

CSC 131-01

Project and MySQL Application Project

05/23/2013

Software Design Document

Deliverable #4

**Table of Contents**

1. **Introduction Page # 3**
2. **Revised Requirements Page # 3**
3. **Data Design Page # 4 - 15**
   * **ERD Diagram**
   * **ERD Entities and Relationship Descriptions**
   * **Sequence Diagrams for Most Critical Use Cases**
   * **Classes**
   * **Class Attributes and Operations**
   * **Expanded and Updated Data Dictionary**
4. **Architecture Design Page # 16 - 19**
   1. **Basic Class Diagram**
   2. **Description of Class Diagram**
5. **Detailed Class Design Page # 20 - 23**
6. **User Interface Design Page # 24 - 31**
7. **Programming Tools Page # 32**
8. **Restrictions, Limitations, and Constraints Page # 32**
9. **References or Resources Used Page # 32**
10. **Team Member’s Roles and Approval Page # 32**
    1. **Team member’s roles in SDD**
    2. **Team member’s signatures and date**
11. **INTRODUCTION**

The purpose of this document is to provide the overall flow and architecture of the project. Included in this document are diagrams that show the entities and relationships between crucial objects and classes.

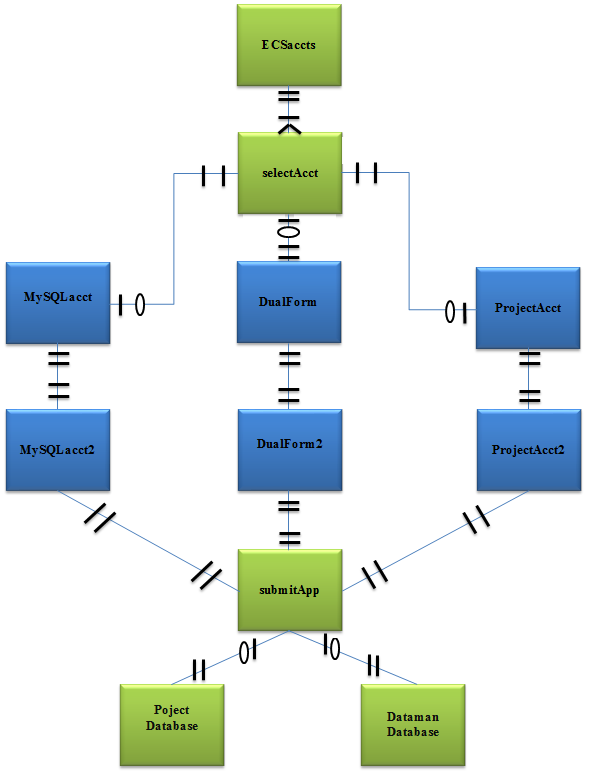
1. **REVISED REQUIREMENTS**

As of current, requirements have not been revised and the project has been following the SRS as accurately as possible.

