GoToTeam SRS

1. **Introduction**
    1. **Purpose**
    This SRS will layout a clear understanding of exactly what is required of the software for the developers, designers, and users. The foundations for the implementation of the software will be explained in this document for the developers. The expected features of the software will also be detailed in the SRS so that the users of the application will understand what will be delivered to them.

    2. **Scope**
        - **Product Information**
        The developers will be producing an application to synchronize folders between a group of computers using peer-to-peer networking. This application will be used when a group of users needs to all have access to the same up-to-date files on multiple machines. Users will not have to manually synchronize each other's files by manual sending files to each other when one of them is changed. The application will automatically synchronize the files on the user's computers when there is a change to one of the files. The application will also allow write access to a file for only one user but gives read access to the other users.

        - **Benefits to Using the Software**
            - Users will not have to manually download and transfer files to other users. They will just have to sign into a web page and load the Client; the software will encapsulate the file transfers between Clients.
            - Users do not have to make backups of their files, since other users will have the same copies of those files.
            - The Server running the software will have less strain on it than a Subversion or Concurrent Version System Server because file transfers are done by peer-to-peer file transfer. There will be less Server traffic generated because file transfers will not be executed through the Server.

        - **Objectives of Software**
        Synchronization of software does not involve the user to manually download the file from the new sources. Another objective is to give a user interface that responds within five seconds after the user clicks on a button.

    1. **Definitions, acronyms, abbreviations**
        - RoR – Ruby on Rails: A programming language that is meant for rapid and easy creation of web servers/services.
        - P2P - Peer-to-Peer
        - DB - Database
        - Team - For the purpose of this document, a team is a group of users that want to share some files between them.
        - Client - For the purpose of this document, "client" refers to the application detailed in section 2.2
        - Server - For the purpose of this document, "server" refers to the Ruby on Rails server, MySQL database, and web services detailed in section 2.1

    2. References

    3. **Overview**
    This document is divided into three major sections: Introduction, Overall Description, and Specific Requirements. The introduction details the purpose and scope of the application suite, and contains an overview of this document. Overall Description describes the product perspective, product functions, user characteristics, assumptions and dependencies, and constraints. Specific Requirements details the external interface requirements, functional requirements, performance requirements, design constraints, software system attributes, and domain requirements. The Overall Description and Specific Requirements contain two subsections; one for the Server and one for the Client.

1. **Overall Description**
    1. **Server**
        1. Product perspective
            1. An active Server with a database and web interface. The web interface handles user authentication, team and file listing, team connectivity, and file locks.
        2. Product functions
            1. Login**,** logout **– User Management**

                User must be able to log into the Server with a set user name (e-mail address) and password.

            2. **Check-in, Check-out,** version control **– File Management**

                Users have the ability to check-out a file, which makes the Server enable that file to have read/write access and force the other files to have only read access.

                The check-in ability is used when the user wants to upload his checked-out file to the other users. The Server will inform the other user Clients to download the new file.

                In order to keep track of changes made to files, a version and revision system is implemented. The Server will have a database of the current version number of each file. Every time a check-in is invoked, it will update the version number.

            3. **Concept of Teams – User Distribution**

                A team is basically a set of users that share a certain folder on their computers to synchronize. Each user will have an entry in the Server's DB where the path of the shared folder for that team is located. A user can be part of multiple teams. A user can unsynchronize a folder by leaving that team.

            4. **P2P Connectivity – File Distribution**

                To send files between Clients, the Server will inform each requester (Client requesting the files) which hosts (Clients with the files) has the requested file. Then the requesters will download files from the hosts within 20 minutes per 1MB of data with Clients on a connection capable of downloading at 100kbps.

        3. **User characteristics**
            1. **Very little/no user training required**

                The user simply informs the Server of which folder to synchronize with a team, and the Server and Client will handle the work of updating, permission settings, and versioning of the files. The user also must check-in and check-out a file using the Server's web site.

