Software Requirements Specification

for

Ask! Forum

Version 1.0

Prepared by

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Instructor: | Assist. drd. ing. Cătălin SBORA |
| Course: | Web Application Design |
| Date: | Mar. 17th, 2020 |

Table of Contents

Table of Contents ii

1 Introduction 3

1.1 Document Purpose 3

1.2 Product Scope 3

1.3 Definitions, Acronyms and Abbreviations 3

1.4 References and Acknowledgments 4

2 Overall Description 5

2.1 End Users and Characteristics 5

2.2 System Stakeholders 5

2.3 Operating Environment 5

3 Specific Requirements 6

3.1 External Interface requirements 6

3.2 Functional requirements 8

3.3 Performance requirements 12

3.4 Design constraints 13

3.5 Software system attributes 13

4 Non-functional Requirements 15

4.1 Software Quality Attributes 15

# Introduction

## Document Purpose

This document describes the requirements for an IT system that helps people in different topics like software development, languages, science, etc. It will explain the purpose, features and the interfaces of the software, it’s role and the constraints under which it must operate. This document is intended for stakeholders and software developers.

## Product Scope

The application has a forum structure and is intended to be a website with the purpose of helping users doing activities of searching and solve their problems in an easy manner such as:

* An user has an account to acces the forum.
* There are 2 types of users: normal user and admin user.
* An admin user can perform important actions during web-site lifetime.
* Whenever a question is posted, a new topic starts.
* Asks something, wait to be answered by others experienced users.
* Every question or article can be commented by an user.
* An article can have comments with up or down votes.
* Every account has a rating level which can increase proportionally with number of up votes for an answer.
* If an user has a question or article with negatively rating, the overall account rating down.
* An user can give up or down votes for a question, article or comment posted.
* An user can receive up or down votes for question, article or comment posted.

## Definitions, Acronyms and Abbreviations

### Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AF | Ask! Forum application |
| JS | JavaScript |
| CSS | Cascading Style Sheets |
| SRS | Software Requirements Specification |
| OS | Operating System |

### Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SW | software |
| REQ | requirement |
| Admin | administrator |

## References and Acknowledgments

1. IEEE Template for System Requirement Specification Documents: https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE412/2-Intermediate\_Project\_Report\_Template-SRS.doc
2. https://stackoverflow.com/
3. https://www.w3schools.com/
4. https://www.tutorialspoint.com/index.htm

# Overall Description

## End Users and Characteristics

There are two types of users that interact with system: normal users and admins. Each of these two types of users has different use of the system so each of them has their own requirements.

**Normal users** can use the website to post questions about anything. For his questions, the user can receive upvotes or downvotes if the other users which view the question consider a good one or irrelevat. For one question can be more answers. Like a question, every answer (response of that) can have upvotes and downvotes but these have not influence the total rating of users.  
The upvotes and downvotes received for a question increase/decrease the total rating of user which post. A normal user can post articles and other users can appreciate them. Monthly, it is made a rank with all users by total rating.

**The admins** also only interact with the web forum. They are managing the overall system so there

is no incorrect information within it. The administrator can manage the information for each topic, can mark an answer as the best, can remove a topic if consider irrelevant and even remove an user which not follow terms and conditions of the forum.

## System Stakeholders

Our definition of a stakeholder represents any person or group who will be affected by the system, directly or indirectly. Stakeholders include end-users who interact with the system and everyone else in an organisation that may be affected by its installation. Other system stakeholders may be engineers who are developing or maintaining related systems, business managers, domain experts.