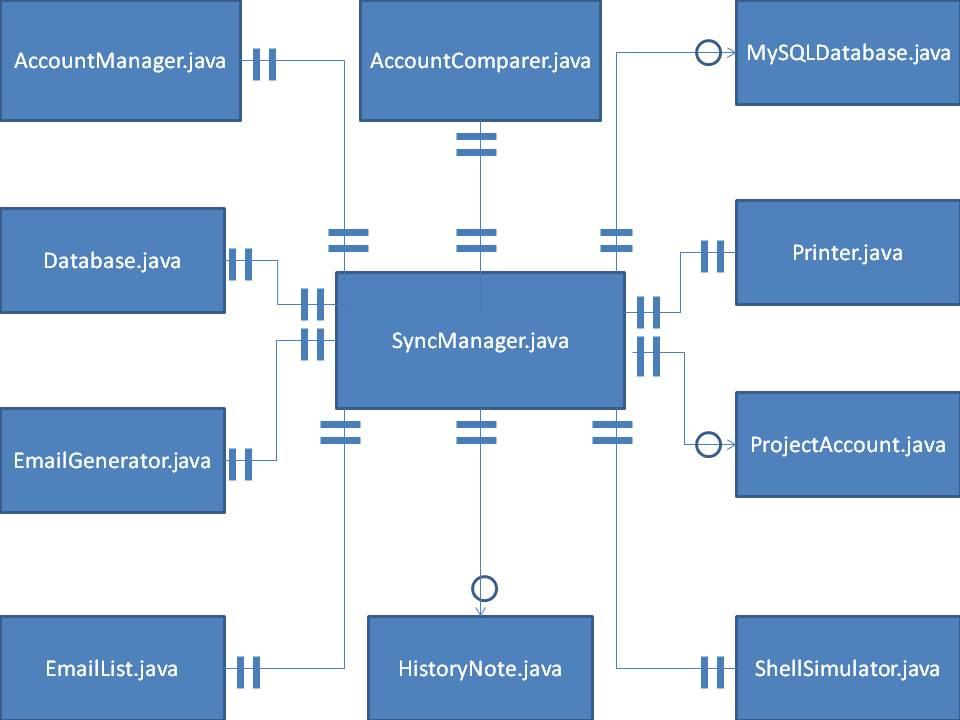
1. **DATA DESIGN**

**ERD Diagrams**

***FRONT END***



***BACK END***

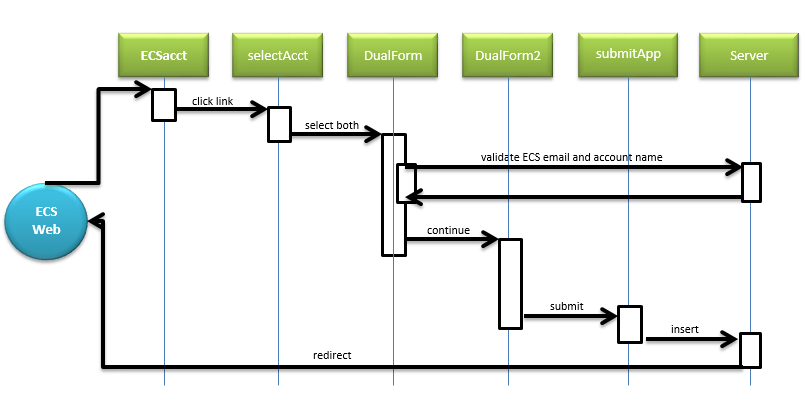


**ERD Entities and Relationship Descriptions**

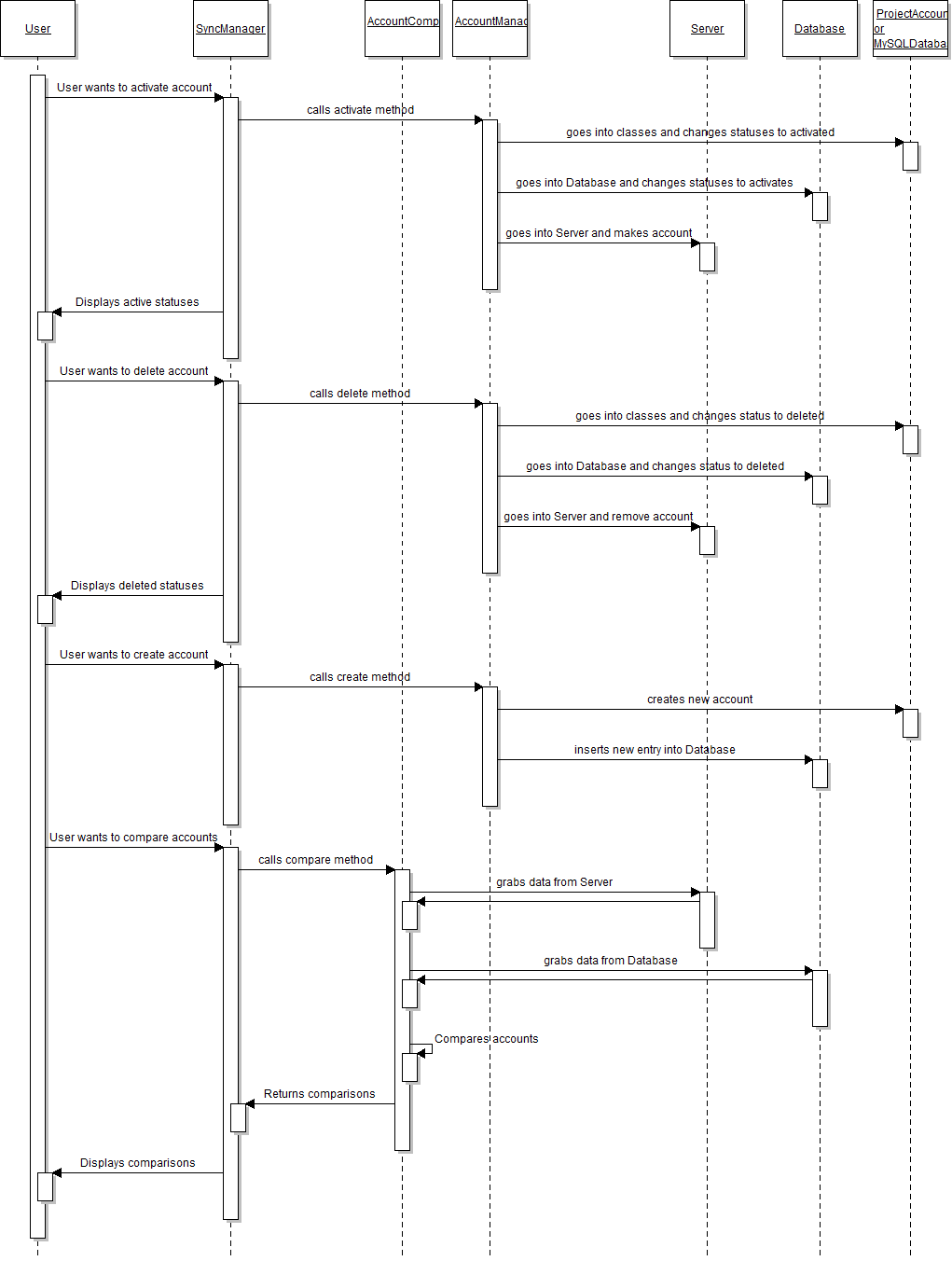
* + - **Front End -** when applying for ECS accounts users have the option to apply for one or two accounts. They may apply for a MySQL account, a Project account, or for both accounts at the same instance. For the most part all the entities have a one-to-one type of relationship, except for the selection of accounts and the databases, where they may exist independently.
    - **Back End** - when running the SyncManager application, all entities must be present and included for best functionality. Some classes, such as MySQLDatabase.java and ProjectAccount.java, are more flexible and theoretically not required for functionality.

**Sequence Diagrams for Most Critical Use Cases**

***FRONT END***



***BACK END***



**Classes**

***FRONT END***

* + - ECSaccts.htm
    - selectAcct.htm
    - ProjectAcct.php
    - ProjectAcct2.php
    - submitProj.php
    - MySQLacct.php
    - MySQLacct2.php
    - submitSQL.php
    - DualForm.php
    - DualForm2.php
    - submitDual.php

***BACK END***

* + - AccountComparer
    - AccountManager
    - Database
    - Email Generator
    - EmailList
    - HistoryNote
    - IOTest
    - MySQLDatabase
    - ProjectAccount
    - Shell Simulator
    - SyncManager
    - Tunnel

**Class Attributes and Operations**

***FRONT END***

* + - select\_account
    - lblmysql
    - mysql
    - validation\_email
    - account\_name
    - quota
    - advisor\_name
    - advisor\_email
    - shell
    - num\_students
    - major
    - group
    - num\_semesters
    - num\_semesters\_other
    - mysql\_host\_location
    - permissions\_radio
    - mysql\_permissions[]
    - comments
    - $host
    - $permissions
    - $expire
    - $year
    - $todayDate
    - HidePart2()
    - mkHidden()
    - chkSelected()
    - enableOther()
    - otherRB()
    - greyOut()
    - hideRed()
    - hideRed2()
    - DualAcctNameCheck()
    - ECSemailValidate()
    - ConnectProjAcct()
    - ConnectSQLdb()
    - sshConnect()
    - SQLacctValidate()
    - ProjAcctValidate()

***BACK END***

Database

* public Database(String dbUser, String dbPassword, String dbName)
* public int countQueryResults (ResultSet newRS)
* public ResultSet getTable(String tableName)
* public ResultSet executeQuery(String sqlQuery)
* public int executeUpdate(String sqlUpdate)
* public int getDeviceCount()
* public boolean connect()
* public boolean isConnected()
* public void closeConnection()
* public Connection returnConnection()

MySQLDatabase

* public void set\_projName(String projName)
* public void set\_server(String server)
* public void set\_semesterCreated(String semesterCreated)
* public void set\_dateExpires(int dateExpires)
* public void set\_dateCreated(int dateCreated)
* public void set\_dateDeleted(int dateDeleted)
* public void set\_status(String status)
* public void set\_advisor\_name(String advisor\_name)
* public void set\_advisor\_email(String advisor\_email)
* public void set\_expiredWarning(String expiredWarning)
* public void set\_DBAccountName(String DBAccountName)
* public void set\_db\_host(String db\_host)
* public void set\_db\_alter(String db\_alter)
* public void set\_db\_index(String db\_index)
* public void set\_db\_references(String db\_references)
* public void set\_db\_grant(String db\_grant)
* public void set\_db\_drop(String db\_drop)
* public void set\_db\_create(String db\_create)
* public void set\_db\_delete(String db\_delete)
* public void set\_db\_update(String db\_update)
* public void set\_db\_insert(String db\_insert)
* public void set\_db\_select(String db\_select)
* public void set\_groupName(String groupName)
* public void set\_historyID(int historyID)
* public void set\_hisDetail(String hisDetail)
* public void set\_hisShortDesc(String hisShortDesc)
* public void set\_hisDate(int hisDate)
* public void set\_uname(String uname)
* public void set\_pass(int pass)
* public void set\_access\_level(String access\_level)
* public void set\_signature(String signature)
* public void set\_peopleID(int peopleID)
* public void set\_deptMajor(String deptMajor)
* public void set\_pNameLast(String pNameLast)
* public void set\_pNameFirst(String pNameFirst)
* public void set\_phoneNum(int phoneNum)
* public void set\_email(String email)
* public String get\_projName()
* public String get\_server()
* public String get\_semesterCreated()
* public int get\_dateExpires()
* public int get\_dateCreated()
* public int get\_dateDeleted()
* public String get\_status()
* public String get\_advisor\_name()
* public String get\_advisor\_email()
* public String get\_expiredWarning()
* public String get\_DBAccountName()
* public String get\_db\_host()
* public String get\_db\_alter()
* public String get\_db\_index()
* public String get\_db\_references()
* public String get\_db\_grant()
* public String get\_db\_drop()
* public String get\_db\_create()
* public String get\_db\_delete()
* public String get\_db\_update()
* public String get\_db\_insert()
* public String get\_db\_select()
* public String get\_groupName()
* public int get\_historyID()
* public String get\_hisDetail()
* public String get\_hisShortDesc()
* public int get\_hisDate()
* public String set\_uname()
* public int get\_pass()
* public String get\_access\_level()
* public String get\_signature()
* public int get\_peopleID()
* public String get\_deptMajor()
* public String get\_pNameLast()
* public String get\_pNameFirst()
* public int get\_phoneNum()
* public String get\_email()

