Requirement Specification

For Developing a Forum

Contents

1. Introduction 3

2. Use cases and User Stories 4

2.1 Use Cases 4

2.2 User Stories 5

3. Functional Requirements 18

4. Non – Functional Requirements 23

1. Introduction

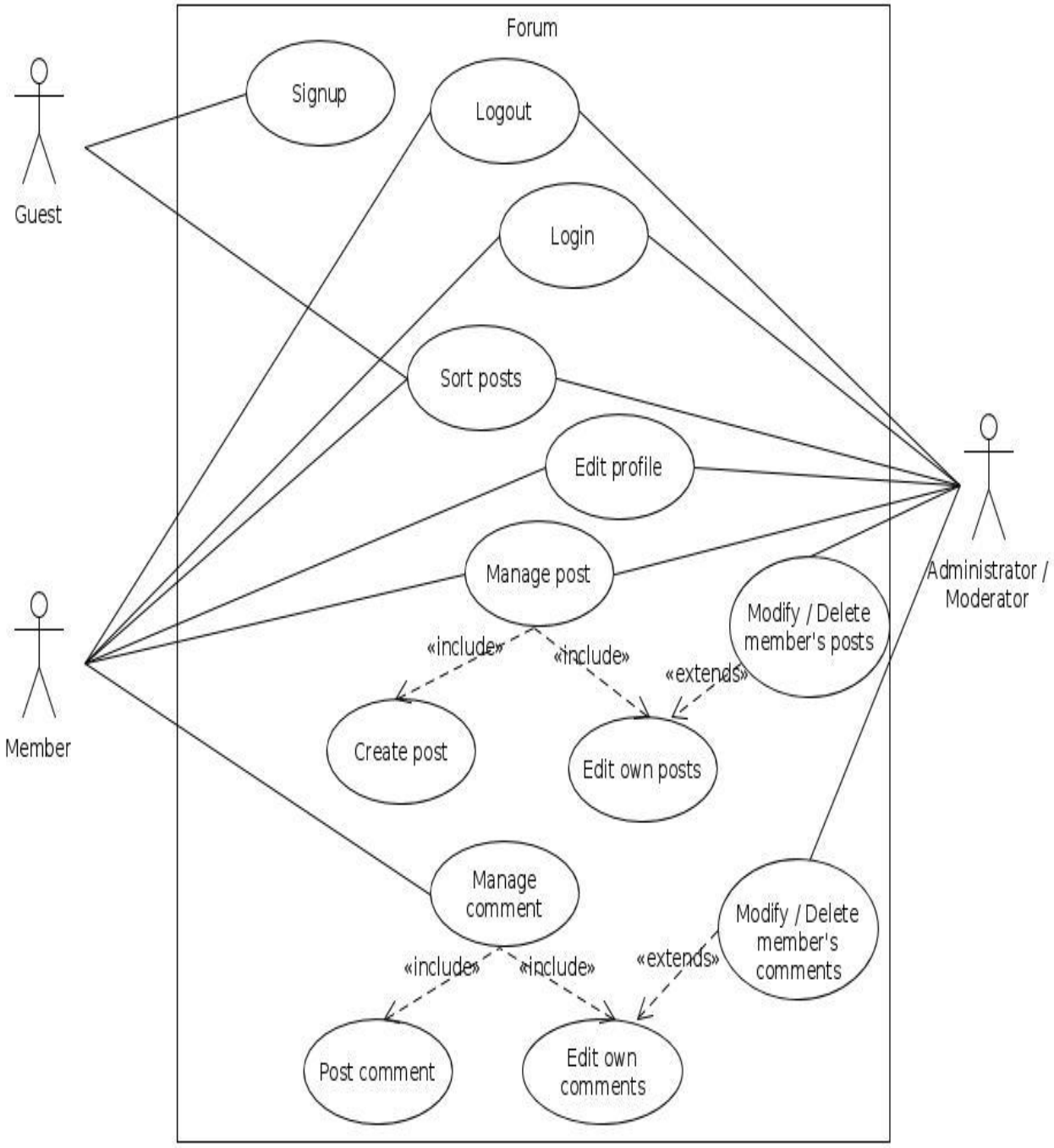
The Requirement Specification focuses on the introduction of the project including the purpose, scope as well as the details of the system requirements.

