

**Software Engineering Design I**

**Winter 2015**

**Assignment 2 – Software Specifications Document**

**Group #1**

**PRMS**

**PROFESSIONAL RESOURCE MANAGEMENT  
SYSTEM**

**Contributing Members:**

**Abad H., Brandon C. and Rafael N.**

**Date Submitted: 2/26/2015**

# Approval

This document has been read and approved by the following team members responsible for its implementation:

Print Name	Signature	Comments

Print Name	Signature	Comments

Print Name	Signature	Comments

# Document Status Sheet

<b>Document Title</b>	Software Specifications Document
<b>Document ID</b>	PRMS/Documents/Product/SSD/1.0.0
<b>Author(s)</b>	A. Hameed, B. Crispino, R. Nazario
<b>Version</b>	1.0.5
<b>Document Status</b>	draft / internally approved / <u>conditionally approved</u> / approved

<b>Version</b>	<b>Date</b>	<b>Author(s)</b>	<b>Comments</b>
0.0.1	29-01-2015	A. Hameed	Document Creation
0.0.2	08-02-2015	R. Nazario	Added Use-Case Diagram and step-by-step descriptions
0.0.3	08-02-2015	A. Hameed, R. Nazario	Added text to chapters 1, 2 and 4
0.0.4	08-02-2015	A. Hameed	Added text to chapters 5 and created project plan and deliverables
0.0.5	08-02-2015	B.Crispino	Added UML Class Diagrams, GUI Design and text to chapters 2 and 4
0.0.6	09-02-2015	A.Hameed	Added Gantt Chart to chapter 5
0.0.7	09-02-2015	A.Hameed, B.Crispino	Document Revision
1.0.0	09-02-2015	A. Hameed	Fixed document status
1.0.1	19-02-2015	A. Hameed	Recreated use case diagrams and re-wrote use case descriptions.
1.0.2	20-02-2015	B. Crispino	Recreated class and sequence diagrams and re-wrote software architecture
1.0.3	22-02-2015	A. Hameed	Made changes to chapters 2, 3 and 5
1.0.4	24-02-2015	R. Nazario	Added text to chapter 4
1.0.5	25-02-2015	A. Hameed	Document Revision Finalized

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Purpose .....	6
1.2	Scope .....	6
1.3	Environment .....	6
1.4	List of definitions and abbreviations .....	6
1.4.1	Definitions.....	6
1.4.2	Abbreviations .....	7
1.5	Documents .....	8
1.5.1	Reference Documents .....	8
1.5.2	Applicable Documents.....	8
<b>2</b>	<b>Specific Requirements.....</b>	<b>9</b>
2.1	Functional Requirements .....	9
2.1.1	Inputs/Outputs.....	9
2.1.2	Computations .....	9
2.1.3	Data Storage.....	9
2.1.4	Synchronization Issues.....	9
2.2	Non-functional Requirements.....	10
2.2.1	System Requirements.....	10
2.2.2	Response Time.....	10
2.2.3	Storage Requirements .....	10
2.2.4	Reliability/Availability .....	10
2.2.5	Privacy/Security/Safety.....	10
<b>3</b>	<b>Development Overview .....</b>	<b>11</b>
3.1	CASE Tools .....	11
3.1.1	Front-end.....	11
3.1.2	Back-end .....	11
3.1.3	Development Platform .....	11
3.1.4	Version/Configuration Control .....	11
<b>4</b>	<b>Product Specifications.....</b>	<b>12</b>
4.1	UML Client-Side Functional Model.....	12
4.1.1	Client-Side Use Case Description.....	13

4.2	UML Server-Side Functional Model.....	16
4.2.1	Client-Side Use Case Description.....	17
4.3	Software Architecture.....	20
4.3.1	Client Side Classes.....	20
4.3.2	Server Side Classes .....	21
4.4	UML Class Diagrams .....	22
4.5	Database Models and Diagrams .....	23
4.6	Sequence Diagrams .....	23
4.7	Communication Protocols / Messaging Formats.....	25
4.8	User Interface (GUI) Design .....	26
<b>5</b>	<b>Project Management Plan .....</b>	<b>28</b>
5.1	Project Organization .....	28
5.1.1	External Structures.....	28
5.1.2	Internal Structures.....	28
5.1.3	Roles and Responsibilities .....	28
5.2	Managerial Process Plans .....	28
5.2.1	Development Effort .....	28
5.2.2	Resource Acquisition Plan.....	30
5.3	Work Plan .....	30
5.3.1	Work Activities and Schedule Allocation.....	30
5.3.2	Project Gantt Chart .....	31
5.3.3	Risk Management Plan .....	32
5.4	Technical Process Plans.....	32
5.4.1	Process Model.....	32
5.4.2	Methods, Tools and Techniques .....	32
5.4.3	Infrastructure Plans .....	32
5.4.4	Configuration Management Plan .....	32
5.4.5	Testing Plan .....	32
5.4.6	Documentation Plan.....	32

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to specify the software requirements and a logical model of the Professional Resource Management System (PRMS) in a clear and consistent manner.

## 1.2 Scope

The software will be used to implement a client-server resource acquisition system. This system allows users to access a database of library resources from which they can acquire articles relevant to their needs. It hides the complexity of a resource acquisition system making the software easy to use. Usability is further increased by offering a front-end GUI for users to access the system.

## 1.3 Environment

The environment identified for the PRMS system is similar to a library system. The software will use keywords to search and acquire matching resources from the database with the capability to perform various sorting algorithms on the data. Being a client-server system, all requests will be sent to a server for acknowledgement.

## 1.4 List of definitions and abbreviations

This section contains the definitions of all used terms, acronyms and abbreviations in this document.

### 1.4.1 Definitions

Algorithm	A process or set of rules to be followed in calculations or other problem-solving operations.
Client	Program that is used by all the users and system admins.
ENGI-3050	The Software Engineering Design I Course ID at Lakehead University.
PRMS Software	A software implementing a client-server resource acquisition system.
Resource	Material which can be readily drawn from a server and displayed to the client for effective use.
Server	A computer or computer program that manages access to a centralized resource or service in a network (i.e. PRMS database)

System Administrator	The system admin oversees the entire PRMS system and has the right to configure the system, to create and remove admins and resources as well as any other high level configuration.
User	Any individual or system administrator with access to the client machine using PRMS.

### 1.4.2 Abbreviations

CASE Tools	Computer-aided Software Engineering Tools
COCOMO	Constructive Cost Model
GUI	Graphical user Interface
IDE	Integrated Development Environment
IP	Internet Protocol
KLOC	Thousands (K) of Lines of Code
LOC	Lines of Code
LU	Lakehead University
PR-<#>	Priority Level, <#> can range from 1 (highest) to 5 (lowest)
PRMS	Professional Resource Management System
SR	Software Requirements
SRD	Software Requirements Document
SVN	Apache Subversion
SWE	Software Engineering
TCP	Transmission Control Protocol
TFS	Team Foundation Server
UML	Unified Modeling Language
XML	eXtensible Markup Language

## 1.5 Documents

### 1.5.1 Reference Documents

[SWE Design I Project Outline]	Naser, H. (2015). Team-project: client-server resource acquisition system [class handout]. Department of Engineering, Lakehead University, Thunder Bay, ON.
[SWE 3670 Textbook]	Schach, S. (2011). <i>Object-Oriented and Classical Software Engineering</i> (8th ed., p. 667). New York: McGraw-Hill Education.
[SWE 3050 Textbook]	Lethbridge, T., & Laganier, R. (2005). <i>Object-Oriented Software Engineering</i> (2nd ed., p. 533). Maidenhead, Berkshire: McGraw-Hill Education.

