CENG491

DETAILED DESIGN REPORT

MISS PROJECT

SoftRunner

Zerrin Bozel 1297571
Burcu Ardıç 1297472
Pınar Çelikoğlu 1297613

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Goals and Objectives

We aim to build an attractive online multi-user Project Management Tool *MissProject* which is easy to use for every kind of users from different backgrounds and important of all, a totally reliable system in which the companies can trust.

With *MissProject*, the quality in project management will increase, and the time, cost and resource management will be optimum. Our goal is creating a satisfaction chain which includes every-one who are somehow related with planning, students, managers, teachers, even housewives, etc...

We claim that this system will increase the number of people using this kind of tools, because as well as providing a practical way of managing projects, our system contains the automatic e-mails that allow the user to keep track of the project without signing in. This way, a registered user can view the project without loosing time.

Finally, we are aiming this project to be used by several people from different backgrounds, not just by the companies. So our GUI will be user-friendly.

1.2 Statement of Scope

- *MissProject* is a powerful, flexible, online, multi-user and multi-project project management tool that can be used to control simple or complex projects.
- After the company buys and installs the system, an account is created for the Configuration Manager of the company. Configuration Manager is responsible for building up the company database; adding/removing users, workgroups, projects and resources to the company database. Adding a project includes assigning a Project Manager to that project. External files can be imported to the system, providing an ease of use for data entry to the database.
- During the project planning phase, it's important for the Project Manager to define the objectives, assumptions, and constraints of the project clearly. After initial planning, the Project Manager can start working with *MissProject.*
- For each project, there is one and only one Project Manager. But, the workload can be too heavy for the Project Manager to make all the edits on the project. So, we decided to

create a role called “Administrator” to help the Project Manager. The difference between Administrator and Project Manager is that Administrator gets the right to edit from the Project Manager. Besides, Project Manager can remove the rights of Administrator (removes the Administrator from the database, so that the user will be a Standard User) in case of a conflict or some sort of managerial issue. The reason of these two different roles is to provide security. (By the way, Standard Users are also added –related- to project by Project Manager and Administrator.)

- Another security related issue is the concurrency problem. In order to preserve the consistency, *MissProject* doesn’t allow two actors to edit the same project at the same time. We achieved this by using 2-state locking mechanism on the database. When someone with edit right (Project Manager or one of the Administrators) requests to update the project, it is controlled whether the project is locked or not. If it is not locked, the request is granted. And until the editing agent leaves the project, the project will be locked. If another update request comes when the project is locked, the request is not granted. Project can be accessed Read-Only. After the editing agent who has the lock saves and quits, the Read-Only views are refreshed after a pop-up message. With efficiency concerns, the projects is not locked when a user with edit right enters a project. The lock mechanism is active only when the user declares that he/she wanted to make an update.

- The project can be divided into subtasks and different people or different resources can be assigned to those subtasks. So, task and resource management appears.

- The task durations and start&finish dates should be entered. This allows the Gantt Chart to form graphically. The tasks are related to each other when necessary, and are assigned to specific dates. Task dependencies can be created between tasks in different projects. Creating dependencies between projects models the interrelationships between different projects.

- The material resources are assigned to each task. Sharing resources is useful for managing resources across multiple projects in which the same material resources will be used. The resource information that has been entered into the system by the Configuration Manager can be assigned to the specific tasks, which have been set up as the work of the project by the Project Manager or Administrators. The most effective way of managing resources on a project is to balance their workloads and track progress on tasks. To achieve this goal,

overallocations or underallocations should be prevented. MissProject provides Project Manager enhanced statistics features

- Since the people are assigned to tasks and start and finish dates are determined, a graphical individual schedule can be created, just like Gantt Chart. Similarly, resource allocation chart can be viewed. These two charts are critical for both individuals and Project Managers.

- At this point in the project planning process, the project scope has been figured out; the task list and estimated task durations have been determined, all resources have been identified and assigned. This information can be used to make cost estimations and identify requirements of the future projects. After completion of the project, *MissProject* will provide companies to benefit from historical data.

- After the project has been scheduled, a notification is sent to the registered project members by e-mail. If the Project Manager or Administrators have changed tasks, resources, or assignments, the most current project information is distributed to the project members via e-mail. As the project progresses, its current status can be reported as notification mails to all of the registered team members.

- Project Manager has the option of arranging and announcing meetings. The best suitable time and place can be determined by using statistics functions. Rooms are handled just like material resources, so that their status can be viewed with resource graph. As soon as the meeting is scheduled time, place and subject is sent to declared group with a meeting notification mail. Meeting agenda and/or meeting deliverables are distributed to all group/project members.

- We are doing our best to make a successful design. Our main considerations are security, correctness and ease of use. They are explained in the Success Criteria section.

1.3 Success Criteria of the Project

Among the success criteria of *MissProject*, security is the first to be mentioned since the information about the projects should not be accessed by rival companies. One of the critical points in online project management is that: The roles and permissions of the user should be identified distinctly for the consistency of the project. The system should employ a well-tested authentication and authorization mechanism.

When all security issues are handled, the response time, ease of use and the attractive interface will move us one step further. To be able to be used by every kind of people from different backgrounds, the system should be clear and intuitive to use. It will not be necessary for the users to have any additional knowledge or training to use *MissProject*. But in case of any problem, we will have a strong *help* facility. Most web-based applications provide users a complicated system, which repel people who have already doubts about internet&security. Our system will consist of very systematic steps, each of them are clear and easy to use, erasing all the doubts in people's minds.

1.4 Design Constraints, Limitations

1.4.1 Design Constraints

System: *MissProject* should provide a secure environment for managing courses. Multiple users should be able to use the system simultaneously; the response time should be reasonably short.

Interface: The end users will use ordinary browsers to login and use the system, therefore the human-system interface should be supported by as many browsers as possible, even the text browsers. Needless to say that the most important constraint about the interface is that it must be user friendly.

1.4.2 Limitations

Time: The time seems to be an important constraint of our project since it must be completed in 9 months and all of the group members have different courses. It is so important to track the project schedule properly.

Employee Skills: The programming skills of the employees are other restrictions. In spite of the fact that it is not so effective as time, it certainly could limit us from doing some additions to the project during its lifetime.

Hardware: The PC's to be used during the implementation of *Miss Project* will have these features minimum: 733MHz Pentium CPU, 128 MB RAM, 32 MB Graphics Card and 15` monitors.

Portability: The software will be implemented under Windows XP and can be executed in Widows 98 and later versions.

Programming Language: We have decided to use .NET as our development platform. During Analysis phase, we were searching about C# and Java, but we could not decide with which one of them we should implement our system. In order to be able to use .NET technologies, we concluded that C# is better. Some of the reasons of our decision for using C# .NET are as follows:

- Staff's being more experienced in C# rather than Java
- The power and flexibility offered by the advanced features and functionality of ASP.NET is much better then anything on the Java side of the world.
- C# took many of the ideas and concepts of Java and made them better.
- Provides a measure of security
- Doesn't require special permission (e.g. unlike CGI)
- Has better interactivity
- Has SQL database access
- Is easier to use than C++, and above all: it is WEB-oriented, and provides an ever growing wealth of class libraries for ever new application domains.

1.4.3 Scheduling Constraint

GANTT_CHART.gif (Gantt chart is given as appendix in hardcopy.)

2. DATA DESIGN

2.1 Data Description

The information of Individuals, Groups, Tasks, Material Resources, Projects, roles and Meeting and their relations are kept and updated when necessary in database.

2.2 Data Objects and Complete Data Model

ER_Diagram.vsd (Revised ER diagrams are given as appendix in hardcopy.)

