**The British College**

**KATHMANDU**

**Coursework Submission Coversheet**

(individual coursework only)

**Faculty of Arts, Environment and Technology**

**LBU Student Id:**

|  |
| --- |
| c7210593 |

**For checking by the student:**

|  |  |  |  |
| --- | --- | --- | --- |
| Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.  Award name: Bsc Hons in Computing  Module code:  Module name: Advance Software Engineering A (ASE)  Module run: 2019    Coursework title: Bug Tracking System  Due Date: 17/5/2019    Module leader: (In LBU): Duncan Mullier  Module tutor: (In TBC): Resham Pun    **TURNITIN** Checked: YES NO ***(please circle)***    Submission date& time: Date: 17/5/2019 Time: Before noon     |  |  |  | | --- | --- | --- | |  | **Total Number of Pages (including this front sheet):** |  |   **Total Word Count:**  **In submitting this form with your assignment, you make the following declaration:**  I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.  Signed: Date:  **You are strongly advised to retain a second copy of your work in case of any query about the assignment.** |

**For completion by the faculty:**

**This mark is provisional and subject to moderation and approval by**

**the relevant**

**examining board**

**T**

**eacher's Feedback**

**Teacher's Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE BRITSH COLLEGE**

SUBMITTED BY: SABINA BHANDARI

L6(Bsc.Hons)

5/17/2019

SUBMITTED TO: RESHAM PUN

Module Tutor

**Introduction:**

Bug Tracking Application is used to aid the software development process. It is used for keeping track of reported software bugs in software development projects along with their solution. It is also known as a type of issue tracking system. Those system which are used by most open source software projects only allow end users for entering bug reports directly by various bug tracking application.

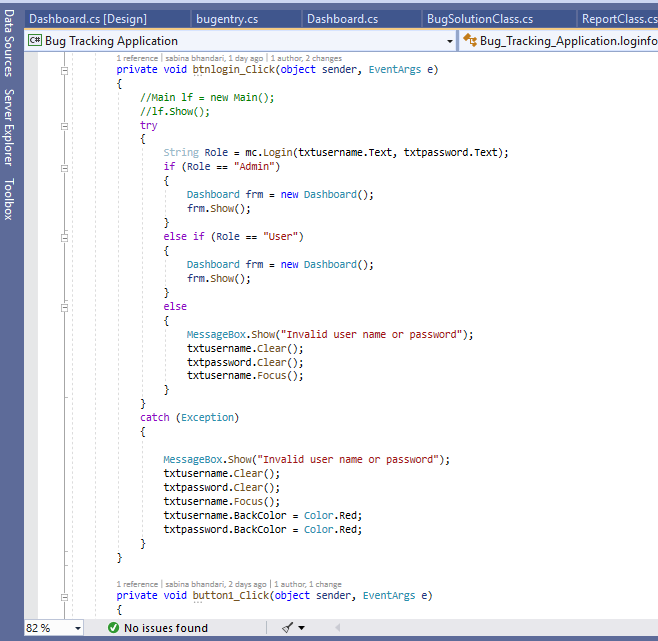
**Bug Tracking Application:**

**Login Page:**

In this application, login forms help to enter to dashboard of system for adding all the information.