Based on this requirements specification, implementation of each function including all of the conditions as well as functional and non-functional requirements, supplied by the customers, will be made.

This document helps the reader to understand the Online Forum of SET Software and Web Application Company by providing details into the product features.

2. Use cases and User Stories

2.1 Use case Diagram



2.2 Use case Diagram

1. USE CASE 1

Name: *Sign up*

Identifier: *UC1*

User story: *US1 – As a guest, I want to sign up so that I can have all of the member's accessibilities*

Inputs:

1. User name.
2. Password.

Outputs:

1. Homepage.

Preconditions:

1. User name must be unique (that is, no one has used the same user name before).

Basic courses:

Actor: Guest	**System**
1. Open the Sign up page	1.1 Display the Sign up page
2. Enter user name and password into the corresponding required boxes	
3. Submit	3.1 Check user's inputs 3.1.1 Check user name's availability 3.1.1 Check password's validation 3.2 Redirect user to home page, with user's account automatically logged in

Alternate courses:

- *Condition 1*: *User name and password are invalid*
  In step 3, if user name and password inputted by guest are invalid, the system will display an error message "Invalid user name or password", and prompt guest to choose a different user name / password to register again.

Post conditions:

1. User's new account must be stored in the database.

2. USE CASE 2

Name: *Sort posts*

Identifier: *UC2*

User story: *US2 – As a guest / members / moderator / administrator, I want to choose the way posts are arranged in the corresponding thread*

Inputs:

1. Way to sort posts.
2. Thread ID.

Outputs:

1. Thread page.

Preconditions:

1. None.

Basic courses:

Actor: Guest / User	**System**
1. Open a thread	1.1 Display all posts in the chosen thread in default order
2. Choose the way to sort the posts	2.1 Extracting each post's information 2.2 Get the sorting order that guest / user chose 2.3 Sort and display the post in the corresponding order

Alternate courses:

- *Condition 1*: *Guest / User can choose to arrange the posts in two (or more) specific orders*
  In step 2, if guest / user has chosen to arrange the posts in two (or more) orders, all the posts will be sorted accordingly, with the high priority orders are considered first (the priority for each sorting method is set by the system).

Post conditions:

1. All posts in the thread must be sorted as guest / user wished.

3. USE CASE 3

Name: *Login / Logout*

Identifier: *UC3*

User story: *US3 – As a member / moderator / administrator, I want to login /logout*

Inputs:

1. User name and password, for Login function.
2. User ID, for Logout function.

Outputs:

1. Homepage, after user has logged in successfully.
2. Front page, after user has logged out successfully.

Preconditions:

1. User name and password must be valid, for Login function.
2. User has not logged in before, for Login function.
3. User must have already registered the corresponding account before, for Login function.
4. User has already logged in, for Logout function.

Basic courses:

Actor: User	**System**
1. Choose Logout action	1.1 Retrieve user object from session 1.2 Erase current user object from session 1.3 Redirect user to the front page
1. Choose Login action	1.2 Redirect user to Login page 1.3 Prompt for User name and Password
2. Enter user name and password	2.1 Validate inputs 2.2 Retrieve matching user from database 2.3 Record the result user object to session 2.4 Display successful message 2.5 Redirect user to the homepage

Alternate courses:

- *Condition 1*: *User enters invalid user name and password*
  In step 2 (for Login action), after validating user's inputs, the system displays a message warning user "Invalid user name or password" and asks user to login again.

Post conditions:

1. User successfully logs in, for Login function.
2. User successfully logs out, for Logout function.

4. USE CASE 4

Name: *Create new post*

Identifier: *UC4*

User story: US4 – *As a member / moderator / administrator, I want to create a new post*

Inputs:

1. Description for post.
2. User ID.
3. Current Thread object.

Outputs:

1. A new post, added into the current thread by the user initiating action.

Preconditions:

1. User must log in as a member / moderator / administrator.
2. The current thread is capable to include a new post.
3. User must be in the front page (where all threads and posts are displayed) to create a new post.