            2. **Must be able to load a web page**

                Since the ability to check-in and check-out a file will be on a web site, the user must have a browser that supports Javascript and HTML 2.0 or greater.

        4. **Constraints**
            1. **Server must support RoR**

                The Server must have enough resources to run Ruby on Rails, a SQL database, and a web server.
        5. **Assumptions and dependencies**
            1. **Internet connectivity**

                Server is expected to have a 95% uptime on a T1 Internet connection.
            2. **Server computer requirements**

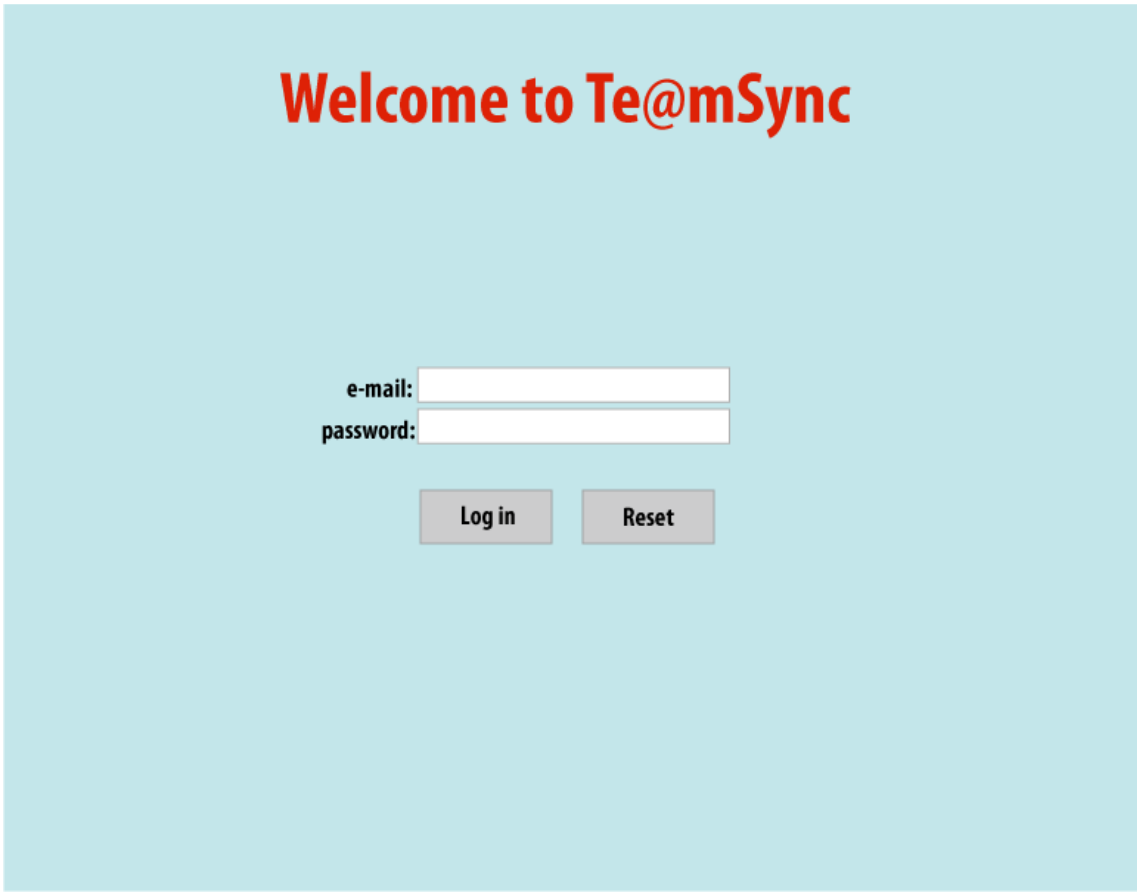
                The computer hosting the website, DB, and web services will be available and accessible except downtime necessary for Server maintenance, hardware upgrades, and software management. The Server is also assumed to have at

least 1GB of memory, a CPU speed of 1.4GHz, and 200MB of free hard disk space for a decent Server response (Clients can communicate with the Server within 30 seconds after establishing a connection). The CPU must also be able to handle at least 200 threads, so at least 100 Clients can use the Server simultaneously. The amount of hard disk space allocated must be sufficient enough to ensure the proper and successful operation of the DB.

