

CSC 131-01

Project and MySQL Application Project

03/19/2013

Deliverable #3

**Table of Contents**

**1. Introduction Page 3**

**1.1 Purpose of This Document**

**1.2 Scope of The Development Project**

**2. General Description Page 4-7**

**2.1 Glossary**

**2.2 User Characteristics**

**2.3 Product Perspective**

**2.4 System Functional Requirements**

**2.5 General Constraints Assumptions**

**2.6 User View of Product Use**

**3. Specific Requirement Page 8-14**

**3.1 Interface Requirement**

**3.1.1 User Interface**

**3.1.2 Hardware Interface**

**3.1.3 Software Interface**

**3.1.4 Communication Interfaces**

**3.2 Detailed Description of Functional Requirements**

**3.2.1 Template For Describing Functional Requirements**

**3.2.2 Data Dictionary**

**3.3 Non-Functional Requirements**

**4. Object Oriented Analysis (OOA) – UML Page 15-18**

**4.1 Use Case Diagram**

**4.2 Use Cases Description**

**4.3 Use Case Diagram List Potential/Analysis Classes**

**5. Special Remarks or Comments Page 18**

**6. References or Resources Used Page 18 Team Member’s Roles and Approval Page 18-19**

**7.1 Team Member’s Roles In Completing The SRS**

**7.2 Team Member’s Signatures and Date Signed**

**1. INTRODUCTION  
*NOTE: General background and reference information***

**1.1 Purpose of this Document**

The purpose of this document is to outline the functions and requirements needed by the software in order to provide better understanding of them and to provide an outline of work required to complete the project.

**1.2 Scope of the Development Project**

**Front End:** To streamline signing up for both an ECS Project account and a MySQL database. More specifically, to combine the two separate links from the ECS Accounts webpage into one link, as well as minimize the signup process by requesting only one data input form instead of two separate ones, when both a Project account and a MySQL database are requested.

The limitations are that Project account names and MySQL database names requested cannot already be in use by either active Project accounts or current MySQL databases, and an active ECS user account is required in order to submit any request. A user can only request a MySQL database by itself, or along with requesting a Project account. For example, a user must not have the option of requesting a MySQL database if they are simultaneously requesting a Part-time Faculty Account.

**Back End:** Create a web application or offline program that ties in the current “snapshot view” of Project accounts and MySQL databases to server execution and vice-versa. The program will gather data from both the current webapps and the current servers in use. It will be able to sync discrepancies between the two by choosing what to copy or delete and to/from which side. Additionally, it will enable seamless “Pending” to “Active” transitions as well as “Active” to “Expired” server movements with minimal user input. The program also needs to be able to run queries and filter results.

**2. GENERAL DESCRIPTION**

***NOTE: This section gives an “executive overview” and is very client-oriented***

**2.1 Glossary (Definitions, Acronyms, and Abbreviations)**

**Account** - see Project Account

**Active** - a supposedly approved and accessible Project account or MySQL database

**Database** - see MySQL Database

**Dataman** - collection of MySQL Databases stored in a database

**Deleted** - an account or database which has been removed from the servers and filed away

**Department** - The ECS Computing Department and/or Lynne, the client

**Expired** - an account or database which has not been used in ‘x’ amount of time

**MySQL Database** - space allocated for student SQL use on a school server, usually associated with a school project

**Pending** - a user has applied for the project account or MySQL database marked as such, which is awaiting final approval

**Project Account** - account created for a group of students, in which they use to complete a school project

**Project Database** - collection of all project accounts

**Server** - the actual accounts and databases found on the school system, such as Athena, Hera, Gaia, etc...

**Validation** - verifying that a user already has an active ECS user account and the requested user name or database name is not currently in use by either a Project account or a MySQL Database

**2.2 User Characteristics**

**Front End**

Students/Applicants - need to be able to apply for accounts and/or databases, various skill levels