## Operating Environment

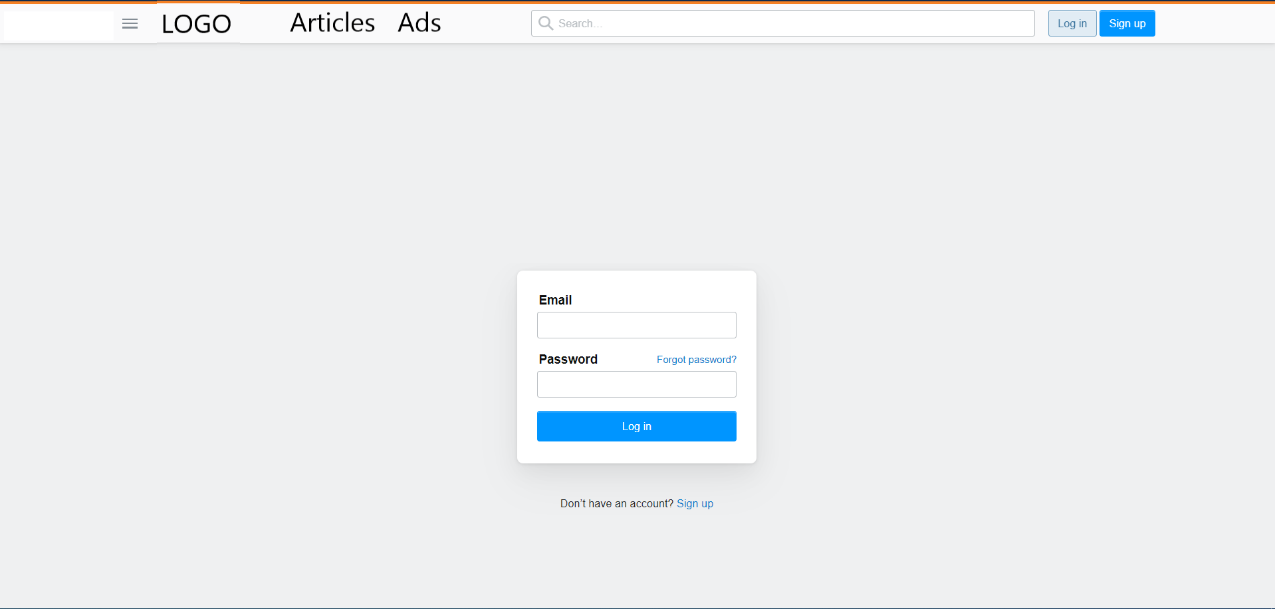
Compatible operating system:

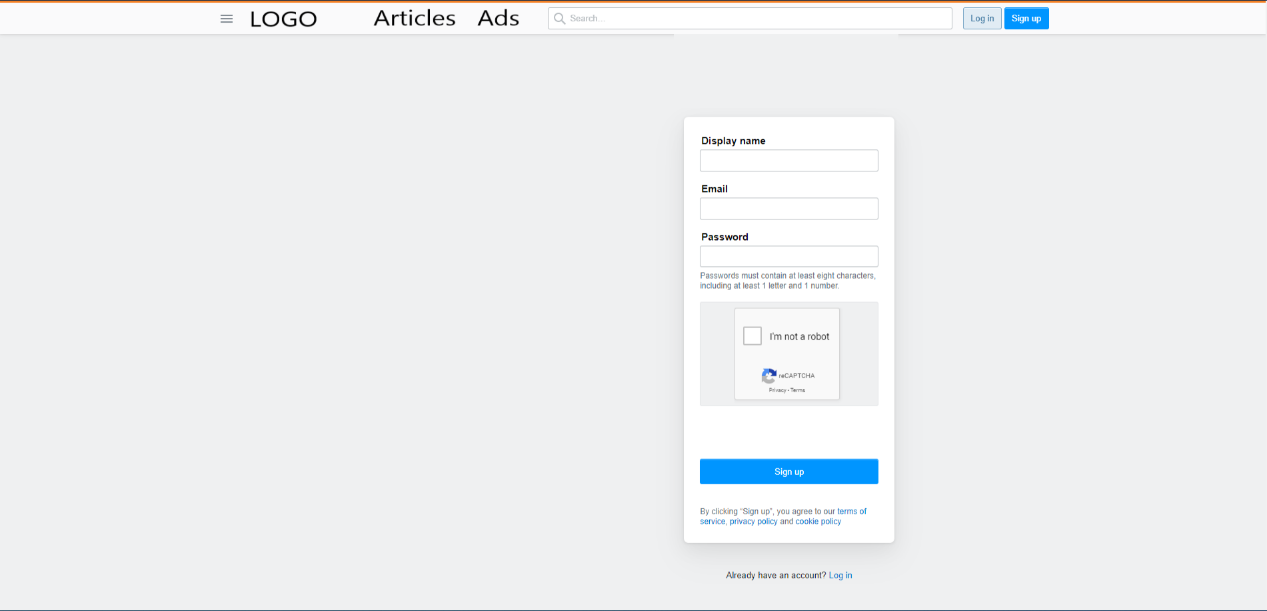
* Windows 7/ 8/ 8.1/ 10
* Mac OS X 10.11: El Capitan (Gala) or latest versions
* Linux

