  3. The security officer gets notification that the alarm has been triggered.
  4. Use case ends when the request is stored in the system
- **Post-conditions:**
  1. The number of requests stored in the system has increased by one.
  2. The request has been saved in the system and marked as abused by user.

Alternative Courses of Action:

1. Security officer checks or tries to find out why the alarm was triggered.

Exceptions:

3. The alarm is not working therefore not sounding or triggering any reports to the security officer.

Related Uses Case: None.

Decision Support

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

Criticality: High. The alarm issue has to be solved in order for the system to come back to normal.

Risk: High.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every month of use.

- Performance: On average, request to the security officer takes about 3 seconds.
- Supportability: The alarm must be compatible to interact with the software.

4. Security: Security officer needs to check the status of the fire alarm and the reasons it was triggered.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/11/2011

Use Case ID: PLIS03 – Guest Spot Abuse.

Use Case Level: High-level

Details:

- **Actor:** Parking User
- **Pre-conditions:**
    1. A car is found to be parked in a guess parking garage that was not assigned to that user or user didn't add time to the parking meter.
- **Description:**
    1. Use case begins when system detects that a guess parking space is taken when it should not be taken.
    2. System sends a notification to the parking administrator.
    3. Use case ends when the unauthorized user is identified and given a ticket
- **Post-conditions:**
    1. System changes the status of that parking space to taken.

Alternative Courses of Action:

None.

Exceptions:

1. The system incorrectly detects a car parked in a guess parking garage.
2. The system notifies the administrator to issue a ticket when there is not an unauthorized user parked in a guess parking space.

Related Uses Case: None.

Decision Support

Frequency: On average, over 20 unauthorized users will be found parked in a guess parking space.

Criticality: High. The system should check for guess parking availability frequently.

Risk: Medium. Implementing this use case requires knowledge of java.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.

5. Security: System will check all the guess parking spaces and compared to the database. If unauthorized user is detected, the parking administrator should give a ticket to the user.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 06/06/2011

Use Case ID: PLIS04 – Unauthorized Handicap User.

Use Case Level: High-level

Details:

- **Actor:** Unauthorized user.
- **Pre-conditions:**
    1. Unauthorized user presses the handicap button.
- **Description:**
    1. Use case begins when user go and park in handicap parking.
    2. The system notifies the security officer of new handicap user is parking.
    3. Use case ends when the security officer validate that the user have a handicap tag and authorized to park in the handicap section.
- **Post-conditions:**
    1. Security officer issue a ticket to the unauthorized user.

Alternative Courses of Action:

1. None

Exceptions:

- None

Related Uses Case:

PLI019.

Decision Support

Frequency: On average, 10 handicap violation per week.

Criticality: High. The system has to notify the security officer that new handicap is parking.

Risk: Low.

Constraints:

- Usability: No previous Training Time.
- Reliability: Mean time to Failure – 5% failures for every 24hrs of use.
- Performance: On average, request takes about 3 seconds.
- Supportability: The screen must display the data clearly and largely enough for the user to read it.
- Security: System will send notification to Security Officer when unauthorized user enters the spot.

Modification History:

Owner: Team 5

Initiation date: 05/15/2011

Date last modified: 05/23/2011

Use Case ID: PLIS05 – Stolen ID Security Alert.

Use Case Level: High-level

Details:

- **Actors:** Unauthorized user, 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 notifies 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.
- **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 presses 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