SyncManager

* public SyncManager()
* public boolean verifyLogin()
* public void testJoin()
* public void createGUI()
* public void windowClosing(WindowEvent e)
* public void searchForPhrase(String keyWord)
* public void populateCompareReport(JLabel[] arr1, JLabel[] arr2, JPanel p1, JPanel p2)
* public LinkedList<String> createNewLineByString(String original)
* public void runCompareReport(JLabel[] arr1, JLabel[] arr2)
* public void insertRow()
* public void deleteRows(JTable jt, Object[][] d)
* public void moveToRightTable(JTable table1, JTable table2, Object[][] d1, Object[][] d2, String moveType)
* public void moveToLeftTable(JTable table1, JTable table2, Object[][] d1, Object[][] d2, String moveType)
* public void clearRow(int rowIndex, Object[][] d)
* public boolean isPartOfArray(int[] arr, int i)
* public void filterDataBy(LinkedList<String> boxesChecked)
* public void unfilterData(String s)
* public void loadData()
* public void clearData(JTable t, Object[][] d)
* public void activateAccounts(Object[][] d)
* public void activateAccounts(String unixCmd)
* public void expireAccounts(Object[][] o)
* public void deleteAccounts(Object[][] o, String unixCmd)
* public void createAccount(String db, Object[] accountInfo)
* public void createMembers(String db, Object[] memberInfo)
* public void showUnixWindow(boolean show, int window)
* public void showMemberWindow(boolean show)
* public void addMoreMembers(int addCount)
* public class KeyPressListener implements KeyListener
* public void keyPressed(KeyEvent e)
* public void keyReleased(KeyEvent e)
* public void keyTyped(KeyEvent e)
* public class TabListener implements ChangeListener
* public void stateChanged(ChangeEvent e)
* public class ClickListener implements ActionListener
* public void actionPerformed(ActionEvent e)
* public class MyTableModel extends AbstractTableModel
* public MyTableModel(int rows)
* public int getColumnCount()
* public int getRowCount()
* public String getColumnName(int col)
* public Object getValueAt(int row, int col)
* public void setValueAt(Object value, int row, int col)
* public static void main(String[] args)

Tunnel

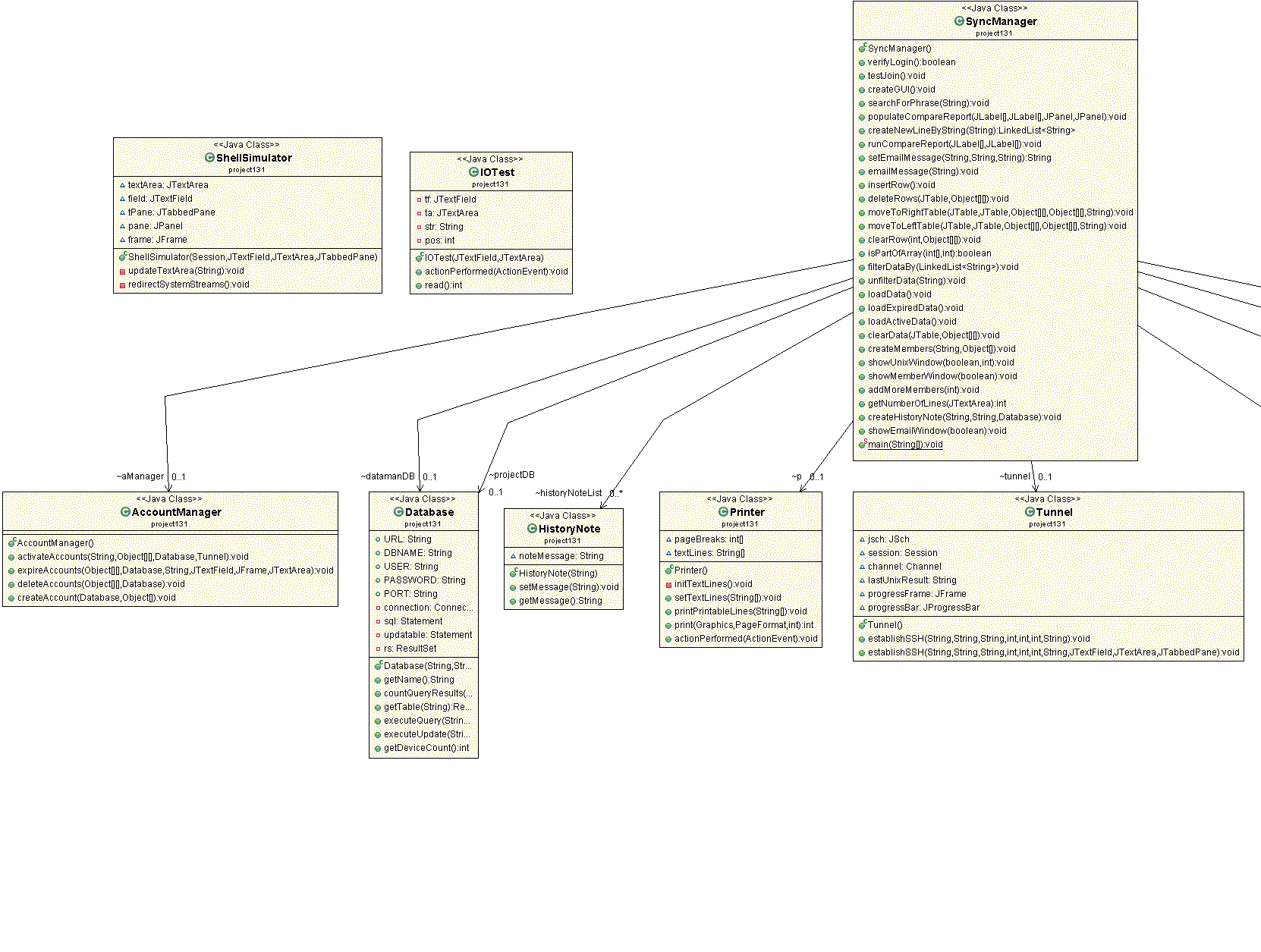
* public void runUnixCommand(String cmd)
* public String getLastUnixResults()
* public void establishSSH(String sshUser, String sshPassword, String sshHost, int sshPort, int localPort, int remotePort, String dbHost)