### 1.5.2 Applicable Documents

[SW Requirements Document]	Hameed, A. Crispino, B., Nazario, R. (2015). PRMS [requirements document]. Department of Engineering, Lakehead University, Thunder Bay, ON.
----------------------------	---



## 2 Specific Requirements

In this section, an overview of the requirements and constraints of the developing product are given. The product will adhere to these requirements. Each of the requirements is preceded by a unique identifier for efficient traceability. (*Refer to chapter 4 for further detail*)

### 2.1 Functional Requirements

#### 2.1.1 Inputs/Outputs

(SR-2101) The PRMS system will have a user-friendly GUI which will enable the user to interact with the server which in turn is connected to a database. The user will specify one or more of the following: document name, document type, subject, publication year and/or author, to search and filter the system for possible resources. The system in turn will acquire found resources and display them to the user who may then sort the results. For example, if the user wants resources regarding ‘Thermodynamics’ by an author ‘Y. Cengel’, they would key in the word ‘Thermodynamics’ as the subject and/or ‘Cengel’ as the author. The server in response to the request will execute a search on the database and acquire all resources marked with the words ‘Thermodynamics’ and/or ‘Cengel’ and display them to the user. The user may then sort acquired results by document type, subject, publication year or author.

#### 2.1.2 Computations

(SR-2102) The PRMS system will allow users to search, filter and sort acquired resources by document name, document type, subject, publication year and author.

#### 2.1.3 Data Storage

(SR-2103) The PRMS system will require a server to keep between 150 and 200 resources and related information within it. A simple hard drive with 256 MB of space will be sufficient storage for this system.

#### 2.1.4 Synchronization Issues

(SR-2104) Synchronization will be checked and clarified in upcoming documentation. There is no possibility to identify any issues in this phase of the project.

#### 2.1.5 Essential UML Diagrams

(SR-2105) Various UML diagrams (use case, class, sequence, state, inheritance/association) are used to give an overview or graphical perspective of how and what will be involved in implementing the complete PRMS system (*see section 4*).

## 2.2 Non-functional Requirements

### 2.2.1 System Requirements

(SR-2201) With PRMS being a client-server system, a dedicated server is required with the ability to handle multiple requests at a given time. A client machine is required to receive acknowledgement from the server and acquire requested resources. Below are some of the recommended system requirements outlined for this system:

- Windows 7 (or higher)
- Intel Core i3 (or higher)
- 4 GB RAM: to ensure the optimal user experience
- 256 MB hard disk space: to provide ample space for an expanding database
- 2 MBit/s network: to minimize lag time during client/server communication

### 2.2.2 Response Time

(SR-2202) In the list of requirements provided there is no specification with regards to the response time. With PRMS being a client-server system, network dependency is a factor and can lengthen resource acquisition time.

### 2.2.3 Storage Requirements

(SR-2203) Customer requirements had no indication with regards to storage requirements. With 150-200 resources required, an approximation of 250MB of storage has been defined.

### 2.2.4 Reliability/Availability

(SR-2204) The PRMS system will be made robust and capable of handling multiple server requests from multiple users simultaneously. As mentioned in 2.2.1, as PRMS is a client-server, it has a dependency on network usage and may incur lag time as a result. In any other scenario, the PRMS system will be made to sustain heavy use ensuring its reliability and availability.

### 2.2.5 Privacy/Security/Safety

(SR-2205) System administrators will have full admin rights to the PRMS software and will solely be in-charge of back-end configurations. General users will have front-end access to the system database, simply to acquire resources for their academic purposes. Each user including the system administrators will be required to login with an authenticated username and password. The login information including user passwords will be marked as hidden for privacy and security purposes. A system timeout can be put in place to logoff a user due to account inactivity. There are no notable safety issues coupled with this software.

## 3 Development Overview

### 3.1 CASE Tools

This section will guide you through a variety of CASE tools which may be used to help implement this project from a development perspective.

#### 3.1.1 Front-end

A GUI will be implemented with an easy-to-use interface allowing general users and system administrators to maneuver through and use the PRMS system appropriately and efficiently. Back-end computations and complex algorithms will not be shown on the front-end of the system. A guided tutorial will be implemented for first-time users.

#### 3.1.2 Back-end

The connection between client-server will be implemented using socket programming. Various searching and sorting algorithms will be put in place to allow for an easier and more efficient user experience.

#### 3.1.3 Development Platform

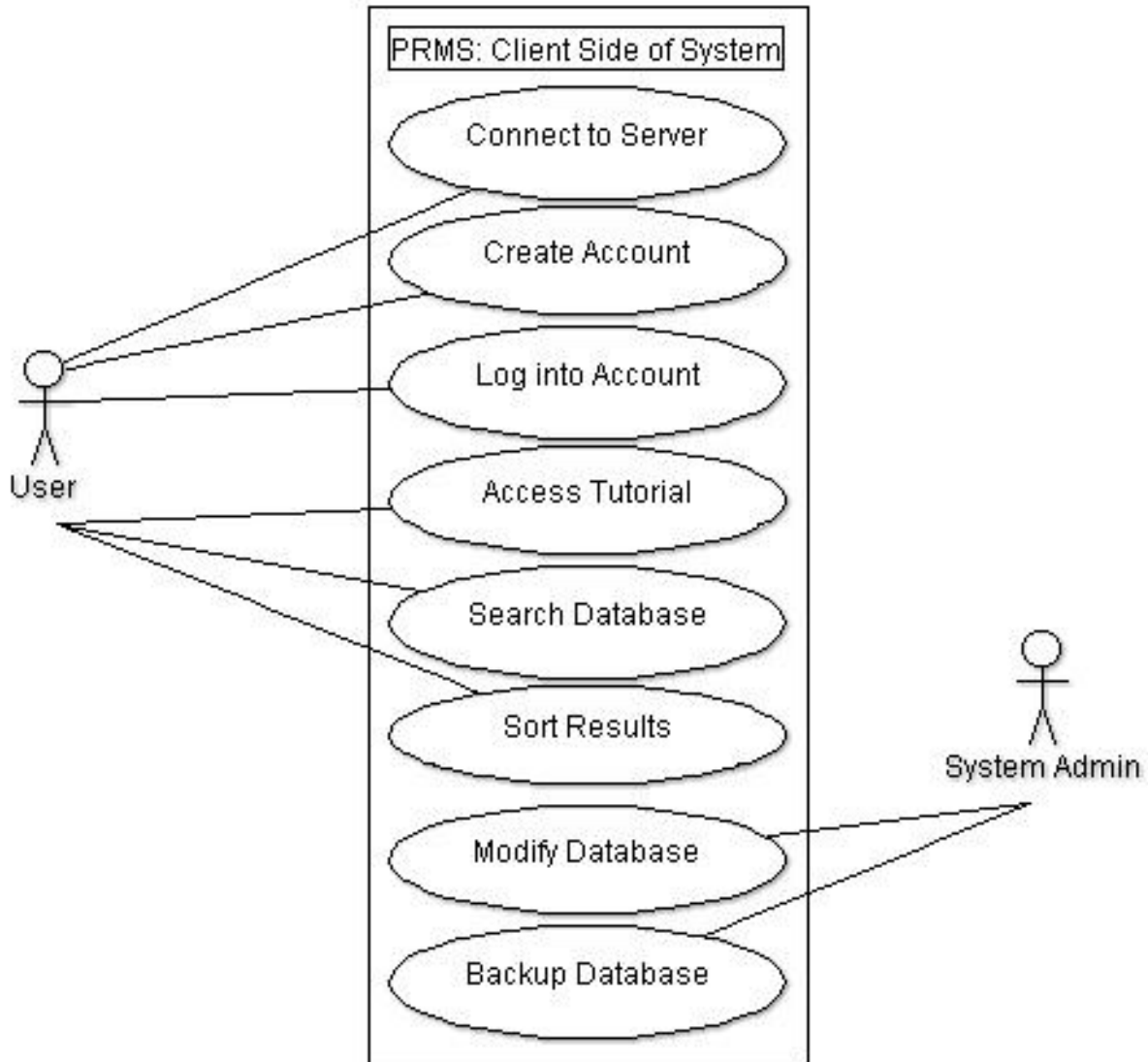
The major development of the PRMS system will be implemented using C++. The database will be a list of resources constructed within an XML document which will then be read into the C++ program to be used.