9.3 Appendix C – User Interfaces

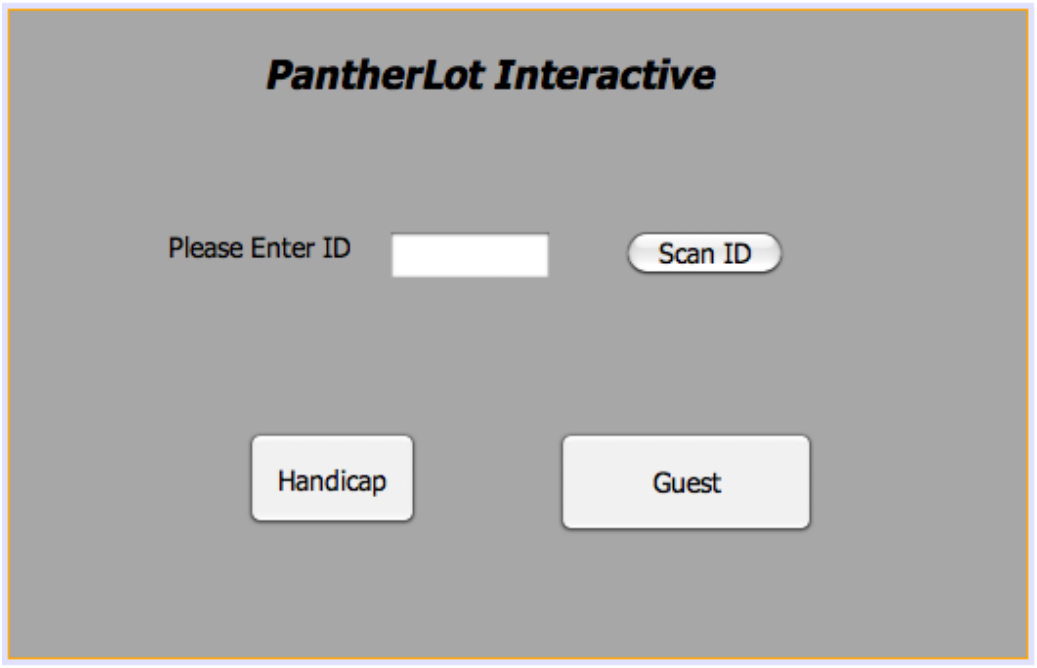

Figure C.1: Parking User Welcome Screen (PLI003)
Description: This UI allows multiple Users to interact with the system according to their needs.

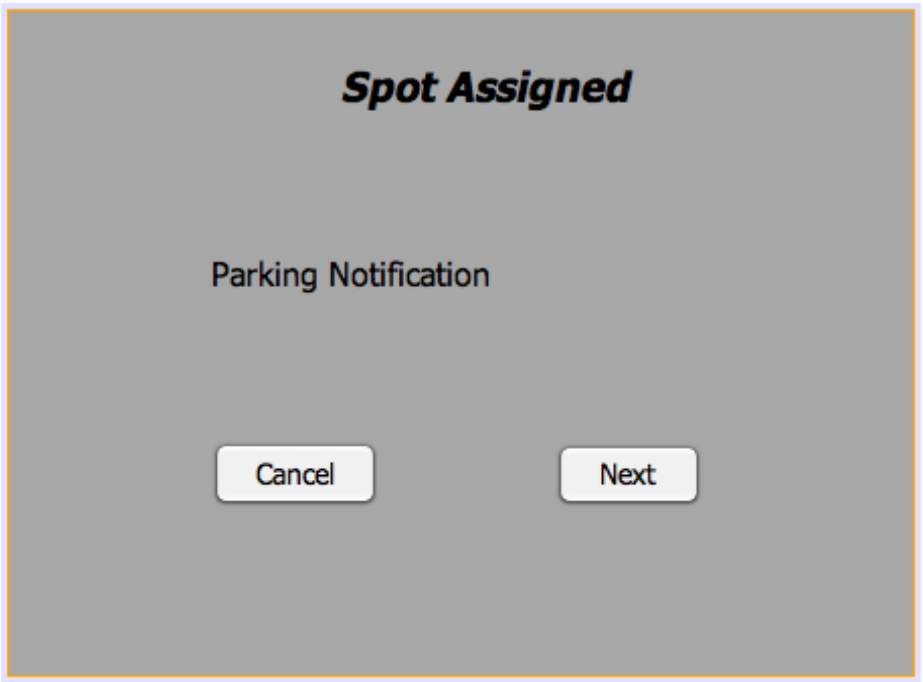

Figure C.2: Parking Spot assigned Display (PLI010)
Description: This UI will prompt after system can determine a parking based on User credentials