2. **Client**
    1. **Product perspective**
       The Client is a system tray application that runs in the background on a Windows machine and maintains an open connection to the Server. It also manages connection between other Clients forming a P2P network.
    2. **Product functions**
       The role of the Client is to respond to commands sent from the Server. Client responsibilities include handling file permissions, maintaining file locality, establishing connections to other Clients, and sending or receiving updated files over these connections. Any errors encountered should be handled gracefully, meaning the Client should remain running while informing the user of the error.
    3. **User characteristics**
       The user need only be able to install a simple executable file and follow setup directions.
    4. **Constraints**
        1. Client users must be running Windows.
        2. File synchronization will not be instantaneous
        3. File synchronization requires a member with the latest version of the file to be online
        4. The Client must be connected in order for a user to check in and out files
    5. **Assumptions and dependencies**
       It is assumed that Windows is the most widely used operating system. Therefore, applications developed for a Windows Platform have the potential to reach the widest audience.

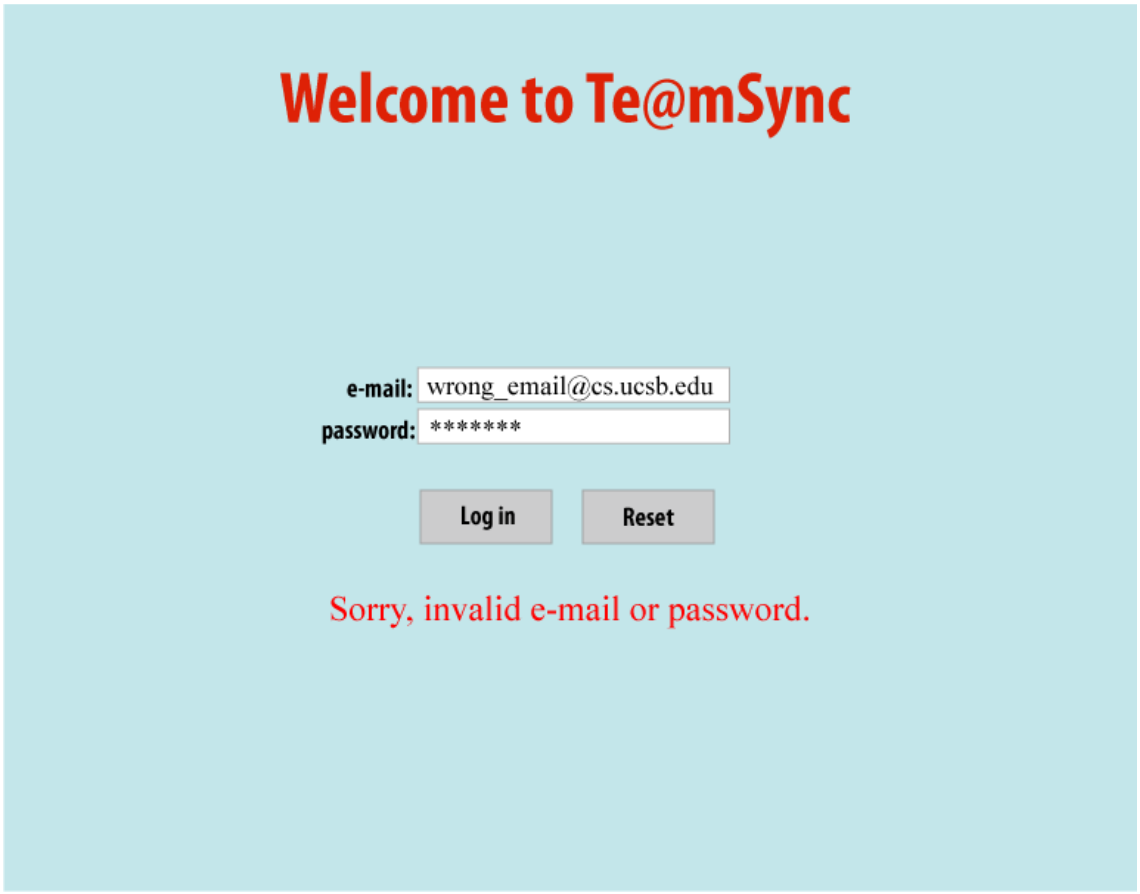
2. Specific Requirements

1. **Server**
    1. **External interface requirements**
        1. **User interfaces**
            1. **Web Based**
                - **Sign-in page**



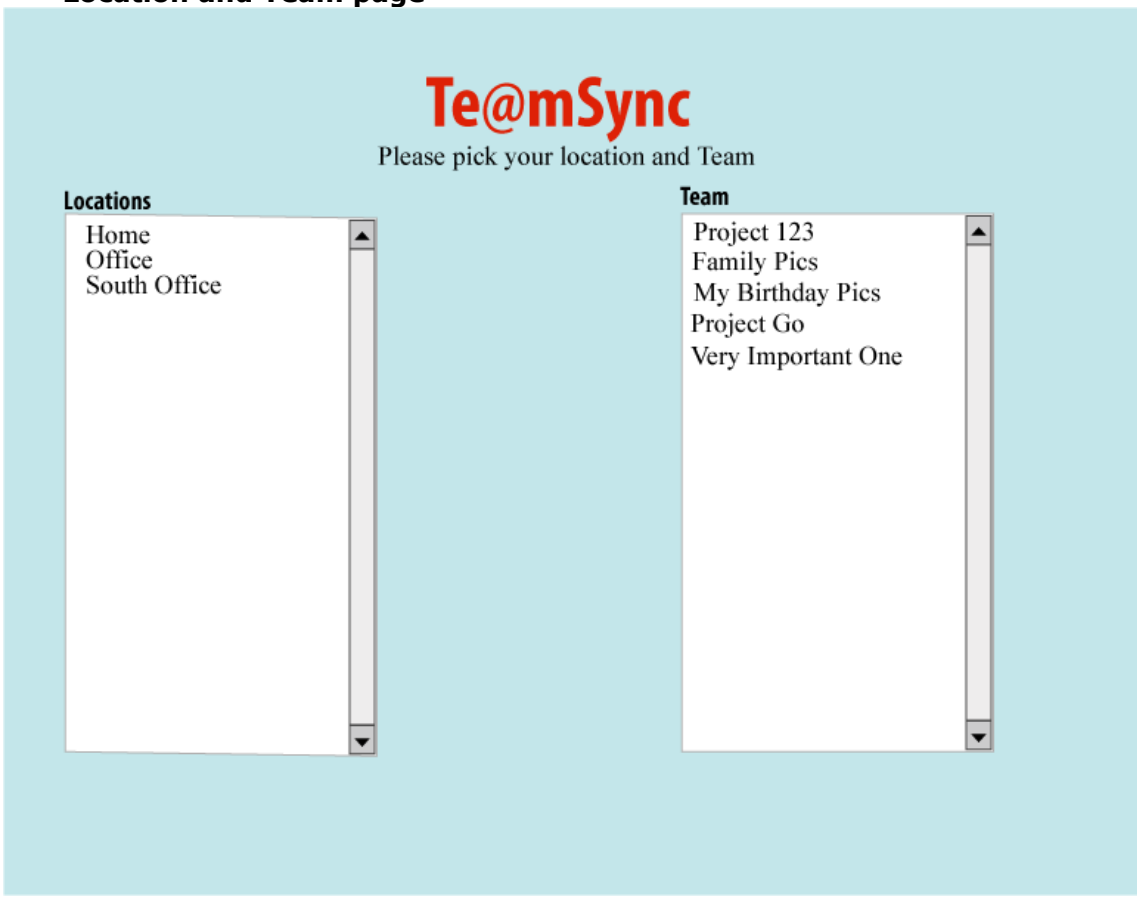
Users will enter the website address of the Server and will be greeted with a log-in page. The page will request the user's e-mail and password. The page will have a "Log in" button and a "Reset" that will erase the text fields