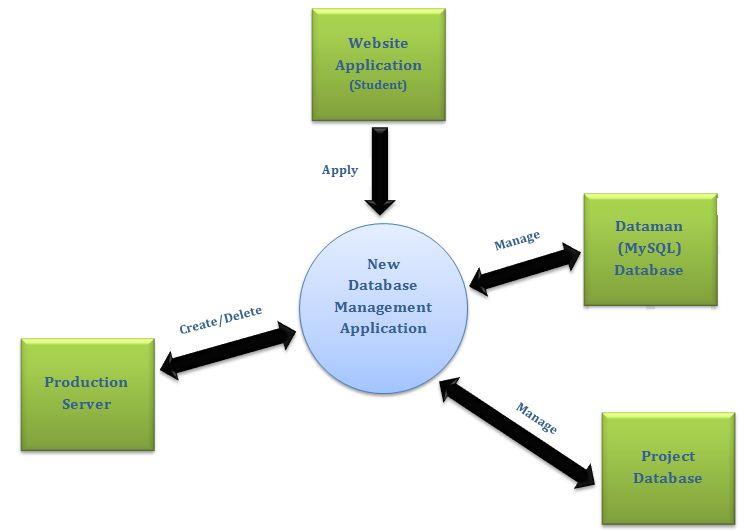
**Back End**

Lynne - Full access, computer guru

ECS Computing Services - Full access, various skill levels

**2.3 Product Perspective**

* + System Context Diagram (SCD)



* + SCD Description

The new database management application is the center system, which will collect applications from the web application. It will also interact with the Dataman database, Project database, and the Production servers by creating/deleting accounts, as well as managing them.

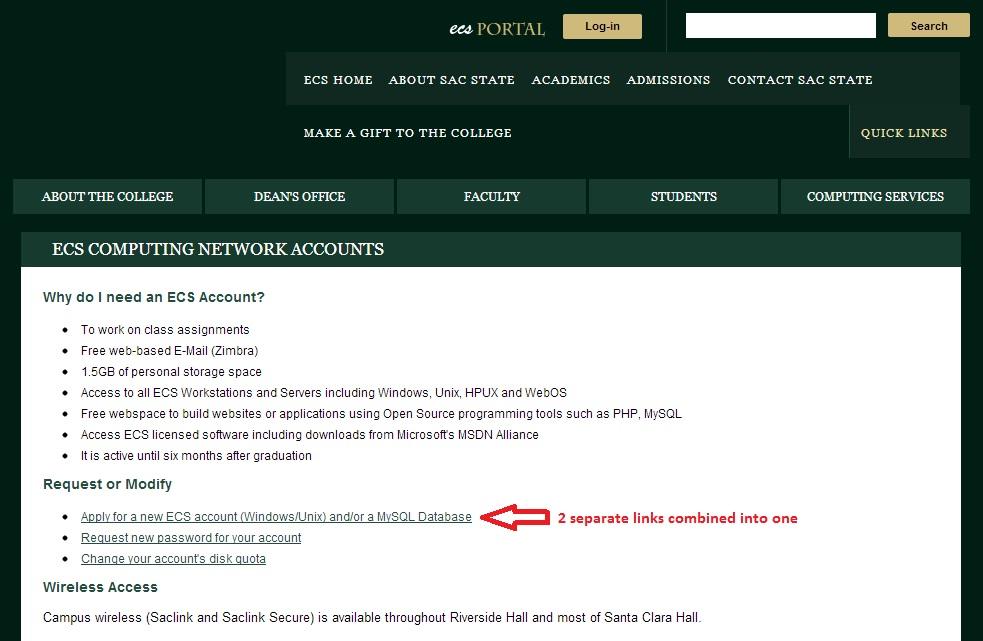
**2.4 System Functional Requirements (FRs)**