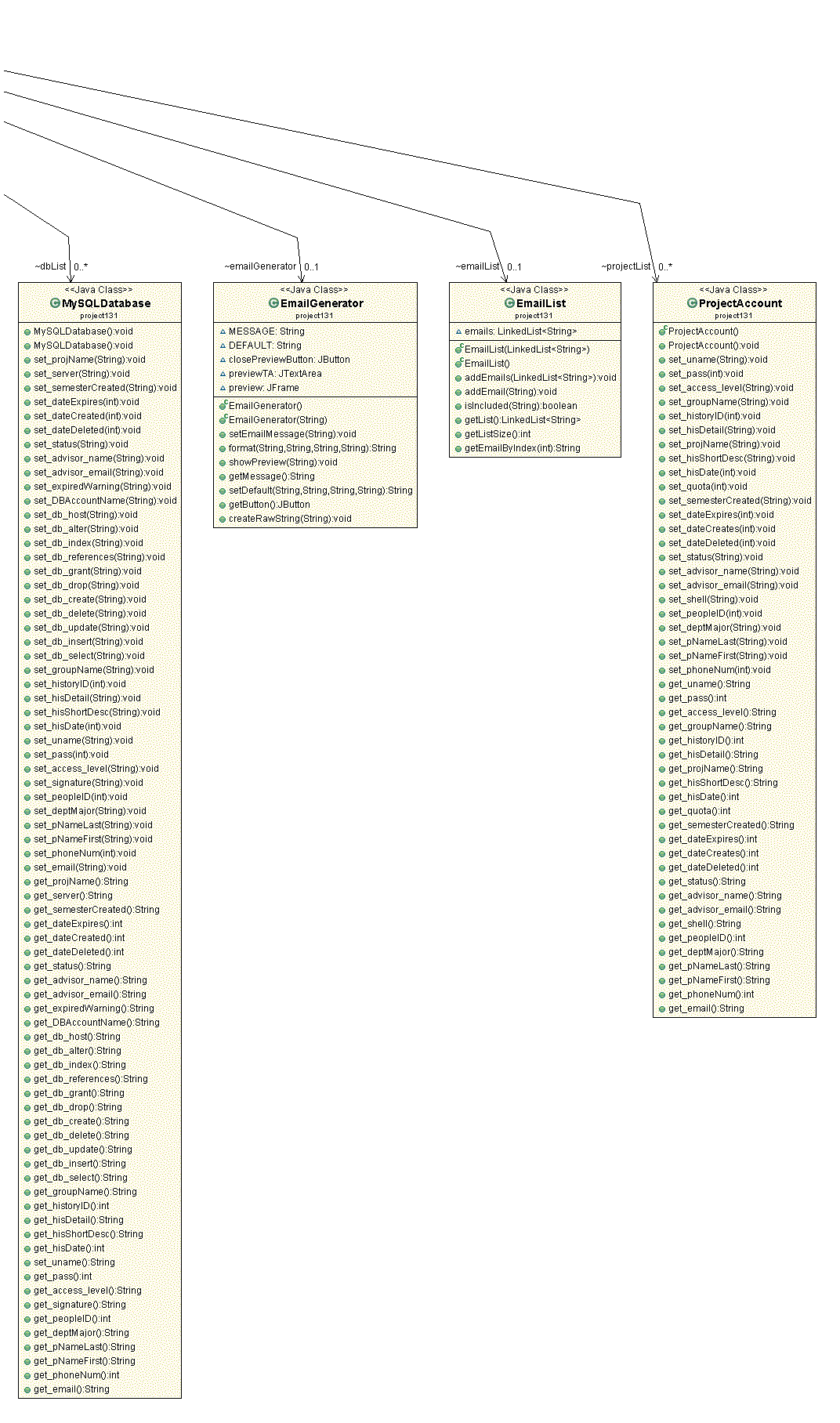
**Expanded and Updated Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item #** | **Name** | **Type** | **Variable**  **Name** | **Size** | **Description** |
| 1 | select account | dropdown | select\_account | 8 options | user selects the account that they want to apply for (i.e. project, student, faculty, MySQL) |
| 2 | Part 2 label | label | lbl | N/A | label that asks the user if they want a MySQL database that becomes visible if user selects project account and vice-versa |
| 3 | Yes or No dropdown | dropdown | yesORno | 2 options | dropdown menu with ‘yes’ or ‘no’ for choosing to apply to both an account and database |
| 4 | ECS email address | text | validation\_email | 30 | a valid ECS email address to confirm the applicant may request accounts and/or databases |
| 5 | account name | text | account\_name | 3-8 letters only | name for the account and/or database |
| 6 | disk quota | int | quota | 4 digits | amount of disk space requested |
| 7 | advisor name | text | advisor\_name | 30 | name of advisor |
| 8 | advisor email | text | advisor\_email | 50 | contact email address of advisor |
| 9 | project shell | dropdown | shell | 5 options | the preferred shell for the account |
| 10 | number of participants | int | num\_students | 2 digits | number of people using the account and/or database |
| 11 | major/department | dropdown | major | 9 options | applicant chooses the major or department that the account will be created under |
| 12 | group | dropdown | group | 11 options | applicant selects what group the account is related to. Student is default |
| 13 | number of semesters | dropdown | num\_semesters | 3 options | 3rd option enables and sends focus to num\_semesters\_other |
| 14 | number of semesters other | int | num\_semesters\_other | 2 digits | for inputting more than 2 semesters of use |
| 15 | host location | radio button | mysql\_host\_location | 2 options | account will be housed on this server. local or remote |
| 16 | permissions | radio button | permissions\_radio | 3 options | all, standard, or other permissions |
| 17 | permissions | array of checkboxes | mysql\_permissions[] | 9 | holds all the selected permissions |
| 18 | comments | text | comments | 500 | comments made during application by applicant |
| 19 | Student First Name | String | fName# | 1 - 15 Characters | the first name of each student involved |
| 20 | Student Last Name | String | lName# | 1 - 15 Characters | the last name of each student involved |
| 21 | Student Phone Number | String | phone# | 10 digits | the phone number of each student involved |
| 22 | Student Email | String | email# | 4 - 20 Characters | the contact email address of each student involved |
| 23 | Student’s major | dropdown | major# | 9 options | to get each student’s individual majors |
| 24 | host location (php) | string | $host | N/A | converts ‘%’ to ‘remote’ and ‘localhost’ to ‘local’ for correct inserting |
| 25 | SQL permissions (php) | array | $permissions | N/A | the 9 different checkboxes |
| 26 | expire | date | $expire | N/A | date that the account should expire |
| 27 | year | date | $year | N/A | used to set correct year of estimated expiration |
| 28 | today | date | $todayDate | N/A | date of submission |