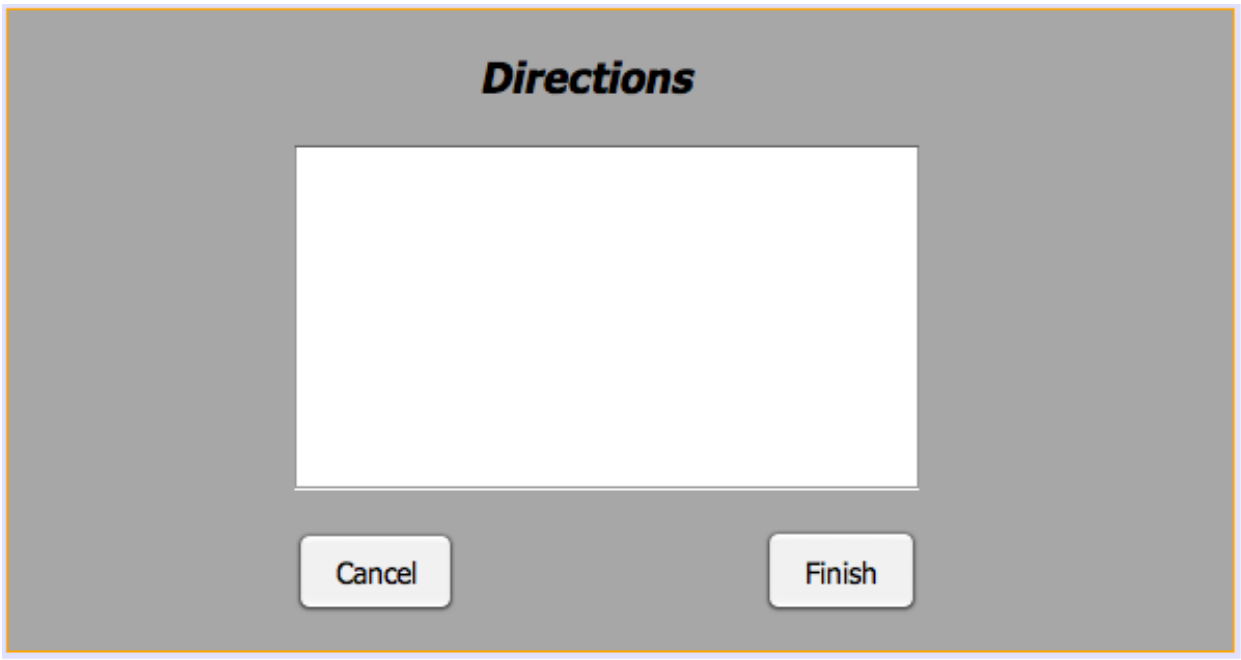

Figure C.3: Display Directions (PLI011)
Description: This UI allows Parking User to review where the assigned parking spot is located.

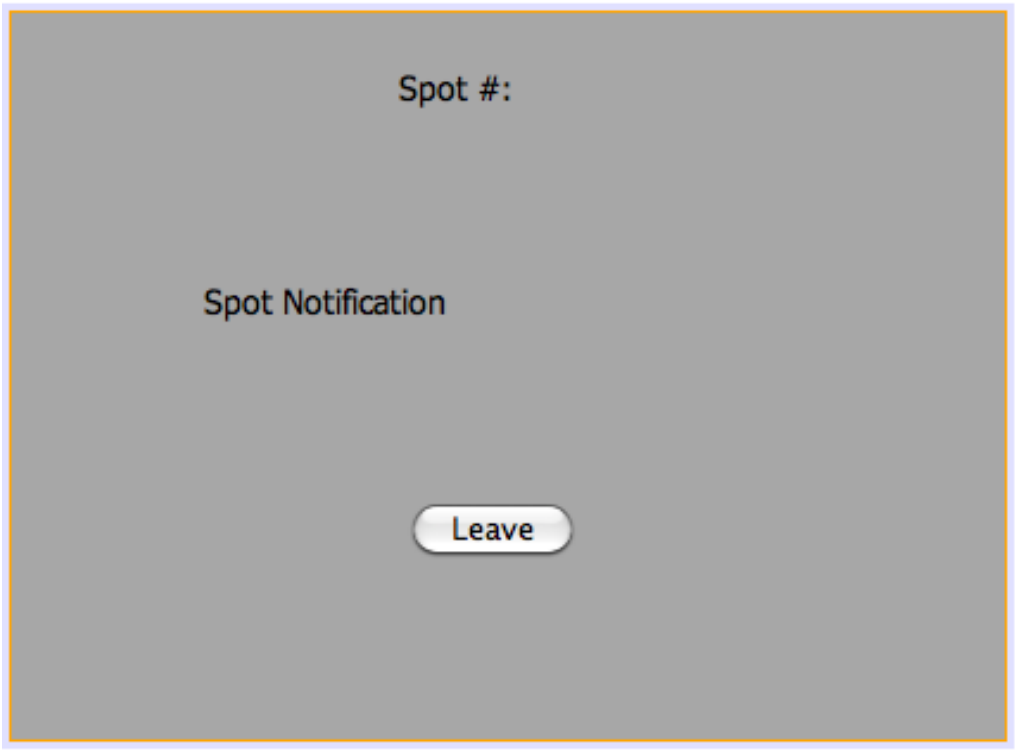

Figure C.4: Correct Parking Spot Scan (PLI012)
Description: This UI will confirm that Parking User is at the right location.