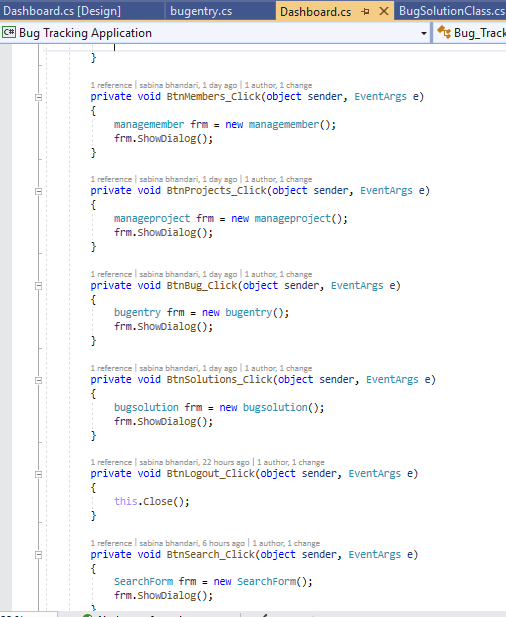
**Code of Login Form**



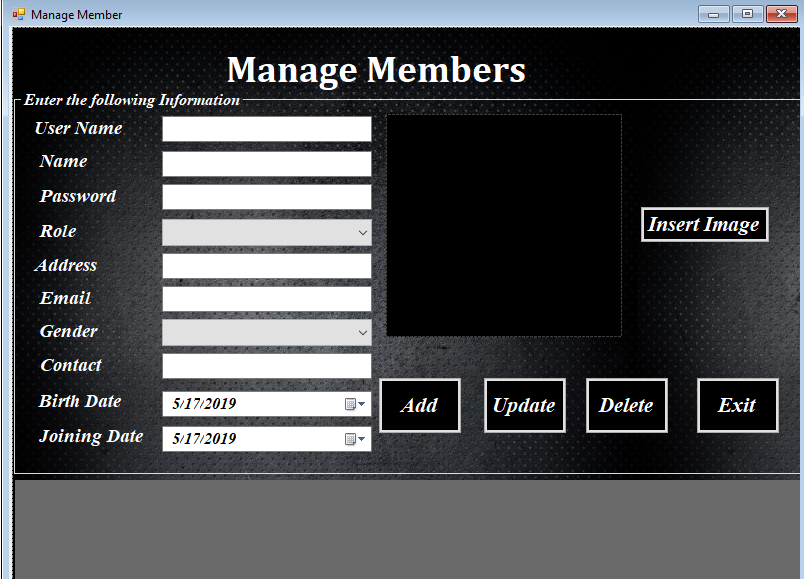
**Dashboard:**



**Code for Dashboard:**



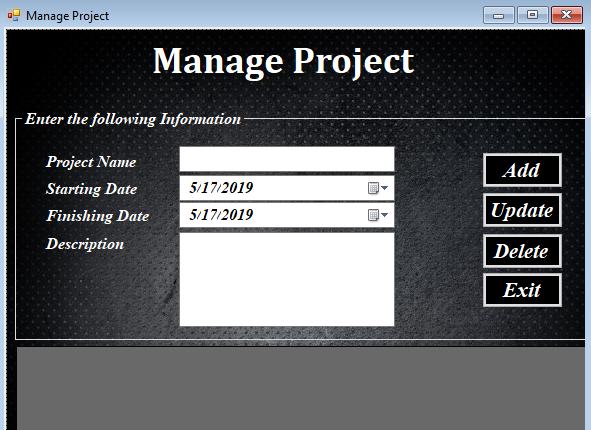
**Manage member:**



**Code for manage member:**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using DataAccessLayer;  using BusinessLogicLayer;  using Microsoft.VisualBasic;  using System.Text.RegularExpressions;  using System.IO;  namespace Bug\_Tracking\_Application  {  public partial class managemember : Form  {  public managemember()  {  InitializeComponent();  txtcontact.MaxLength = 10;  }  //acccessing data from various classes  BusinessLogicClass blc = new BusinessLogicClass();  MemberClass mc = new MemberClass();  HelperClass hc = new HelperClass();  public int MemberId;  //close the window  private void btnexit\_Click(object sender, EventArgs e)  {  this.Close();  }  //retrieve the data on DataGridView  private void managemember\_Load(object sender, EventArgs e)  {  dgvmembers.DataSource = mc.GetAllUsers();  }  //adding the data to display on datagridview and store to database  private void AddButton\_Click(object sender, EventArgs e)  {  if (txtusername.Text == "")  {  MessageBox.Show("Provide Username: Full information required");  }  if (txtmembername.Text == "")  {  MessageBox.Show("Provide Member Name: Full information required");  }  if (txtpassword.Text == "")  {  MessageBox.Show("Provide Password: Full information required");  }  if (cmbrole.SelectedIndex == -1)  {  MessageBox.Show("Provide Role: Full information required");  }  if (txtaddress.Text == "")  {  MessageBox.Show("Provide Address: Full information required");  }  if (txtemail.Text == "")  {  MessageBox.Show("Provide Email: Full information required");  }  if (cmbgender.SelectedIndex == -1)  {  MessageBox.Show("Provide Gender: Full information required");  }  if (txtcontact.Text == "")  {  MessageBox.Show("Provide Contact: Full information required");  }  if (dtpbirthdate.Text == "")  {  MessageBox.Show("Provide Birth Date: Full information required");  }  if (dtpjoiningdate.Text == "")  {  MessageBox.Show("Provide Join Date: Full information required");  }  if (btnbrowse.Text == "")  {  MessageBox.Show("Provide Image: Full information required");  }  else if (DublicateUser() == true)  {  MessageBox.Show("Member with same name already exists");  txtusername.Clear();  txtusername.Focus();  }  { CreateUser(); }  }  //create user  private void CreateUser()  {  //try catch exception  try  {  bool res = blc.MemberTable(0,  txtusername.Text,  txtmembername.Text,  txtpassword.Text,  cmbrole.Text,  txtaddress.Text,  txtemail.Text,  cmbgender.Text,  Convert.ToInt32 (txtcontact.Text),  Convert.ToDateTime (dtpbirthdate.Text),  Convert.ToDateTime(dtpjoiningdate.Text),  HelperClass.imageConverter(picmembers),  1);  if (res == true)  {  //display message of adding memeber in database  MessageBox.Show("Success to Add member");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  else  {  //display error message as data cannot be stored  MessageBox.Show("Couldn't Add selected member");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //helps in data store as if users have same information  public bool DublicateUser()  {  int x = 0;  try  {    for (int i = 0;i<dgvmembers.Rows.Count;i++)  {  if (txtusername.Text == dgvmembers.Rows[i].Cells["UserName"].Value.ToString())  x = 1;  }    }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  if (x == 1)  return true;  else  return