Basic courses:

Actor: *User*	**System**
1. Click 'Create Post' button	1.1 Request description for the post
2. Enter description	2.1 Create a new post with the given inputs 2.2 Add the created post into the current thread, and store it in the database 2.3 Display message notifying the post has been created successfully 2.4 Redirect user to the new post's page

Alternate courses:

- *Condition 1: The current thread is not able to include anymore posts (out of memory)*
  In step 2, after user has submitted the description, the system displays the warning message, asks user to do this later, and notifies the system owner about the situation (to upgrade database for threads).

- *Condition 2*: *The description inputted is invalid*
  In step 2, after user has submitted the description, the system displays a warning message for invalid input, and prompts user for another one.

Post conditions:

1. A new post is created and stored in the database. It must also be added to the current thread, and assigned the current user's ID.

5. USE CASE 5

Name: *Edit post*

Identifier: *UC5*

User story: *US5 – As a member / moderator / administrator, I want to edit my own posts*

Inputs:

1. User ID.
2. New description for post.
3. Current Post object.

Outputs:

1. The post with updated content according to user.

Preconditions:

1. User must be authorized to edit the post (that is, user must be the owner of the post).
2. The post must still exist in the database.
3. User must log in as a member / moderator / administrator.
4. User must be in the post page to edit the post.

Basic courses:

Actor: *User*	**System**
1. Click 'Edit Post' button	1.1 Check user authorization 1.2 Display current content of the post, and request new description for it
2. Enter new description	2.1 Update the content of the post according to the given input 2.2 Display message notifying that the change has been made successfully 2.3 Refresh the page and display the post with the newly modified content

Alternate courses:

- *Condition 1: User is not authorized to edit the post*
  In step 1, after user has clicked the 'Edit Post' button, the system displays a message warning user that he/she is not allowed to edit this post.

- *Condition 2: The post is not existed in the database (e.g. removed by administrator)*
  In step 1, after user has clicked the 'Edit Post' button, the system displays a message warning user that the post has been removed (e.g. by administrator).

Post conditions:

1. The post is modified accordingly in the database.

6. USE CASE 6

Name: *Post comment*

Identifier: *UC6*

User story: *US6 – As a member / moderator / administrator, I want to post a new comment*

Inputs:

1. Content of the comment.
2. User ID.
3. Current Post object.

Outputs:

1. The current post with the new comment added.

Preconditions:

1. The post needed to add comment into must exist in the database.
2. User must log in as a member / moderator / administrator.

Basic courses:

Actor: *User*	**System**
1. Choose and open a post page	1.1 Display the post page
2. Enter a comment in the text box	
3. Submit	3.1 Create a new comment with the given inputs 3.2 Add the comment into the current post, and store the comment in the database 3.3 Display comment's content in a block

Alternate courses:

- *Condition 1: The current post needed to add comment into had been removed*
  In step 1, after user has chosen the post, the system displays a message notifying user that the post had been removed.

Post conditions:

1. The comment is added into the respective post successfully.
2. The comment is stored in the database successfully.

7. USE CASE 7

Name: *Edit comment*

Identifier: *UC7*

User story: *US7 – As a member / moderator / administrator , I want to edit my own comments*

Inputs:

1. New content for the comment.
2. User ID.
3. Current comment object.

Outputs:

1. The comment with new content updated.

Preconditions:

1. The comment to be modified must exist in the database.
2. The post containing the comment must exist in the database.
3. User must log in to the system.
4. User must be the owner of the comment.

Basic courses:

Actor: User	**System**
1. Choose and open a Post page	1.1 Display the Post page
2. Choose a comment to edit	2.1 Check and validate user's ID 2.2 Display current content of the comment 2.3 Prompt for modified content
3. Enter the modified content of the comment to the display box	
4. Submit edited comment	4.1 Update the comment accordingly 4.2 Record the update in the database 4.3 Display the modified comment in the Post page

Alternate courses:

- *Condition 1*: *User is not allowed to edit the comment*
  In step 2, after checking user's ID and seeing that this user is not allowed to edit the comment, the system disables the Edit function and displays a warning message: "You are not allowed to modify this comment".