2.3 Data Dictionary

1. Individual Table

- The information of the individuals can be changed through Update Personal Information function.
- Attributes of the Individual table are as follows;

iid	: Integer
username	: String[10]
password	: String[10]
name	: String[40]
Mail_adress	: String[30]

UNIQUE(username)
PRIMARY KEY: iid

2. Group Table

- Attributes of the Group table are as follows;

gid	: Integer
name	: String[10]
status	: Integer
proffesion	: String[20]
team_leader	: Integer

PRIMARY KEY: gid
FOREIGN KEY: team_leader REFERENCES Individuals(iid)

3. Member Table

- member table contains the relation between the individuals and the groups.
- Attributes of the member table are as follows;

gid : Integer
iid :Integer
PRIMARY KEY: (gid, iid)
FOREIGN KEY: gid REFERENCES Group(gid)
FOREIGN KEY: iid REFERENCES Individuals(iid)

4. Task Table

- The information of the tasks can be changed through update process.
- Attributes of the Task table are as follows;

tid : Integer
name : String[30]
project_id : Integer
dto : Date[DD/MM/YYYY]
dfrom : Date[DD/MM/YYYY]
duration : String[10]
priority : Integer
prerequisite_id : Integer
UNIQUE(name, project_id)
PRIMARY KEY: gid
FOREIGN KEY: project_id REFERENCES Project(pid)
FOREIGN KEY: prerequisite_id REFERENCES Task(tid)

5. Assignedto Table

- Assignedto table shows the relation between the tasks and the groups, which groups are assigned to which task.
- Attributes of the Assignedto table are as follows;

gid : Integer

tid : Integer

PRIMARY KEY: (gid, tid)

FOREIGN KEY: gid REFERENCES Group(gid)

FOREIGN KEY: tid REFERENCES Task(tid)

6. Material_resources Table

- Parts for the Material_resources table are as follows;

mid : Integer

description : String[30]

type : String[20]

serial_number : String[20]

PRIMARY KEY: (mid)

7. Reserve Table

- Reserve Relation shows the relation between the Tasks and the Material Resources.
- Parts for the Reserve Relation are as follows;

tid : Integer

mid : Integer

dfrom : DateTime[DD/MM/YYYY HH:MM]

dto : DateTime[DD/MM/YYYY HH:MM]

PRIMARY KEY: (mid, tid)

FOREIGN KEY: mid REFERENCES MaterialResources(mid)

FOREIGN KEY: tid REFERENCES Task(tid)

8. Project Table

- Attributes of the Project Table are as follows;

pid : Integer

name : String[20]

dto : Date[DD/MM/YYYY]

dfrom : Date[DD/MM/YYYY]

duration : String[10]

project_manager_id : Integer

PRIMARY KEY: (pid)

FOREIGN KEY: project_manager_id REFERENCES Individuals(iid)

9. Role Table

- Attributes of the Role table are as follows;

individual_id : Integer

project_id : Integer

role : String[25]

PRIMARY KEY: (individual_id, project_id)

FOREIGN KEY: individual_id REFERENCES Individuals(iid)

FOREIGN KEY: project_id REFERENCES Project(pid)

10. Meeting Table

- We have to obtain a Meeting Table in order to arrange and record meetings. There is no need for a room because, rooms are in the Material_resources table. So, place references Material_resources table. The status of the room at the meeting time will be checked with an assertion.

meet_id : Integer

room_id : Integer

subject : String[25]

group_id : Integer

dfrom : DateTime[DD/MM/YYYY HH:MM]

dto : DateTime[DD/MM/YYYY HH:MM]

PRIMARY KEY: (meet_id)

FOREIGN KEY: room_id REFERENCES Material_resources(mid)

FOREIGN KEY: group_id REFERENCES Group(gid)

11. Company Table

- Parts for the Company information are as follows;

name : String[20]

Configuration_manager_id : Integer

PRIMARY KEY: (name)

FOREIGN KEY: Configuration_manager_id REFERENCES Individuals(iid)

2.4 XML

We will benefit from XML especially in Import&Export functions. We are using data oriented XML to read XML file data into the database and write out database data into the XML document.

```
<Company>
    <Company Name></Company Name>
    <Configuration Manager></Configuration Manager>
    <Individual>
        <iid></iid>
        <username> </username>
        <password> </password>
        <name> </name>
        <mail_adress> </mail_adress>
    </Individual>
    <Group>
        <gid></gid>
        <name></name>
        <status></status>
        <proffesion></proffesion>
        <team_leader></team_leader>
    </Group>
    <member>
        <gid>< gid>
        <iid></iid>
```

```
</member>
<Task>
    <tid></tid>
    <name></name>
    <project_id></project_id>
    <dto></dto>
    <dfrom></dfrom>
    <duration></duration>
    <priority></priority>
    <prerequisite_id></prerequisite_id>
</Task>
<Assignedto>
    <gid></gid>
    <tid></tid>
</Assignedto>
<Material_resources>
    <mid></mid>
    <description></description>
    <type></type>
    <serial_number></serial_number>
</Material_resources>
<Reserve>
    <tid></tid>
    <mid></mid>
    <dfrom></dfrom>
    <dto></dto>
</Reserve>
<Project>
    <pid></pid>
    <name></name>
    <dto></dto>
    <dfrom></dfrom>
    <duration></duration>
    <project_manager_id></project_manager_id>
</Project>
```

```
    <Role>
        <individual_id></individual_id>
        <project_id></project_id>
        <role></role>
    </Role>

    <Meeting>

        <meet_id></meet_id>
        <room_id></room_id>
        <subject></subject>
        <group_id></group_id>
        <dfrom></dfrom>
        <dto></dto>
    </Meeting>
</Company>
```

3. ARCHITECHTURAL DESIGN

3.1 Review of Data Flow Diagrams

DFD.vsd (Revised Data Flow Diagrams are given as appendix in hardcopy.)

3.2 Static View of Design

We examined the design patterns in C# and decided to use the Abstract Factory method. Below is the class diagram of *Miss Project*. The classes and the methods to be implemented are shown in this graph.

ClassDiagram.vsd (Class Diagram is given as appendix in hardcopy.)

3.2.1 Classes in Class Diagram

3.2.1.1 Authentication Class

Definition:

This class is used to accomplish user and login operations, and detect the roles of the user in the registered projects .

Responsibilities:

- Login and Sending a password.
- Creating a new password if the password is forgotten.

Operations:

- DetectRole(Individual person, Project project): string;
  Detects the roles of the user in each of the registered projects
- Login(string username, string password): bool
  Checks if the username and password are valid.

3.2.1.2 Individual Class

Definition:

The instances of this class contain the information about all users.

Responsibilities:

- Creating an object with the inputs and commands taken from the user during signing up, adding user/Administrator, Update User Information processes.

- Inserting/Removing this object into/from the database and Update the information of this object in the database.

Operations:

- AddUser(Individual person): void; Arguments that are taken from the Configuration or Project Managers.

  Inserts the information take from the Configuration or Project Managers into the database and creates an account.
- RemoveUser(Individual person): void; Arguments that are taken from the Configuration or Project Managers.

  Removes the object that is created from the database.
- UpdateUserInfo(Individual person, Individual newPerson): void;

  Creates a new object from the user's commands and updates the information of the individual in the database according to the new information.
- Individual(int indId, string username, string password, string name, string email):

  Creates an Individual object from the commands of the user.

3.2.1.3 Project Class

Definition:

The instances of this class contain the information about projects.

Responsibilities:

- Creating an object by the input and commands taken from the Configuration Manager during adding a project.

Operations:

- AddProject(Project project): void; Arguments that are taken from the Configuration Manager.

  Adds a project into the Project Table in the database according to the commands of the Configuration Manager.
- RemoveProject(Project project): void; Arguments that are taken from the Configuration Manager.

  Removes a project from the database according to the commands of the Configuration Manager.
- AddAdmin(Project project, Individual admin): void

  Adds an Administrator to a specific project according to the commands of the Project Manager.

- RemoveAdmin(Project project, Individual admin): void

  Removes an Administrator from a specific project according to the commands of the Project Manager.
- AddStUser(Project project, Individual admin) : void

  Adds a Standard User to a specific project according to the commands of the Project Manager.
- RemoveStUser(Project project, Individual admin) : void

  Removes a Standard User from a specific project according to the commands of the Project Manager.
- LockProject(Project project, Individual user) : void

  If the user is a Project Manager or Administrator of the specified project, this function locks the project and the project becomes read only for other users.
- CostEstimate(Project project) : double

  Estimates the cost of a specific project.
- Project(int pid, string name, string fromDate, string toDate, string duration, Individual manager)

  Creates a Project object from the commands of the user.

3.2.1.4 Task Class

Definition:

The instances of this class contain the information about tasks

Responsibilities:

- Creating an object with the input and commands taken from the Project Manager or Administrator and doing task operations.

Operations:

- AddTask(Task task, Project project): void

  Creates a Task object from the arguments that are taken from the Project Manager or Administrator and assign this task to a specific project.
- RemoveTask(Task task, Project project): void

  Creates a Task object from the arguments that are taken from the Project Manager or Administrator and remove this task from a specific project.
- AssignPriority(Task task, int priority): void

  Creates a Task object from the arguments that are taken from the Project Manager or Administrator and assign a specific priority to this task.

- Task(int taskId, string name, int projectId, string toDate, string fromDate, string duration, int priority, int prerequisiteId)
  Creates a Task object from the commands of the Project Manager or Administrator.

3.2.1.5 Resource Class

Definition:

The instances of this class contain the information about resources of the company and projects.

Responsibilities:

- Creating an object by the input and commands taken from the Configuration Manager during adding resources or updating resources.