false;  }  //Browse image in button click  private void btnbrowse\_Click(object sender, EventArgs e)  {  try  {  try  {  OpenFileDialog ofd = new OpenFileDialog();  if (ofd.ShowDialog() == DialogResult.OK)  {  picmembers.Image = Image.FromFile(ofd.FileName);  }  else  {  MessageBox.Show("Please select a profile picture");  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrive all data from datagridview to the details entry section on a single click  private void dgvmembers\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  try  {  MemberId = Convert.ToInt32(dgvmembers.SelectedRows[0].Cells["MemberId"].Value.ToString());  txtusername.Text = dgvmembers.SelectedRows[0].Cells["UserName"].Value.ToString();  txtmembername.Text = dgvmembers.SelectedRows[0].Cells["Name"].Value.ToString();  txtpassword.Text = dgvmembers.SelectedRows[0].Cells["Password"].Value.ToString();  cmbrole.Text = dgvmembers.SelectedRows[0].Cells["Role"].Value.ToString();  txtaddress.Text = dgvmembers.SelectedRows[0].Cells["Address"].Value.ToString();  txtemail.Text = dgvmembers.SelectedRows[0].Cells["Email"].Value.ToString();  cmbgender.Text = dgvmembers.SelectedRows[0].Cells["Gender"].Value.ToString();  txtcontact.Text = dgvmembers.SelectedRows[0].Cells["Contact"].Value.ToString();  dtpbirthdate.Text = dgvmembers.SelectedRows[0].Cells["DOB"].Value.ToString();  dtpjoiningdate.Text = dgvmembers.SelectedRows[0].Cells["DOJ"].Value.ToString();  MemoryStream memoryStream = new MemoryStream((byte[])dgvmembers.SelectedRows[0].Cells["Image"].Value);  picmembers.Image = Image.FromStream(memoryStream);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //update the date entered into the database  private void btnupdate\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.MemberTable(MemberId,  txtusername.Text,  txtmembername.Text,  txtpassword.Text,  cmbrole.Text,  txtaddress.Text,  txtemail.Text,  cmbgender.Text,  Convert.ToInt32(txtcontact.Text),  Convert.ToDateTime(dtpbirthdate.Text),  Convert.ToDateTime(dtpjoiningdate.Text),  HelperClass.imageConverter(picmembers),  2);  if (res == true)  {  MessageBox.Show("Success to Update Member");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  else  {  //display error message as data cannot be updated  MessageBox.Show("Couldn't Update selected member");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // delete the data entered into the database  private void btndelete\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.MemberTable(MemberId,  txtusername.Text,  txtmembername.Text,  txtpassword.Text,  cmbrole.Text,  txtaddress.Text,  txtemail.Text,  cmbgender.Text,  Convert.ToInt32(txtcontact.Text),  Convert.ToDateTime(dtpbirthdate.Text),  Convert.ToDateTime(dtpjoiningdate.Text),  HelperClass.imageConverter(picmembers),  3);  if (res == true)  {  //display message of successfully deleted  MessageBox.Show("Success to Delete Member");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  else  {  //display error message as data cannot be deleted  MessageBox.Show("Couldn't delete selected memeber");  dgvmembers.DataSource = mc.GetAllUsers();  HelperClass.makeFieldsBlank(grpContainer);  picmembers.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private void Txtemail\_TextChanged(object sender, EventArgs e)  {  Regex regex = new Regex(@"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$");  bool isValid = regex.IsMatch(txtemail.Text.Trim());  if (!isValid==false)  {  MessageBox.Show("Invalid Email.");  }  }  }  } |