- **Log-in page error**



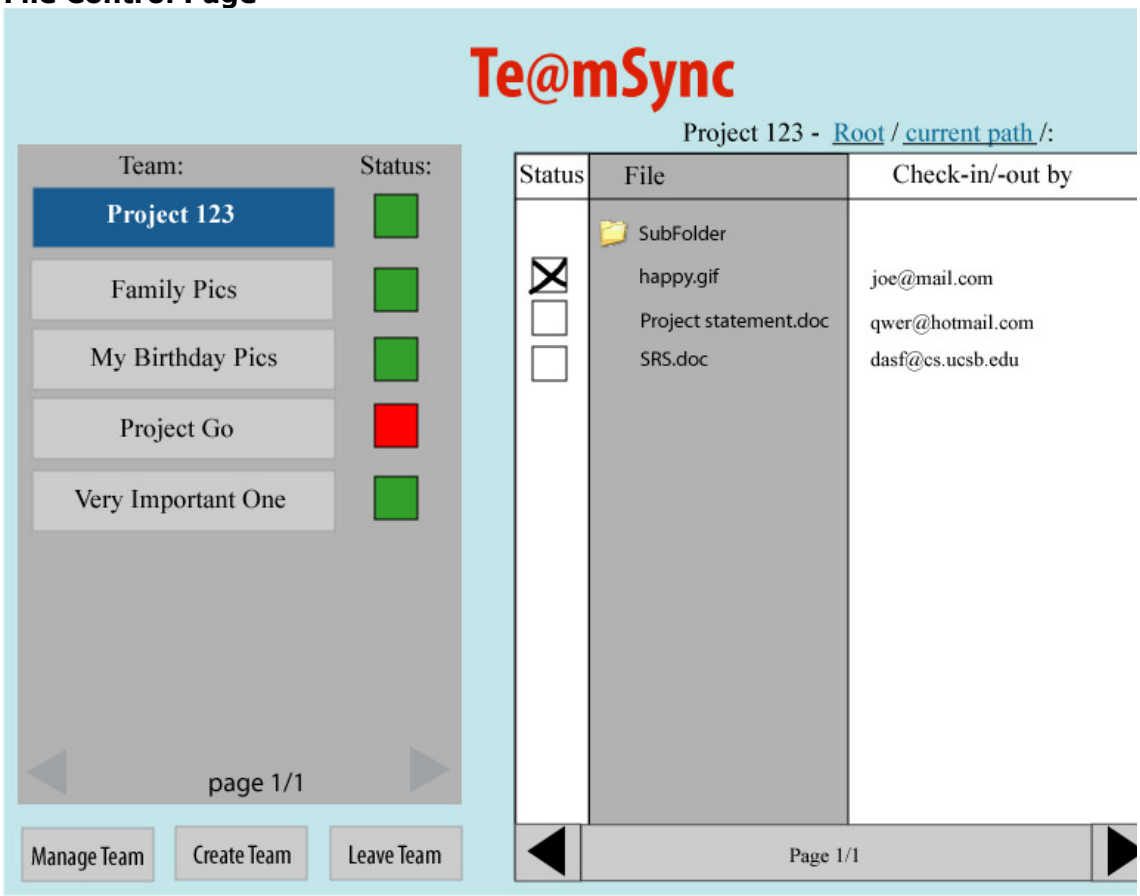
If a user enters incorrect information, the Server will indicate the error with a large red message.

- **Location and Team page**



After logging into the web site, the Location and Team page will be displayed. On the left side is a panel with a list of all locations the user created. On the right, there will be a list of all active teams the user is a part of. After the user selects a location and team, a new page will load where the user can check-in/-out files and select teams.