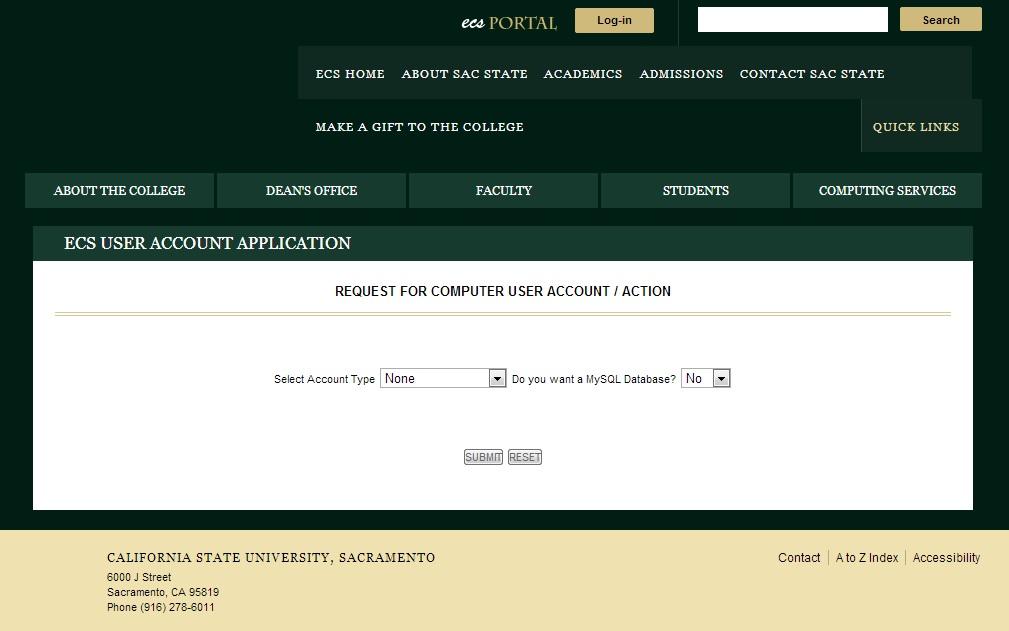
* + Front End Functional Requirements
    1. Combine MySQL database application link with the ECS account application link
    2. Merging of both the MySQL database account request and Project account request into one user application
    3. Validate Project account names to avoid duplication
    4. Validate MySQL database names to avoid duplication
    5. Validate user’s email address
    6. Insert new MySQL database into existing files, used by back end application, with “Pending” status
    7. Insert new Project accounts into existing files, used by back end application, with “Pending” status
    8. send email to ECS Computing Department and/or Lynne notifying them of new accounts/databases to approve
  + Back End Functional Requirements
    1. Approve “Pending” status Project accounts, marking them as “Active”
    2. Approve “Pending” status MySQL databases, marking them as “Active”
    3. Send emails to person(s) who requested the accounts and/or databases once approved
    4. Process expired Project accounts, moving them to archive folders and changing status from “Active” to “Expired”
    5. Process expired MySQL databases, moving them to archive folders and changing status from “Active” to “Expired”
    6. Compare “Active” status accounts and databases in the application to actual accounts and databases found on the server
    7. Add missing accounts to server side or application side
    8. Run reports and queries to filter the information in the program for further use

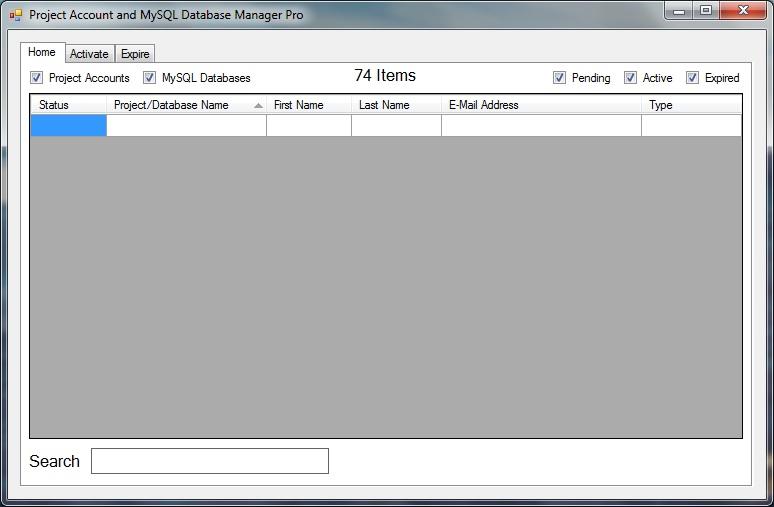
**2.5 General Constraints and Assumptions**

* + Need root access to use the application
  + Database is already configured for use
  + Stable internet connection may be maintained

**2.6 User View of Product Use**







**3. SPECIFIC REQUIREMENTS**

***NOTE: Technical information needed to design the software***

**3.1 Interface Requirements**

**3.1.1 User Interface**

The front end will be a web page utilizing HTML and PHP.