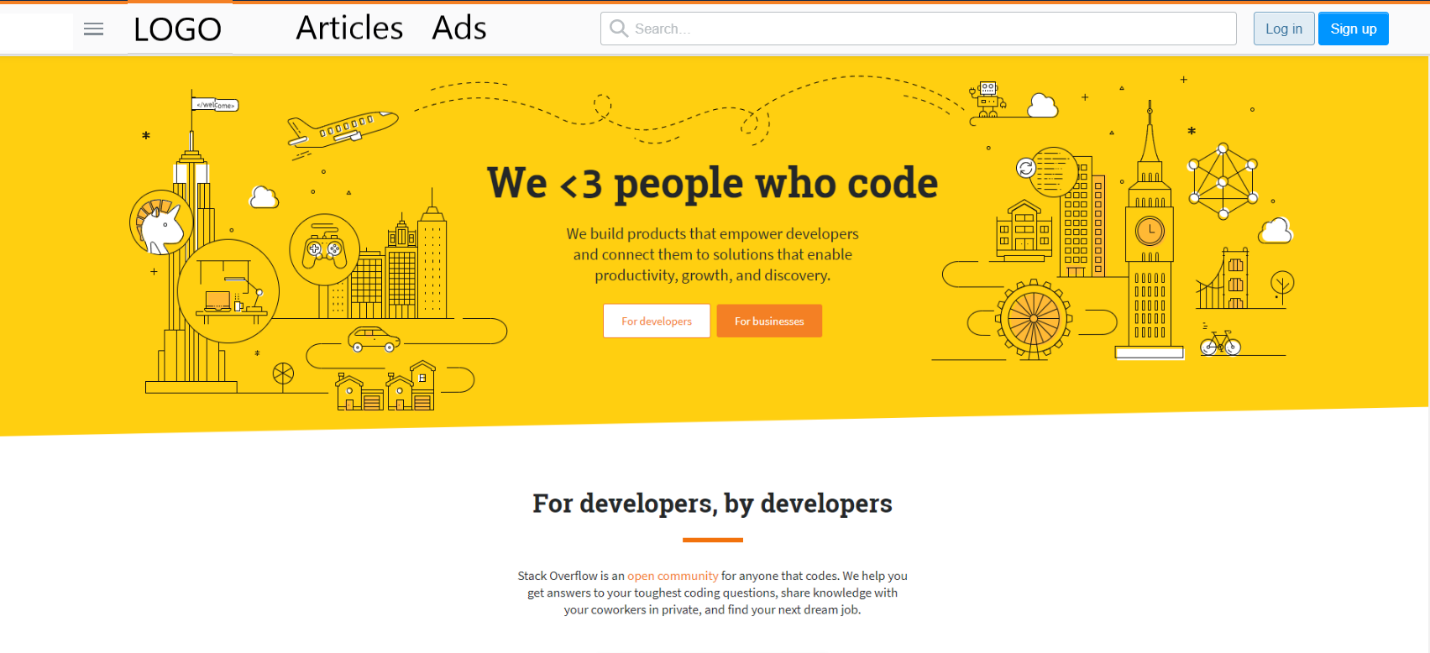
# Specific Requirements

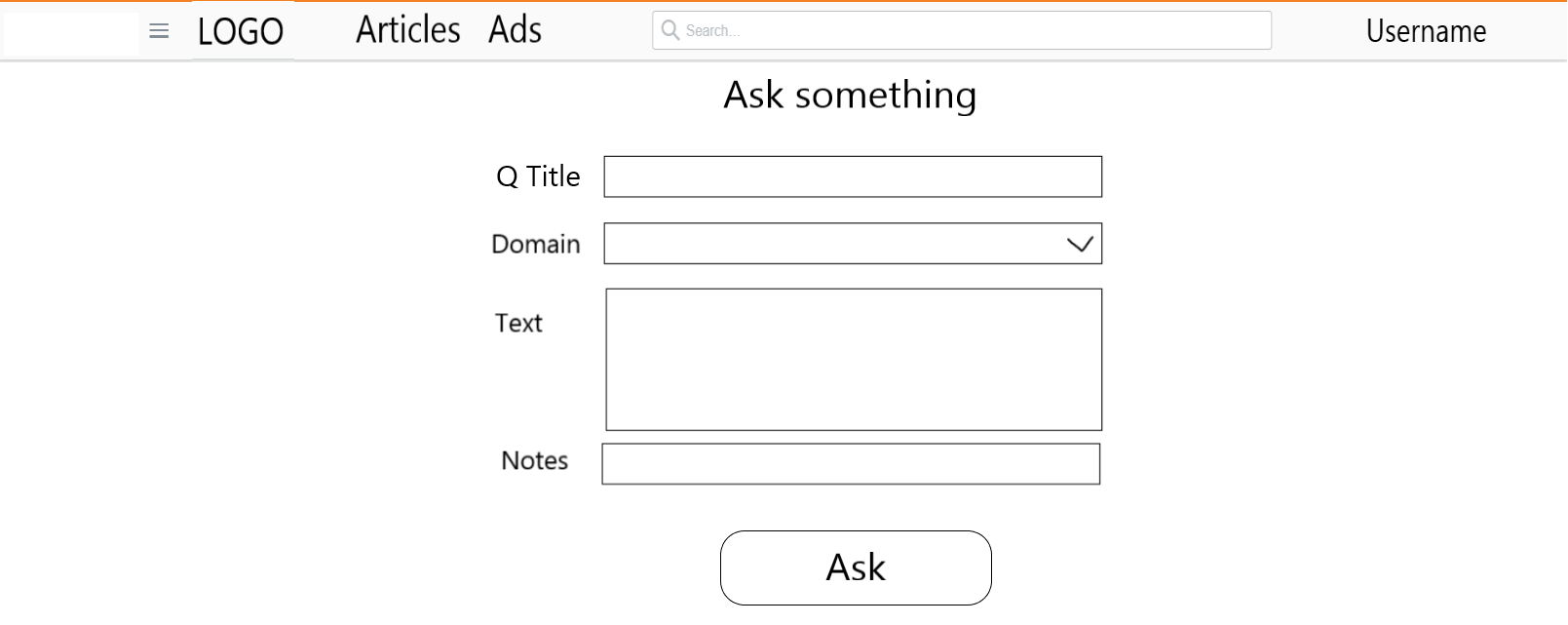
## External Interface requirements

### User interfaces











### Hardware interfaces

**Hardware minimum:**

Ensure that the system onto which you install Discovery Studio meets these minimum requirements:

* OS: Windows 7 64 Bit Service Pack 1
* 32-bit Intel® Pentium® 4 or compatible processor running at 2 GHz or greater
* 2 GB RAM
* Sound Card: 100% DirectX 10 compatible
* Disk space: 25 GB for client components
* Internet access is helpful

### Software interfaces

The communication between the database and the web portal consists of operation concerning both reading and modifying the data using Entity Framework.