2. **File Control Page**



The left area is the list of teams that the user is a member of and the synchronization status of the user's local folders. Red indicates the folder is outdated and is in the process of downloading updates, and Green indicates it is up-to-date. The right area is the file browser that provides the user an interface to check-in and check-out. An empty box indicates the file is not locked, a checked box indicates the file is in read-write mode for the current user, and a crossed-out box indicates the file is being revised by another user. The file browser indicates which user has the file checked-in or out. If a file is not currently checked-out, the last user to use it is listed. The user is able to scroll through a list of teams by clicking on the arrow on the bottom, if multiple pages are required to display all the teams. This also applies to the the file browser on the right. The top of the file browser are web links and the user can traverse through the parent directories, if they are shared. The browser is similar to the Windows environment.

- **Creating a Team**
  A user would click on "Create Team" and a web page would load giving the user options to add team members by their e-mail address. Then the user would click on "Finish" to generate the new team. After creation, it will generate a web page that will allow users to sign in and join the team. If the user is already signed in, then it will just add the user to the team. The Server will also generate a web page that tells the Team Leader what link to send to team members.

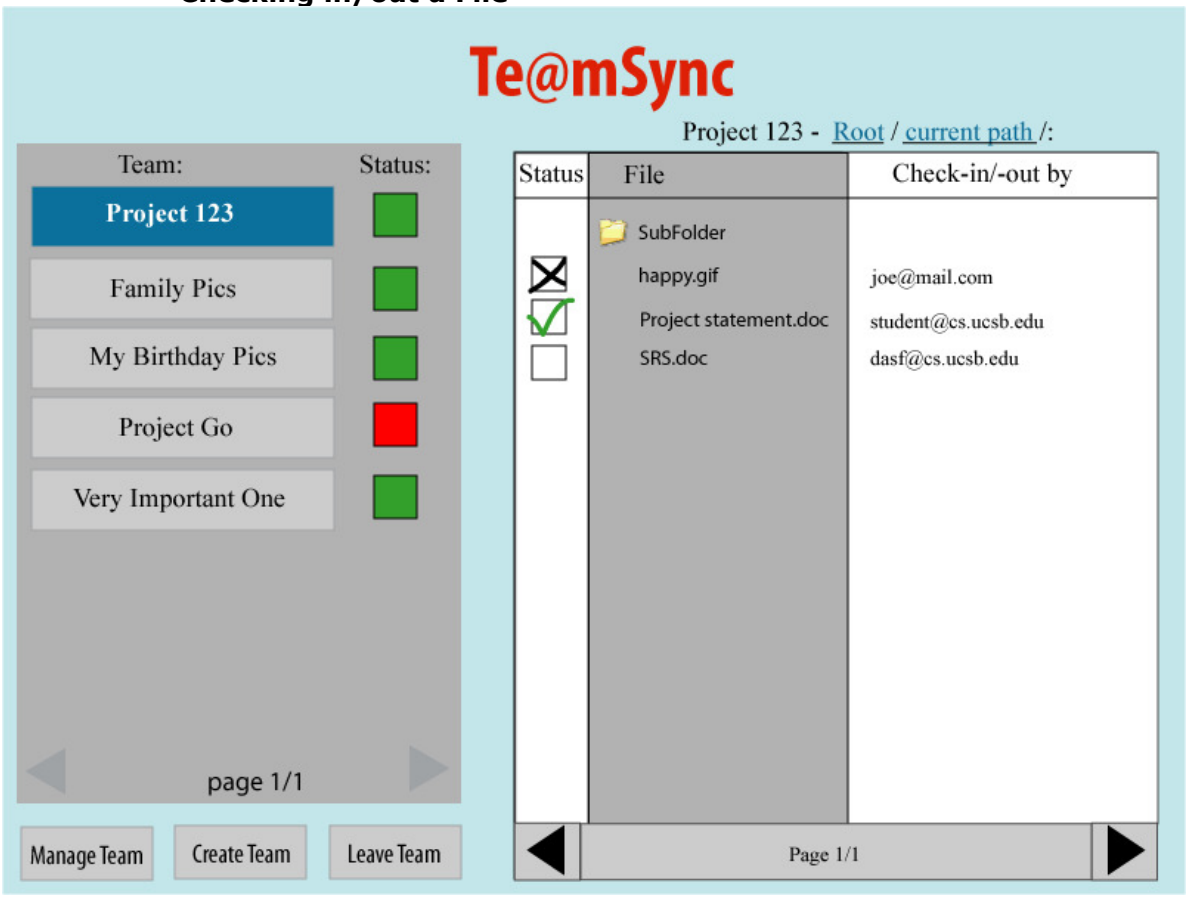
- **Joining a Team**

After a Team Leader sends out the link to the join page, the future team member will click on the link and visit the page. It will prompt users to sign in and join the team. If the user is already signed in, then it will just add the user to the team.

- **Leaving a Team**
  The user selects a Team on the left text box, then clicks on the "Leave Team" button. A pop up will ask if the user, "Are you sure you want to leave Team <insert name>?". Then the user clicks on "OK" or "Cancel".

- **Team Management**
  A team leader would be able to modify members of the team by clicking on "Manage Team". A new page loads and it would have a list of current members where the leader can remove members or add more members by their e-mail addresses. If a user is not a team leader, the user would not have the "Manage Team" button accessible on the control page.

- **Checking in/out a File**



- Once a user selects a team on the left text box and click on the check box next to the file to check-out. After clicking on the empty check box, the image will turn into a checked box. If the user has a file already checked-out, the user can click on the check mark to check-in the file.

2. Hardware Interfaces

A network interface card is required for the server.

3. Software Interfaces

RoR, MySQL, Apache are required for the server.

4. Communication Interfaces

1. Sockets, TCP/IP