#### 3.1.4 Version/Configuration Control

The project will be document-oriented and every change will be tracked by group members. Revision control tools such as SVN, TFS or GIT will be used to ensure program traceability and group/task effectiveness.

## 4 Product Specifications

### 4.1 UML Client-Side Functional Model



#### 4.1.1 Client-Side Use Case Description

##### **Brief Description**

The **Connect to Server** use case is a process where the user will enter the IP address and port number of the server they wish to establish a connection with.

##### **Step-by-Step Description**

For initial set-up of client system:

1. User will be prompted to establish connection to a server.
  - 1.1. User will be requested to enter:
    - 1.1.1. Server IP address
    - 1.1.2. Server port number (i.e. 0 to 65535)
2. A request to establish connection with the specified server will made.

##### **Brief Description**

The **Create Account** use case will have the user enter an email address and password and send these parameters to the server to create the user account.

##### **Step-by-Step Description**

For every user:

1. User will be prompted to create an account.
  - 1.1. User will be requested to create:
    - 1.1.1. Username: valid lakeheadu email address ([xxx@lakeheadu.ca](mailto:xxx@lakeheadu.ca))
    - 1.1.2. Password: can be alphanumeric, min. 6 characters
2. A request to create account with the specified information will be made to the server.

##### **Brief Description**

The **Log into Account** use case will require the user to enter their registered email address and password to request the server for access into the system.

##### **Step-by-Step Description**

For every registered user:

1. User will be prompted to log in to their account.
  - 1.1. User will be requested to enter:
    - 1.1.1. Registered '@lakeheadu.ca' username ([xxx@lakeheadu.ca](mailto:xxx@lakeheadu.ca))
    - 1.1.2. Their personal password
2. Server request to validate login information and grant necessary access will be made.
3. Server will check account for general or system admin privileges as well.

**Brief Description**

The **Access Tutorial** user case will request the server for access to the software tutorial.

**Step-by-Step Description**

For every registered user:

1. User must be logged into their account
  - 1.1. User will be given an option to access/view the PRMS system tutorial
2. A request to the server will be made to acquire the tutorial material for user

**Brief Description**

The **Search Database** use case will require the user to enter search parameter(s)/keyword(s) and various data filtering options, all of which will be sent to the server to acquire related resources.

**Step-by-Step Description**

For every registered user:

1. User will be prompted to search and filter through resources by specifying some or all of the following:
  - 1.1. Document name (i.e. 'C++ How To Program')
  - 1.2. Document type (i.e. 'Textbook')
  - 1.3. Subject (i.e. 'Programming')
  - 1.4. Publication year (i.e. '2012')
  - 1.5. Author (i.e. 'Deitel')
2. The search/filter specifications will be sent to the server to acquire all related resources

**Brief Description**

The **Sort Results** use case will require the user to specify their sorting requirements which will be sent to the server to be applied and acquire updated resources.

**Step-by-Step Description**

For every registered user:

1. Upon acquiring search results, user will be shown various options to sort by:
  - 1.1. Document type
  - 1.2. Subject
  - 1.3. Publication year
  - 1.4. Author
2. A request to the server will be made to sort the results based on the specified option.

**Brief Description**

The **Modify Database** use case will enable a system administrator only, to request access and perform modifications to the system database.

**Step-by-Step Description**

For system administrator only:

1. User must be logged in and have system administrator privileges.
  - 1.1. System admin will have access to the server database with which they may:
    - 1.1.1. Create/Add new resource(s) to the database
    - 1.1.2. Edit/Modify existing resource(s) within the database
    - 1.1.3. Delete/Remove existing resource(s) from the database
2. All modifications made will be sent to the server to request for implementation.

**Brief Description**

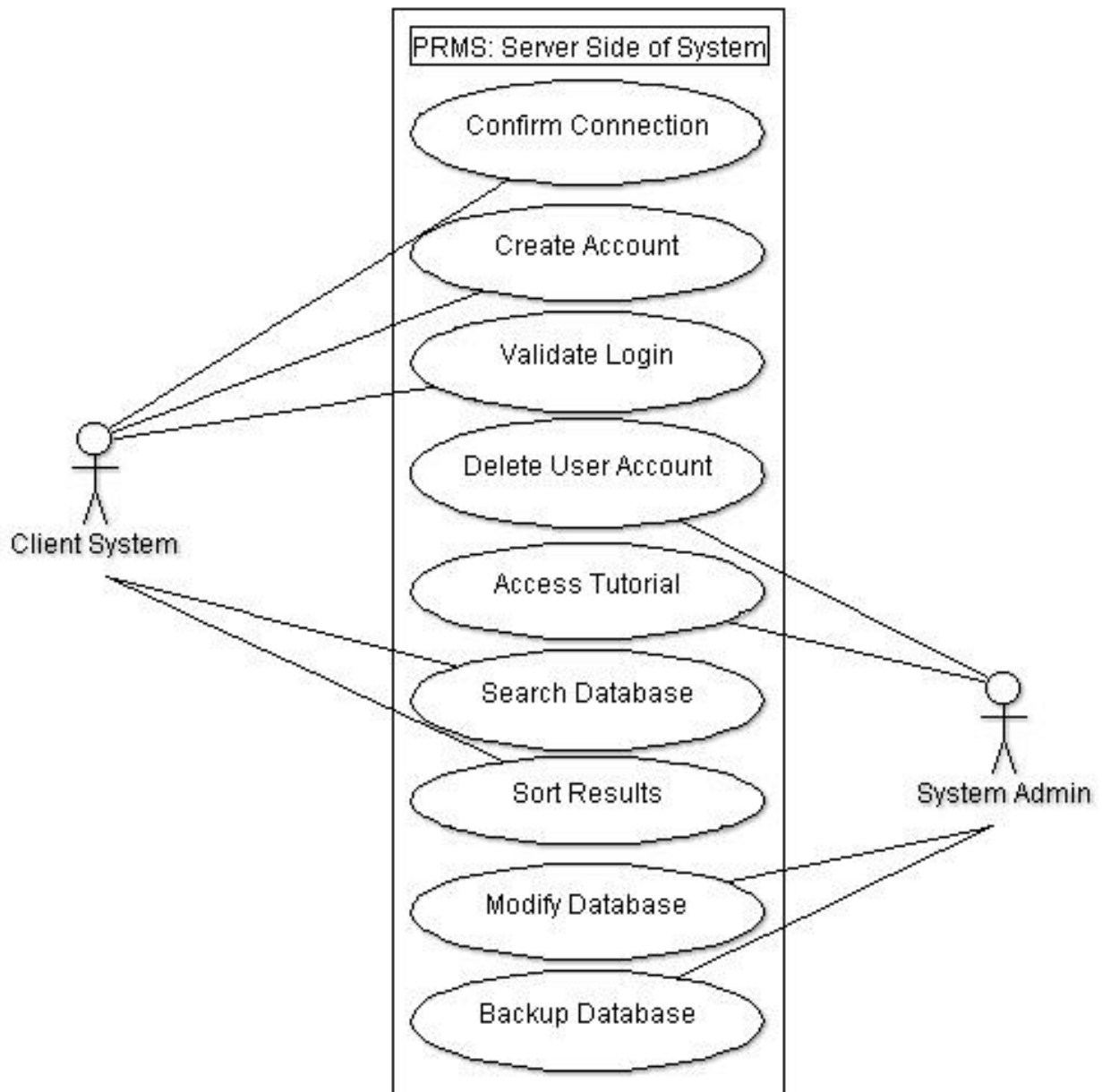
The **Backup Database** use case will enable a system administrator only, to request the server access to the database in order to create a backup file.