The back end will be a stand-alone executable file coded in Java.

**3.1.2 Hardware Interface**

N/A

**3.1.3 Software Interface**

Web browser that is Java-capable and has internet access, as well as Apache HTTP Server, MySQL, and PHP software suite (*AMP*).

**3.1.4 Communication Interfaces**

Internet Access

ECS Server Connectivity

Web-to-Database Connection

**3.2 Detailed Description of Functional Requirements**

**3.2.1 Template for Describing Functional Requirements**

**Front End**

* + 1. Combine MySQL database application link with the ECS account application link

Purpose/Description

The purpose of this functional requirement is to make the website more efficient as well as make it easier for the user to navigate through.

Inputs

* + - * + “Apply for a MySQL Database” link on ECS Accounts web page
        + “Apply for an ECS Account (Windows/Unix)” link on ECS Accounts web page

Processing

Combine the two links into one link, which then redirects to a webpage which more specifically asks the user what they want to apply for

Outputs

* + - * + Only one link, “Apply for an ECS Account and/or a MySQL Database”
    1. Merging of both the MySQL database account request and Project account request into one user application.

Purpose/Description

Currently, when a user applies for a Project account on the ECS website, they are not prompted to apply for a MySQL database. As an Engineering and Computer Science student at Sac State, many students commonly need both a Project account and a MySQL database. The issue with the current functionality is that when a student applies for both a Project account and a MySQL database, they have to fill out two different online applications. When a user applies for a Project account on the ECS website, the system *should* ask if they want to apply for a MySQL database as well. And vice versa. This functionality will allow users to apply for a Project account and a MySQL database via a single process.

Inputs

* + - * + User input via form submission on the ECS web page when applying for a Project account
        + User input via form submission on the ECS web page when applying for a MySQL database
        + User input to apply for a MySQL database when applying for a Project account
        + User input to apply for a Project account when applying for a MySQL database

Processing

When a user additionally applies for a ECS project account or MySQL database, the system will take in the inputted items from the application form and begins the process to create both an ECS project account and a MySQL database. When the user chooses to apply for both, the system will choose the web form that contains the information needed about both a project account and a MySQL database.

Outputs

* + - * + When a user chooses to apply for both an ECS project account and a MySQL database, the system outputs a form that has information for both
        + When a user submits their form to apply for both an ECS project account and a MySQL database, the system will tell the back-end to create both
        + When a user finishes the form and submits their form to apply for both an ECS project account and a MySQL database, the system outputs a summary form that confirms the user’s submission
    1. Validate Project account names to avoid duplication

Purpose/Description

When a user applies for a Project account, their requested Project account name must be checked for validity before entering the ECS databases. If a Project account name is already in use, the user must be notified that they cannot use that same Project account name and must specify another.

Inputs

The user’s requested Project account name.

Processing

Upon submission of the user’s application, the website will check the requested Project account name with all “Active” status Project account names to make sure the requested name is unique and not a duplication

Outputs

If the name is found to be currently in use, an alert will be displayed telling the user that the name is in use and they must change the requested name, then resubmit the application. If the name is currently unused, the user will see that their application was successfully submitted to the department

* + 1. Validate MySQL database names to avoid duplication

<See iii, but replace all instances of “Project account” with “MySQL Database”>

* + 1. Validate user’s email address

Purpose

To make sure that a person requesting an account has a valid school email address

Inputs

The email address typed into the web form application

Processing

Compare the input email address with all Sac State known email addresses when the user submits their application