The Server will always be actively listening on a certain port for Clients. Once the Client has made a connection to the Server, the Server will create another

socket specifically for the Client. The Server will then be able to establish and maintain an active TCP/IP connection with the Client as long as the Client is online. This connection will serve as the control connection to the Client. It will inform the Clients to change file permissions, that a file is out of date and needs to download it, and where to download the files. Another TCP/IP connection necessary will be for the web server. This connection will also be active to ensure reliability and to enable users to interact with their team folders.

2. Functional requirements

The Server will be interacting with Clients by listening to their actions in the web-based interface. When the Client connects to the web server, the Server will provide a login website where the Client will enter an e-mail address and password. The Server will then verify with the DB to authenticate the Client and then prompt the Client with another website. The website will enable Clients to view all team memberships, specific team directories, local synchronization status of all folders, files that are checked-in and checked-out and by whom, and more options and preferences to manage, create, and leave Teams. The Server will also be interacting with Clients through a seamless connection. This connection will allow the Server to change local file permissions for each Client when files are being checked-in or checked-out, to verify the Client has the most recent up-to-date files, and to communicate the IP addresses of other Clients if files need to be downloaded.

3. Performance requirements

The Server is expected to have a 95% uptime on a T1 connection. The computer hosting the website, DB, and web services will be available and accessible except downtime necessary for server maintenance, hardware upgrades, and software management. The Server is also assumed to have at least 1GB of memory, a CPU speed of 1.4GHz, and 200MB of free hard disk space for a decent Server response (Clients can communicate with the Server within 30 seconds after establishing a connection). The CPU must also be able to handle at least 200 threads, so at least 100 Clients can use the Server simultaneously. The amount of hard disk space allocated must be sufficient enough to ensure the proper and successful operation of the DB. Communication between web services, the database, and sockets will take less than five seconds. Communication using the socket between the Client and Server will take no more than 30 minutes.

4. Design constraints

The Server will be running a RoR Server, web services, MySQL database, TCP/IP connections and protocol. The developers will have to code in such a way that all components of the Server will revolve around the RoR Server. This Server will be communicating with all components built on RoR framework to ensure synchronization and communication within the period of 30 minutes.