**Step-by-Step Description**

For system administrator only:

1. User must be logged in and have system administrator privileges.
  - 1.1. System admin will have read/write access to the database on the server.
2. A request to backup database will be sent to the server.

## 4.2 UML Server-Side Functional Model





#### 4.2.1 Client-Side Use Case Description

##### **Brief Description**

The **Confirm Connection** use case receives a request to establish a connection from a client.

##### **Step-by-Step Description**

For initial set-up of individual client systems:

1. Server will receive a request to establish a connection from client system.
  - 1.1. Server will auto-learn:
    - 1.1.1. Client system IP address
    - 1.1.2. Client system port number (i.e. 0 to 65535)
2. Client's request will be acknowledged and a connection will be established.

##### **Brief Description**

The **Create Account** use case will receive a valid request from the client system to create an account for a user, which will be used to access the PRMS system.

##### **Step-by-Step Description**

For every user:

1. Server will receive a request to create a user account from a client system.
  - 1.1. Client system will provide the user's:
    - 1.1.1. Username: valid lakeheadu email address ([xxx@lakeheadu.ca](mailto:xxx@lakeheadu.ca))
    - 1.1.2. Password: can be alphanumeric, min. 6 characters
  - 1.2. System administrators may grant admin access to the account being created.
2. Server will create the account and send confirmation to the client system.

##### **Brief Description**

The **Validate Login** use case will receive an email address and password from the client system, which upon validation will allow the user access to their account.

##### **Step-by-Step Description**

For every registered user:

1. Server will receive a request from the client system to validate user login.
  - 1.1. Client system will provide the user's:
    - 1.1.1. Registered '@lakeheadu.ca' username ([xxx@lakeheadu.ca](mailto:xxx@lakeheadu.ca))
    - 1.1.2. Valid account password
2. Server will validate the user's login as well as access rights and grant/deny entry into the system accordingly.

**Brief Description**

The **Delete User Account** use case enables the system administrator to remove a specific user account from the PRMS system.

**Step-by-Step Description**

For system administrator only:

1. User must be logged in and have system administrator privileges.
  - 1.1. System admin will have access to all stored user accounts within which they may delete any and all that they require.
2. Server stored accounts list will be updated following any modification.

**Brief Description**

The **Access Tutorial** use case will grant the client system, access to the tutorial upon request which will display to the user the basic functionality and a 'how-to' of the PRMS system.

**Step-by-Step Description**

For every registered user:

1. Server will receive a request from the client system to access the PRMS tutorial.
  - 1.1. Account type of the user requesting help will be found.
2. Access to the tutorial corresponding to the user's account type (general user or system administrator) will be granted to the client system.

**Brief Description**

The **Search Database** use case receives certain parameter(s)/keyword(s) from the client system which will be used to search and acquire resources from the database and be sent back to the client system.

**Step-by-Step Description**

For every registered user:

1. Server will receive a search request with one or more possible filters specified:
  - 1.1. Document name (i.e. 'C++ How To Program')
  - 1.2. Document type (i.e. 'Textbook')
  - 1.3. Subject (i.e. 'Programming')
  - 1.4. Publication year (i.e. '2012')
  - 1.5. Author (i.e. 'Deitel')
2. Server will run a search algorithm on the imported database with the specified parameter(s) and acquire all found results.
3. Acquired results will be sent back to the client system to be displayed to user.

**Brief Description**

The **Sort Results** use case will receive sorting specifications from the client system which will be applied to the previously acquired search results. The updated/sorted set of resources will then be sent back to the client system.

**Step-by-Step Description**

For every registered user:

1. Server will receive a request with one of the following sorting options specified:
  - 1.1. Document type
  - 1.2. Subject
  - 1.3. Publication year
  - 1.4. Author
2. Server will run a sort algorithm on the client systems acquired results.
3. Sorted list of resources will be sent back to the client system and displayed to the user.

**Brief Description**

The **Modify Database** use case will enable a system administrator only to perform modifications (add/edit/delete) to the database.

**Step-by-Step Description**

For system administrator only:

1. Server will receive a request to implement one or more of the following functions on the imported database:
  - 1.1. Create/Add new resource(s) to the database
  - 1.2. Edit/Modify existing resource(s) within the database
  - 1.3. Delete/Remove existing resource(s) from the database
2. All modifications will be implemented and a confirmation will be sent back to the client system for the system administrator.

**Brief Description**

The **Backup Database** use case will enable a system administrator only to access and perform a backup of the system database.

**Step-by-Step Description**

For system administrator only:

1. Server will receive a request to back up the current database.
2. Server will create an output .XML file containing all existing resources and save it within a secure folder on the client system.

## 4.3 Software Architecture

Below are detailed definitions of the UML Class diagrams displayed in section 4.4.

### 4.3.1 Client Side Classes

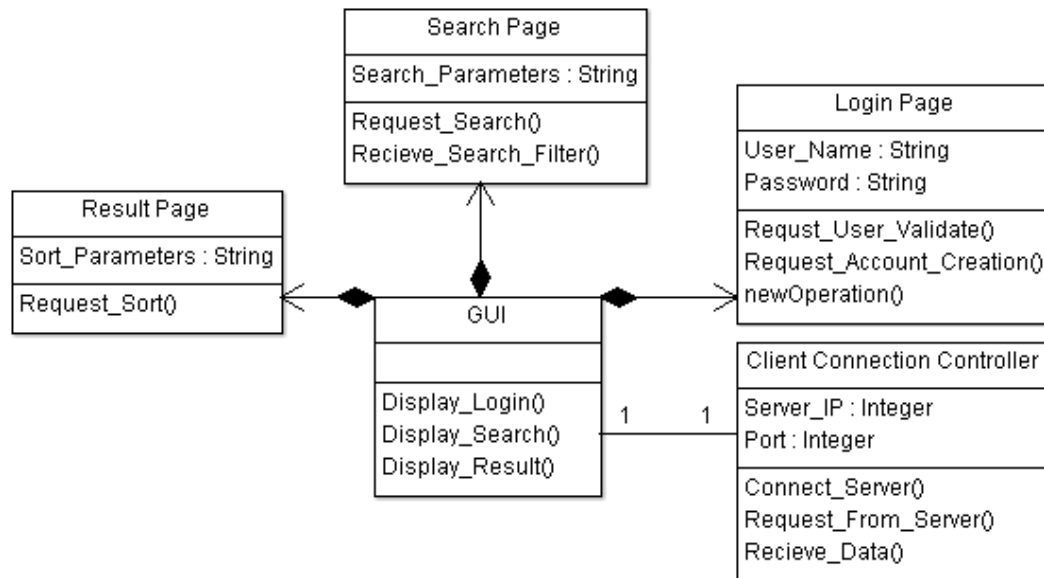
<u>Class</u>	<u>Definition</u>
GUI	<ul style="list-style-type: none"><li>• connects the client to the various pages of the system</li><li>• without the GUI, none of the pages described below (Login, Search, Result) would exist</li></ul>
Login Page	<ul style="list-style-type: none"><li>• receives user login information: username and password</li><li>• requests validation of authentication from the server</li><li>• request to create account is made in the absence of validation</li></ul>
Search Page	<ul style="list-style-type: none"><li>• receives search/filter parameters from the client: document name, document type, subject, publication year, author</li><li>• requests all resources relevant to the search/filter parameters from database stored on server</li></ul>
Result Page	<ul style="list-style-type: none"><li>• displays data acquired from the server</li><li>• request to sort by a single parameter: document name, document type, subject, publication year, or author, is sent to server and sorted data is again displayed</li></ul>
Client Connection Controller	<ul style="list-style-type: none"><li>• connects to server by IP address and port number</li><li>• sends requests and receives data to and from the server</li></ul>