Operations:

- AddResource(Resource resource): void
  Creates a Resource object from the arguments that are taken from the Configuration Manager and inserts this object into database, Resource Table.
- RemoveResource(Resource resource): void
  Creates a Resource object from the arguments that are taken from the Configuration Manager and removes this object from the database.
- AllocateResource(Resource resource, Task task): void
  Creates a Resource object from the arguments that are taken from the Administrator or Project Manager, assigns this resource to a specific task in the project, and inserts this relation into the database.
- ChangeStatus(Resource resource, int status): void
  Creates a Resource object from the arguments that are taken from the Administrator or Project Manager, changes the status of the resource according to the user's command(Free/Reserved/In use) and updates the database.
- Resource(int ResourceId, string description, string type, string serialNumber, int status)
  Creates a Resource object from the commands of the user.

3.2.1.6 Group Class

Definition:

The instances of this class contain the information about workgroups of the projects.

Responsibilities:

- Creating an object by the input and commands taken from the Project Manager and Admin during adding groups and assigning groups to tasks.

Operations:

- AddUserToGroup(Individiual ind, Group group): void
  Creates an Individual object from the arguments that are taken from the Project Manager and Administrator, and adds this individual to a workgroup. Also the database is updated.
- AssignGroup(Task task, Group group): void
  Assigns a group to a task by the commands that are taken from the Project Manager and Administrator, and updates the database.
- Group(int groupId, string name, int Status, string profession, int teamLeader)
  Creates a Group object from the commands of the Project Manager and Administrator.

3.2.1.7 Meeting Class

Definition:

The instances of this class contain the information about the meetings of the company.

Responsibilities:

- Creating an object by the input and commands taken from the Project Manager during arranging meetings and arrange the meeting in the most convenient time and place.

Operations:

- ArrangeRoomDate(int RoomId, string from, string to): int
  Arranges the optimum meeting according to the given date and duration, and inserts the meeting into the database. The id of the meeting is returned.
- Meeting(int meetId, int roomId, string from, string to, int groupId, string subject)
  Creates the meeting object.

3.2.1.8 Statistics Class

Definition:

The instances of this class contain the information about statistics of the company and projects.

Responsibilities:

- Creating the necessary graphs and reports according to the commands of the user..

Operations:

- DrawGanttChart(Project project):void.
  Takes the information that the user enters and generates the Gantt chart.
- DrawResourceGraph(Project project):void
  Takes the information that the user enters and generates the ResourceGraph.
- GenerateReport(….):void; Arguments taken from the user.
  Takes the information that the user enters and generates a report.
- CustomFilter(…):void; Arguments taken from the user for filtering.
  Takes the information that the user enters makes the project to be viewed according to the user's request.

3.2.1.9 XMLFile Class

Definition:

The instances of this class contain the information about the imported and exported files or projects.

Responsibilities:

- Creating an object by the commands taken from the Project Manager or Administrator during importing or exporting.

Operations:

- XMLGenerate(Project project): XMLFile
  Returns an object that has the information of the exported subproject.
- XMLParse(XMLFile file): void
  Imports a subproject by parsing the XMLFile object, and updates the database.
- XMLFile(…): Arguments that are taken from the project according to the commands of the users.
  Creates an XMLFile object from the commands of the user.

3.2.1.10 IDBConnector Interface

Definition:

The methods of interface is used to connect to any type of database and make transactions according to the users' commands. In *MissProject* SQL will be used, so the SqlDBConnector implemented inherits these methods.

Responsibilities:

- Opening a connection to the database.
- Beginning, committing and rolling back transactions.
- Executing the queries.
- Closing the connection to the database.

Operations:

- Open(): void

  Opens the connection
- Close(): void

  Closes the connection
- BeginTransaction() : void

  Starts the transaction.
- CommitTransaction(): void

  This operation is used to commit transactions.
- RollbackTransaction(): void

  This operation is used to rollback transactions.
- ExecuteDataSet(string commandText): DataSet
- ExecuteNonQuery(string commandText): void
- ExecuteReader(string commandText): IDataReader
- ExecuteScalar(string commandText): object

These operations are used to execute the queries.

3.2.1.11 SqlDBConnector Class

Definition:

This class inherits the methods in the IDBConnector interface and is used to connect to the SQL Database and make transactions according to the users' commands.

Responsibilities:

- Opening a connection to the database.
- Beginning and committing and rolling back transactions.
- Executing the queries.
- Closing the connection to the database.

Operations:

- Open(): void

  Opens the connection

- Close(): void

  Closes the connection
- BeginTransaction() : void

  Starts the transaction.
- CommitTransaction(): void

  This operation is used to commit transactions.
- RollbackTransaction(): void

  This operation is used to rollback transactions.
- ExecuteDataSet(string commandText): DataSet
- ExecuteNonQuery(string commandText): void
- ExecuteReader(string commandText): IDataReader
- ExecuteScalar(string commandText): object

  These operations are used to execute the queries.
- SqlDBConnector(string SqlServerName, string InitialCatalog, string UserID, string Password)

  This is the constructor of the class, and has the parameters to construct a connection string.

3.3. Use Cases

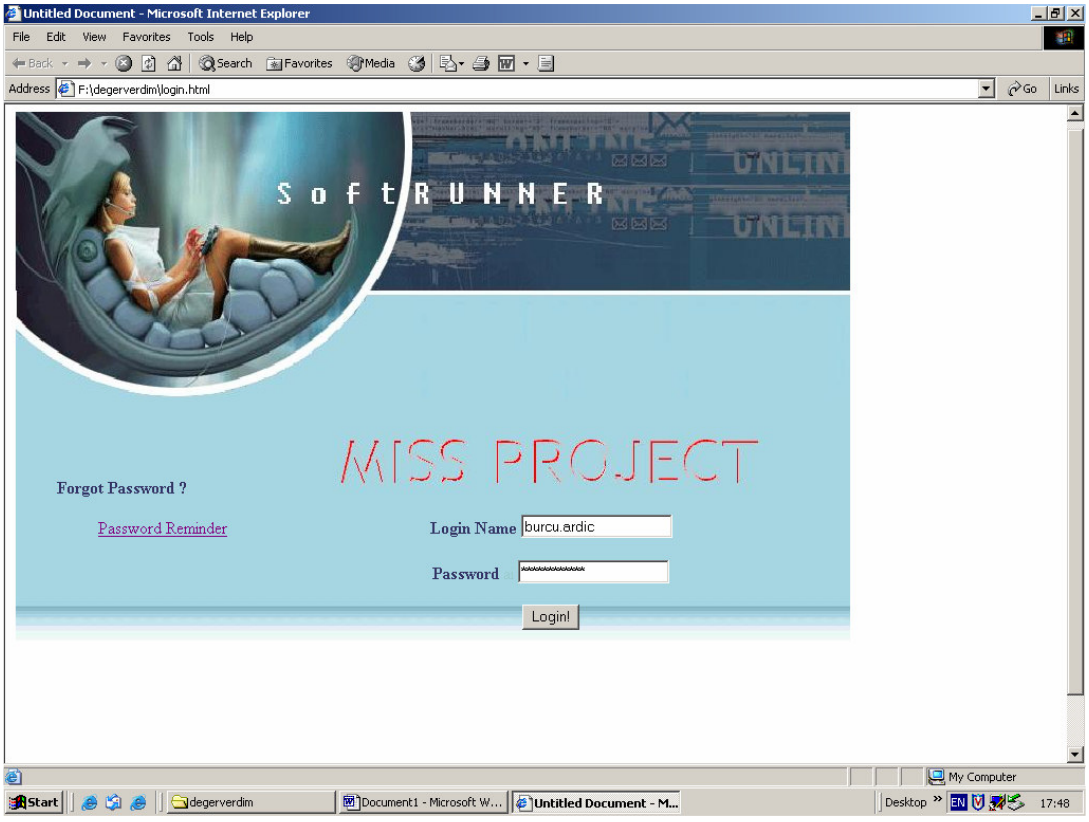
UseCase.gif (Use Case Diagrams are given as appendix in the hard copy.)