Outputs

If a match has been found, the email address has been verified, and the application will proceed by telling the user that their application was successfully submitted and to wait for an approval by email. Otherwise, an alert will notify the user that their email address was not found on record and that they must provide a valid Sac State email address in order to submit the application

* + 1. Insert new MySQL database into existing files, used by back end application, with “Pending” status

Purpose

To automate creating the new MySQL database entry in the database file used by the backend application

Inputs

A successfully submitted application for a MySQL database

Processing

The user’s inputted application data, such as database name, email address, first name, last name, etc, will be transferred and inserted into the dataman file as a new MySQL database entry

Outputs

A new entry in the dataman database with status of “Pending” as well as an email sent to the ECS Computing Dept to let them know that an application has been received

* + 1. Insert new Project accounts into existing files, used by back end application, with “Pending” status

<See vi, but replace all instances of “MySQL database” with “Project account” and all instances of “dataman” to “projects”>

* + 1. send email to ECS Computing Department and/or Lynne notifying them of new accounts/databases to approve

Purpose

To let the department know that someone has applied for a new account and/or database so that they may approve it

Inputs

Successfully submitted application request for a MySQL database and/or a Project account

Processing

A standardized email will be sent to the department and will include some brief information from the submitted application

Outputs

The email being sent is the output, but no visual indicator to the user. The output is verified once the department finds a the new email in their inbox

**Back End**

* + 1. Approve “Pending” status Project accounts, marking them as “Active”

Purpose/Description

Change status of Project accounts from “Pending to “Active” in the program and create the accounts on the server

Inputs

“Pending” status Project accounts derived from a search or filter of the available accounts

Processing

After the user confirms activating the accounts or databases, an email will be sent to the account requester, accounts will be marked as “Active”, and will be created on the server with Linux commands

Outputs

A new Project account on the server. These are created/automated using standard Linux commands.

* + 1. Approve “Pending” status MySQL databases, marking them as “Active”

<See ii, but replace all instances of “Project account” with “MySQL Database”>

* + 1. Send emails to person(s) who requested the accounts and/or databases once approved

Purpose/Description

To let the requestor know that the Project account or MySQL database has been activated and can now be put to use

Inputs

Project account name or MySQL database name taken from the approved entry in projects file or damaman file, as well as the email address associated with the account or database

Processing

Once an account or database has been approved by the ECS department, a standardized email shall be created and sent to the email address found on file which is associated with the current account or database

Outputs

No tangible output. Process should be automated, but the email will have been successful when it is received by the requestor

* + 1. Process expired Project accounts, moving them to archive folders and changing status from “Active” to “Expired”

Purpose/Description

Change status of Project accounts from “Active” to “Expired” in the program and remove them from the server

Inputs

“Active” status Project accounts derived from a search or filter of the available accounts

Processing

After the user confirms expiring the accounts, the accounts will be copied to a semester folder, removed from the server, and then marked as “Expired”

Outputs

Accounts copied into a storage folder/location as well as removal from the active server. This is accomplished using standard Linux commands

* + 1. Process expired MySQL databases, moving them to archive folders and changing status from “Active” to “Expired”

<See ii, but replace all instances of “Project account” with “MySQL Database”>

* + 1. Compare “Active” status accounts and databases in the application to actual accounts and databases found on the server

Purpose/Description

To expose any and all accounts/databases that are either in the program as “Active”, but not in the server as such, or vice-versa

Inputs

All “Active” accounts found in both the server and the program

Processing

Upon opening the program a query will result in all “Active” accounts/databases that do not have a matching account or database found on the other end. For example, if an account is active in the program, but not found on the server, it will show up on this list. Once displayed to the user, that user can determine if they want to sync the active accounts/databases individually to the missing area (program or server) or just delete them as extraneous and unnecessary accounts or databases

Outputs

New accounts or databases in program to match ones already in the server or new accounts/databases on server to match the ones in the program or removal of extraneous accounts/databases if not needed. If no accounts/databases are found by the query, program will resume normally without a prompt for syncing.

