Software Requirements Specification

for

Lions Final Exam Scheduler (L.I.F.E.S.)

Version 2.2

Prepared by



|  |  |  |
| --- | --- | --- |
| Team Leader | Scott Smoke | ssmoke@una.edu |
| SQA | Riley Smith | msmith9@una.edu |
| SQA | Jordan Beck | jbeck@una.edu |
| Engineer | Joshua Ford | jford@una.edu |
| Technical Writer | Jeffrey Allen | jallen2@una.edu |

Instructor: Patricia Roden, Ph.D

Course: CS455 Software Engineering

|  |  |
| --- | --- |
|  |  |
|  |  |

Contents

RevisioNs iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 References and Acknowledgments 2

1.6 Overview 2

2 Overall Description 2

2.1 Product Perspective 2

2.2 Product Functionality 2

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 4

3 Specific Requirements 4

3.1 External Interface Requirements 4

3.2 Functional Requirements 5

4 Non-functional Requirements 7

4.1 Safety and Security Requirements 7

4.2 Software Quality Attributes 7

4.3 Input File Format 8

4.4 output File Format 9

4.5 Error Format 9

5 Deliverables 9

6 client- developer agreement 11

Appendix A – Interface prototypes 12

APPENDIX B - CODING STANDARDS 18

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Jeffrey Allen | Initial draft  Added document’s purpose  Added definitions and acronyms section  Added references and acknowledgements  Added user documentation  Defined intended audience  Added overview of document  Added Product Overview  Added application perspective  Added product functionality  Edited References  Added line numbers  Added Name of Application  Removed unnecessary sections | 2/20/15 |
| 1.1 | Scott Smoke, Jeffrey Allen | Made revisions to every section added in sections 4.3 and 4.4 | 2/27/15 |
| 1.2 | Riley Smith | Added screen shots, fixed grammatical errors | 2/27/15 |
| 1.3 | Scott Smoke | Changed section 2.5, 4.1, and 3.1.1. Also removed a non important screen shot. Fixed table of contents, font, and spacing | 2/27/15 |
| 1.4 | Jeffrey Allen | Updated section 3 to reflect that of IEEE Std 830-1998 standard.  Updated font size and font type  Updated Table of Contents  Updated definitions, acronym, abbreviations | 2/28/15 |
| 1.5 | Scott Smoke | Modified 2.5, 3.2. Added new requirement in functional requirements section. | 3/4/15 |
| 1.6 | Scott Smoke, Jeffrey Allen | made changes to denote that two different versions will be delivered.  Added in the input and output files in the appendix.  Added information to the deliverables chapter | 3/5/15 |
| 1.7 | Scott Smoke, Jeffrey Allen,  Riley Smith | Updated definitions, acronym, abbreviations  Updated 2.2 Product Functionality  Updated 2.3 Users and Characteristics  Added Screenshots for Version 1 and 2  Updated fonts  Updated Figure references | 3/6/15 |
| 2.1 | Jeffrey Allen | Added Appendix B: Coding Standards  Updated Coding Standards | 3/9/15 |
| 2.2 | Scott Smoke | Fixed inconsistent usage with the acronyms. Made changes to the coding standards and added more acronyms. | 3/9/15 |

# Introduction

This section gives a scope description and overview of everything included in this S.R.S. document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

## Document Purpose

The purpose of this document is to give a detailed description of the requirements for the Lions Final Exam Scheduler (L.I.F.E.S.) software package. It will illustrate the purpose and provide a complete declaration for the functionality of the system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a client for its approval and to be used as reference for the Tune Squad during development. Upon agreement of this requirements specification document between the client, Dr. Patricia Roden, and the Tune Squad, both parties will provide their signatures (see section 6).

## Product Scope

L.I.F.E.S.is a desktop application with a graphical user interface that will allow for a user in the University of North Alabama’s Office of the Registrar to generate a final exam schedule. The application will make use of the data from a previous similar semester to determine trends in popular class times. The L.I.F.E.S. application will be available via CD-ROM or flash drive to anyone that wishes to generate a final exam schedule.

The goal of this application is for a user to be able to generate a final exam schedule and display the schedule in a similar format presented by UNA website as of the year 2015. Previously, a person had to compose these schedules by hand, which could take days or even weeks to produce a schedule with minimal conflicts. This software attempts to speed up the process by providing a single user with an easy and intuitive work environment by providing a small subset of data to produce a high quality exam schedule.