### 4.3.2 Server Side Classes

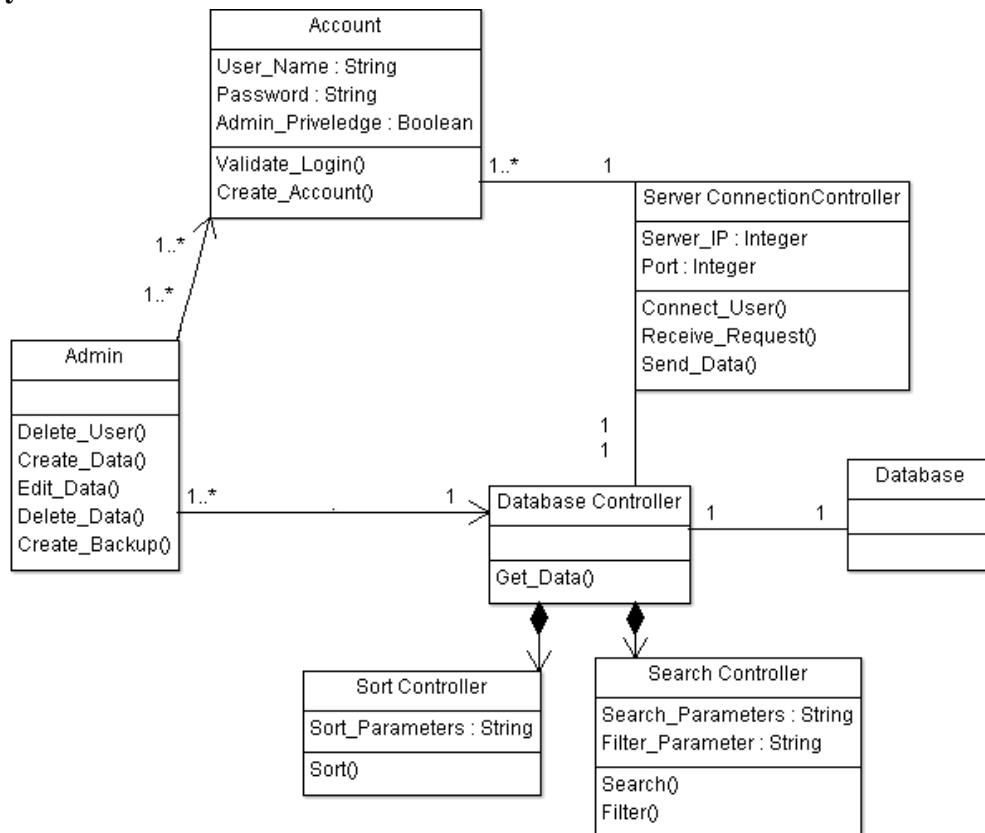
<u>Class</u>	<u>Definition</u>
Server Connection Controller	<ul style="list-style-type: none"><li>• connects to client by IP address and port number</li><li>• receives requests and sends data to and from the client</li></ul>
Account	<ul style="list-style-type: none"><li>• contains a list of user/admin accounts</li><li>• validates login requests and creates new accounts</li></ul>
Admin	<ul style="list-style-type: none"><li>• deletes user accounts</li><li>• modifies (create/edit/delete) data within the database</li><li>• creates backups of current database</li></ul>
Database Controller	<ul style="list-style-type: none"><li>• gets all data that need to be filtered/sorted and sends to client</li><li>• without the Database Controller, neither the Search Controller nor the Sort Controller would exist.</li></ul>
Database	<ul style="list-style-type: none"><li>• contains all data/resources for the PRMS system</li></ul>
Sort Controller	<ul style="list-style-type: none"><li>• applies sorting algorithm to sort data using one of the following parameters: document type, subject, publication year, or author</li><li>• received from user and sorted data sent back to client</li></ul>
Search Controller	<ul style="list-style-type: none"><li>• sends request to grab all the data from database</li><li>• applies search/filtering algorithm using parameters: document name, document type, subject, publication year, and author</li><li>• Acquired data sent back to the client</li></ul>

## 4.4 UML Class Diagrams

### Client System:



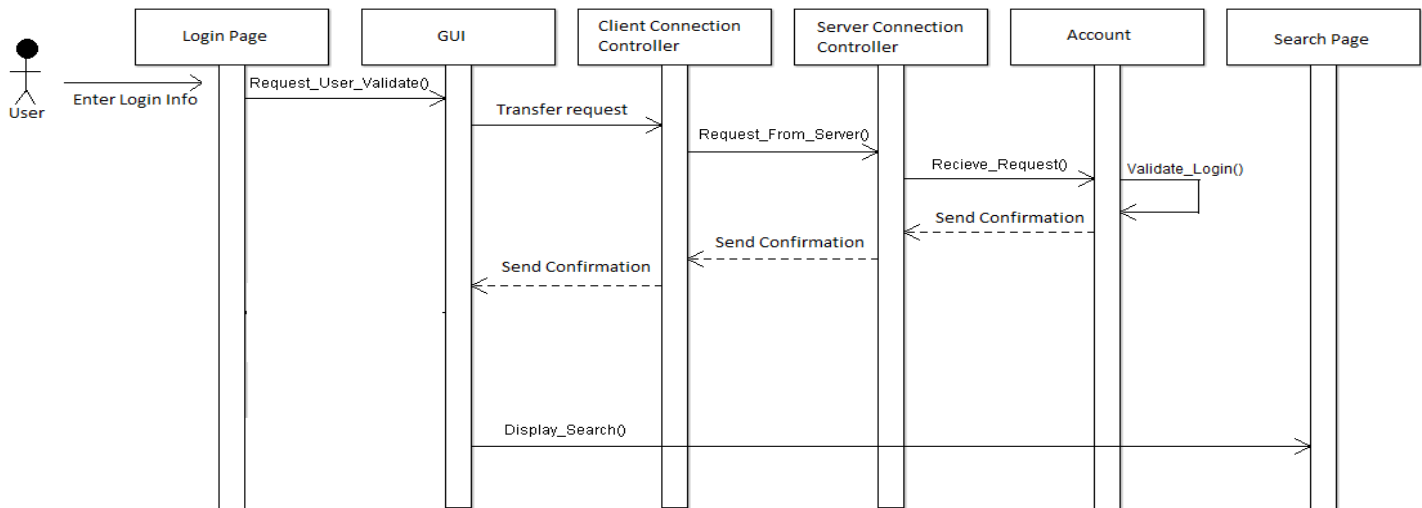
### Server System:



## 4.5 Database Models and Diagrams

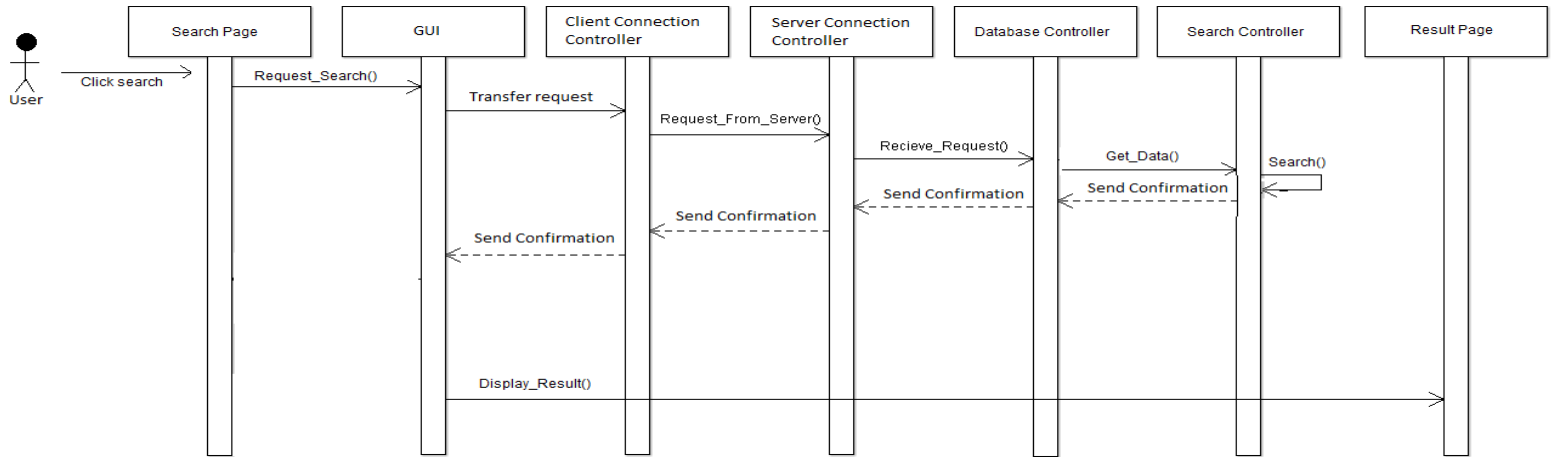
The database will be a simple XML document consisting of all the data (resources for the PRMS system). This document/database will simply be imported/read into the C++ program on the server side where it will be used and modified as per user request.

## 4.6 Sequence Diagrams



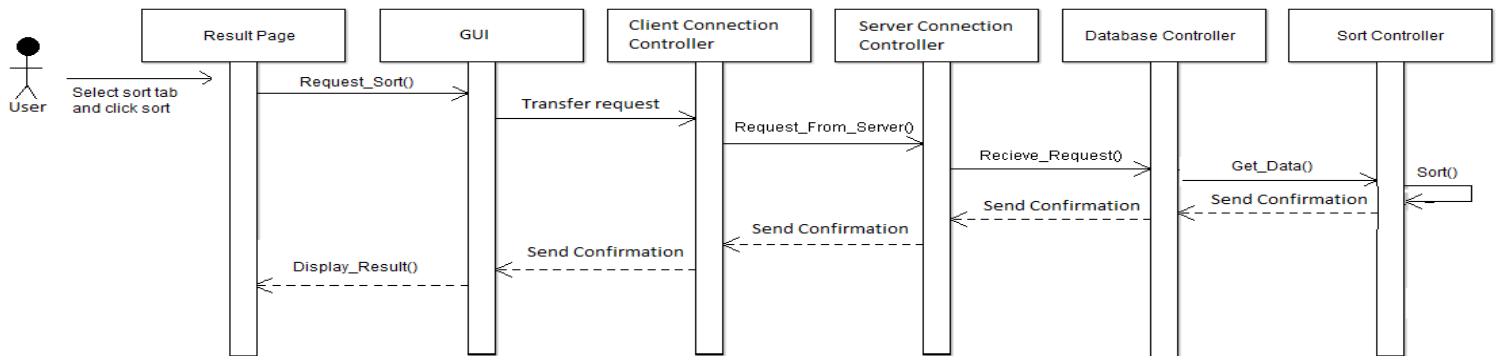
### Sequence: User Login Flow

1. User inputs their login information in Login Page on client system
2. Login Page sends the GUI a request to validate/authenticate user
3. GUI transfers the request to Client Connection Controller
4. Client Connection Control accesses the server via Server Connection Controller
5. Server Connection Controller then forwards the request to Account where the user information is validated/authenticated
6. After validating, Account sends a confirmation to the Server Connection Controller which forwards it to Client Connection Controller after which the user is shown GUI
7. Once validated the GUI accesses the Search Page to be displayed to the user



#### Sequence: User Search Flow

1. User inputs search parameters and clicks search
2. Search Page sends the GUI a request to search
3. GUI transfers the request to Client Connection Controller
4. Client Connection Control accesses the server via Server Connection Controller
5. Server Connection Controller then forwards the request to Database Controller
6. Database Controller sends request to Search Controller where it searches the database
7. After searching, Search Controller sends data to the Server Connection Controller which forwards it to Client Connection Controller after which the user is shown GUI
8. Once searched, the GUI accesses the Result Page to be displayed to the user



#### Sequence: User Sort Flow

1. User inputs sort parameters and clicks sort
2. Result Page sends the GUI a request to sort
3. GUI transfers the request to Client Connection Controller
4. Client Connection Control accesses the server via Server Connection Controller
5. Server Connection Controller then forwards the request to Database controller
6. Database controller send request to Sort controller where it sorts the data
7. After searching, Sort Controller sends data to the Server Connection Controller which forwards it to Client Connection Controller after which the user is shown GUI
8. Once sorted, the GUI accesses the Result Page to be displayed to the user



## 4.7 Communication Protocols / Messaging Formats

The communication between the client and the server will be implemented by the use of Socket Programming. This technique will allow many clients to access the server simultaneously simply by having each client establish a connection with the same server. This is done by the user inputting the server's IP address and port number on the client system (a process which only needs to be done once during initial setup). The server will auto-learn each client's IP address and port number and confirm the connection.

The system of the client and the server will start to change protocols and messages. All the messages sent will be done following the simple format of having the first and last line as a full line of + symbols; this separation must also be done if the message has more than 1 line. Also, at the time of the server to send back a message showing sources, it must show the count of sources that the server are sending in the message, and the number of count of the message in comparison to the total. The following example is the message that must be send by the server after a search operation:

```
+-----+
|                                     |
|                               Source number A of Z founds                |
|                                     |
+-----+
|                                     |
|                               Document Name                                |
|                                     |
+-----+
|                                     |
|                               Author                                        |
|                                     |
+-----+
|                                     |
|                               Subject                                       |
|                                     |
+-----+
|                                     |
|                               Document Type                                |
|                                     |
+-----+
|                                     |
|                               Year of Publication                          |
|                                     |
+-----+
```

