**Project Design Document**

Group 9:

Anton Alexander, Eric Monk, Jay P. Boring, Sallie Kihu

University of Maryland University College

**Author Note:**

Revision: 4.0

Date: 02/08/2015

Class #: CMSC 495 6380

**Revision History**

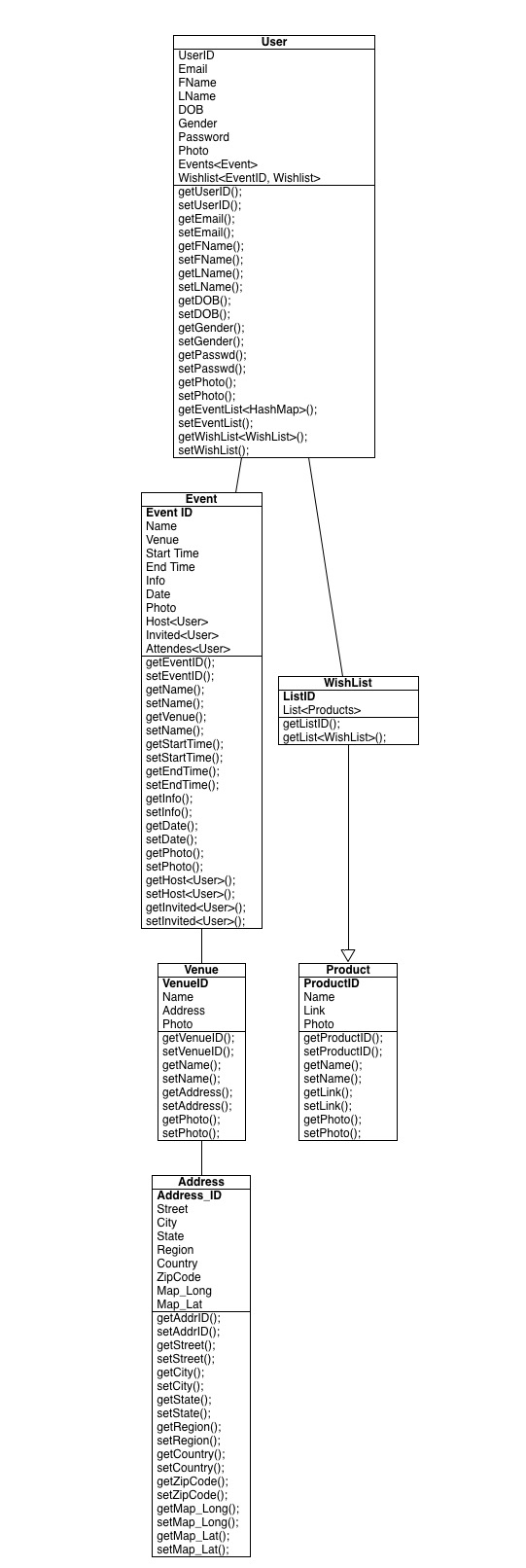
|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Name of editor** | **Description** |
| 1 | 02/04/15 | Jay Boring | Original Document |
| 2 | 2/8/15 | Sallie Kihu | Trace Diagrams, Event Class Pseudocode |
| 3 | 2/8/15 | Anton Alexander | Class Diagrams |
| 4 | 2/8/15 | Anton Alexander, Eric Monk, Jay Boring | Psuedocode update, UML diagrams, Final Edits, updates |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Design**

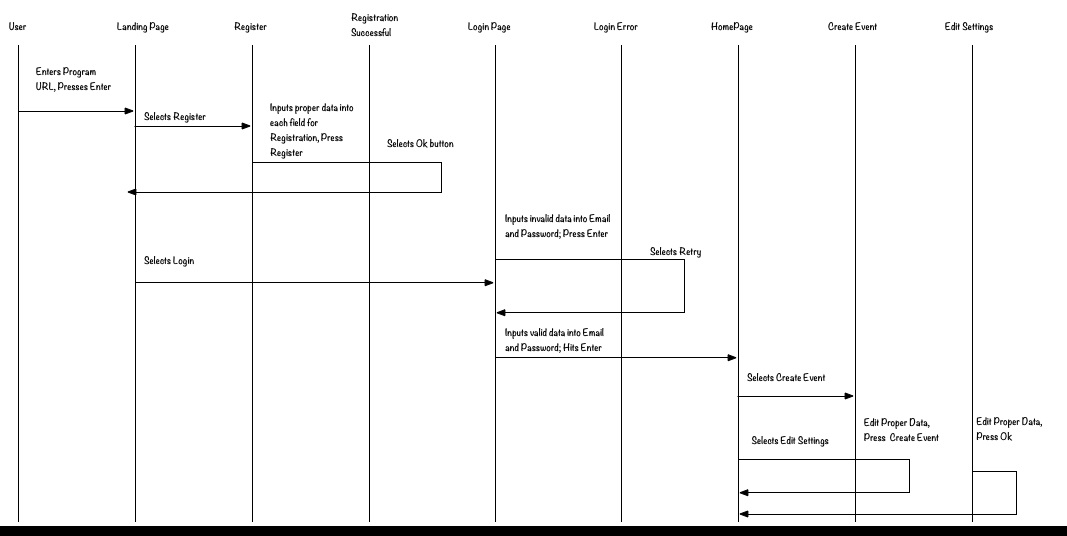
The project is to create an application that keeps track of your loved ones special dates (e.g. anniversaries, birthdays, etc.) and their wish lists.