- *Condition 2*: *The modified content violates the rules*
  In step 4, after user has submitted the modified content which violates the forum's rules, the system displays a message warning user: "This content is against the rules of the forum. Please submit another description" and waits for user to input another description.

Post conditions:

1. The content of the comment is modified accordingly.

2. The edited comment must be valid and not violate any rules.

8. USE CASE 8

Name: *Edit profile*

Identifier: *UC8*

User story: *US8 – As a member / moderator / administrator, I want to edit my own profile*

Inputs:

1. User ID.
2. New user name (if any).
3. New password (if any).
4. New profile picture (if any).
5. New profile description (if any).

Outputs:

1. Profile with updated content according to user's inputs.

Preconditions:

1. User must be the profile owner.
2. User must have already logged in.
3. User must be in his / her Profile Page to perform Edit profile function.

Basic courses:

Actor: User	**System**
1. Choose Edit profile function	1.1 Redirect user to the Edit profile page 1.2 Prompt for inputs
2. Enter new content of attributes that user wants to change	2.1 Record the inputs 2.2 Retrieve user ID from session 2.3 Perform modifications for the corresponding user based on user ID 2.4 Display successful message 2.5 Redirect user to the Profile page

Alternate courses:

Post conditions:

1. The user's profile is updated accordingly in the database.

9. USE CASE 9

Name: *Delete members' comment*

Identifier: *UC9*

User story: *US9 – As a moderator / administrator, I want to delete other members' comments if they violated the forum's regulations*

Inputs:

1. User ID (of moderator / administrator).
2. Comment ID.
3. ID of user that posted the comment.
4. Reason to delete the comment.

Outputs:

1. Notification mail.

Preconditions:

1. User must be a moderator / an administrator. In case of moderator, user must have privilege over the post which contains the comment he / she wants to delete.
2. User must have already logged in.
3. The member's comment must violate forum's regulations.
4. The member's comment and its post must still exist in the database.

Basic courses:

Actor: Moderator / Administrator	**System**
1. Click 'Delete comment' button of the violated comment	1.1 Retrieve user's ID and validate his / her authorization to delete the comment. 1.2 [Authorized] Ask moderator / administrator for the reason to delete the comment (list some options to choose)
2. Choose the option specifying the correct reason	2.1 Record the reason chosen 2.2 Retrieve the selected Comment object, and erase it from its post and the database 2.3 Send a notification message to the owner of the comment, specifying why his / her comment is removed 2.4 Redirect to the current Post page

Alternate courses:

- *Condition 1*: *Moderator does not have authorization to delete the comment*
  In step 1, if moderator does not have authorization to delete the selected comment, the system will display a warning message “Permission denied: You do not have privilege to perform this action”.

- *Condition 2*: *The selected comment have already been removed from the database*
  In step 2, if the selected comment to be deleted has already been removed, the system will notify the moderator / administrator “This comment does not exist”, and reload the Post page to update the list of current existed comments.

Post conditions:

1. The comment is removed from the corresponding post and database successfully.
2. The owner of the comment is reported about his / her comment being deleted.

10. USE CASE 10

Name: *Delete members' post*

Identifier: *UC10*

User story: *US10 – As a moderator / administrator, I want to delete other members' posts if they violated the forum's regulations*

Inputs:

1. User ID (of moderator / administrator).
2. Post ID.
3. ID of user that created the post.
4. Reason the delete the post.

Outputs:

1. Notification mail.

Preconditions:

1. User must be an administrator, or a moderator that have the authorization in managing the post.
2. User must have already logged in.
3. The post must violate the forum's regulations.
4. The post must still exist in the database.

Basic courses:

Actor: Moderator / Administrator	**System**
1. Click 'Delete post' button of the violated post	1.1 Retrieve user's ID and validate his / her authorization to delete the post. 1.2 [Authorized] Ask moderator / administrator for the reason to delete the post (list some options to choose)
2. Choose the option specifying the correct reason	2.1 Record the reason chosen 2.2 Retrieve the selected Post object, and erase it from its thread and the database 2.3 Send a notification message to the owner of the post specifying why his / her post is removed 2.4 Redirect to the corresponding Thread page

Alternate courses:

- Condition 1: Moderator does not have authorization to delete the comment
  In step 1, if moderator does not have authorization to delete the selected post, the system will display a warning message “Permission denied: You do not have privilege to perform this action”.