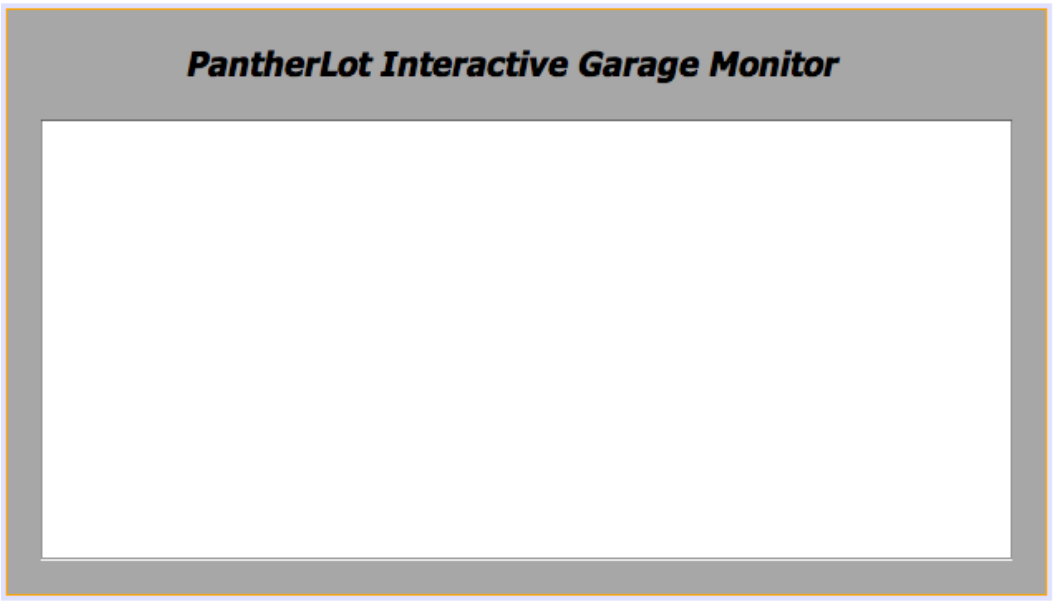

Figure C.5: Stolen ID Security Alert (PLIS05)
Description: This UI will be used by the Security Officer to do multiple tasks in the system.

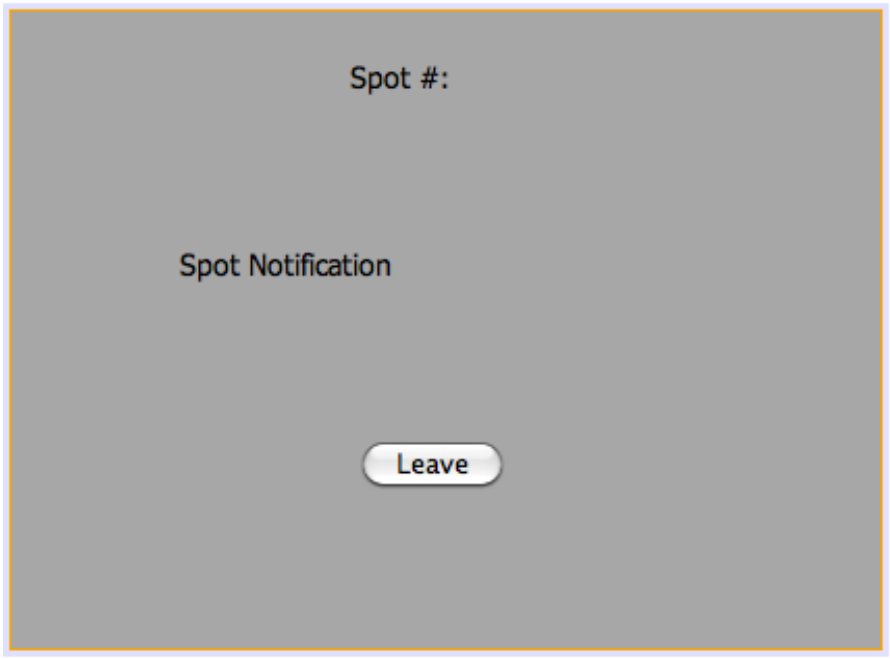

Figure C.6: Wrong parking security notification (PLIS06)
Description: This UI will be used by the Security Officer to do multiple tasks in the system.