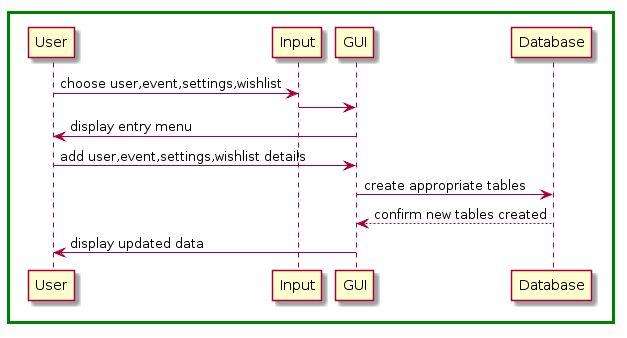
From the project analysis, we proceed with the following design:

**Class Diagram**

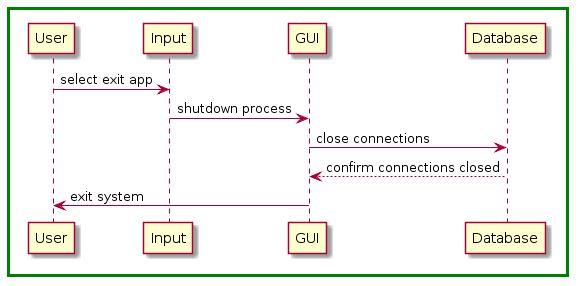


**Event Trace Diagram**

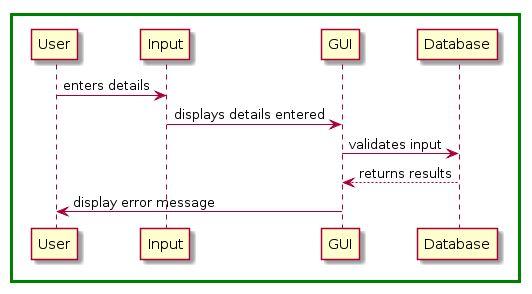


**Normal Operation Diagram**

**Shut-down Diagram**



**Error-Handling Diagram**



**Event Class Pseudocode:**

public class GUIprogram {

LinkableHashMap <email, User> myUsers = new LinkableHashMap <email, User>();

LinkableHashMap <email, Events> myEvents = new LinkableHashMap <email, Events>();

LinkableHashMap <email, Wishlist> myWishlist = new LinkableHashMap <email, Wishlist>();

void main (){

GUIprogram program = new GUIprogram();

program.createGUI();

}

CreateGUI(){

// Create instance of main class

GUILanding landing = new GUIlanding();

// Display the landing page

landing.display();

// Listener for login button

actionListener.Login(){

// Remove landing frame

landing.nodisplay();

GUILogin login = new GUILonging ();

login.display();

Boolean login = false;

// Loop until correct user info is entered

while (login == false){

// Display login frame

// Lister to login button

actionListener.login(){

// If the user is in the system exit loop, else dislay error and loop

if (myUsers.contains(email)){

...

//create instance of homepage and display

GUIhomepage homepage = new GUIhomepage ();

homepage.display();

// Listen if user clicks create event

actionListener.createvent{

GUIcreateEvent creatEvent = new GUIcreateEvent();

//Register listener

actionListener.create{

int dataValidate = validate(creatEvent.data);

if (dataValidate==0){

User newUser = new User(register.data);

myUsers.add(newUser);

JOptionPane.("Succes event created");

//return to login screen

else if (dataValidate ==1){

JOptionPane.("Invalid inputs");

}

else {

JOptionPane.("event already in database");

}

}

};

// Cancel button listener

actionListener.cancel{

System.exit(0);

};

};

// Listens if user clicks edit settings

actionListener.editSettings{

// create instance of register frame

GUIeditSettings editSettings = new GUIeditSettings ();

editSettings.display();

//Register listener

actionListener.update{

int dataValidate = validate(editSettings.data);

if (dataValidate==0){

User newUser = new User(editSettings.data);

myUsers.add(newUser);

JOptionPane.("Succes user updated");

//return to login screen

else if (dataValidate ==1){

JOptionPane.("Invalid inputs");

}

else {

JOptionPane.("no updates");

}

};

// Cancel button listener

actionListener.cancel{

System.exit(0);

};

}

};

// listens if user clicks event on JTable

actionListener.clickEvent{

GUIcreateEvent event = new GUIcreateEvent();

event.display();

actionListen.editEvent{

GUIcreateEvent editEvent = new GUIcreateEvent();

//Register listener

actionListener.edit{

int dataValidate = validate(editEvent.data);

if (dataValidate==0){

User newUser = new User(register.data);

myUsers.add(newUser);

JOptionPane.("Success event updated");

}

else if (dataValidate ==1){

JOptionPane.("Invalid inputs");

}

else {

JOptionPane.("no updates");

}

};

// Cancel button listener

actionListener.cancel{

System.exit(0);

};

};

};

}

else{

// display error redraw screen

JOptionPane.display ("Invalid Input")

login.reDraw();

}

};

// Listen to canel button and exit system

actionListener.cancel(){

System.exit(0);

};

}

actionListener.Register(){

// remove login frame

login.noDisplay();

// create instance of register frame

GUIRegister register = new GUIRegister ();

register.display();

//Register listener

actionListener.register{

int dataValidate = validate(register.data);

if (dataValidate==0){

User newUser = new User(register.data);

myUsers.add(newUser);

JOptionPane.("Succes user created");

//return to login screen

else if (dataValidate ==1){

JOptionPane.("Invalid inputs");

}

else {

JOptionPane.("user already in database");

}

};

// Cancel button listener

actionListener.cancel{

System.exit(0);

};

}

}

}

**Unresolved risks**

Risk 1: Password cracking due to brute force attacks and session hijacking

Mitigation: Password Encryption; Making passwords not visible when typing in for user.

Risk 2: Interception of transmitted data; including personal data such as birthdays and anniversary dates.

Mitigation: Encrypted data during transmission and storage.

.