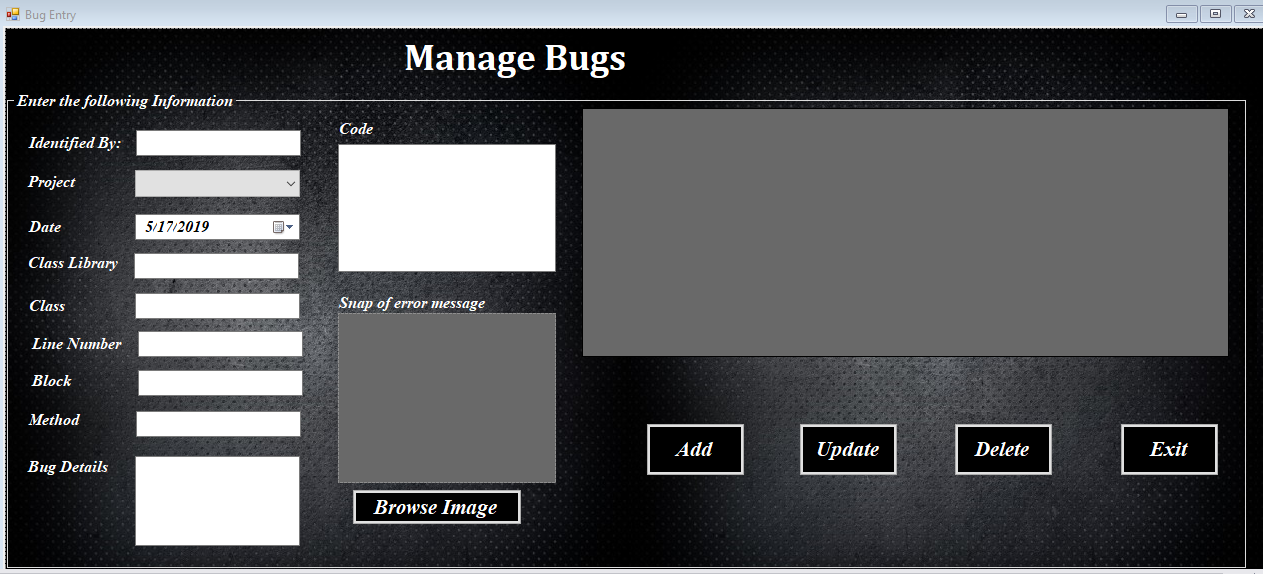
**Manage Project:**



**Code for manage project:**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using BusinessLogicLayer;  using DataAccessLayer;  namespace Bug\_Tracking\_Application  {  public partial class manageproject : Form  {  public manageproject()  {  InitializeComponent();  }  //acccessing data from various classes  BusinessLogicClass blc = new BusinessLogicClass();  ProjectClass pc = new ProjectClass();  HelperClass hc = new HelperClass();  public int ProjectId;  private void btnexit\_Click(object sender, EventArgs e)  {  this.Close();  }  //adding the data to display on datagridview and store to database  private void btnadd\_Click(object sender, EventArgs e)  {  if (txtprojectname.Text == "")  {  MessageBox.Show("Provide Projectname: Full information required");  }  if (dtpstartingdate.Text == "")  {  MessageBox.Show("Provide Starting Date: Full information required");  }  if (dtpfinishingdate.Text == "")  {  MessageBox.Show("Provide Finishing Date: Full information required");  }  if (txtdescription.Text == "")  {  MessageBox.Show("Provide Description: Full information required");  }  else if (DublicateProject() == true)  {  MessageBox.Show("Project with same name already exists");  txtprojectname.Clear();  txtprojectname.Focus();  }  { CreateProject(); }  }  private void CreateProject()  {  //try catch exception  try  {  bool res = blc.ProjectTable(0,  txtprojectname.Text,  Convert.ToDateTime(dtpstartingdate.Text),  Convert.ToDateTime(dtpfinishingdate.Text),  txtdescription.Text,  1);  if (res == true)  {  //display message as added project  MessageBox.Show("Add to Create Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  else  {  //display error message as data cannot be added  MessageBox.Show("Couldn't Add selected Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }    public bool DublicateProject()  {  int x = 0;  try  {  for (int i = 0; i < dgvprojects.Rows.Count; i++)  {  if (txtprojectname.Text == dgvprojects.Rows[i].Cells["ProjectName"].Value.ToString())  x = 1;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  if (x == 1)  return true;  else  return false;  }  //retrive all data from datagridview to the details entry section on a single click  private void dgvprojects\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  try  {  ProjectId = Convert.ToInt32(dgvprojects.SelectedRows[0].Cells["ProjectId"].Value.ToString());  txtprojectname.Text = dgvprojects.SelectedRows[0].Cells["ProjectName"].Value.ToString();  dtpstartingdate.Text = dgvprojects.SelectedRows[0].Cells["StartingDate"].Value.ToString();  dtpfinishingdate.Text = dgvprojects.SelectedRows[0].Cells["FinishingDate"].Value.ToString();  txtdescription.Text = dgvprojects.SelectedRows[0].Cells["Description"].Value.ToString();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrieve the data on DataGridView  private void manageproject\_Load(object sender, EventArgs e)  {  dgvprojects.DataSource = pc.GetAllProjects();  }  //update the date entered into the database  private void btnupdate\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.ProjectTable(ProjectId,  txtprojectname.Text,  Convert.ToDateTime(dtpstartingdate.Text),  Convert.ToDateTime(dtpfinishingdate.Text),  txtdescription.Text,  2);  if (res == true)  {  //display message as updated project  MessageBox.Show("Success to Update Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  else  {  //display error message as data cannot be updated  MessageBox.Show("Couldn't Update selected Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // delete the data entered into the database  private void btndelete\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.ProjectTable(ProjectId,  txtprojectname.Text,  Convert.ToDateTime(dtpstartingdate.Text),  Convert.ToDateTime(dtpfinishingdate.Text),  txtdescription.Text,  3);  if (res == true)  {  //display message as deleted project  MessageBox.Show("Success to Delete Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  else  {  //display error message as data cannot be deleted  MessageBox.Show("Couldn't Delete selected Project");  dgvprojects.DataSource = pc.GetAllProjects();  HelperClass.makeFieldsBlank(grpContainer);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }    }  } |