## 4.8 User Interface (GUI) Design

### Request Server Connection Page:

Professional Resource Management System

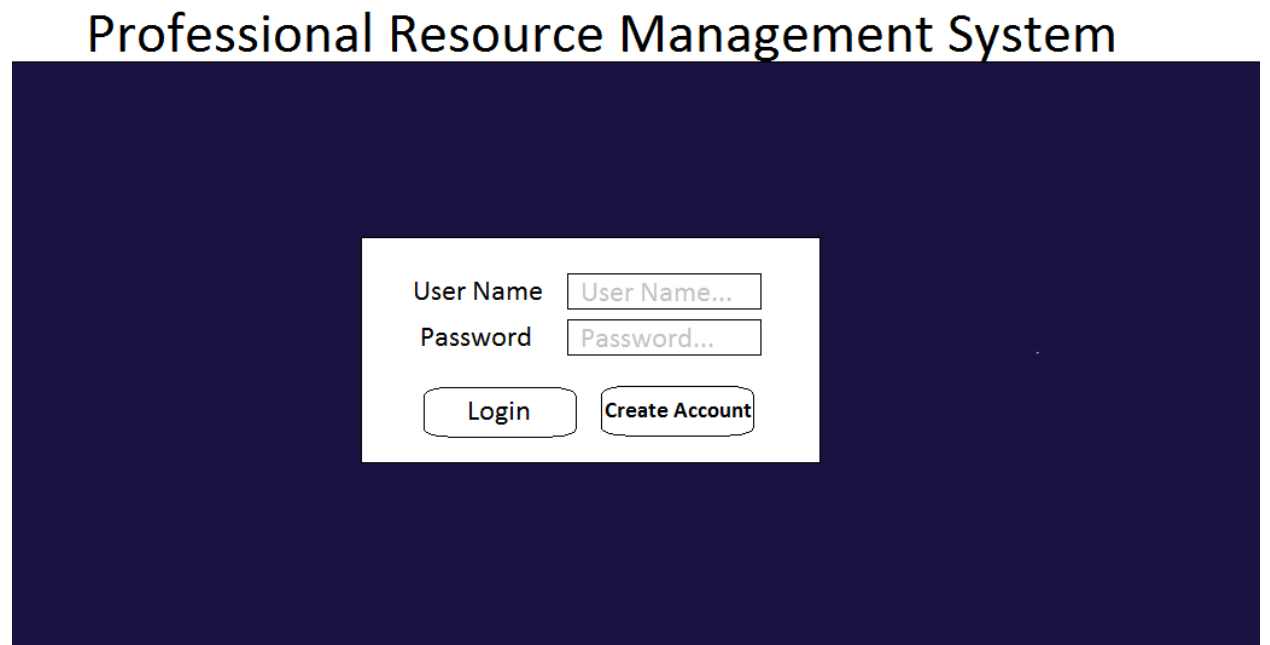


The image shows a screenshot of a web application interface for a 'Professional Resource Management System'. The background is a solid dark blue. In the center, there is a white rectangular box containing the connection form. The form has two labels, 'Server IP' and 'Server Port', each followed by a text input field. The first input field contains the placeholder text 'IP Address...' and the second contains 'Port Number...'. Below these fields is a single button labeled 'Request Connection'.

Server IP	<input type="text" value="IP Address..."/>
Server Port	<input type="text" value="Port Number..."/>
<input type="button" value="Request Connection"/>	

### Login Page:

Professional Resource Management System



The image shows a screenshot of a web application interface for a 'Professional Resource Management System'. The background is a solid dark blue. In the center, there is a white rectangular box containing the login form. The form has two labels, 'User Name' and 'Password', each followed by a text input field. The first input field contains the placeholder text 'User Name...' and the second contains 'Password...'. Below these fields are two buttons: 'Login' and 'Create Account'.

User Name	<input type="text" value="User Name..."/>
Password	<input type="text" value="Password..."/>
<input type="button" value="Login"/>	<input type="button" value="Create Account"/>

## Search Page:

# Professional Resource Management System

[Logout](#)[View Tutorial](#)

Document Name

Document Type  ▼

Subject  ▼

Publication Year  ▼

Author

## Results Page:

# Professional Resource Management System

[Logout](#)

Title	Author	Subject	Document Type	Date of Publication
Example: A Book of Examples	Dr.Example	Subject Example	Textbook	1992
Why Examples are Important	Dr.Example	Subject Example	Article	2004
A Theory on Examples	Dr.Example	Subject Example	Paper	1999

## 5 Project Management Plan

The plan presented here is for the development of the PRMS System by a small software team consisting of three individuals: Abad Hameed, the project leader and two software engineers, Brandon Crispino and Rafael Nazario.

### 5.1 Project Organization

#### 5.1.1 External Structures

All the work on this project will be performed by Abad Hameed, Brandon Crispino and Rafael Nazario. The team will meet weekly with the client to report progress and discuss possible changes or modifications.

#### 5.1.2 Internal Structures

The development team consists of Abad Hameed, Brandon Crispino and Rafael Nazario.

#### 5.1.3 Roles and Responsibilities

Abad Hameed, Brandon Crispino and Rafael Nazario will all perform the design workflow as well as begin with implementing the product. Abad will oversee the product progress and match it with the plan and deliverables as well as construct and develop the client/server socket program. Brandon will develop the GUI for the various pages required as well as assist with the resource acquisition from the database. Rafael will develop the searching and sorting algorithms required to be performed on the acquired resources.

### 5.2 Managerial Process Plans

#### 5.2.1 Development Effort

For the purpose of calculating the development effort, the intermediate Constructive Cost Model (COCOMO) approach has been utilized. The following is the intermediate COCOMO calculation in regards to the PRMS system:

Formulae:

1. Nominal Effort =  $A \cdot (\text{KLOC})^B$ , in person-months
2. Effort Multiplier = product of 15 ratings
3. Est. Project Effort = Nominal Effort \* Effort Multiplier

**Mode:** Organic                      therefore, **A=2.4, B=1.05**

LOC estimate: 3000 (**KLOC = 3**)

**Nominal Effort** =  $2.4 * (3)^{1.05} = 7.60656$  (person-months)

Development Effort Multipliers for PRMS:

Cost Drivers	Rating	
Required software reliability	Nominal	1.00
<i>Database size</i>	<i>Low</i>	<i>0.94</i>
<i>Product complexity</i>	<i>High</i>	<i>1.15</i>
<i>Execution time constraint</i>	<i>High</i>	<i>1.11</i>
Main storage constraint	Nominal	1.00
Virtual machine volatility	Nominal	1.00
Computer turnaround time	Nominal	1.00
Analyst capabilities	Nominal	1.00
<i>Applications experience</i>	<i>Low</i>	<i>1.13</i>
<i>Programmer capability</i>	<i>Low</i>	<i>1.17</i>
<i>Virtual machine experience</i>	<i>N/A</i>	-
Programming language experience	Nominal	1.00
<i>Use of modern programming practices</i>	<i>High</i>	<i>0.91</i>
Use of software tools	Nominal	1.00
<i>Required development schedule</i>	<i>Very Low</i>	<i>1.23</i>
<b>Effort Multiplier:</b>		<b>1.77566</b>

**Estimated Effort for Project** =  $7.60656 * 1.77566 = 13.51$  person-months

This estimation results in an approximate team member effort of:

$13.51$  person-months / 3 persons = **4.50 person-months per team member**

Which, given 1.5 months until project deadline, further approximates each member doing the work of:  $4.50$  person-months per team member / 1.5 months = **3.00 persons**.