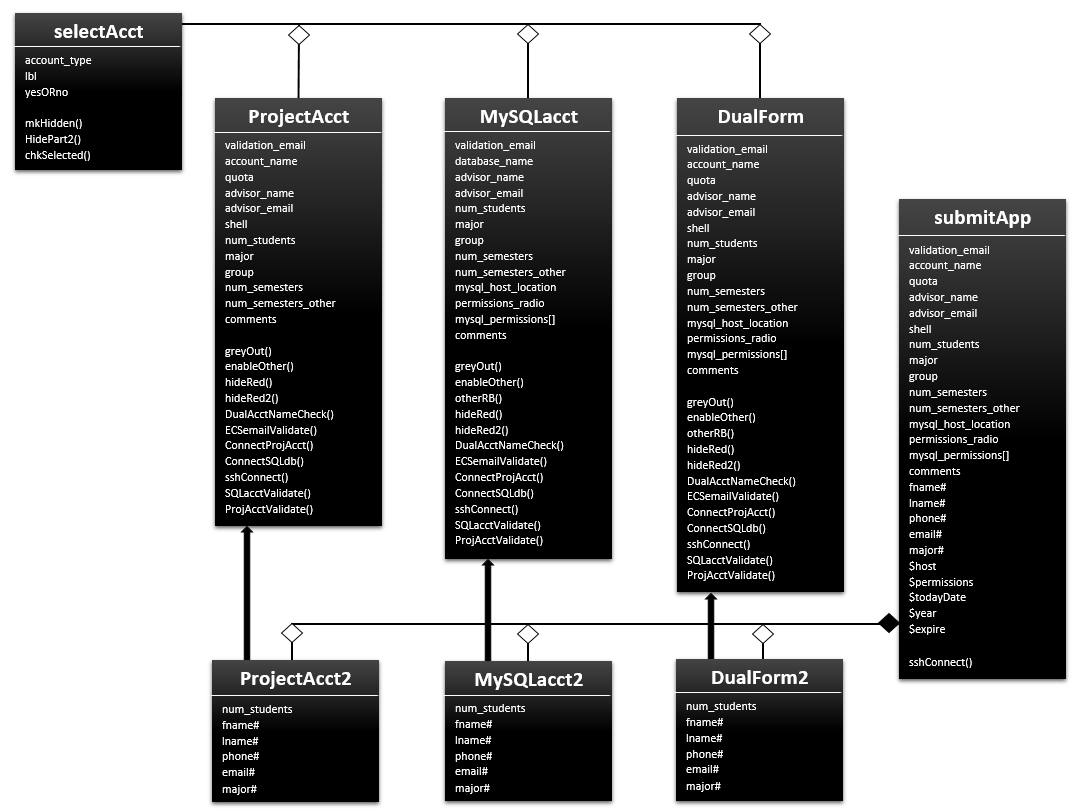
1. **ARCHITECTURE DESIGN**

***Backend Class Diagram***





***Front End Class Diagram***



* + **Description of Class Diagrams**
    - **Backend:** The backend of our application is utilized and displayed as a top-down approach. The SyncManager class is the main class that instantiates and calls upon all the helper methods. Indicated by the class diagram, the SyncManager class calls upon each lower class to do different functions. For example, SyncManager will instantiate and call upon an instance of AccountManager to deal with the Project Accounts and the MySQL Accounts. SyncManager will also instantiate a Database class object to deal with database data and run the database SQL queries. The SyncManager class points to each of the lesser classes to perform different functions that help it out.
    - **Front End:** Our webpage uses a combination of top down and horizontal partitioning. The user chooses where the website will take them based on their needs. All the webpages maintain a similar look and feel and each one utilizes slightly different versions of similar functions. If the user wants a Project account, they choose it from the dropdown and click submit. This redirects them to the ProjectAcct.php webpage, which is a sister page of MySQLacct.php and DualForm.php. After filling out the form, the applicant is sent to the correct submittal webpage and their application is processed by being inserted into the correct databases on the server. There are many forms and pages one can go to from the selectAcct class, hence the three open diamonds and lack of a solid diamond. Those three forms are only some of the classes that the selectAcct class can take an applicant to. Each of those are parents to the ones below them. Finally, submitApp requires at least one of the three form number 2s.

1. **DETAILED CLASS DESIGN**

***BACKEND***

class SyncManager{ //Most important class!

createGUI(){

* + - Instantiates and creates the GUI
    - GUI window is separated into four tabs that represents a function that will be made by the user
      * Creates necessary GUI components appropriate to each tab
      * Activate Accounts, Expire Accounts, Create Accounts, and Compare/Sync Accounts
    - Checks for correct login credentials
    - if login is correct, create a Tunnel
      * Tunnel establishes an SSH connection to the server
    - else kill application
    - Create a Database and a Database connection
    - Create the EmailGenerator
    - Create the AccountComparer
    - Create the HistoryNote Editor and create HistoryNote Window
    - Create the EmailList Class and the EmailList window

}

loadData(){

* + - Calls upon the Database to access its data
    - Loads all the account information and data into each Table within each Tab

}

// Show different windows....

showWindows(){

* + - Show specific windows when needed or when appropriate.
    - Email window, unix window, member window... etc.

}

createHistoryNote(){

* + - Creates a new instance of HistoryNote and puts its information into the database

}

class ClickListener{

actionPerformed(){

* + - Tells the program that something has been clicked

}

class KeyListener{

keyPressed(){

* + - Tells the program that something has been typed

}

class Tablistener{

stateChanged(){

* + - Tells the program that the user has changed tabs

}

class AccountComparer{

compareAccounts(AccountData1, AccountData2){

* + - Compares AccountData1 against AccountData2 and checks whether they are synced
    - Shows in green the accounts that are properly synced across Database and Server
    - Shows in red the accounts that are NOT synced across Database and Server

}

}

class AccountManager{

createsAccount(Account information);

expiresAccount(Account information);

activateAccount(Account information);

deleteAccoutn(Account information);

}

class Database{

connect(Host, Password, Port);

executeQuery(Query);

countQueryResults();

}

class EmailGenerator{

createEmail(message);

formatEmail(message);

}

class HistoryNote{

// this is a history note that displays information about a history item and the info goes into the databases

}

class MySQLDatabase{

// this is a class representation of a MySQLDatabase account

// this class stores all the data about a MySQLDatabase

}

class ProjectAccount{

// this is a class representation of a ProjectAccount

// this class stores all the data about a ProjectAccount

}

class ShellSimulator{

// Creates a simulation of a unix shell

// Run lives commands onto this shell

}

class Tunnel{

// Creates a Tunnel that facilitates an SSH connection between Java and Unix Servers

}

***FRONT END***

* ECSaccts.htm
  + user clicks account and/or database link
    - redirect to selectAcct.htm
* selectAcct.htm
  + on form load, hide the “Do you also want a MySQL/Project” (Part2) label and dropdown
  + on selection from the account drop down menu
    - if (account\_type == “Project”), show the Part2 label and dropdown,
    - else if (account\_type == “MySQL”, show the Part2 label and dropdown
    - else hide them
  + onsubmit()
    - if account selected == “Project” and yesORno == “Yes”)
      * redirect to DualForm.php
    - else if Project and mysql == “No”
      * redirect to ProjectAcct.php
    - else if account selected == “MySQL” and yesORno == “Yes”
      * redirect to DualForm.php
    - else if account selected == “MySQL” and yesORno == “Yes”
      * redirect to MySQLacct.php
* DualForm.php
  + on form load
    - disable num\_semesters\_other so user cannot have One or Two semesters selected as well as a number in that field. It’s “One”, “Two”, OR “Other”, not AND
    - it gets enabled and focused upon when user selects “Other” from num\_semesters dropdown
  + validate email
    - connects to server (ECSemailValidate.php)
    - runs ypcat passwd | grep <username>
    - if result = “”, username not found, display “Invalid Email!”
      * remove “Invalid Email!” text when users changes field
  + validate account name
    - check for account name (AcctNameCheck.js)
      * connect to Project database (ConnectProjAcct.php)
      * check for duplicate account name (ProjAcctValidate.php)
        + iterate through account\_name column and compare strings
        + if duplicate found, display “Invalid account name!”

remove “Invalid Email!” text when users changes field

* + - * connect to Dataman database (ConnectMySQL.php)
      * check for duplicate account name (MySQLacctValidate.php)
        + iterate through database\_name column and compare strings
        + if duplicate found, display “Invalid account name!”