## Intended Audience

This S.R.S. is intended to be read by the client professor, V.P.A.A., the individuals located in the Office of the Registrar that will be using this application, and the Tune Squad development team.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Admin/Administrator | System administrator who is given specific permission for managing and controlling the system |
| Developer | A member of the Tune Squad |
| CSV | Comma Separated Values |
| GUI | Graphical User Interface |
| IEEE | Institute of Electrical and Electronics Engineers |
| ISO | International Organization for Standardization |
| L.I.F.E.S. | Lions Final Exam Scheduler |
| L.I.F.E.S.V1 | The version of Lions Final Exam Scheduler that **does not** contain distinct levels of users, such as an administrative an general user |
| L.I.F.E.S.V2 | The version of Lions Final Exam Scheduler that **does** contain distinct users, such as an administrative and general user |
| PDF | Portable Document Format |
| S.R.S. | Software Requirements Specification |
| Time Constraint File | First input data file of this final exam scheduling application |
| Total Enrollment File | Second input data file of this final exam scheduling application |
| UNA | University of North Alabama |
| User | Someone who interacts with the desktop application |
| V.P.A.A. | Vice President of Academic Affairs |
| Wish | A desirable level of achievement that may not be attainable through available means contained in a language statement |

## References and Acknowledgments

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

[2] University of North Alabama Official Colors, http://www.una.edu/graphic-standards/print-web/

## Overview

Section 2 of this S.R.S. describes the application’s functionality at a high-level. In section 3, a detailed description of its requirements is presented, including functional requirements. Located in section 4 are the non-functional requirements such as security, error handling, and help messages.

# Overall Description

## This section of the S.R.S. describes application at a high-level. The L.I.F.E.S. application will be explained in its context to show how users interact with it and introduce its basic functionality.

## Application Perspective

L.I.F.E.S. is a desktop application for the University of North Alabama. Its only dependency is an external printer in the event a user wishes to print a final exam schedule generated by the application.

## Product Functionality

There are two different versions of the L.I.F.E.S. application. The first version is known as L.I.F.E.S.V1 and the second is known as L.I.F.E.S.V2. The first version will allow a user to generate final exam schedules without having to authenticate themselves as valid users. L.I.F.E.S.V2 will allow a user to generate final exam schedules only after they have successfully authenticated themselves as either a general or administrative user.

The result of generating a final exam schedule will be based on two separate data files provided by the user after the user enters the semester to schedule. The first file provides the application with details that define the timing constraints of the exam schedule. This file is aptly named the time constraint file. This file can either be created manually by using the application or it can be loaded from a separate input file. The second file must be one that is loaded from a separate file, which will provide the application with the data gathered from a previous semester. This file is known as the class enrollment file.

When these two files are successfully loaded into the application, a user will be able to view a list of popular class times by enrollment. The list of classes by popularity of enrollment will be displayed in a descending order. When a user decides to generate a final exam schedule after inputting a valid time constraint and class enrollment file, the resulting schedule will immediately be displayed in a similar format the University’s website presents the schedule of final exams (see Appendix A, Figure 1). A user will be able to either view a single date of class schedules or the schedule in its entirety.

An administrative user that is operating L.I.F.E.S.V2 will have the option to finalize a schedule that has been generated. This will label a selected final exam schedule as “Administratively Approved,” and will apply a seal of approval to all printed documents associated with the finalized schedule.

Any user that has questions about the application can refer to the help menu. This menu will aid users with operating the application.

## Users and Characteristics

The users of this application will be the V.P.A.A. and individuals at the Office of the Registrar. All users should be familiar with opening desktop applications and maneuvering file systems. In L.I.F.E.S.V1, there is only one category of user known as a general user. In L.I.F.E.S.V2, there are two categories of users, which include the general user from the first version and also adds one more category of user known as an administrative user. Privileges are the separating factor between these two different categories of users. There can only exist a maximum of two administrators.

Users in both versions of the application will be able to open a schedule, view a schedule by day or by week, generate a schedule, reschedule a schedule that was generated, modify a schedule, and view total enrollments by their class times. L.I.F.E.S.V2 requires all users to authenticate themselves as valid users. An administrator must verify general users in L.I.F.E.S.V2 if they wish to access all their privileges.

An administrative user in L.I.F.E.S.V2 has all of the same privileges as the general users. Additionally, they are able to create and delete general user accounts, reset passwords, finalize exam schedules, and unlock accounts.