5. Software system attributes

1. **Portability**
   The Server will be very portable and scalable because of RoR and it is capable of running on at least one operating system.
2. **Security**
   The web services will be using SSL to provide an additional layer of protection when communicating with the RoR Server. Passwords will also be using a MD5 hash to encrypt passwords. Traffic between Server and Client through the socket will be tunneled through SSL.
3. **Reliability**
   The Server will run for at least one day.

6. **Domain requirements**

Users must use their e-mail addresses and passwords at least a length of eight and must contain numbers and characters. No more than 100 simultaneous socket

connections. Users will not be able to interfere with team directories they do not have access to. Users can have at most 10 team memberships. Users will be able to only view one directory at a time. One user will be restricted to at most three simultaneous log-ins.

2. **Client**
    1. **External interface requirements**
        1. **User interfaces**
        The Client application only requires the User to input a username, password, and maximum number of concurrent external connections (minimum 2: Server and another Client). The User is only required to enter this information once, during installation. Information can be changed through a pop-up window initiated by the user from the system tray.
        2. **Software interfaces**
        The Client will interface with the webserver though an open socket and TCP/IP protocol. Likewise when sending files between Clients it will interface between the Clients using sockets.
        3. **Communication Interfaces**
        Sockets, TCP/IP

    2. Functional requirements
    The Client will start upon Windows startup and immediately try to establish a connection to the Server. In the case of a connection error the program will give an error message and state that the application cannot run without a valid connection. This is how the Server will keep track of who is online and who is offline. The Server will then issue commands to the Client. First in will get a list of the files that the Client has in it's shared folder and check them against the list that the Server owns. If there is a new file on the Client that the Server doesn't know about, it will add it to the list and send out a message to all other users to notify them that they need the file. It will then check the files that the Server and the Client have in common and see if the Client has the most up to date files. If the Client does not have the latest version of the file, the Server will notify it of the IP and port of a user who has the most up to date file, and the Client will connect to that other user and retrieve that up to date file. If no other users have the most up to date file, that user will not be able to checkout the file for modification.

    Once the Client has made the connection with the Server and established that it has the most up to date files, the user will be able to "check out" the file via the web interface. When this happens, the Server will notify all of the other Clients and then the user that wishes to check out the file will change the file permissions to read/write in order to modify them. When the user is done with the file and checks it back in via the web interface, the Client program will change the file back to read access.

    3. Performance requirements
    Since the Client application will be running in the background, when it is idle or communicating with the Server it should never take more than 5% of the CPU load and use less than 20 Kb of memory. The Client will only handle a set number of connections to other Clients within the range of 1-10 set by the user. This does not include the connection that is always on to the Server.

    4. Design constraints
    The Client application will be written in C++. It will initially only be for the Windows platform. We will be using the winsock2 to open sockets and TCP/IP to communicate between other Clients and the Server.

    5. Software system attributes
        1. Reliability
        Potential errors include:
        Broken connection to Server - the Client would inform the User of the connection loss, close the socket, kill the thread, and attempt to reconnect every 5 minutes until a new connection can be established.

Broken connection to another Client - the Client would inform the User of the connection loss, close the socket, kill the thread, and inform the Server of the error. To avoid incomplete files due to connection loss, the Client will save a temporary until the entire file transfer is completed and verified.
File System error - the Client would inform the User of the error encountered, suggests a possible fix, then awaits user confirmation before continuing.

2. **Availability**
   By default the Client will start on Windows startup. This functionally can be turned off, and the Client can always be started manually through a shortcut on the Desktop. The User initiates interaction with the Client by double clicking the Desktop icon or right clicking on the System Tray icon. Errors will be displayed via pop up windows from the System Tray.
3. **Security**
   All sent passwords will be encrypted Client-side with an MD5 hash. The Client will keep a limited 1 Mb log of all of the file transfers that take place between itself and other Clients.
4. **Portability**
   The code is broken into two key components: thread synchronization and sockets. The thread portion of the code is written entirely for the windows platform. Windows requires a method call to start and close sockets, while all other portions of the sockets are portable

3. **Appendices**