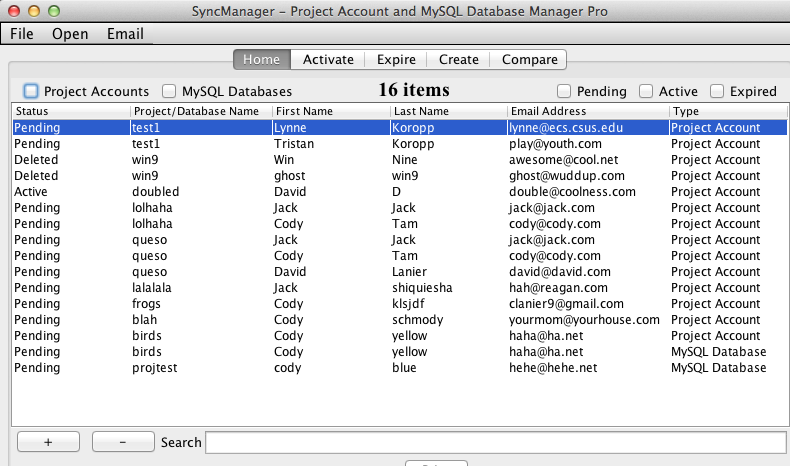
remove “Invalid Email!” text when users changes field

* + if num\_semesters == “Other”
    - enableOther()
      * enables the num\_semesters\_other text field and gives it focus
  + onsubmit()
    - verifyFormInfo()
      * UserAccount.js
* DualForm2.php
  + for loop to populate HTML text fields for each student
  + onsubmit()
    - verifyFormInfo()
      * UserAccount.js
* submitApp.php
  + connect to project database
  + insert project info
  + insert student info
  + connect to dataman database
  + insert database info
  + insert student info
  + generate text for email
    - info from application
    - for loop of student info
    - send to admin
  + redirect automatically to ECS Webpage

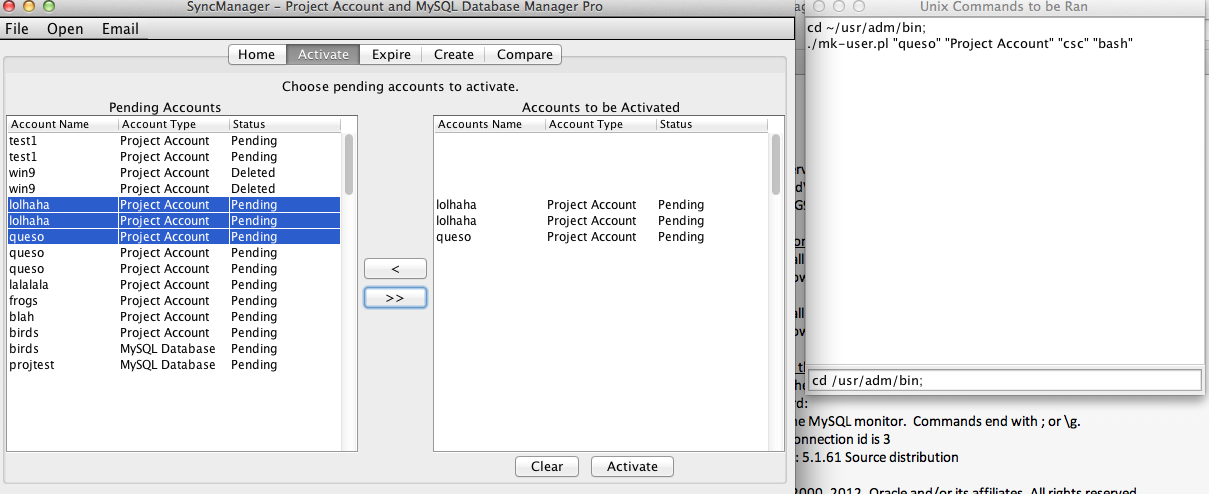
1. **USER INTERFACE DESIGN**

***BACKEND***

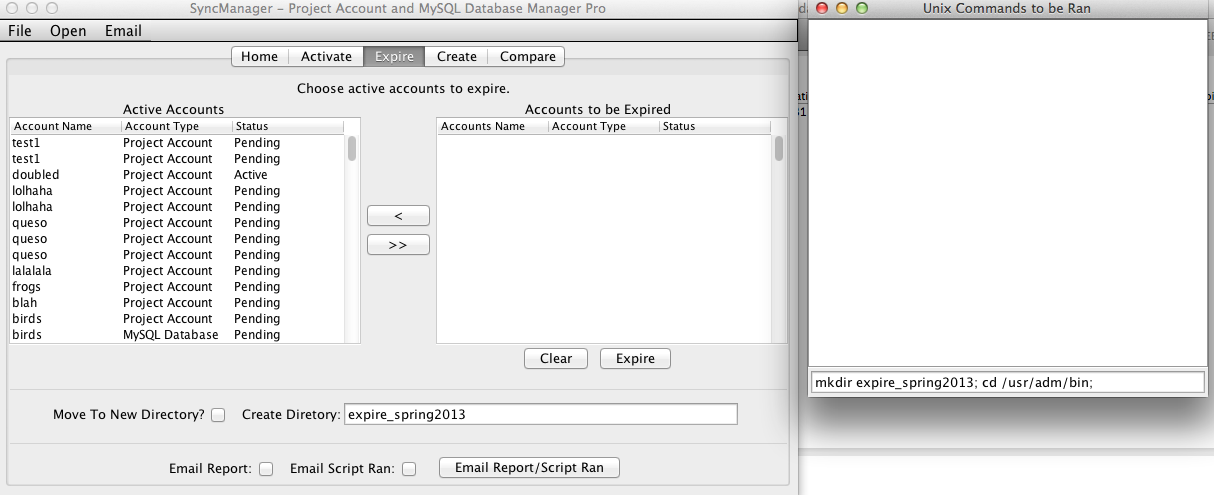
**Home Screen - See outstanding account information**