### Communications interfaces

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems for website.

## Functional requirements

This section includes the requirements that specify all the fundamental actions of the software system.

### User Class 1 - The User

#### Functional requirement 1.1

ID: FR1

TITLE: User registration

DESC: Given that a user enter on website, then the user should be able to register. The user must provide a name which display on website, password and e-mail address.

RAT: In order for a user to register on the web forum.

DEP: -

ID: FR2

TITLE: User log-in

DESC: Given that a user has registered, then the user should be able to log in to the web forum.

The log-in information will be stored on the e-mail and in the future the user should be logged in

automatically.

RAT: In order for a user to register on the forum.

DEP: FR1

ID: FR3

TITLE: Retrieve password

DESC: Given that a user has registered, then the user should be able to retrieve his/her password by e-mail.

RAT: In order for a user to retrieve his/her password.

DEP: FR1

ID: FR4

TITLE: Search for a specific question

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to search for a question.

RAT: In order for a user to search for a question.

DEP: FR1

ID: FR5

TITLE: Search for a specific article

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to search for an article.

RAT: In order for a user to search for an article.

DEP: FR1

ID: FR6

TITLE: Search for an ad.

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to search for an ad.

RAT: In order for a user to search for a ad.

DEP: FR1

ID: FR7

TITLE: Add an article

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to add a new article.

RAT: In order for a user to add a new article.

DEP: FR1

ID: FR8

TITLE: Add a new question.

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to add a new question.

RAT: In order for a user to add a new question.

DEP: FR1

ID: FR9

TITLE: Add an answer for a given question.

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to add an answer for a specific question.

RAT: In order for a user to add a new answer into a topic.

DEP: FR8

ID: FR10

TITLE: Add an ad.

DESC: Given that a user is logged in to the web forum, then the first page that is shown should be the home page. The user should be able to add a new ad.

RAT: In order for a user to add a new ad.

DEP: FR1

ID: FR11

TITLE: Give an upvote/downvote to a question.

DESC: The user should be able to give a vote (up or down) to a question.

RAT: In order for a user to give a vote.

DEP: FR9

ID: FR12

TITLE: Give an upvote/downvote to an answer.

DESC: The user should be able to give a vote (up or down) to an answer.

RAT: In order for a user to give a vote.

DEP: FR8

ID: FR13

TITLE: Give an upvote/downvote to an article.

DESC: The user should be able to give a vote (up or down) to an article.

RAT: In order for a user to give a vote.

DEP: FR7

### User Class 2 – The Admin

ID: FR14

TITLE: Admin registration

DESC: The admin should be able to register at first use of site. He must provide a name which display on website, password and e-mail address.

RAT: In order for an admin to register on the web forum.

DEP: -

ID: FR15

TITLE: Admin log-in

DESC: Given that a admin has registered, then the admin should be able to log in to the web forum.

The log-in information will be stored on the e-mail and in the future the user should be logged in

automatically.

RAT: In order for an admin to register on the forum.

DEP: FR13

ID: FR16

TITLE: Retrieve password

DESC: Given that an admin has registered, then the admin should be able to retrieve his/her password by e-mail.

RAT: In order for an admin to retrieve his/her password.

DEP: FR13

ID: FR17

TITLE: Manage recently questions in forum.

DESC: The admin can remove a question if that not follow the rules of forum.

RAT: In order to manage questions in forum.

DEP: FR14, FR8

ID: FR18

TITLE: Manage recently articles in forum.

DESC: The admin can remove an article if that not follow the rules of forum.

RAT: In order to manage articles posted in forum.

DEP: FR14, FR7

ID: FR19

TITLE: Manage recently ads in forum.

DESC: The admin can remove an ad if that not follow the rules of forum.

RAT: In order to manage new ads in forum.

DEP: FR14, FR10

ID: FR20

TITLE: Manage recently answers in forum.

DESC: The admin can remove a question if that not follow the rules of forum (content, language).