9.4 Appendix D – Detailed Class Diagram

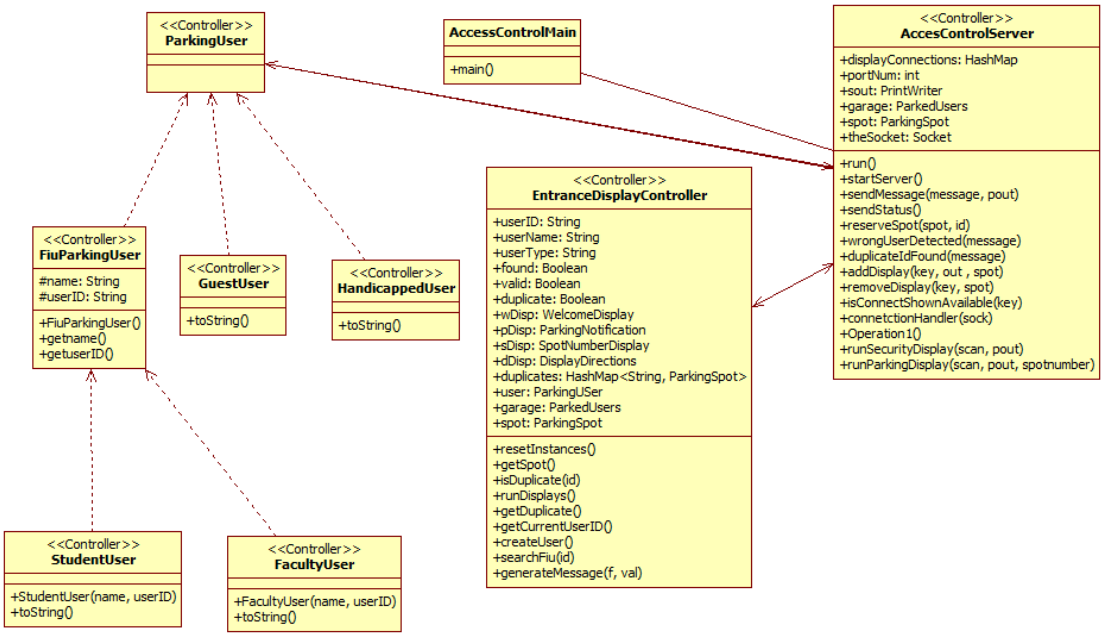

Fig D.1 PantherLot Interactive Controller Subsystem Detailed Class Diagram

This is the controller subsystem, here is the layer that contains the server communicating to all the clients and it also contains all the entity objects.

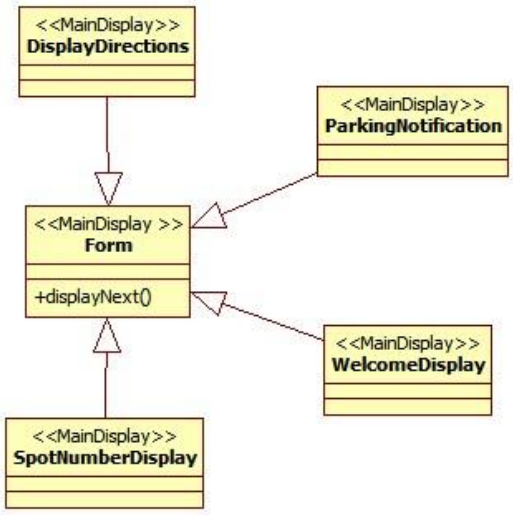

Fig D.2 PantherLot Interactive MainDisplay Subsystem Detailed Class Diagram

This is the subsystem that contains the GUI for the main display subsystem, all of the frame screens that the user sees are created here.

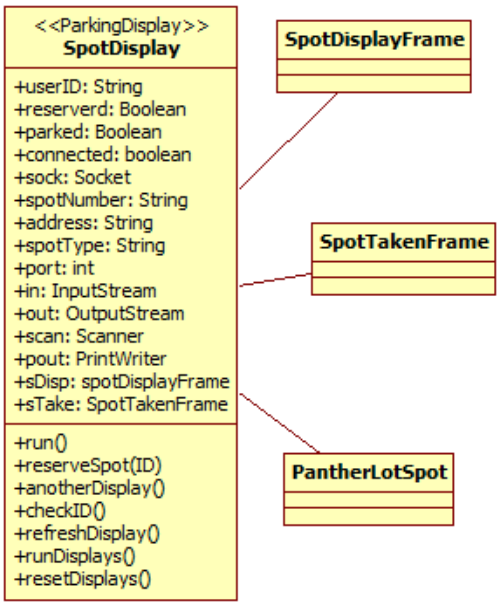

Fig D.3 PantherLot Interactive ParkingDisplay Subsystem Detailed Class Diagram

This is the subsystem that handles the client connection from each parking spot to the main server and it also contains the GUI classes for the Parking Spot Display.