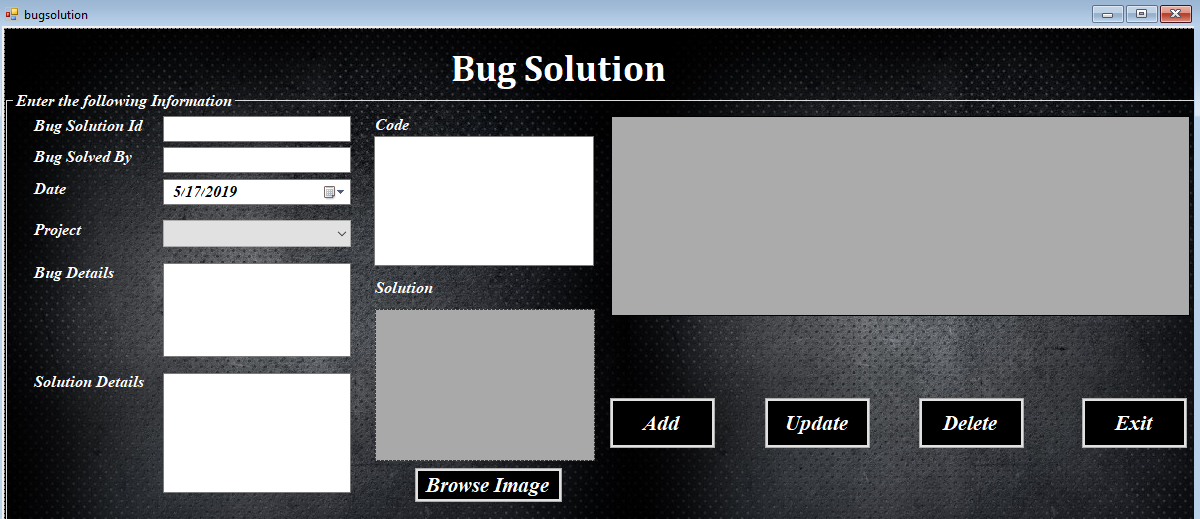
**Manage Bug entry:**



**Code for Bug entry:**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using Microsoft.VisualBasic;  using BusinessLogicLayer;  using DataAccessLayer;  using System.IO;  namespace Bug\_Tracking\_Application  {  public partial class bugentry : Form  {  public bugentry()  {  InitializeComponent();  }  //acccessing data from various classes  BusinessLogicClass blc = new BusinessLogicClass();  HelperClass hc = new HelperClass();  ProjectClass pc = new ProjectClass();  BugEntryClass bec = new BugEntryClass();  public int BugId;  //close the form  private void btnexit\_Click(object sender, EventArgs e)  {  this.Close();  }  //adding the data to display on datagridview and store to database  private void btnadd\_Click(object sender, EventArgs e)  {  if (dtpdate.Text == "")  {  MessageBox.Show("Provide Date: Full information required");  }  else if (txtclasslibrary.Text == "")  {  MessageBox.Show("Provide Class Library: Full information required");  }  else if (txtblock.Text == "")  {  MessageBox.Show("Provide Block: Full information required");  }  else if (txtidentifiedby.Text == "")  {  MessageBox.Show("Provide Identified by: Full information required");  }  else if (txtclass.Text == "")  {  MessageBox.Show("Provide Class: Full information required");  }  else if (txtlinenumber.Text == "")  {  MessageBox.Show("Provide Line Number: Full information required");  }  else if (cmbproject.SelectedIndex == -1)  {  MessageBox.Show("Provide Project: Full information required");  }  else if (txtmethod.Text == "")  {  MessageBox.Show("Provide Method: Full information required");  }  else if (txtbugdetails.Text == "")  {  MessageBox.Show("Provide Bug Details: Full information required");  }  else if (txtcode.Text == "")  {  MessageBox.Show("Provide Code: Full information required");  }  else if (btnbrowse.Text == "")  {  MessageBox.Show("Provide Image: Full information required");  }    { CreateBugs(); }  }  //create bugs to fill the empty space  private void CreateBugs()  {  //try catch exception  try  {  bool res = blc.BugTable(0,  Convert.ToDateTime(dtpdate.Text),  txtclasslibrary.Text,  txtblock.Text,  txtidentifiedby.Text,  txtclass.Text,  txtlinenumber.Text,  cmbproject.Text,  txtmethod.Text,  txtbugdetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugs),  1);  if (res == true)  {  //display message of successfully added  MessageBox.Show("Added to Entry Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  else  {  //display error message as data cannot be stored  MessageBox.Show("Couldn't Add data to Entry Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrive all data from datagridview to the details entry section on a single click  private void dgvbugs\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  try  {  BugId = Convert.ToInt32(dgvbugs.SelectedRows[0].Cells["BugId"].Value.ToString());  dtpdate.Text = dgvbugs.SelectedRows[0].Cells["Date"].Value.ToString();  txtclasslibrary.Text = dgvbugs.SelectedRows[0].Cells["ClassLibrary"].Value.ToString();  txtblock.Text = dgvbugs.SelectedRows[0].Cells["Block"].Value.ToString();  txtidentifiedby.Text = dgvbugs.SelectedRows[0].Cells["IdentifiedBy"].Value.ToString();  txtclass.Text = dgvbugs.SelectedRows[0].Cells["Class"].Value.ToString();  txtlinenumber.Text = dgvbugs.SelectedRows[0].Cells["LineNumber"].Value.ToString();  cmbproject.Text = dgvbugs.SelectedRows[0].Cells["Project"].Value.ToString();  txtmethod.Text = dgvbugs.SelectedRows[0].Cells["Method"].Value.ToString();  txtbugdetails.Text = dgvbugs.SelectedRows[0].Cells["BugDetails"].Value.ToString();  txtcode.Text = dgvbugs.SelectedRows[0].Cells["Code"].Value.ToString();  MemoryStream memoryStream = new MemoryStream((byte[])dgvbugs.SelectedRows[0].Cells["Snap"].Value);  picbugs.Image = Image.FromStream(memoryStream);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrieve the data on DataGridView  private void bugentry\_Load(object sender, EventArgs e)  {  dgvbugs.DataSource = bec.GetAllBugs();  cmbproject.DataSource = pc.GetAllProjects();  cmbproject.DisplayMember = "ProjectName";  cmbproject.ValueMember = "ProjectName";  cmbproject.SelectedIndex = -1;  }  //Browse image in button click  private void btnbrowse\_Click\_1(object sender, EventArgs e)  {  try  {  try  {  OpenFileDialog ofd = new OpenFileDialog();  if (ofd.ShowDialog() == DialogResult.OK)  {  picbugs.Image = Image.FromFile(ofd.FileName);  }  else  {  MessageBox.Show("Please select a Bug picture");  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // update the data entered into the database  private void Btnupdate\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugTable(BugId,  Convert.ToDateTime(dtpdate.Text),  txtclasslibrary.Text,  txtblock.Text,  txtidentifiedby.Text,  txtclass.Text,  txtlinenumber.Text,  cmbproject.Text,  txtmethod.Text,  txtbugdetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugs),  2);  if (res == true)  {  //display message of successfully updated  MessageBox.Show("Success to Update Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  else  {  //display error message as data cannot be updated  MessageBox.Show("Couldn't success to Update Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // delete the data entered into the database  private void Btndelete\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugTable(BugId,  Convert.ToDateTime(dtpdate.Text),  txtclasslibrary.Text,  txtblock.Text,  txtidentifiedby.Text,  txtclass.Text,  txtlinenumber.Text,  cmbproject.Text,  txtmethod.Text,  txtbugdetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugs),  3);  if (res == true)  {  //display message of successfully deleted  MessageBox.Show("Success to Delete Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  else  {  //display error message as data cannot be deleted  MessageBox.Show("Couldn't success to Delete Bugs");  dgvbugs.DataSource = bec.GetAllBugs();  HelperClass.makeFieldsBlank(grpContainer);  picbugs.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  }  } |