RAT: In order to manage answers in forum.

DEP: FR14, FR9

ID: FR21

TITLE: Remove an active user.

DESC: The admin can remove an active user if he does not follow the rules of forum.

RAT: In order to manage active users and their activity in forum.

DEP: FR1

ID: FR22

TITLE: Remove an inactive user.

DESC: The admin can remove an if he not follow the rules of forum and he does not have any activity in the last 3 months.

Activity means at least a login process made, a question posted, an answer, an article or promote an ad.

RAT: In order to manage inactive users in forum.

DEP: FR1

ID: FR23

TITLE: Closed topic if is complete and correct.

DESC: The admin can close a topic if the question has at least one answer which respond complete and correct.

DEP: FR8

## Performance requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance.

### Prominent search feature

ID: QR1

TITLE: Prominent search feature

DESC: The search feature should be prominent and easy to find for the user.

RAT: In order to for a user to find the search feature easily.

DEP: none

### Usage of the search feature

ID: QR2

TITLE: Usage of the search feature

DESC: The different search options should be evident, simple and easy to understand.

RAT: In order to for a user to perform a search easily.

DEP: none

### Usage of the result in the list view

ID: QR3

TITLE: Usage of the result in the list view

DESC: The results displayed in the list view should be user friendly and easy to understand. Selecting an element in the result list should only take one click.

RAT: In order to for a user to use the list view easily.

DEP: none

### Response time

ID: QR4

TAG: ResponseTime

GIST: The fastness of the search

SCALE: The response time of a search

METER: Measurements obtained from 1000 searches during testing.

MUST: No more than 2 seconds 100% of the time.

WISH: No more than 1 second 100% of the time.

### System dependability

ID: QR5

TAG: SystemDependability

GIST: The fault tolerance of the system.

SCALE: If the system loses the connection to the Internet the system gets some

strange input, the user should be informed.

METER: Measurements obtained from 1000 hours of usage during testing.

MUST: 100% of the time.

## Design constraints

This section includes the design constraints on the software caused by the hardware.

### Application memory usage

ID: QR6

TAG: ApplicationMemoryUsage

GIST: The amount of Operate System memory occupied by the application.

SCALE: MB.

METER: Observations done from the performance log during testing

MUST: No more than 20 MB.

PLAN: No more than 16 MB

WISH: No more than 10 MB

Operate System: DEFINED: The Windows/Mac/Linux OS which the application is running on.

MB: DEFINED: Megabyte.

## Software system attributes

The requirements in this section specify the required reliability, availability, security and maintainability of the software system.

### Reliability

ID: QR7

TAG: SystemReliability

GIST: The reliability of the system.

SCALE: The reliability that the system gives the right result on a search.

METER: Measurements obtained from 1000 searches during testing.

MUST: More than 98% of the searches.

PLAN: More than 99% of the searches.

WISH: 100% of the searches.

### Availability

ID: QR8

TAG: SystemAvailability

GIST: The availability of the system when it is used.

SCALE: The average system availability (not considering network failing).

METER: Measurements obtained from 1000 hours of usage during testing.

MUST: More than 98% of the time.

PLAN: More than 99% of the time.

WISH: 100% of the time.

ID: QR9

TITLE: Internet Connection

DESC: The application should be connected to the Internet.

RAT: In order for the application to communicate with the database.

DEP: none

### Security

ID: QR10

TAG: AdminLoginAccountSecurity

GIST: Security of accounts.

SCALE: If an admin tries to log in to the web portal with a non-existing account then the admin should not be logged in. The admin should be notified about log-in failure.

METER: 1000 attempts to log-in with a non-existing user account during testing.

MUST: 100% of the time.

ID: QR11

TAG: UserCreateAccountSecurity

GIST: The security of creating account for users of the system.

SCALE: If a user wants to create an account and the desired user name is occupied, the user should be

asked to choose a different user name.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time.