## 5.2.2 Resource Acquisition Plan

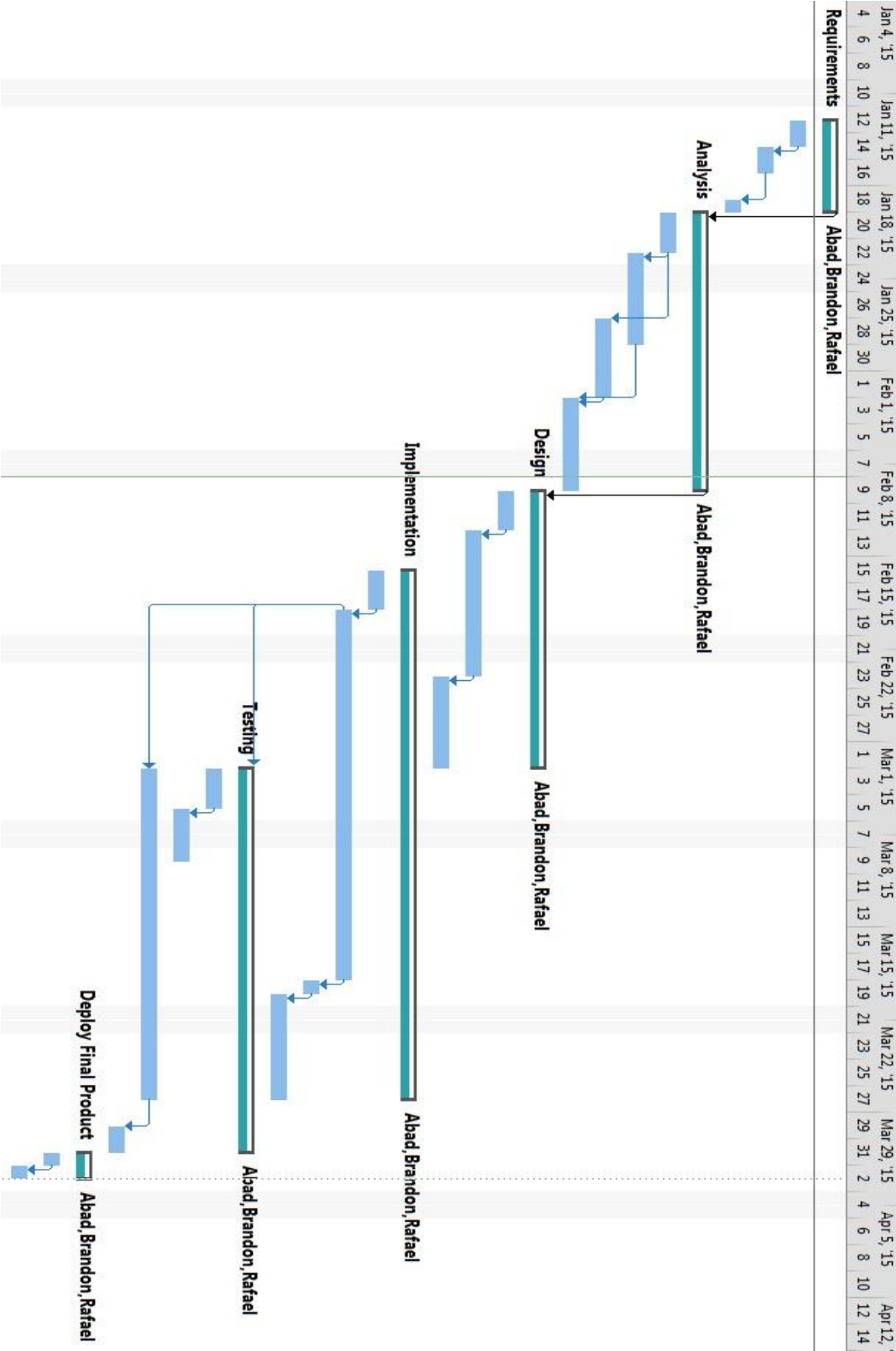
All required CASE tools, hardware and software for the project is already available. The product will be delivered installed on two separate desktop/laptop machines either of which will perform the client/server responsibilities, respectively.

## 5.3 Work Plan

### 5.3.1 Work Activities and Schedule Allocation

Task Mode ▾	Task Name ▾	Duration ▾	Start ▾	Finish ▾
★	▸ <b>Requirements</b>	<b>5 days</b>	<b>Tue 1/13/15</b>	<b>Mon 1/19/15</b>
→	Requirements elicitation	2 days	Tue 1/13/15	Wed 1/14/15
→	Draw up initial requirements	2 days	Thu 1/15/15	Fri 1/16/15
→	Refine Requirements	1 day	Mon 1/19/15	Mon 1/19/15
★	▸ <b>Analysis</b>	<b>15 days</b>	<b>Tue 1/20/15</b>	<b>Mon 2/9/15</b>
→	Analyze client needs	3 days	Tue 1/20/15	Thu 1/22/15
→	Draw up initial specifications	5 days	Fri 1/23/15	Thu 1/29/15
→	Create project management plan	4 days	Wed 1/28/15	Mon 2/2/15
→	Refine specifications document & SPMP	5 days	Tue 2/3/15	Mon 2/9/15
★	▸ <b>Design</b>	<b>15 days</b>	<b>Tue 2/10/15</b>	<b>Mon 3/2/15</b>
→	Finalize design details with client	3 days	Tue 2/10/15	Thu 2/12/15
→	Draw up initial design document	7 days	Fri 2/13/15	Mon 2/23/15
→	Refine design document	5 days	Tue 2/24/15	Mon 3/2/15
★	▸ <b>Implementation</b>	<b>30 days</b>	<b>Mon 2/16/15</b>	<b>Fri 3/27/15</b>
→	Confirm roles and responsibilities	3 days	Mon 2/16/15	Wed 2/18/15
→	Begin development	20 days	Thu 2/19/15	Wed 3/18/15
→	Warm-up demo for client	1 day	Thu 3/19/15	Thu 3/19/15
→	Finalize development	6 days	Fri 3/20/15	Fri 3/27/15
★	▸ <b>Testing</b>	<b>21 days</b>	<b>Tue 3/3/15</b>	<b>Tue 3/31/15</b>
→	Draw up testing document	3 days	Tue 3/3/15	Thu 3/5/15
→	Refine testing document	2 days	Fri 3/6/15	Mon 3/9/15
→	Regularly test new development	19 days	Tue 3/3/15	Fri 3/27/15
→	Finalize Integration/Regression tests	2 days	Mon 3/30/15	Tue 3/31/15
★	▸ <b>Deploy Final Product</b>	<b>2 days</b>	<b>Wed 4/1/15</b>	<b>Thu 4/2/15</b>
→	Set up product for client use	1 day	Wed 4/1/15	Wed 4/1/15
→	Final demo for client	1 day	Thu 4/2/15	Thu 4/2/15

5.3.2 Project Gantt Chart



### 5.3.3 Risk Management Plan

There are various resource acquisition systems in the industry which the new product can be compared with for the developers and customer to better understand the system. This, however, does not disregard the extensive testing that the new product will undergo. The client is assumed to be inexperienced with computers and special attention will therefore be given to the analysis and design workflows to make the product as user-friendly and efficient as possible.

In order to prevent any potential risks and faults in the product design, extensive testing will be performed in every workflow and more specifically the design workflow. Each member will be responsible to test their own code first and then their colleagues. Albeit there are slim chances, in case of a hardware failure, the product will simply be installed on another machine. Should there be a fault with any CASE tools, the specific tool will be replaced.

## 5.4 Technical Process Plans

### 5.4.1 Process Model

The Unified Process will be used as it allows for each increment of our product development to be followed by iterations through all workflows. This ensures that development is consistent and meets all requirements and specification guidelines and that all milestones are attained.

### 5.4.2 Methods, Tools and Techniques

The workflows will be performed in accordance to the Unified Process. The implementation of the product will be in C++.

### 5.4.3 Infrastructure Plans

The product will be developed using Visual Studio for programming, MS Project for planning and ArgoUML for diagrams and infrastructure.

### 5.4.4 Configuration Management Plan

Apache Subversion (SVN) is the version control tool which will be used for all the artifacts.

### 5.4.5 Testing Plan

The testing workflow will be performed in accordance to the Unified Process.

### 5.4.6 Documentation Plan

Documentation will be produced as specified in the Unified Process.