## Operating Environment

L.I.F.E.S. will be operated in the office setting of UNA’s Office of the Registrar. The application will be operated on an HP Pro running a 64-bit running a Windows 7 operating system.

## Design and Implementation Constraints

This hardware platform this application will be executed upon is a HP Pro desktop computer with a Windows 7 operating system installed.

### *Programming Language*

Creating a user interface, which is both effective and easily navigable, will pose a difficult challenge. This is why the Tune Squad proposes that L.I.F.E.S. should be developed using the C# programming language. In a meeting with a client that was conducted by the Tune Squad on February 24th, the client verbally agreed with the Tune Squad that C# will be a suitable language for the project. The coding standards that will be used are described in appendix B.

## User Documentation

The user manual for this application can be found electronically by first clicking the tab labeled “Help” in the menu bar of the application, and then clicking “User’s Manual”. A hard copy of the manual will also accompany the final submission of the project.

# Specific Requirements

Descriptions of all functional requirements are described in this section. There are detailed descriptions of all the interfaces in the system and all of the application’s features.

## External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the application. It also contains a description of the hardware and software interfaces. Basic prototypes of the user interface are also provided in this section.

### User Interfaces

A first-time user of L.I.F.E.S.V2 should be presented with a login screen when he/she opens the application (See Appendix A, figure 5). If the user has not been registered by the V.P.A.A., he/she should not be able to have access to any other interface. A returning user should be able to login directly when the application is opened.

The main interface will have a scrollable table that will display a generated schedule for viewing either the entire schedule or a single day. Above the table there will be two buttons to select the two input files. Below the table there will be a button to generate an exam schedule and a button to reschedule the currently displayed schedule and a swap button. The main interface will contain a menu bar with the following menu options: File, View, Help, and in L.I.F.E.S.V2 there will include a Log Out option (See Appendix A, Figure 6). Under the file option there will be the sub options: new, open, save, save as, print, and close.

All colors and logos used in L.I.F.E.S. will adhere to the University of North Alabama graphics standards. Examples of these interfaces are in the appendix.

#### Login Dialog (Exclusive to L.I.F.E.S.V2)

The login dialog will ask for the username and password and will have the buttons login and cancel. If a user attempts to log into L.I.F.E.S. with invalid credentials, an error message will be displayed. The message will explain to the user that either an invalid username or password has been entered.

#### Open File Dialog

The open file dialog will prompt the user to enter a file name or search for the required file. This dialog will contain the buttons open and cancel.

#### Save File Dialog

The save file dialog will prompt a user to associate a name with a file and specify the format of the file to be saved. The save dialog will contain the buttons save and cancel.

#### Print Dialog

The print dialog will display a list of printers available to the computer. The user must select a printer that he/she wishes to print final exam schedules.

#### Error Dialog

The error dialog will display the error and have an “OK” button that will close the message.

#### Data input dialog

The data input dialog will prompt the user to enter the data that would otherwise be in the first file that is required for the exam schedule. This information is the data that will be used to generate the schedule. The fields are number of days to schedule, beginning time of the first exam of the day, length of time for each exam, length of time between exams, and length of time for a lunch period. See Appendix A figure 7.

### Hardware Interfaces

The product will be able to communicate with printers through libraries available to the C# programming language. A printer will allow hard copies of final exam schedules to be printed. Either a local or networked printer will be suitable. A printer capable of printing black and white will be sufficient.

### Software Interfaces

The application will be able to communicate with the Windows 7 operating system and link with libraries in order to access a printer.

## Functional Requirements

The L.I.F.E.S.application will allow its users to create a final exam schedule that will make use of data from previous similar semesters to determine trends in popular class times. There are two different versions of this application that are only separated by the types of users each version supports. L.I.F.E.S.V1 only has one category of user, known as a general user. L.I.F.E.S.V2 on the other hand is a superset of L.I.F.E.S.V1 because it extends the functionality of it by adding one other category of user, known as an administrative user, and authentication system that all users must operate in order to gain access to all other functionality.

### General User

All the functionality listed beneath will appear in both versions of this application. L.I.F.E.S.V1 will not require a user to authenticate their selves in order to access all functionality of the application that is under the General User section.

#### Identification & Authentication (Exclusive to L.I.F.E.S.V2)

The user will be prompted to enter a username and password. The username must be the users’ University of North Alabama email address. The password must be between seven and nine characters long. The password must start with an alphabetical character followed by any number of alphanumeric characters and may contain any of the following \*, #, or $. The password is not case sensitive.