**Bug Solution:**



**Code for Bug Solution:**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using BusinessLogicLayer;  using DataAccessLayer;  using System.IO;  namespace Bug\_Tracking\_Application  {  public partial class bugsolution : Form  {  public bugsolution()  {  InitializeComponent();  }  //acccessing data from various classes  BusinessLogicClass blc = new BusinessLogicClass();  HelperClass hc = new HelperClass();  ProjectClass pc = new ProjectClass();  BugEntryClass bec = new BugEntryClass();  BugSolutionClass bsc = new BugSolutionClass();  public int BugSolutionId;  //adding the data to display on datagridview and store to database  private void btnadd\_Click(object sender, EventArgs e)  {  if (txtbugsolvedby.Text == "")  {  MessageBox.Show("Provide Bug Solved By: Full information required");  }  if (dtpdate.Text == "")  {  MessageBox.Show("Provide Date: Full information required");  }  if (cmbproject.Text == "")  {  MessageBox.Show("Provide Project: Full information required");  }  if (txtbugdetails.Text == "")  {  MessageBox.Show("Provide Bug Details by: Full information required");  }  if (txtsolutiondetails.Text == "")  {  MessageBox.Show("Provide Solution Details: Full information required");  }  if (txtcode.Text == "")  {  MessageBox.Show("Provide Code: Full information required");  }  if (btnbrowse.Text == "")  {  MessageBox.Show("Provide Image: Full information required");  }  { CreateBugSolution(); }  }  //create bug solution  private void CreateBugSolution()  {  try  {  bool res = blc.BugSolutionTable(0,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  1);  if (res == true)  {  //display message of entred bug solution data  MessageBox.Show("Success to Entry Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be stored  MessageBox.Show("Couldn't success to Entry Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrive all data from datagridview to the details entry section on a single click  private void dgvbugsolutions\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  try  {  BugSolutionId = Convert.ToInt32(dgvbugsolutions.SelectedRows[0].Cells["BugSolutionId"].Value.ToString());  txtbugsolvedby.Text = dgvbugsolutions.SelectedRows[0].Cells["BugSolvedBy"].Value.ToString();  dtpdate.Text = dgvbugsolutions.SelectedRows[0].Cells["Date"].Value.ToString();  cmbproject.Text = dgvbugsolutions.SelectedRows[0].Cells["Project"].Value.ToString();  txtbugdetails.Text = dgvbugsolutions.SelectedRows[0].Cells["BugDetails"].Value.ToString();  txtsolutiondetails.Text = dgvbugsolutions.SelectedRows[0].Cells["SolutionDetails"].Value.ToString();  txtcode.Text = dgvbugsolutions.SelectedRows[0].Cells["Code"].Value.ToString();  MemoryStream memoryStream = new MemoryStream((byte[])dgvbugsolutions.SelectedRows[0].Cells["Snap"].Value);  picbugsolutions.Image = Image.FromStream(memoryStream);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrieve the data on DataGridView  private void bugsolution\_Load(object sender, EventArgs e)  {  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  cmbproject.DataSource = pc.GetAllProjects();  cmbproject.DisplayMember = "ProjectName";  cmbproject.ValueMember = "ProjectName";  cmbproject.SelectedIndex = -1;  }  //Browse image in button click  private void btnbrowse\_Click(object sender, EventArgs e)  {  try  //try catch exception  {  try  {  OpenFileDialog ofd = new OpenFileDialog();  if (ofd.ShowDialog() == DialogResult.OK)  {  picbugsolutions.Image = Image.FromFile(ofd.FileName);  }  else  {  MessageBox.Show("Please select a Solution picture");  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //close the form  private void btnexit\_Click(object sender, EventArgs e)  {  this.Close();  }  // update the data entered into the database  private void Btnupdate\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugSolutionTable(BugSolutionId,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  2);  if (res == true)  {  //display message of updating the bug solution data  MessageBox.Show("Success to Update Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be updated  MessageBox.Show("Couldn't success to Update Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // delete the data entered into the database  private void Btndelete\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugSolutionTable(BugSolutionId,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  3);  if (res == true)  {  //display message of deleting the bug solution data  MessageBox.Show("Success to Delete Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be deleted  MessageBox.Show("Couldn't success to Delete Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  }  } |