4. INTERFACE DESIGN

4.1. Description of the User Interface

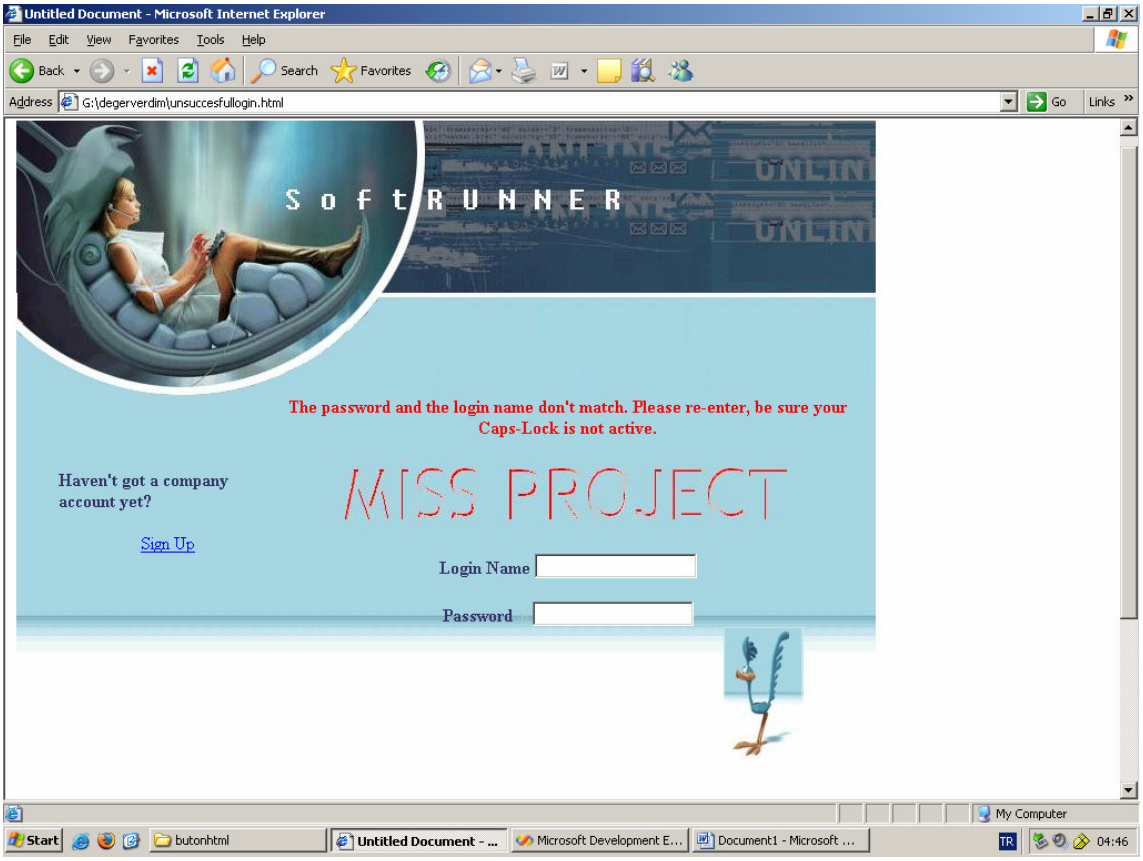
4.1.1. Login Interface:

To ensure the security, the system is required to make a password authorization. The user enters his/her login name (which is unique) and the password.



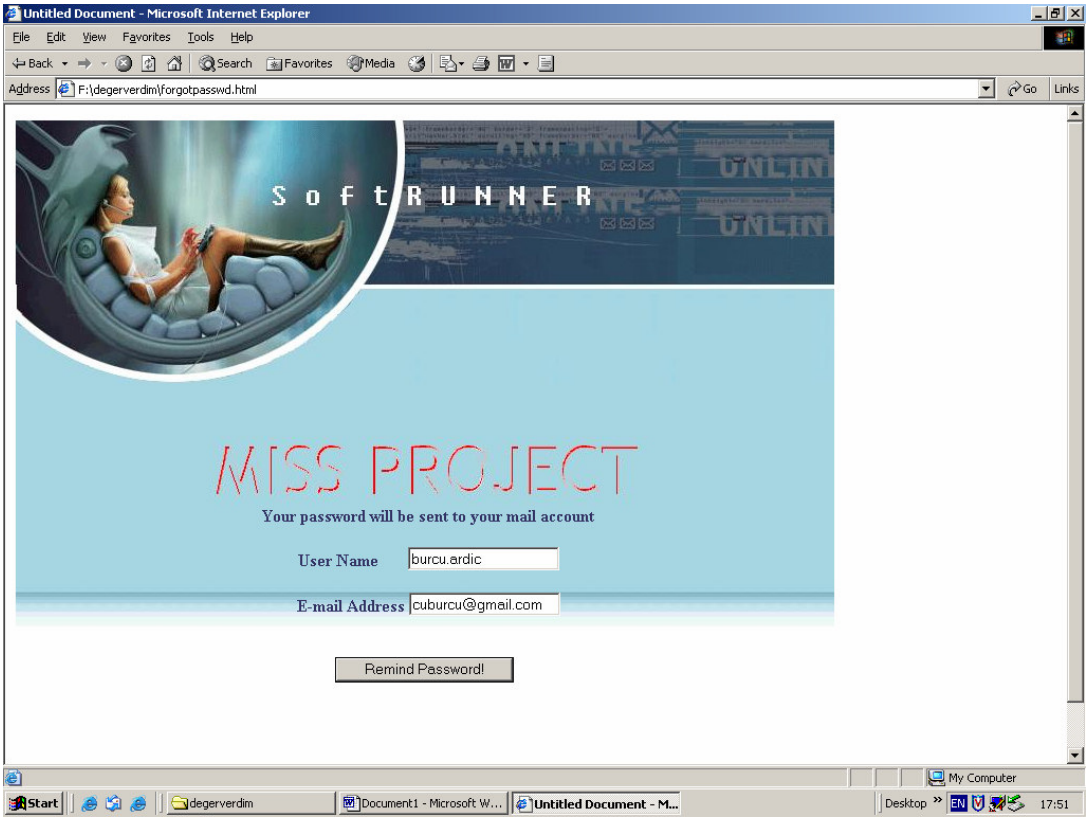
4.1.2. Unsuccessful Login Interface:

This page is displayed when the user enters his/her username/ password, or if he/she is not registered.



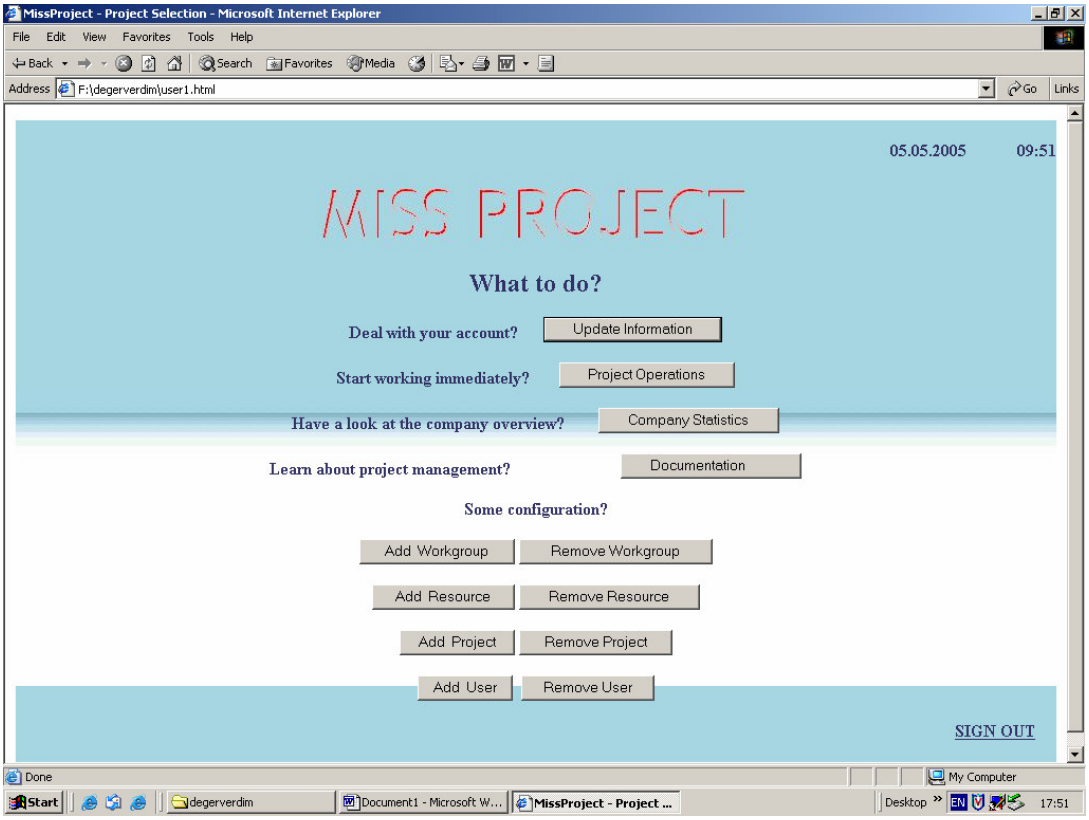
4.1.3. Forgot Password Interface:

There is no need to panic if you have forgot your password.