Post conditions:

1. The post is removed successfully from the database.
2. All of the comments in the post are deleted accordingly.
3. The owner of the post is notified about his / her post being deleted.

3. Functional Requirements

USE CASE 1: SIGN UP

1. **The scope of the work:**
   - This occurs in sprint 4 in the process.
   - 4 tasks are needed for this function.
   - 39 hours of effort are needed.

2. **Functional and Data requirements:**

   a) Functional Requirement:
   - Shall display a page requiring user's information.
   - Shall validate username and password.
   - Shall create new account in database after successful registration.
   - Shall display the confirmation of the registration result.

   b) Data Requirement:
   - Account information must be valid, which means still available.

USE CASE 2: SORT POST

1. **The scope of the work:**
   - This occurs in sprint 6 in the process.
   - 4 tasks are needed for this function.
   - 28 hours of effort are needed.

2. **Functional and Data Requirements:**

   a) Functional Requirements:
   - Shall allow user to choose among different sorting options.
   - Shall display posts correctly in the corresponding order (latest first, most popular first, etc).

   b) Data Requirements:
   - View count of each post.
   - Rate of each post.
   - Date and time that the post is created.
   - Post's ID

USE CASE 3: LOGIN / LOGOUT

1. **The scope of the work:**
   - This occurs in sprint 4 in the process.
   - 7 tasks are needed for this function.
   - 39 hours of effort are needed.

2. **Functional and Data Requirements:**

a) Functional Requirements:

i. Login
   - Shall display a page prompting for username and password to login.
   - Shall validate inputs (username, password) with the database.
   - Shall notify user if the login attempt is unsuccessful (username and password do not match, etc).
   - Shall redirect user to the main page after logging in successfully.

ii. Logout
   - Shall display the logout icon on screen only when users have already logged in.
   - Shall erase user object from the current session and close all user's activities after user has logged out.
   - Shall redirect user back to the login page after user has logged out successfully.

b) Data Requirements:
- The user account must be valid for user to be able to login.

USE CASE 4: CREATE POST

1. **The scope of the work:**
   - This occurs in sprint 1 of the process.
   - 7 tasks are needed for this function.
   - 35 hours of effort are needed.

2. **Functional and Data Requirements:**

a) Functional Requirements:
- Shall display "Create new post" button in a thread.
- Shall display all requirements of creating a new post for members to fill out.
- Shall check for post contents violation.
- Shall receive all the information that members having filled out and save into database of that thread.
- Shall create a new legal post after having performed all the checking.
- Shall store the created post in the database, and register it with the corresponding thread.

b) Data Requirements:

- The content of post cannot have inappropriate characters.
- The content of post cannot violate any of the rules.

USE CASE 5: EDIT POST

1. **The scope of the work:**
   - This occurs in sprint 1 of the process
   - 7 tasks are needed for this function.
   - 35 hours of effort are needed.

2. **Functional and Data Requirements:**

   a) Functional Requirements:

   - Shall check user authorization on the post
   - Shall allow the user to re-type the content of the post in a text box.
   - Shall display the post in a block.
   - Shall modify the post in the database accordingly.

   b) Data Requirements:

   - The content of post cannot have inappropriate characters after being modified.
   - The content of post cannot violate any of the rules after being modified.

USE CASE 6: POST COMMENT

1. **The scope of the work:**
   - This occurs in sprint 2 of the process.
   - 7 tasks needed for this function.
   - 36 hours of effort is needed for this function.

2. **Functional and Data Requirements:**

   a) Functional Requirements:

   - Shall check user privilege (guest is not allowed).
   - Shall allow the user to type the comment in a text box.
   - Shall display the comment in a block.
   - Shall store the comment in the database.
   - Shall register the comment with the corresponding post.

   b) Data Requirements:

   - The comment cannot have inappropriate characters.
   - The content cannot violate any of the rules.