* + 1. Add missing accounts to server side or application side

Purpose/Description

To effectively sync what is found on the server to what is found in the projects and dataman files, to prevent further problems

Inputs

A list of Project accounts or MySQL databases that are found on the server but not in the program or vice versa

Processing

If an account or database was found in the program files, but not on the server it will be displayed on the left side of the screen, and if an account or database was found on the server, but no matching entry is found in the files, it will be displayed on the right. The application user can then choose to delete or sync, individually, to the server the items on the left, and then can choose to delete or sync, individually, the items from the server to the files. There will also be an option to sync all both ways, effectively creating the missing entries on the other ends. A confirm window should probably be shown to prevent accidental deletion or syncing.

Outputs

A small new window which gives a summary of the syncing or deletion or both

* + 1. Run reports and queries to filter the information in the program for further use

Purpose/Description

To look up specific accounts or databases to find out more information about them or to perform actions on them

Inputs

The user will input their search criteria into the program as well as if they want to find accounts and/or databases. For example, one could search by Project Account name or the last name of a student, or even by filling out multiple fields to filter results

Processing

Once the user has filled out all fields used in the search and clicked the filter button, a list of all accounts and/or databases that match the criteria will be displayed as a list. If no results are found, the program will tell the user as such

Outputs

Accounts and/or databases in the program that match the search criteria. The user can then click individual entries to see a detailed view and perform several actions or select one or multiple entries from the list and perform actions on them all at once

**3.2.2 Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item Number | Data Item Name | Data Item Type | Alias / Variable Name | Data Item Type Size | How Data Item Is Used |
| 1 | Project Account Type | String | accountType | 4 - 17 Characters | The type of user account the user is applying for |
| 2 | Project Account Name | String | accountName | 1 - 15 Characters | The desired name for the project account |
| 3 | Both Applications Chosen | Boolean | bothAppsChosen | True or False | The indicator that decides wether the user wants to apply for a Project account and a MySQL Database |
| 4 | Selected Major | String | major | 3 Characters | To ensure that a Project account or a MySQL database is necessary for their major |
| 5 | Advisor Name | String | advisorName | 1 - 15 Characters | To verify which advisor instructed the user to obtain a MySQL database or Project Account |
| 6 | Advisor Email | String | advisorEmail | 1 - 20 Characters | To have the direct contact email of the advisor who instructed students to apply for an account |
| 7 | Number Of Participants | Integer | participantNum | 1 - 4 Non-Decimal Numbers | To verify how many people are involved with this account |
| 8 | Number Of Active Semesers | Integer | activeSemNum | 2 - 10 Non-Decimal Numbers | To know how long the account must be active and running |
| 9 | User Comments | String | userComments | 0 - 250 Characters | To allow applicants to write a miscellaneous comment or feedback |
| 10 | Applicant First Name | String | applicantFName | 1 - 15 Characters | To have the first name of the person who is applying for this account |
| 11 | Applicant Last Name | String | applicantLName | 1 - 15 Characters | To have the last name of the person who I applying for this account |
| 12 | Applicant Phone Number | String | applicantPhone | 1 - 15 Characters | To have a direct phone number to contact the applicant who is in charge of the account |
| 13 | Applicant Email | String | applicantEmail | 1 - 20 Characters | To have a direct email address to contact the applicant who is in charage of the account |
| 14 | MySQL Database Name | String | dbName | 1 - 15 Characters | To designate an alias to the database when it is created |
| 15 | MySQL Host Location | String | dbHostLoc | 1 - 20 Characters | To designate a location on the server where the MySQL database will reside |
| 16 | MySQL Permissions | Boolean | dbPermissions | True of False | To know which permissions the user chose for the MySQL database |