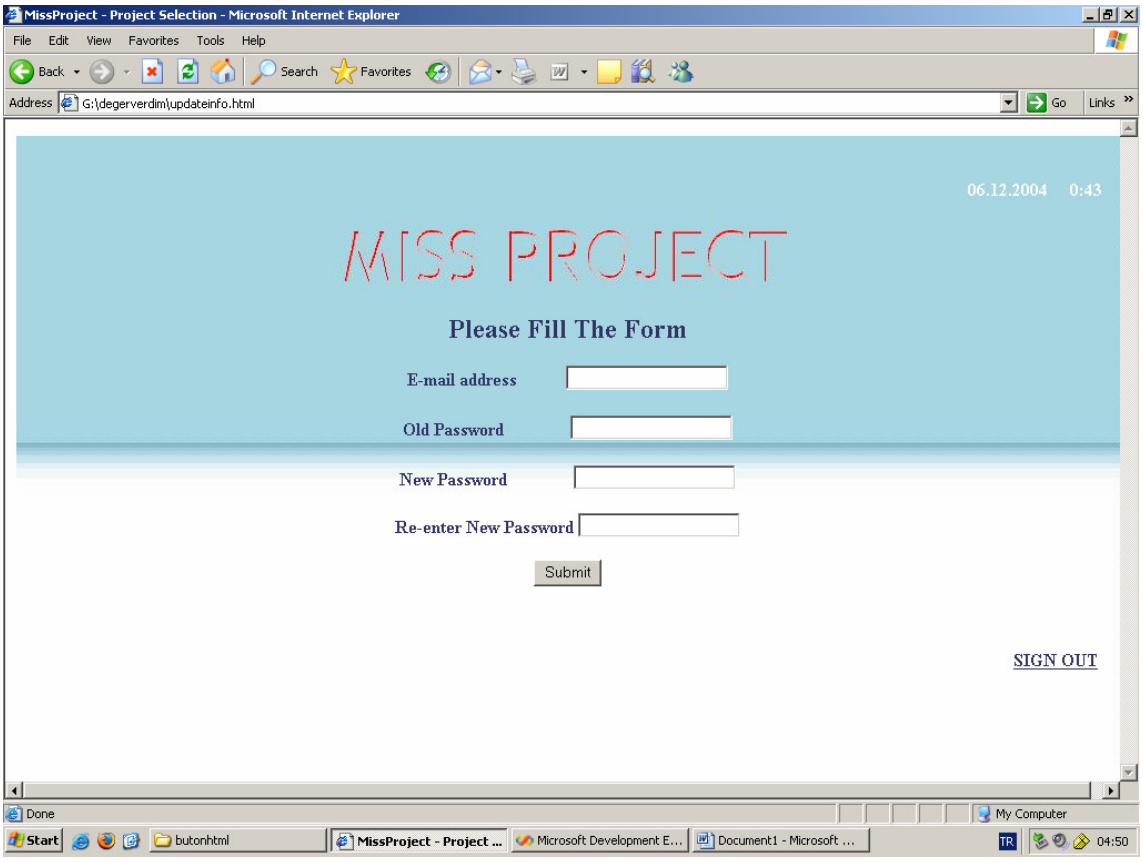
4.1.4. Operation Selection Interface:

After successful login, the user is directed to the Operation selection page of his/her company. He/she may update personal information by clicking the “Update Information” button. He/she can enter the Project Selection Page by choosing “Project Operations”. Or he/she can look at general company statistics or read the tutorials by clicking “Documentation” button. If the current user is the configuration manager of the company, configuration buttons are enabled, else disabled.



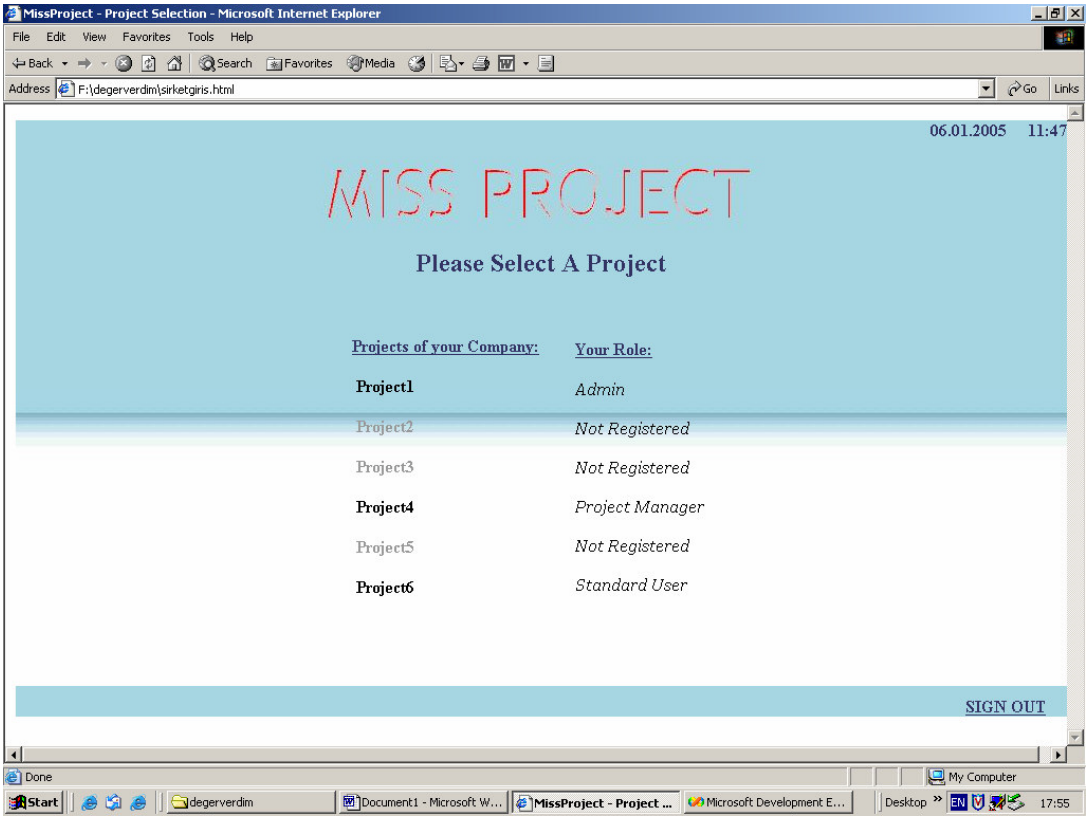
4.1.5. Update Personal Information Interface:

The user cannot change his/her username, name and company.



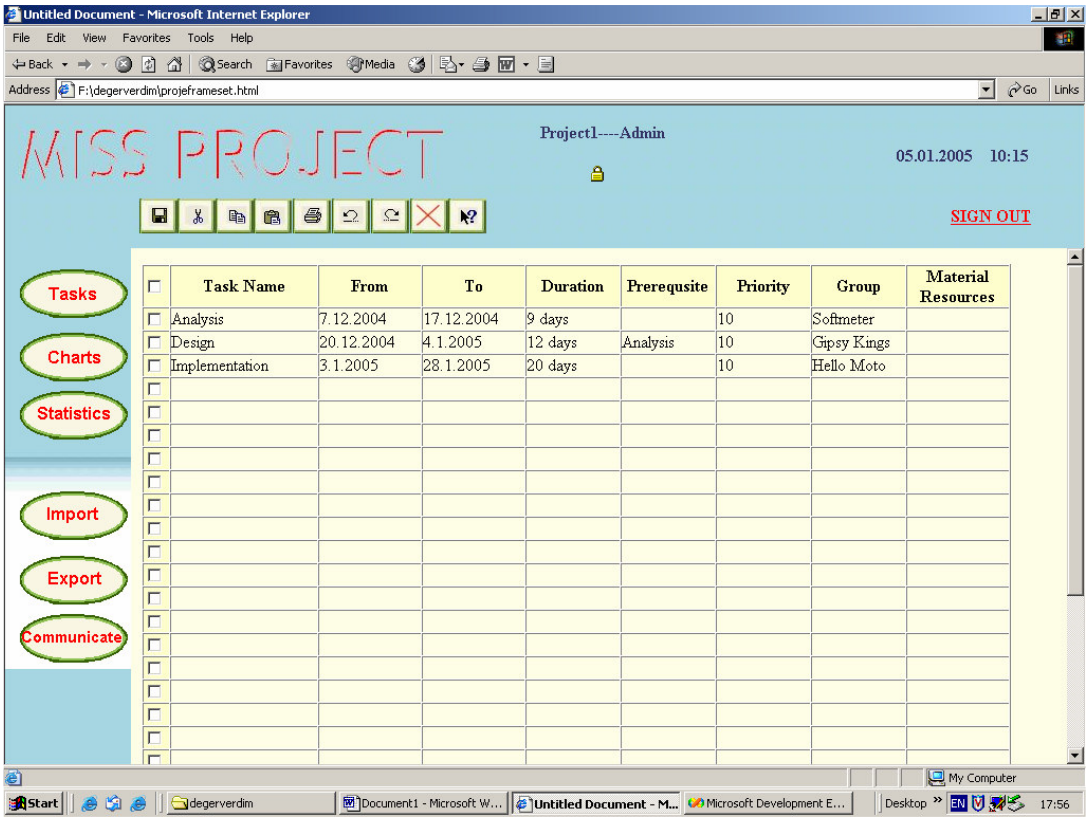
4.1.6. Project Selection Interface:

User selects the project in order to work on. He/she can only enter the projects that he/she has right to. A user can be an admin in one project, the project manager of another project, a standard user who has only view right in another project, and even not registered to a fifth one.