### Maintainability

ID: QR12

TITLE: Application extendibility

DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

RAT: In order for future functions to be implemented easily to the application.

DEP: none

ID: QR13

TITLE: Application testability

DESC: Test environments should be built for the application to allow testing of the applications different functions.

RAT: In order to test the application.

DEP: none

### Portability

ID: QR14

TITLE: Application portability

DESC: The application should be portable with different OS like Windows/Mac/Linux.

RAT: The adaptable platform for the application to run on.

DEP: none

# Non-functional Requirements

## Software Quality Attributes

### Software Quality Attributes

In order to guarantee a much more cleaner and readable code, we have chosen a set of well-defined rules such as:

1. **One Class per File**: Source files should contain one class definition per source file. Said differently, each class definition will exist within its own file. The stem of the file name must be the same name as the name used in the class declaration. For example, the class definition for a class named Bank will have a file name of Bank.cs.
2. **Indentation**:Indentation is constructed with tabs, not spaces. Typically, tabs are set to be displayed as white space with a width of four characters.
3. **Line Length Optimizing** for down level tools and editors such as Notepad should not impact code style. 80 character lines are a recommendation, not a hard and fast rule.
4. **Comments**: C# programs can have two kinds of comments: implementation comments and documentation comments. Implementation comments are those found in C++, which are delimited by /\*...\*/, and //. Implementation comments are meant for commenting out code or for comments about the particular implementation. Doc comments are meant to describe the specification of the code, from an implementation-free perspective, to be read by developers who might not necessarily have the source code at hand.
5. **Number Per Line**: One declaration per line is recommended since it encourages commenting.
6. **Initialization**: We initialize local variables where they're declared. The only reason not to initialize a variable where it's declared is if the initial value depends on some computation occurring first. In this case declare the int and initialize it with an appropriate value.
7. **Placement**: We have decided to put declarations only at the beginning of blocks. (A block is any code surrounded by curly braces "{" and "}".) Don't wait to declare variables until their first use; it can confuse the unwary programmer and hamper code portability within the scope.
8. **Class and Interface Declarations**: When coding C# classes and interfaces, the following formatting rules should be followed:
   1. No space between a method name and the parenthesis "(" starting its parameter list
   2. Open brace "{" appears at the beginning of the line following declaration statement and is indented to the beginning of the declaration.
   3. Closing brace "}" starts a line by itself indented to match its corresponding opening statement. For null statements, the "}" should appear immediately after the "{" and both braces should appear on the same line as the declaration with 1 blank space separating the parentheses from the braces.
   4. Methods are separated by two blank lines.
   5. Every interface name should start with an uppercase letter “I”.
9. **Properties**: If the body of the get or set method of a property consists of a single statement, the statement is written on the same line as the method signature. White space is inserted between the property method (get, set) and the opening brace. This will create visually more compact class definitions.
10. **Simple Statements**: Each line should contain at most one statement.
11. **return Statements**: A return statement with a value should not use parentheses unless they make the return value more obvious in some way.
12. **if** statements always use braces {}.
13. A **try-catch** statement may also be followed by finally, which executes regardless of whether or not the try block has completed successfully.
14. A **keyword** followed by a parenthesis should be separated by a space.

* **Naming Rules**: Naming rules make programs more understandable by making them easier to read.

1. **Names of methods**: should contain active verb forms and imperatives (DeleteOrder, OpenSocket). It is not necessary to include the noun name when the active verb refers directly to the containing class.
2. In method overloading, all overloads should perform a similar function.
3. **Collections** should be named as the plural form of the singular objects that the collection contains. A collection of Book objects is named Books.
4. **Boolean variable** names should contain “Is” or “is” which implies Yes/No or True/False values, such as isFound, or isSuccess.
5. **Temporary variables** should always be used for one purpose only; otherwise, several variables should be declared.
6. **Pascal case**: The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. You can use Pascal case for identifiers of three or more characters. For example: BackColor.
7. **Camel case**: The first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized. For example: backColor.