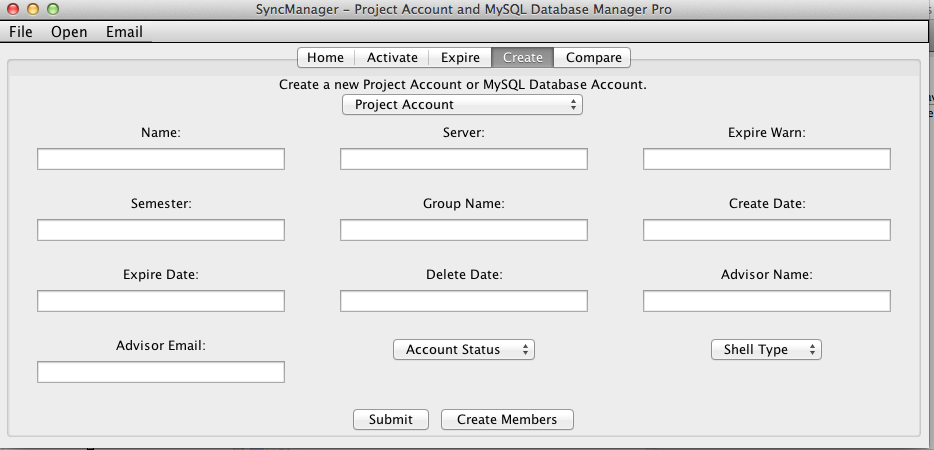
**Activate Accounts - Change “Pending” Accounts to Active**



**Expire Accounts - Change Expired Accounts to Deleted**



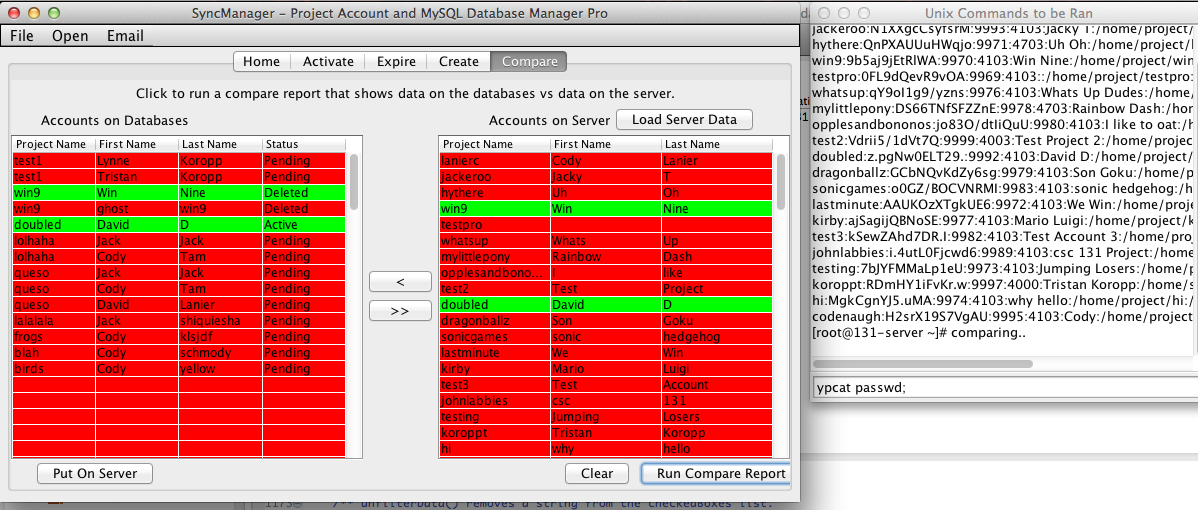
**Create Project/MySQL Accounts - Create Project Account or MySQL Database Account**



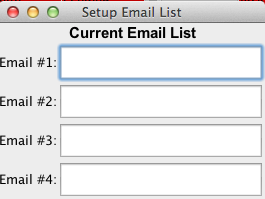
**Add Members - Add Members to the corresponding Project Account or MySQL Account**



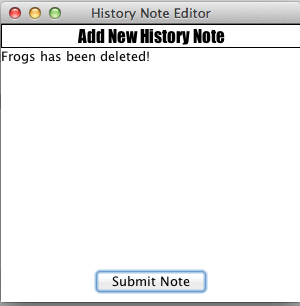
**Compare Accounts - Compare MySQL Accounts or Project Accounts such that they are synced. Synced, meaning every account that is active must be on the server. And every account that is expired must be removed from the server.**



**Setup the Email List - Edit who will be emailed whenever desired**

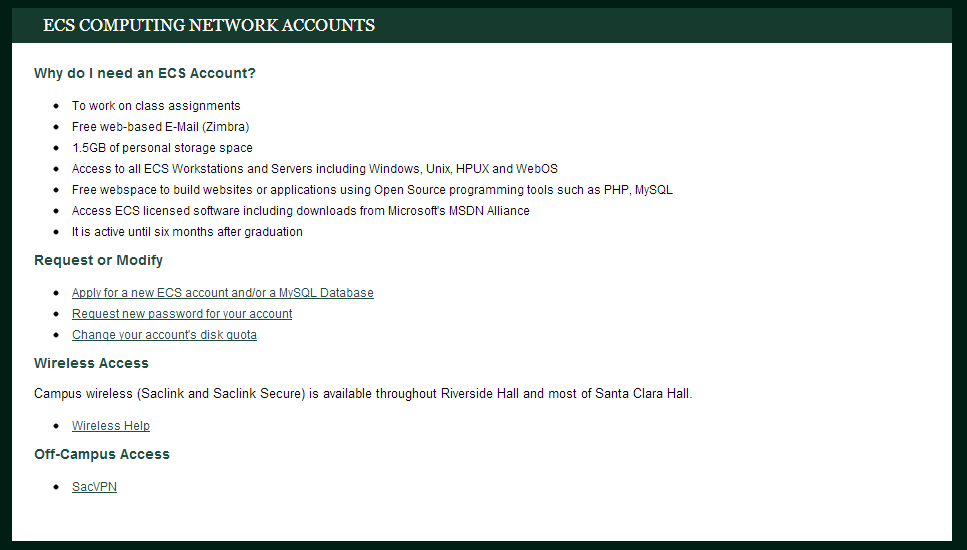


**Add History Notes - Whenever something has been done inside the application, the user can create a history note where its information is stored in the database.**