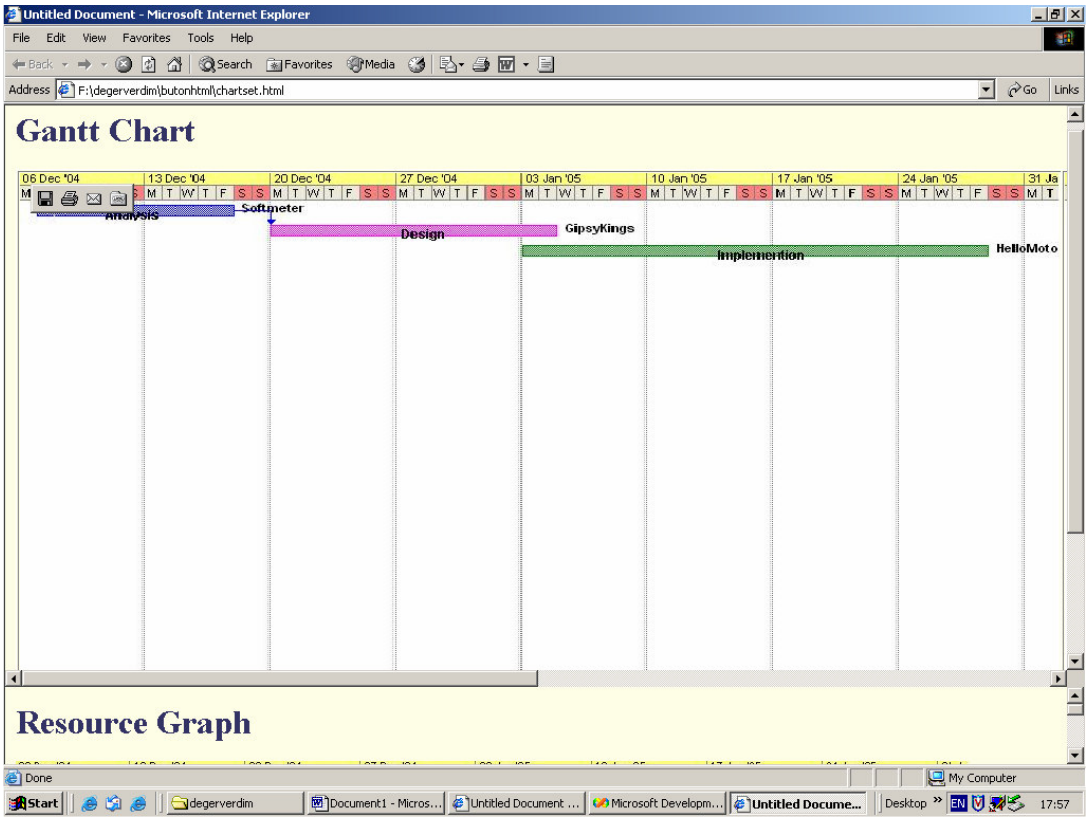
4.1.7. Tasks Interface:

If the project is locked, there is a lock icon under the name of the project. It means the user can not make any modifications even if he/she has right to.



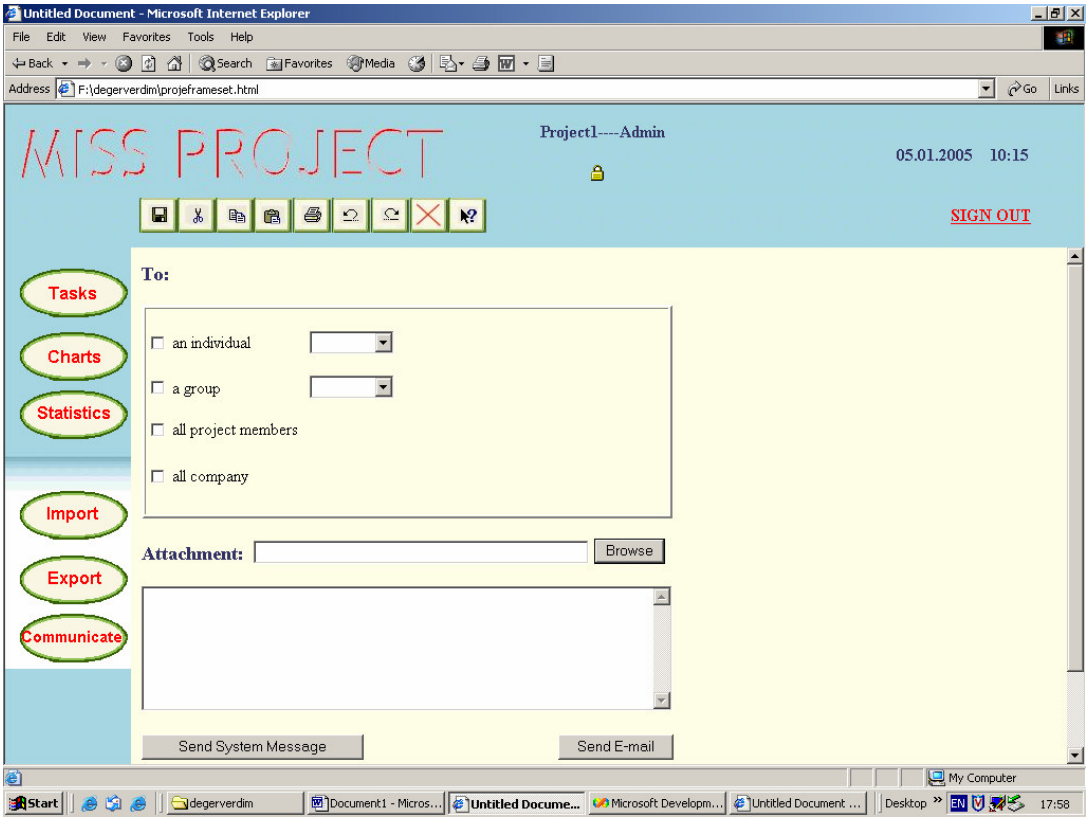
4.1.8. Charts Interface:

A new window is opened when Charts button at the project page is clicked. The advantage of a new window is flexibility and ease of use. By this way, the project manager can view the Gantt Chart at the same time he/she is adding a new task. Both Gantt Chart and Resource Graph is presented in this page.



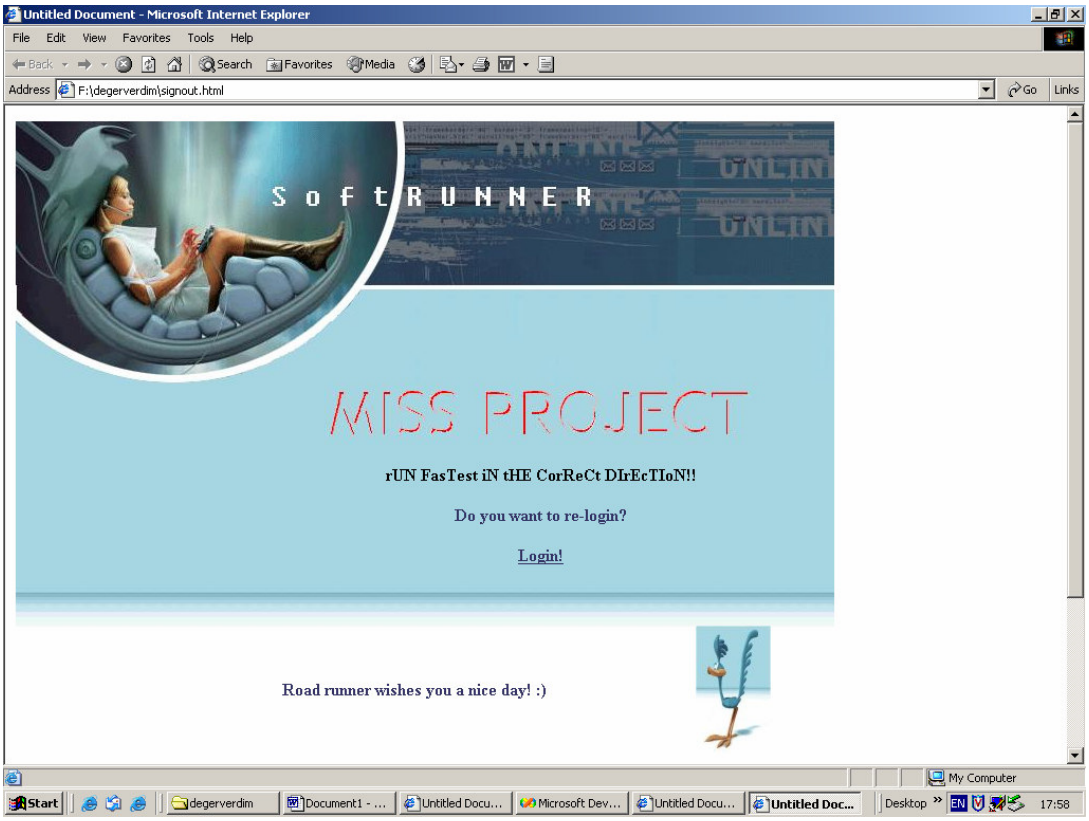
4.1.9. Communication Interface:

Different mailing options are present.