**3.3 Non-Functional Requirements (NFRs)**

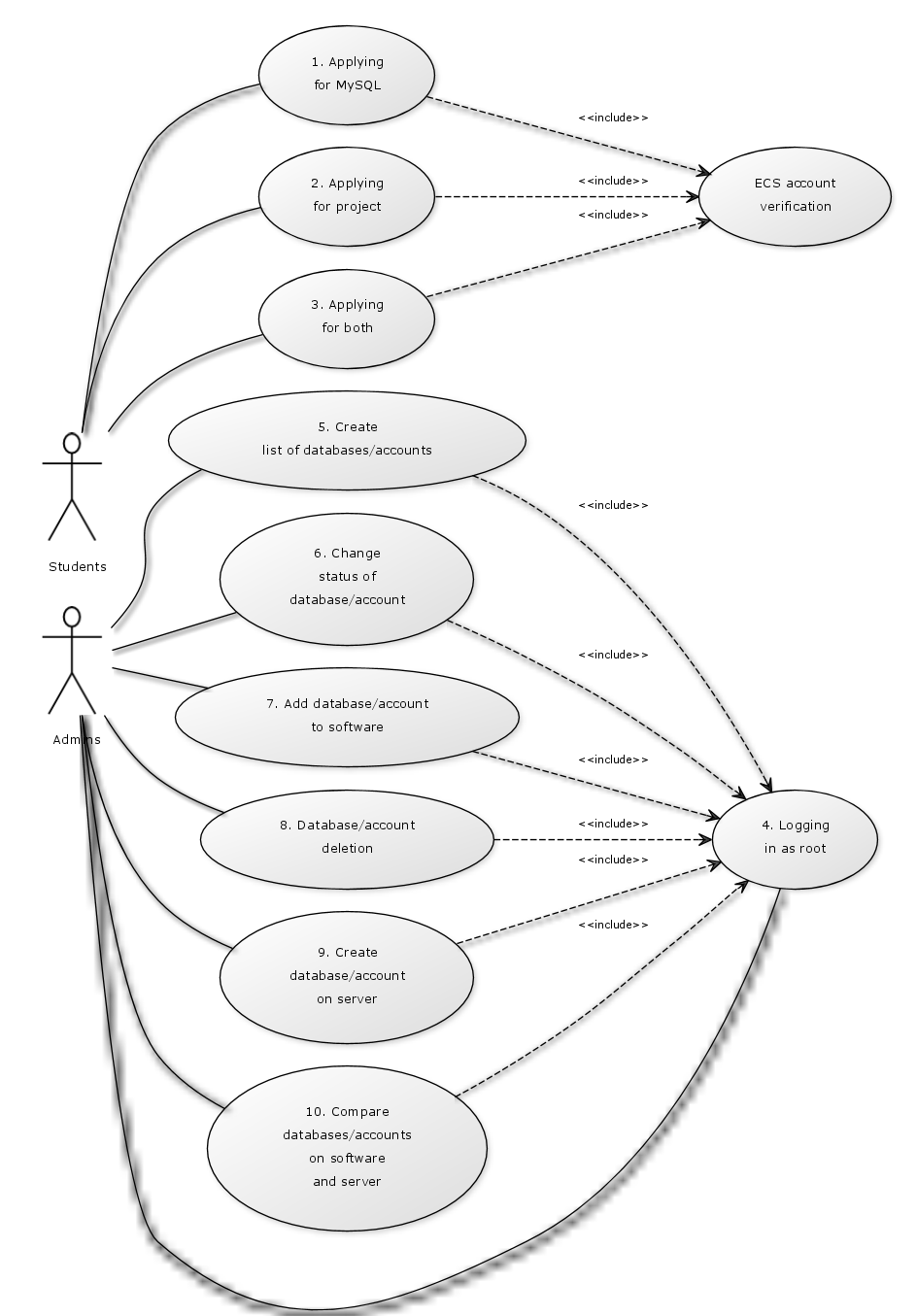
Some of the non-functional requirement that we would like to succeed

with, but are not limited to include:

* + - Accuracy of information
    - usability
    - efficiency
    - maintainability
    - reliability
    - security
    - performance
    - portability

**4. OBJECT ORIENTED ANALYSIS (OOA) - UML**

***NOTE: Use any other additional models needed to document your work in this section***

**4.1 Draw a Complete “Use Case” Diagram**

**4.2 Describe each Use Case**

**Use Case Name:** Applying for MySQL

**Use Case Number:** 1

**Author(s):** Le and Cody Lanier

**Actor(s):** Students

**Overview:** This use case demonstrates the possible processes the frontend goes through when an applicant applies for a MySQL database