#### Open Schedule

A user will be able to open a previously generated final exam schedule. They will be able to choose the entire schedule or a single day.

#### Generate Schedule

A user must provide a valid time constraint file along with a valid student enrollment file in order to generate a final exam schedule.

#### Reschedule

When a user generates a final exam schedule, the option to rerun the final exam schedule will be available.

#### Modify Schedule

A General User will be able to swap final exam time slots. The final exam times will be first selected and then swapped. The final exam times to swap will be highlighted.

#### Save Schedule

The General User will be able to save the file as either a PDF, CSV, or Plain Text file.

#### Print Schedule

The General User will be able to print the generated schedule in portrait or landscape form.

#### View Schedule

The General User will be able to view a single day or the entire final exam schedule in the GUI.

#### View Class times by enrollment

The General User will be able to view a list of popular class time by total enrollment. This list will be displayed in descending order.

**3.2.10 Manual Data Input**

The General user will be able to manually input the data that otherwise would be in the time constraints file. The user will use the manual data interface that is described in section 3.1.1.6.

### Administrative User (Exclusive to L.I.F.E.S.V2)

Administrators have all privileges of a General User in addition to the following privileges described in this section. At maximum there can only exist two administrative users in the application.

#### Create/Delete General Users

An Administrator will have to create accounts for this software.

#### Reset Passwords

An Administrator will be able to change a user’s password at any time.

#### Finalize Exam Schedule

An Administrator will finalize a final exam schedule making it so that no more changes can be made.

#### Unlock Account

An Administrator will be able to unlock an account after it has been locked due to too many failed long in attempts.

# Non-functional Requirements

## Safety and Security Requirements

Administrative and general users are the two types of users that can be authenticated in this application. Both types of users will be required to provide login credentials, including a username and password.

1. Usernames will be the user’s UNA email address
2. Passwords will consist of 7-9 characters with the first character being alphabetic and the rest being alphanumeric or the symbols \*, #, $. Passwords are not case sensitive
3. The list of username and passwords will be stored in an encrypted text file.
4. A general user that enters invalid login credentials four consecutive times will be locked out of the application to where it can only be unlocked by an administrative user
5. In the event a user enters the wrong login credentials, an error message will display whether the username or password was incorrect
6. Only an administrative user will be allowed to finalize exam schedules

## Software Quality Attributes

|  |  |
| --- | --- |
| Flexibility: | The program will allow for manual modification to the exam schedule. |
| Usability: | The user interface will be designed to be easy to learn and use. There will also be an included user manual in the program’s help menu. |
| Reliability: | The program will generate an exam schedule that will always work, having no time conflicts. |
| Testability: | The software will be written with testability in mind. Each module of the software will be written with test cases in mind to allow for the finding of faults. |
| Availability: | The software will be available on a CD-ROM and/or flash-drive. |
| Maintainability: | The software will achieve maintainability through the use of modules. |

## Input Data Files

L.I.F.E.S. uses the ISO 8601 24-hour format as a standard convention to represent time.

### Time Constraint File

The first file provides the application with details that define the time constraints of the final exam schedule. This file can either be created manually by using the application, or it can be loaded from a separate input file. An example of this file can be found in *Appendix A*, labeled *Figure 2*.

#### Number of days to schedule

This integer value will be between 3 and 5.

#### Beginning time of the first exam of the day

The beginning time of each exam is 0700. The final time for exams is either 1700 or 1715.

#### Length of time for each exam

The minimum is one hour fifteen minutes for each exam, and there is no maximum. Every exam can begin on the quarter hour, half hour, or hour.

#### Length of time between exams

The minimum is ten minutes and the maximum is thirty minutes.

#### Length of time for a lunch period

This value will be optional, and there are no limits for this value.

### Total Enrollments File

The format of this file is a CSV file with the first column specifying the day and time of the class, and the second column containing the total number of students in the class (See *Appendix A, Figure 3*).

## Output File

The first line will contain the semester and the year. The second line will specify the class data file used to generate the schedule. Then, all the data from the first file or the manually inputted data will be included in this file. Lastly, the schedule that was generated from all the previous information will be displayed using standard times. The format of this file will either be plain text, PDF, or CSV (See Appendix A, Figure 4).

## Input Errors

Every error dealing with the two input files will be given in a pop up dialog that will list the line number and the type of error.