4.1.11. Sign Out Interface:

The user can sign out in every step. He/she may want to login with a different user name.



4.2. Human-machine Interface Design Rules

The following rules are considered while designing the user interfaces:

- The interfaces are made as user-friendly as possible
- The interfaces are consistent. Although every page has a similar look, the information content is well designed.
- Displaying unnecessary or extra information as well as displaying less information than needed is avoided.
- There is a generic error page. When an error occurs, the description of the error along with the possible cause of it and suggestions/warnings to the user are printed. This explanatory error page makes the system user aware of what is happening.

5. PROCEDURAL DESIGN

5.1 System Modules

5.1.1. Company Management Module

- Login
- Forgot Password
- Update Personal Information
- Add/Remove User
- Add/Remove Work Group
- Assign Individuals to workgroups
- Add/Remove resources to company database
- Change status of the resource(free/reserved /in use)

5.1.2. Project Management Module

- Add/Remove Project, assign project manager to project.
- Add/remove task to/from the database
- Assign tasks and administrators to project
- Assign priority, individual/workgroup, resource to tasks
- Prevent resource conflicts
- Arrange the deadlines
- Cost Estimation

5.1.3. Communication Module

- Send deliverables(meeting reports, document,..)
- Send notification mails
- Send System messages(to an individual, to workgroups, to all projet members, to all company members)
- Meeting Arrangement

5.1.4. Statistics Module

- Individual Schedule
- Gantt Chart Generation
- Resource Graph Generation
- Report Generation
-Custom filter of the project

5.2. Pseudo Code of System

5.2.1 Company Management Module

```
//Fundamental parts of company management module are
//”add project”, “delete project”, “update company
//information”

BEGIN company.management.module
CASEOF button.PRESSED(b1)
     WHEN b1==add.project
          BEGIN
          show.add.project.typing.screen;
          IF assign.project.manager.button.PRESSED
               THEN BEGIN
               CASE OF button.PRESSED(b2)
                    WHEN b2==select.from.users.list
                         BEGIN
                         show.users.of.company.list;
                         assign.selected.user.to.project;
                    redisplay.add.project.typing.screen;
                         END
                    WHEN b2==add.new.user
                         BEGIN
                         show.add.user.typing.screen;
                         record.new.user.to.database;
                         assign.added.user.to.project;
                    redisplay.add.project.typing.screen;
                         END
                    END CASE
               ENDIF
          record.new.project.to.database;
```

```
          redisplay.configuration.manager.main.page;
          END
     WHEN b1==delete.project
          BEGIN
               show.remove.project.selection.screen;
               remove.selected.project.from.database;
               redisplay.configuration.manager.main.page;
               END
     WHEN b1==update.company.information
          BEGIN
               show.update.company.information.typing.screen;
               record.new.information.to.database;
               redisplay.configuration.manager.main.page;
               END
     WHEN b1==update.personal.information
          BEGIN
               show.update.personal.information.typing.screen;
               record.new.information.to.database;
               redisplay.configuration.manager.main.page;
               END
          END CASE
    END
```

5.2.2 Project Management Module

```
//Fundamental parts of project management module are
//"update project details", "add/remove administrators"

BEGIN project.management.module
IF project.operation.button.PRESSED
THEN BEGIN
CASEOF button.PRESSED(bP)
     WHEN bP == edit.project.details
          BEGIN
          IF technical.processes.button.PRESSED;
               THEN BEGIN
          CASEOF button.PRESSED(b1)
               WHEN b1 == add.task.button.PRESSED
                    BEGIN
```

```
            show.new.task.typing.screen;
            record.new.task.to.database;
            redisplay.project.main.page;
            END
       WHEN b1 == remove.task.button.PRESSED
            BEGIN
            show.tasks.selection.screen;
            remove.selected.task.from.database;
            redisplay.tasks.selection.screen;
            END
       WHEN b1 == add.resource.button.PRESSED
            BEGIN
            show.new.resource.typing.screen;
            record.new.resource.to.database;
            redisplay.project.main.page;
            END
       WHEN b1 == remove.resource.button.PRESSED
            BEGIN
            show.reseource.selection.screen;
            remove.selected.resource.from.database;
            redisplay.resources.selection.screen;
            END
        WHEN b1 == add.group.button.PRESSED
            BEGIN
            show.new.group.typing.screen;
            record.new.group.to.database;
            redisplay.group.main.page;
            END
        WHEN b1 == remove.group.button.PRESSED
            BEGIN
            show.group.selection.screen;
            remove.selected.group.from.database;
            redisplay.groups.selection.screen;
            END
       END CASE
       END
  ELSEIF assign.roles.button.PRESSED
       THEN BEGIN
            IF add.admin.button.PRESSED
            THEN BEGIN
```

```
     IF select.from.users.list.button.PRESSED
          THEN BEGIN
          display.users.of.company.list.screen;
          assign.selected.user.as.admin;
          redisplay.add.admin.selection.screen;
          END
     ELSEIF add.new.user.button.PRESSED
          THEN BEGIN
          display.add.new.user.typing.screen;
          record.new.user.to.database;
          assign.added.user.to.project;
          redisplay.add.admin.typing.screen;
          END
          ENDIF
     END
     ELSEIF add.standard.user.button.PRESSED
     THEN BEGIN
     IF select.from.users.list.button.PRESSED
          THEN BEGIN
          display.users.of.company.list.screen;
          assign.selected.user.as.standard.user;
redisplay.add.standard.user.selection.screen;
          END
     ELSEIF add.new.user.button.PRESSED
          THEN BEGIN
          display.add.new.user.typing.screen;
          record.new.user.to.database;
          assign.added.user.to.project;
     redisplay.add.standard.user.typing.screen;
          END
          ENDIF
     END
     ELSEIF remove.user.button.PRESSED
          THEN BEGIN
          display.users.of.project.list;
          remove.selected.user.from.project;
          redisplay.remove.user.screen;
          END
ENDIF
END
```

```
          ENDIF
     WHEN bP == view.button
          BEGIN
               CASEOF button.PRESSED(b2)
          WHEN b2 == tasks
               BEGIN
                    display.tasks.list.screen;
               END
          WHEN b2 == resources
               BEGIN
               display.resources.list.screen;
               END
          WHEN b2 == groups
               BEGIN
               display.groups.list.screen;
               END
               END CASE
          END
     END CASE
     END
ELSEIF update.personal.information.button.PRESSED
     THEN BEGIN
     show.update.personal.information.typing.screen;
     record.new.information.to.database;
     redisplay.project.manager.main.page;
     END
END
```

5.2.3 Statistics Module

```
//Fundamental parts of statistics module are “Gantt Chart”,
//“Resource Graph”, “Reports”, “Individual Scheduling” and
//“Custom Filter”


BEGIN statistics.module
CASEOF button.PRESSED(b1)
     WHEN b1 == draw.Gantt.Chart.button.PRESSED
          BEGIN
```

```
          display.Gantt.Chart;
          END
     WHEN b1 == draw.Resource.Graph.button.PRESSED
          BEGIN
          display.Resource.Graph;
          END
     WHEN b1 == draw.Resource.Graph.button.PRESSED
          BEGIN
          display.Resource.Graph;
          END
     WHEN b1 == show.individual.schedule.button.PRESSED
          BEGIN
          display.individual.schedule;
          END
     WHEN b1 == custom.filter.button.PRESSED
          BEGIN
          display.statistics.with.desired.criteria;
          END
END CASE
END
```

5.2.4 Communication Module

```
//Fundamental parts of statistics module are “e-mail
//notifications”, “System Messages” and  “Meeting
//Arrangements”

BEGIN communication.module
IF any.change.on.project
     THEN BEGIN
     send.notification.via.email.to.all.users.of.project;
     END
CASEOF button.PRESSED(b1)
     WHEN b1 == send.system.message.button.PRESSED
          BEGIN
          display.system.message.typing.screen;
          IF to.a.user.selected
               THEN BEGIN
               send.message.to.that.user;
```

```
                END
            ELSEIF to.a.workgroup.selected
                THEN BEGIN
            send.message.to.all.participants.of.that.workgroup;
                END
            ELSEIF to.a.project.team.selected
                THEN BEGIN
            send.message.to.all.participants.of.that.project;
                END
            ELSEIF to.all.company.selected
                THEN BEGIN
                send.message.to.all.employees.of.that.company;
                END
            END
        WHEN b1 == arrange.meeting.button.PRESSED
            BEGIN
            IF valid.project.manager
                THEN BEGIN
                arrange.appropriate.datetime.and.room;
                send.notifications.to.participants.of.meeting;
                END
            END
END CASE
END
```

6. CLASSES OF TEST

6.1. Interface Testing

We will test our interfaces in the means of;

- Design
- Toolbars
- Buttons
- Menu items in order to check whether they work properly, and they are displayed correctly or not.