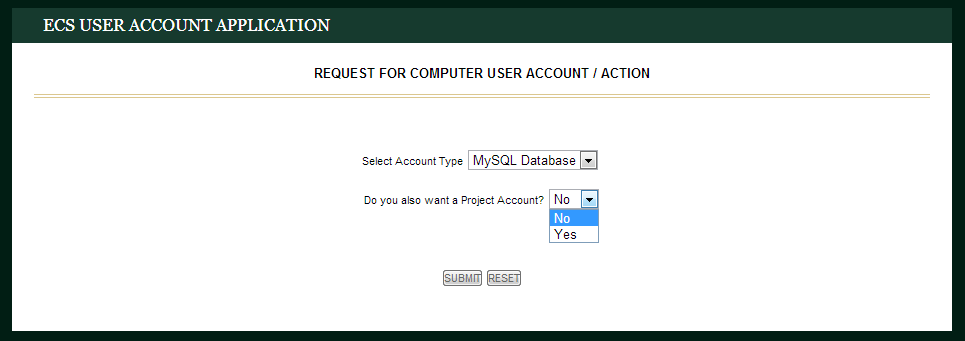


***FRONT END***

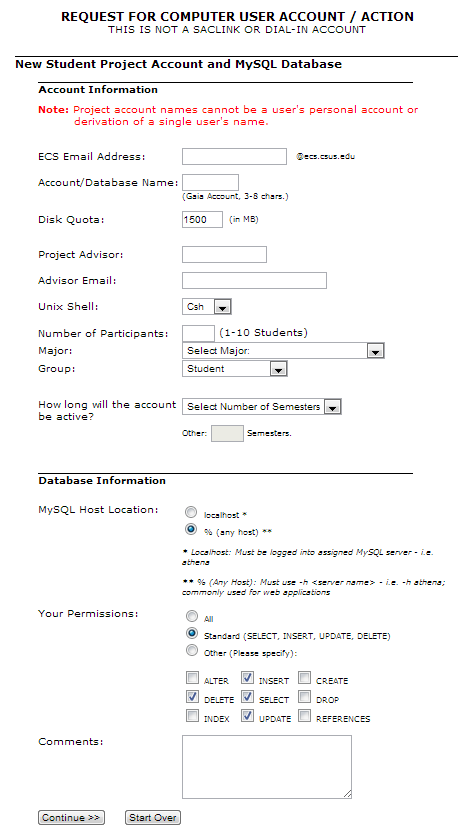
**The Combined Links**



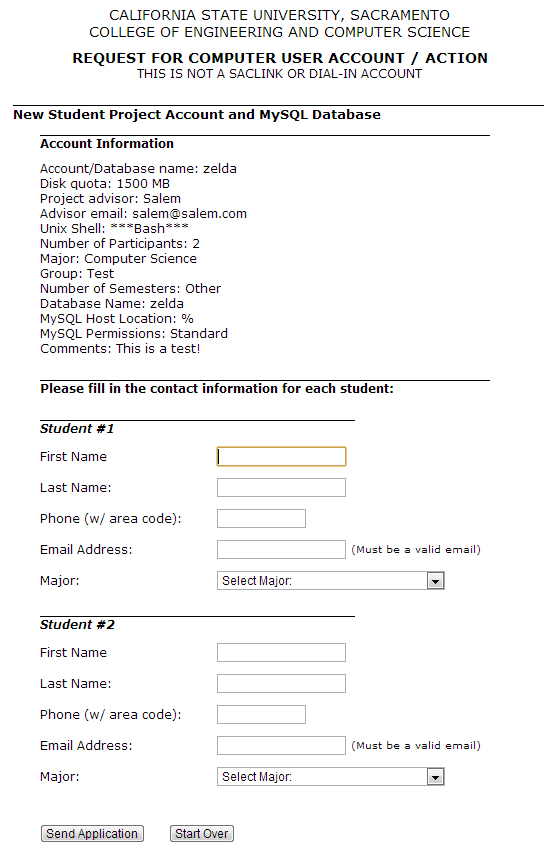
**Selecting Accounts**



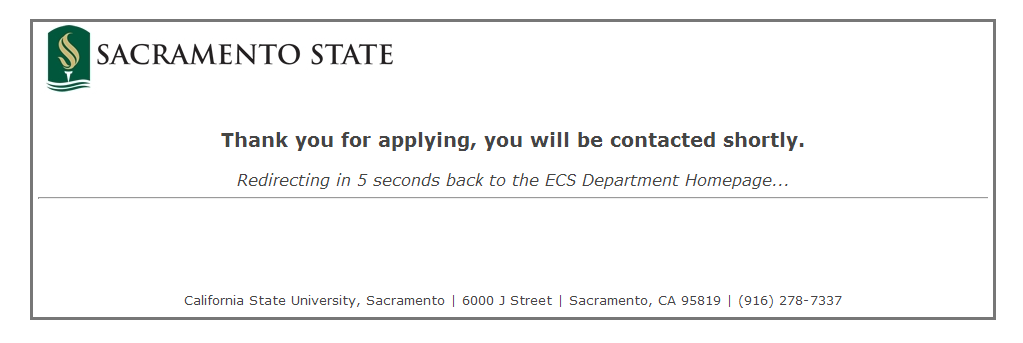
**The Main Application Form**



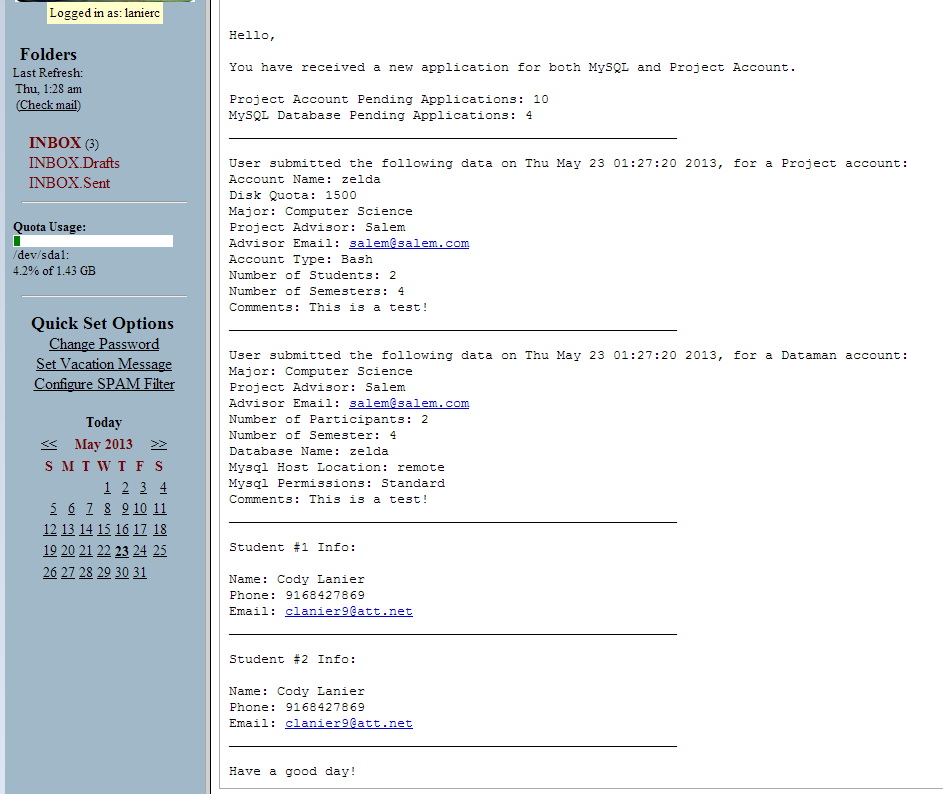
**Student Contact Info Form**



**Submit and Redirect Page (insert into dataman and project happen here)**



**Admin email received after application submitted**



**PROGRAMMING TOOLS**

**Back End** - Java, JDBC (Java External Library), JSCH (Java External Library)

**Front End**- PHP, LAMP, CSS, HTML, JQuery, AJAX, JavaScript

**Other Tools**: Eclipse, Dropbox, MAMP, WAMP, SACVPN, Notepad++

1. **RESTRICTIONS, LIMITATIONS, AND CONSTRAINTS**

N/A

1. **REFERENCES OR RESOURCES USED**

www.w3schools.com

www.php.net

www.jcraft.com/jsch/

www.mysql.com

www.youtube.com

1. **TEAM MEMBER’S ROLES AND APPROVAL**
   * **Team member’s roles in SDD**

John - 1, 2, 3d

David - 7, 6i, 5i, 4ai, 4bi, 3d, 3f

Jack - 3a, 3b, 3c

Cody - 3d, 3e, 3f, 4a, 4b, 5, 6

* + **Team member’s signatures and date**

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

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

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

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

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