**Business Logic Layer:**

|  |
| --- |
| Business logic class:  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using BusinessLogicLayer;  using DataAccessLayer;  using System.IO;  namespace Bug\_Tracking\_Application  {  public partial class bugsolution : Form  {  public bugsolution()  {  InitializeComponent();  }  //acccessing data from various classes  BusinessLogicClass blc = new BusinessLogicClass();  HelperClass hc = new HelperClass();  ProjectClass pc = new ProjectClass();  BugEntryClass bec = new BugEntryClass();  BugSolutionClass bsc = new BugSolutionClass();  public int BugSolutionId;  //adding the data to display on datagridview and store to database  private void btnadd\_Click(object sender, EventArgs e)  {  if (txtbugsolvedby.Text == "")  {  MessageBox.Show("Provide Bug Solved By: Full information required");  }  if (dtpdate.Text == "")  {  MessageBox.Show("Provide Date: Full information required");  }  if (cmbproject.Text == "")  {  MessageBox.Show("Provide Project: Full information required");  }  if (txtbugdetails.Text == "")  {  MessageBox.Show("Provide Bug Details by: Full information required");  }  if (txtsolutiondetails.Text == "")  {  MessageBox.Show("Provide Solution Details: Full information required");  }  if (txtcode.Text == "")  {  MessageBox.Show("Provide Code: Full information required");  }  if (btnbrowse.Text == "")  {  MessageBox.Show("Provide Image: Full information required");  }  { CreateBugSolution(); }  }  //create bug solution  private void CreateBugSolution()  {  try  {  bool res = blc.BugSolutionTable(0,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  1);  if (res == true)  {  //display message of entred bug solution data  MessageBox.Show("Success to Entry Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be stored  MessageBox.Show("Couldn't success to Entry Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrive all data from datagridview to the details entry section on a single click  private void dgvbugsolutions\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  try  {  BugSolutionId = Convert.ToInt32(dgvbugsolutions.SelectedRows[0].Cells["BugSolutionId"].Value.ToString());  txtbugsolvedby.Text = dgvbugsolutions.SelectedRows[0].Cells["BugSolvedBy"].Value.ToString();  dtpdate.Text = dgvbugsolutions.SelectedRows[0].Cells["Date"].Value.ToString();  cmbproject.Text = dgvbugsolutions.SelectedRows[0].Cells["Project"].Value.ToString();  txtbugdetails.Text = dgvbugsolutions.SelectedRows[0].Cells["BugDetails"].Value.ToString();  txtsolutiondetails.Text = dgvbugsolutions.SelectedRows[0].Cells["SolutionDetails"].Value.ToString();  txtcode.Text = dgvbugsolutions.SelectedRows[0].Cells["Code"].Value.ToString();  MemoryStream memoryStream = new MemoryStream((byte[])dgvbugsolutions.SelectedRows[0].Cells["Snap"].Value);  picbugsolutions.Image = Image.FromStream(memoryStream);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //retrieve the data on DataGridView  private void bugsolution\_Load(object sender, EventArgs e)  {  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  cmbproject.DataSource = pc.GetAllProjects();  cmbproject.DisplayMember = "ProjectName";  cmbproject.ValueMember = "ProjectName";  cmbproject.SelectedIndex = -1;  }  //Browse image in button click  private void btnbrowse\_Click(object sender, EventArgs e)  {  try  //try catch exception  {  try  {  OpenFileDialog ofd = new OpenFileDialog();  if (ofd.ShowDialog() == DialogResult.OK)  {  picbugsolutions.Image = Image.FromFile(ofd.FileName);  }  else  {  MessageBox.Show("Please select a Solution picture");  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  //close the form  private void btnexit\_Click(object sender, EventArgs e)  {  this.Close();  }  // update the data entered into the database  private void Btnupdate\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugSolutionTable(BugSolutionId,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  2);  if (res == true)  {  //display message of updating the bug solution data  MessageBox.Show("Success to Update Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be updated  MessageBox.Show("Couldn't success to Update Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  // delete the data entered into the database  private void Btndelete\_Click(object sender, EventArgs e)  {  try  {  bool res = blc.BugSolutionTable(BugSolutionId,  txtbugsolvedby.Text,  Convert.ToDateTime(dtpdate.Text),  cmbproject.Text,  txtbugdetails.Text,  txtsolutiondetails.Text,  txtcode.Text,  HelperClass.imageConverter(picbugsolutions),  3);  if (res == true)  {  //display message of deleting the bug solution data  MessageBox.Show("Success to Delete Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  else  {  //display error message as data cannot be deleted  MessageBox.Show("Couldn't success to Delete Bug Solutions");  dgvbugsolutions.DataSource = bsc.GetAllBugSolutions();  HelperClass.makeFieldsBlank(grpContainer);  picbugsolutions.Image = null;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  }  } |

**Data Access Layer:**

|  |
| --- |
| **Member Class:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using System.Threading.Tasks;  using System.Data.SqlClient;  namespace DataAccessLayer  {  public class MemberClass  {  SqlConnection conn = new SqlConnection(ConnectionClass.ConnectionString);  public int MemberTable(int MemberId,  String UserName,  String Name,  String Password,  String Role,  String Address,  String Email,  String Gender,  int Contact,  DateTime DOB,  DateTime DOJ,  byte[] Image,  int Mode)  {  try  {  SqlCommand cmd = new SqlCommand("SP\_ManageMembers", conn);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@MemberId", MemberId);  cmd.Parameters.AddWithValue("@UserName", UserName);  cmd.Parameters.AddWithValue("@Name", Name);  cmd.Parameters.AddWithValue("@Password", Password);  cmd.Parameters.AddWithValue("@Role", Role);  cmd.Parameters.AddWithValue("@Address", Address);  cmd.Parameters.AddWithValue("@Email", Email);  cmd.Parameters.AddWithValue("@Gender", Gender);  cmd.Parameters.AddWithValue("@Contact", Contact);  cmd.Parameters.AddWithValue("@DOB", DOB);  cmd.Parameters.AddWithValue("@DOJ", DOJ);  cmd.Parameters.AddWithValue("@Image", Image);  cmd.Parameters.AddWithValue("@Mode", Mode);  conn.Open();  int result = cmd.ExecuteNonQuery();  conn.Close();  return result;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  public DataTable GetAllUsers()  {  try  {  DataTable dt = new DataTable();  SqlCommand cmd = new SqlCommand("Select \* from MemberTable", conn);  conn.Open();  SqlDataReader dr = cmd.ExecuteReader();  dt.Load(dr);  conn.Close();  return dt;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  public string Login(String UserName, String Password)  {  try  {  String Role = "";  DataTable dt = new DataTable();  SqlCommand cmd = new SqlCommand("Select Role from MemberTable where UserName=@UserName and Password=@Password", conn);  cmd.CommandType = CommandType.Text;  cmd.Parameters.AddWithValue("@UserName", UserName);  cmd.Parameters.AddWithValue("@Password", Password);  conn.Open();  SqlDataReader dr = cmd.ExecuteReader();  dt.Load(dr);  conn.Close();  Role = dt.Rows[0]["Role"].ToString();  return Role;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  }  } |