6.2. Black Box Testing

In this method, the system is considered as a black box, that is we only provide it with some inputs and get the corresponding output, no knowledge about internal process is assumed. Black box testing is applied to the interfaces, the descriptions are listed below.

Login Screen: The testing of the login module is easy. All possible combinations of a correct user-id, a wrong user-id, a correct password and a wrong password are entered through the interface. The output is observed. If an erroneous user-id/password pair is entered, the *authentication failure* page is printed, otherwise the user is successfully logged into the system.

Configuration Manager/Project Manager/Administrator/Standard User Main Interfaces: Almost no error can occur through these interfaces since the only input from the user through these interfaces is a click to a button or a link. In other words, the user is restricted with the input capability offered by the interface. But there is a constraint that all the links and the buttons should not be enabled for all users. So the outputs for such conditions must be checked properly. All the links will be followed and checked whether it sends the correct page or not.

For the other interfaces that needs typing some data, or entering some inputs such as Configuration Manager's "Add Project" interface; all the entries will be tested according to the valid inputs. Intentionally wrong inputs will be entered, e.g. a date in a wrong format, and checked whether it sends an error page or not. The content of the error page will be checked

to see if it contains the proper error description and proper solution(s). If all the values are entered correctly, we will check whether the system prints a page that indicates success or not.

The Black Box Testing will be applied to the all interfaces, and the results will be documented with the test report after the implementation phase.

6.3. White Box Testing

Since our system is a web-based application, we cannot simply write some scripts to test the internal processes and functions.

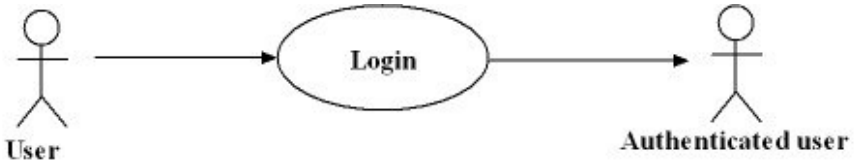
Instead, some software testing tools will be used for automating the white box testing. The class of the tools we use is named as *Capture/Replay Tools*. A capture/replay tool looks like an ordinary browser with an extra capability of *capturing* the events from the beginning of the session. It saves the links you have followed, the passwords you have entered and the values you have filled in a form. Using one of the *Capture/Replay Tools* available off-the-shelf, about 20 test cases are prepared. The test cases are designed carefully considering the internal processes and functions of the system. With this test cases almost all the loops and program structures of the system are explored. While testing the system by a capture/replay tool, some breakpoints are set to inspect the values of the variables and some parameters to verify the proper functioning of the main and sub routines.

7. CONCLUSION

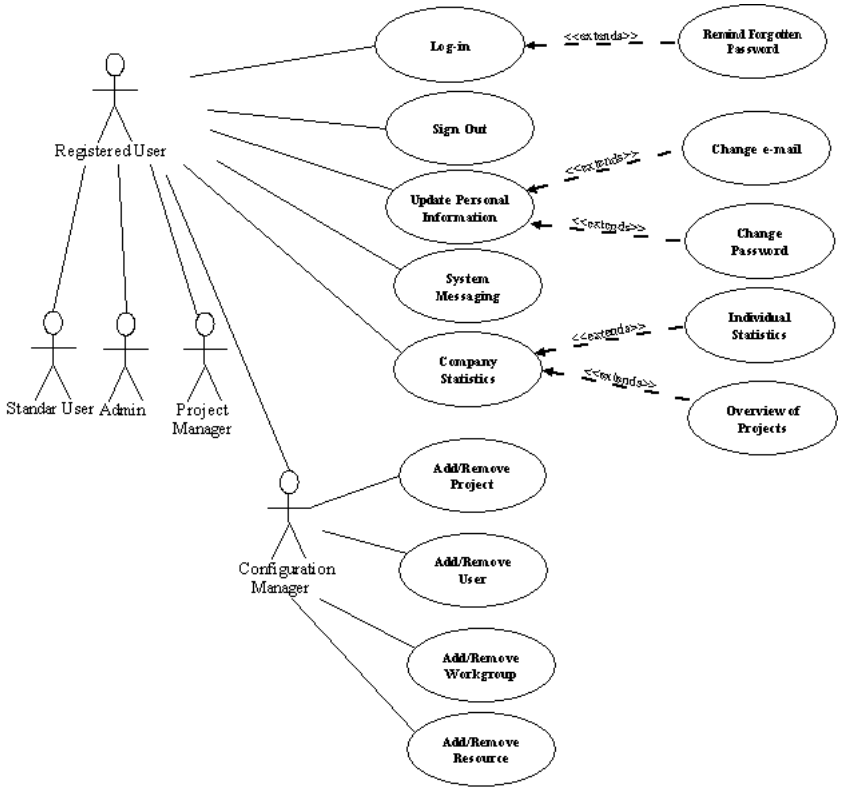
With *MissProject*, the end-users will be introduced to an easy, flexible (also even fun?) way of tracking effort on tasks. Also, our tool will provide continuous feedback of team status to both managers and customers; this will accelerate the improvement of project. Additionally, team communication through a shared view of activities will increase a lot. Planning metrics will be collected and calculated easily.

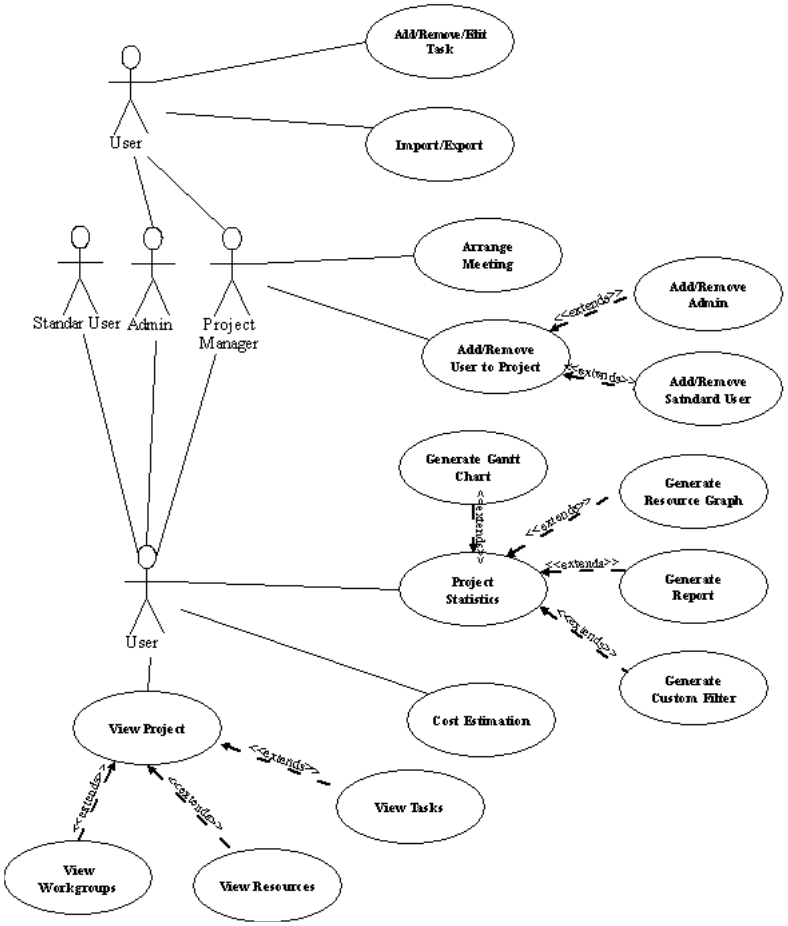
APPENDIX A

Use Cases:



Login


Registered User
Standar User
Admin
Project Manager
Configuration Manager
Log-in
<<extends>>
Remind Forgotten Password
Sign Out
Update Personal Information
Change e-mail
Change Password
System Messaging
Company Statistics
Individual Statistics
Overview of Projects
Add/Remove Project
Add/Remove User
Add/Remove Workgroup
Add/Remove Resource


User
Standar User
Admin
Project Manager
Add/Remove/Edit Task
Import/Export
Arrange Meeting
Add/Remove User to Project
Add/Remove Admin
Add/Remove Saindard User
Generate Gantt Chart
Generate Resource Graph
Project Statistics
Generate Report
Generate Custom Filter
Cost Estimation
View Project
View Tasks
View Workgroups
View Resources