**References:**

**Related Use Cases:** 2, 3

**Typical Flow Description:**

1. Applicant wants MySQL database

2. Applicant fills out webform

3. Validate applicant has an active ECS account

4. Confirm entered information

5. Submit application

6. “Pending” status MySQL database entry created in software

**Alternative Flow Description:**

1. Invalid ECS account found

2. Display to user that incorrect account information was entered

3. Re-prompt for username and password

**Use Case Name:** Applying for project account

**Use Case Number:** 2

**Author(s):** Le

**Actor(s):** Students

**Overview:** This use case demonstrates the possible processes the front end goes through when an applicant applies for a project account

**References:**

**Related Use Cases:** 1, 3

**Typical Flow Description:**

1. Applicant wants project account

2. Applicant fills out webform

3. Validate applicant has an active ECS account

4. Confirm entered information

5. Submit application

6. “Pending” status project account entry created in software

**Alternative Flow Description:**

1. Invalid ECS account found

2. Display to user that incorrect account information was entered

3. Re-prompt for username and password

**Use Case Name:** Logging in as root

**Use Case Number:** 4

**Author(s):** Le and Lanier

**Actor(s):** Admins

**Overview:** This use case demonstrates the possible processes the back end goes through when the user desires root access. This is required for most functions provided by back end.

**References:**

**Related Use Cases:**

**Typical Flow Description (steps):**

1. Perform backend action which requires root access

2. Type in username and password

3. Validate username and password

4. Log in

**Alternative Flow Description (steps):**

1. Validation fails

2. Display to user that information entered is invalid

3. Prompt again for username and password

**Use Case Name:** Add database/account to software

**Use Case Number:** 7

**Author(s):** Le

**Actor(s):** Admins

**Overview:** This use case demonstrates the possible processes the back end goes through when the user wants to add either a database or an account entry to the software

**References:**

**Related Use Cases:** 1, 3, 10

**Typical Flow Description (steps):**

1. Student/applicant successfully submitted application for account and/or database

2. Create new entry in program with status “Pending”

3. Fill this new entry with information from submitted application

4. Send email to applicant and ECS Computing Services alerting them of a pending account and/or database

**Alternative Flow Description (steps):**

1. Account and/or database found on server, but not in program

2. Alert program user of mismatch

3. Create new entry in program with status “Active”

4. Fill this new entry with information from the server account

5. Display that the sync was a success

**Use Case Name:** Compare databases/accounts on software and server

**Use Case Number:** 10

**Author(s):** Le

**Actor(s):** Admins

**Overview:** This use case demonstrates the possible processes the back end goes through when the user wants compare existing databases/accounts on both the application and the server

**References:**

**Related Use Cases:** 7

**Typical Flow Description (steps):**

1. User wants to compare databases or accounts on program and server

2. Program connects to server and reads live databases and/or accounts

3. Information from both server and program are displayed for the user

**Alternative Flow Description (steps):**

1. Program is unable to connect to server to read databases and/or accounts

2. Prompt for verification that a connection to desired server is available

3. Attempt to connect again after confirmation

**4.3 List Potential/Analysis Classes based on Problem Statement & Use Case Diagram**

N/A

**5. SPECIAL REMARKS OR COMMENTS**

N/A

**6. REFERENCES OR RESOURCES USED**

N/A

**7. TEAM MEMBERS’ ROLES AND APPROVAL**

**7.1 Roles**

**All:** 2.1, 2.2, 3.3

**Cody:** 1.2, 2.4, 2.6, 3.2.1, 4.2

**David:** 2.5,3.2.1,3.2.2

**Jack:** TofC**,** 2.3, 2.4, 3.3

**John:** 1.1, 3.1, 4.1, 4.2

**7.2 Signatures and Date Signed**

(>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)>

(>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)>

(>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)>

(>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)>

(>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)> (>’.’)>