5 Deliverables

L.I.F.E.S.V1 and L.I.F.E.S.V2 will be delivered on April 28th, 2015. Both versions of the application will be delivered via CD-ROM and/or a flash drive. The application will be able to install itself on the computer. Along with the application, everything pertaining to the development will be delivered. This includes:

(1) Test files

(2) Design documents

(3) Emails concerning the development

(4) Requirements documentation

(5) Test code

(6) Entire application code for both versions

(7) Printed and digital user manual

(8) Executables for both versions

(9) Source code for both versions

(10) Team meeting minuets

(11) Scratch code

(12) CD-ROM/Flash drive for installing both versions

(13)Test files

6 Client-Developer Agreement

|  |  |  |
| --- | --- | --- |
| Application deliver date April 28. |  |  |
| *Patricia L. Roden, Ph.D, Client* |  | *Date* |
|  |  |  |
| *Scott Smoke, Team Leader* |  | *Date* |
|  |  |  |
| *Riley Smith, SQA* |  | *Date* |
|  |  |  |
| *Jordan Beck, SQA* |  | *Date* |
|  |  |  |
| *Joshua Ford, Engineer* |  | *Date* |
| *Jeffrey Allen, Technical Writer* |  | *Date* |

Appendix A – Interface Prototypes

Figure 1 – Schedule Display Guideline

**

Figure 2 – Time Constraint File

**

Figure 3 – Total Enrollment File



Figure 4 – Output File



Figure 5 – L.I.F.E.S.V2 Login Mock-up

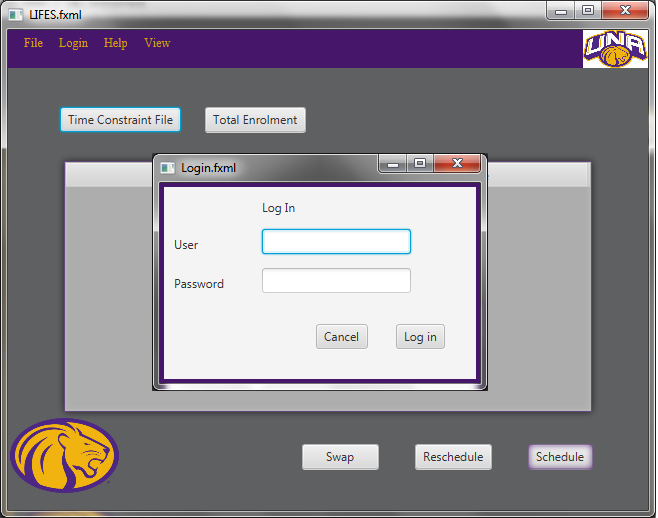


Figure 6 – L.I.F.E.S. Mock-Up

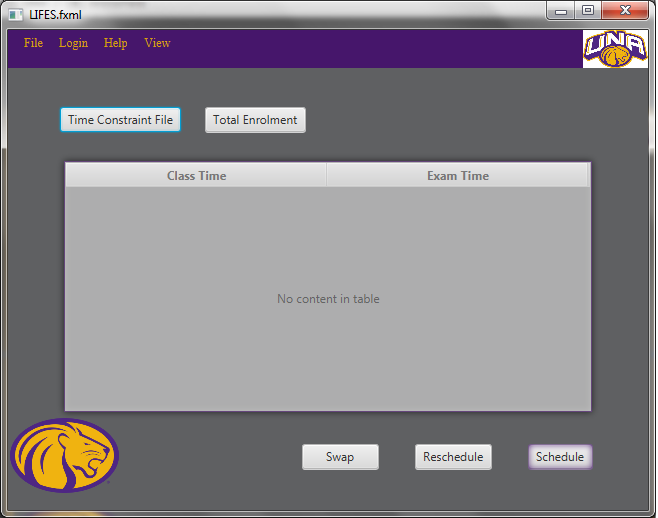
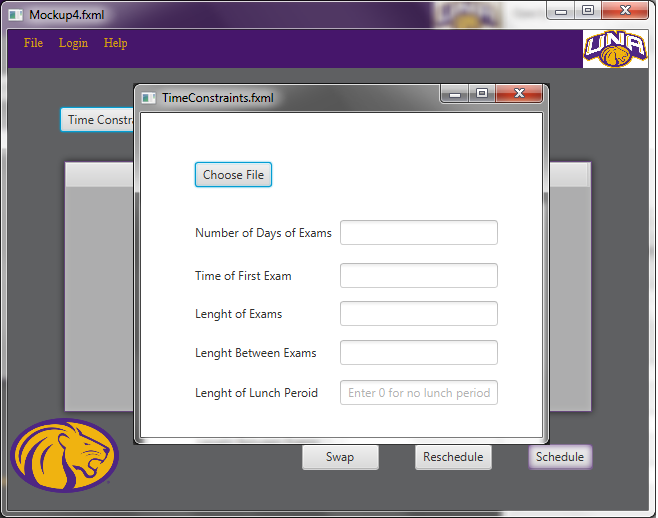
**