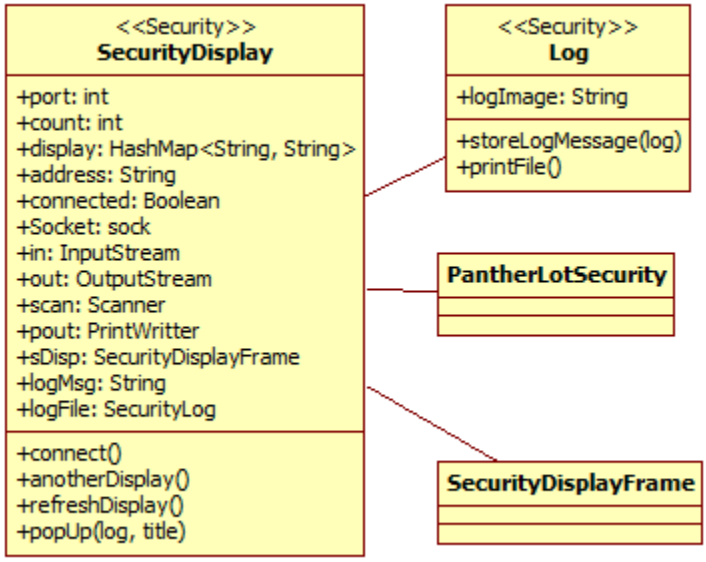

Fig D.4 PantherLot Interactive SecurityDisplay Subsystem Detailed Class Diagram

This is the subsystem that contains the GUI for the security display and this subsystem also handles the connection to the Main server.

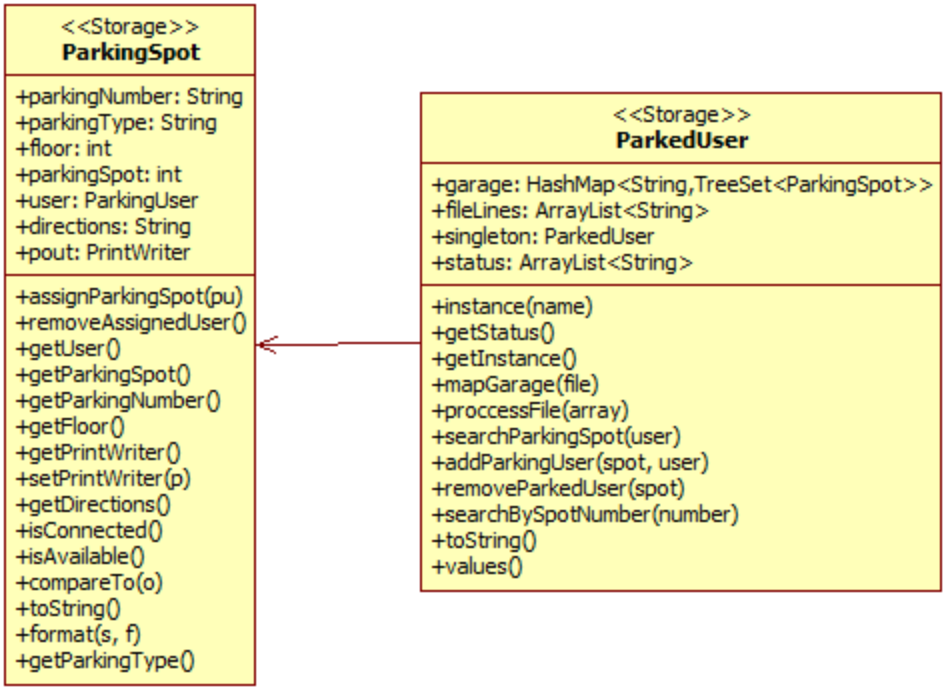

Fig D.5 PantherLot Interactive Storage Subsystem Detailed Class Diagram

This is the subsystem that handles the storage of data, here is where the garage file is mapped and the parking spot objects are created

9.5 Appendix E – Class Interfaces

Link to PantherLot Interactive Main Subsystem javadoc: PantherLotMain

Link to PantherLot Interactive Security Subsystem javadoc: PantherLotSecurity

Link to PantherLot Interactive Spot Subsystem javadoc: PantherLotSpot

9.7 Appendix G – Diary of meeting and tasks