USE CASE 7: EDIT COMMENT

1. **The scope of the work:**
   - This occurs in sprint 2 of the process.
   - 7 tasks needed for this function.
   - 36 hours of effort is needed for this function.

2. **Functional and Data Requirements:**
   a) Functional Requirements:
   - Shall check user authorization on the comment.
   - Shall allow the user to re-type the comment in a text box.
   - Shall display the comment in a block.
   - Shall modify the comment in the database.

   b) Data Requirements:
   - The comment cannot have inappropriate characters after being modified.
   - The content cannot violate any of the rules after being modified.

USE CASE 8: EDIT PROFILE

1. **The scope of the work:**
   - This occurs in sprint 5 of the process.
   - 5 tasks needed for this function.
   - 32 hours of effort is needed for this function.

2. **Functional and Data Requirements:**
   a) Functional Requirements:
   - Shall display the “edit profile” button on screen when users have already logged in.
   - Shall display the profile page after click on “edit profile” button.
   - Shall display all of the current information (name, avatar …) of the user and the confirm button.
   - Shall update new information in the account database after users have confirmed their new edited information.

   b) Data Requirements:
   - All user data must be updated and stored in the database accordingly.

USE CASE 9: DELETE VIOLATED COMMENT

1. **The scope of the work:**
   - This occurs in sprint 3 of the process.
   - 7 tasks needed for this function.
   - 34 hours of effort is needed for this function.

2. **Functional and Data Requirements:**
   a) Functional Requirements:
   - Shall check the user authorization on deleting the comment.
   - Shall show the delete button on the comment.
   - Shall display successful or error notification after having deleted comment.
   - Shall alert the owners about the removal of their comments via emails.

   b) Data Requirements:
   - The deleted comment shall be remove from the database.

USE CASE 10: DELETE VIOLATED POST

1. **The scope of the work:**
   - This occurs in sprint 3 of the process.
   - 7 tasks needed for this function.
   - 34 hours of effort is needed for this function.

2. **Functional and Data Requirements:**
   a) Functional Requirements:
   - Shall check the user authorization on deleting the post.
   - Shall show the delete button on the post.
   - Shall display successful or error notification after having deleted post.
   - Shall alert the owners about the removal of their posts via emails.

   b) Data Requirements:
   - The deleted post shall be remove from the database.

4. Non - Functional Requirements

1. USABILITY

The important functions of the system are prioritize based on the usage patterns.

Frequently used functions such as complex and critical functions should be tested many times.

Should be easy for beginners to use the basic functions.

System documentation should be provided for users to help them understand and quickly approach the system.

2. LEGAL

The privacy of every user must be committed that will be protected and cannot be used by the company without user's permission.

All other integrating software must be announced in order to avoid copyright issues and must have permission from the other parties.

3. RELIABILITY

Should have long MTBF (mean time between failures).

Data created in the system will be retained for a number of years without being changed by the system.

Error messages for every failure must be clear so that it is easy to understand and to find solutions.

Should have some error detection and correction mechanisms for dealing with software failures.

4. SECURITY

User's privacy information should be encrypted before being stored in the main database server.

Should have a medium level of security to protect access privilege of all type of user.

Policy of privacy should be given out to restrict the third retrieval of customers' information.

Session and its timeout should be re-set each time a user logins.

5. PORTABILITY

Should be able to access from multiple devices (computer, smartphones, tablets, etc).

The statistics about different devices and browsers used to access the website should be documented to improve user's experience.

6. MAINTAINABILITY

User feedback should be taken biannually to help modify some current functions and develop some new functions which meet the world trends.

The database should be backed-up biannually to avoid data corruption or server breakdown.

The system should be designed and implemented in a clear and well-defined manner for further maintenance.

Critical sections of the source code should be noted and commented appropriately for future enhancements.

7. PERFORMANCE

Response time of login function should be adequately fast.

Time for system to update and retrieve data from database must be small.

Loading time must be fast and data of all components must be create correctly from database.

Should be able to handle multiple requests from many different users as the same period of time with smallest delay time.

Should be able to handle multiple modifications from the database without "lost-update" issues.