Figure 7 – Manual Data Input

**

Appendix B – Coding Standards



**Coding Standards**

University of North Alabama

CS 455

Software Engineering Project

Spring 2015

**Overview**

These coding standards have been developed by the Tune Squad to aid in developing a reliable and robust program. The standards focus on the C# programming language. Included in these standards are details about: (1) File organization; (2) Document Styles; (3) General Code Style; (4) Statements; (5) Comments; (6) Naming Conventions. Following the details about the styles is an example. Every developer working on the L.I.F.E.S. project will read the coding standards and review the example before any code is written.

GitHub shall be used for source code revision control. The Tune Squad’s repository on Github will be a point of reference for all testing guidelines.

The default font for text shall be Times New Roman Size 12 (Excluding the Specification Document). Source code placed in documents shall be in Consolas size 10.

**File Organization**

C# Source Files

Every class will be contained in a separate file. For example, class ExampleClasswill be contained in a file named ExampleClass.cs.

Directory Layout

There will be a directory created for every namespace. For example, FirstNameSpace.SecondNameSpacewill be contained inside of FirstNameSpace/SecondNameSpace.

**General**

A tab shall equal 4 spaces.

No line shall be longer than 80 characters.

All variables shall be declared before used.

All function/methods will begin with a capital letter.

All variables will begin with a lowercase letter.

All global variables, constants, and defined names shall be in all caps. Multiple words shall be separated with an underscore.

int CONTANT\_VAR = 1;

**Statements**

Write only one statement per line.

Write only one declaration per line.

If continuation lines are not indented automatically, indent them one tab stop (four spaces).

There will be one blank line before a method header and the method will follow directly after the header without a blank line.

Trailing braces shall be used everywhere on a new line (if, else-if, else, functions, structures, typedef, class definitions, etc.)

if (true)   
{

}

The else statement shall start on a new line after the last closing brace with one blank line in-between.

if (x)   
{

}   
  
else if (y)   
{

}   
  
else   
{

}

Parenthesized expressions will not be padded with spaces.

if (x)   
{

}

And

x = y \* (z + 5);

Not

if ( x )   
{

}

And

x = y \* ( z + 5 );

There will be exactly one space between operators.

x = y + z \* a;

not

x=y+z\*a;

**Commenting**

Place the comment on a separate line, not at the end of a line of code.

Begin comment text with an uppercase letter.

End comment text with a period.

Insert one space between the comment delimiter (//) and the comment text, as shown in the following example.

// This is a sample comment.

The use of multi-line comments is allowed.

/\*

\* Multi-line comments

\* can be used.

\*/

All methods will have a header of the following format:

/\*

\* Method: Name

\* Parameters: number and type of parameters

\*

\* Description: Describe the purpose of the method.

\*

\*/

All classes will have a header of the following format:

/\*

\* ClassName.cs

\*

\* Author:

\* Date:

\*

\* Description:

\*

\*/

**Naming**

All namespace, class, and method names will use PascalCasing. This convention capitalizes the first character of each word. For example:

public int MethodName()

File names shall also use PascalCasing. For example:

FileName.cs

Namespace components will be separated with periods.

 Microsoft.Office.PowerPoint

All variable names will use camelCasing. This convention ensures that the first letter of the name is lower case and each successive word starts with an upper case letter. For example:

int varName;

Ordering of class variables and methods shall be as follows:

1. public variables
2. protected variables
3. private variables
4. public methods
5. protected methods
6. private methods

Use predefined type names instead of system type names.

// Correct  
string firstName;  
int lastIndex;  
bool isSaved;

// Avoid  
String firstName;  
Int32 lastIndex;  
Boolean isSaved;

**Example Brace Placement**

namespace WhereIsTheBracket

{

public enum Test

{

TestMe,

TestYou

}

public class TestMeClass

{

Test test;

void DoSomething()

{

if (test == Test.TestMe)

{

// ...stuff gets done

}

else if (test == Test.TestThat)

{

// ...other stuff gets done

}

else

{

// ...some other stuff gets done

}

}

}

}