|  |
| --- |
| **Project Class:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using System.Data.SqlClient;  using System.Threading.Tasks;  namespace DataAccessLayer  {  public class ProjectClass  {  SqlConnection conn = new SqlConnection(ConnectionClass.ConnectionString);  public int ProjectTable(int ProjectId,  String ProjectName,  DateTime StartingDate,  DateTime FinishingDate,  String Description,  int Mode)  {  try  {  SqlCommand cmd = new SqlCommand("SP\_ManageProjects", conn);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@ProjectId", ProjectId);  cmd.Parameters.AddWithValue("@ProjectName", ProjectName);  cmd.Parameters.AddWithValue("@StartingDate", StartingDate);  cmd.Parameters.AddWithValue("@FinishingDate", FinishingDate);  cmd.Parameters.AddWithValue("@Description", Description);  cmd.Parameters.AddWithValue("@Mode", Mode);  conn.Open();  int result = cmd.ExecuteNonQuery();  conn.Close();  return result;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  public DataTable GetAllProjects()  {  try  {  DataTable dt = new DataTable();  SqlCommand cmd = new SqlCommand("Select \* from ProjectTable", conn);  conn.Open();  SqlDataReader dr = cmd.ExecuteReader();  dt.Load(dr);  conn.Close();  return dt;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  }  } |

|  |
| --- |
| **Bug Entry Class:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using System.Threading.Tasks;  using System.Data.SqlClient;  namespace DataAccessLayer  {  public class BugEntryClass  {  SqlConnection conn = new SqlConnection(ConnectionClass.ConnectionString);  public int BugTable(int BugId,  DateTime Date,  String ClassLibrary,  String Block,  String IdentifiedBy,  String Class,  String LineNumber,  String Project,  String Method,  String BugDetails,  String Code,  byte[] Snap,  int Mode)  {  try  {  SqlCommand cmd = new SqlCommand("SP\_ManageBugs", conn);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@BugId", BugId);  cmd.Parameters.AddWithValue("@Date", Date);  cmd.Parameters.AddWithValue("@ClassLibrary", ClassLibrary);  cmd.Parameters.AddWithValue("@Block", Block);  cmd.Parameters.AddWithValue("@IdentifiedBy", IdentifiedBy);  cmd.Parameters.AddWithValue("@Class", Class);  cmd.Parameters.AddWithValue("@LineNumber", LineNumber);  cmd.Parameters.AddWithValue("@Project", Project);  cmd.Parameters.AddWithValue("@Method", Method);  cmd.Parameters.AddWithValue("@BugDetails", BugDetails);  cmd.Parameters.AddWithValue("@Code", Code);  cmd.Parameters.AddWithValue("@Snap", Snap);  cmd.Parameters.AddWithValue("@Mode", Mode);  conn.Open();  int result = cmd.ExecuteNonQuery();  conn.Close();  return result;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  public DataTable GetAllBugs()  {  try  {  DataTable dt = new DataTable();  SqlCommand cmd = new SqlCommand("Select \* from BugTable", conn);  conn.Open();  SqlDataReader dr = cmd.ExecuteReader();  dt.Load(dr);  conn.Close();  return dt;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  }  } |

|  |
| --- |
| **Bug Solution Class:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using System.Threading.Tasks;  using System.Data.SqlClient;  namespace DataAccessLayer  {  public class BugSolutionClass  {  SqlConnection conn = new SqlConnection(ConnectionClass.ConnectionString);  public int BugSolutionTable(int BugSolutionId,  String BugSolvedBy,  DateTime Date,  String Project,  String BugDetails,  String SolutionDetails,  String Code,  byte[] Snap,  int Mode)  {  try  {  SqlCommand cmd = new SqlCommand("SP\_ManageBugSolutions", conn);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@BugsolutionId", BugSolutionId);  cmd.Parameters.AddWithValue("@BugsolvedBy", BugSolvedBy);  cmd.Parameters.AddWithValue("@Date", Date);  cmd.Parameters.AddWithValue("@Project", Project);  cmd.Parameters.AddWithValue("@BugDetails", BugDetails);  cmd.Parameters.AddWithValue("@SolutionDetails", SolutionDetails);  cmd.Parameters.AddWithValue("@Code", Code);  cmd.Parameters.AddWithValue("@Snap", Snap);  cmd.Parameters.AddWithValue("@Mode", Mode);  conn.Open();  int result = cmd.ExecuteNonQuery();  conn.Close();  return result;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  }  public DataTable GetAllBugSolutions()  {  try  {  DataTable dt = new DataTable();  SqlCommand cmd = new SqlCommand("Select \* from BugSolutionTable", conn);  conn.Open();  SqlDataReader dr = cmd.ExecuteReader();  dt.Load(dr);  conn.Close();  return dt;  }  catch (Exception ex)  {  throw ex;  }  finally { conn.Close(); }  } |

**Testing:**

**Unit Testing:**

Unit testing is the method which is checked whether the each and every control of the form are working or not. The given table used for unit testing shows the working of the different controls of various forms are success or failure.

**Login Form:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.  N | Test  Description | Intended Result | Actual Output | Action /  Remarks |
| 1. | Null Space  Validation | In case of blank field message should be displayed. | The Login wasn’t successful in case of blank field. If username or  password wasn’t entered. | The test was successful. |
| 2. | Trying to access with wrong username and password | If the Login Username and  Password is wrong  the login shouldn’t be success. | The Login wasn’t success in case of wrong username or password. | The test was successful. |
| 3. | The Login should be successful if correct user name and password is entered. | The login should provide access to main form after correct information was provided. | The log in was successful after correct username and password was entered in textbox. | The test was successful. |

**Manage Member:**

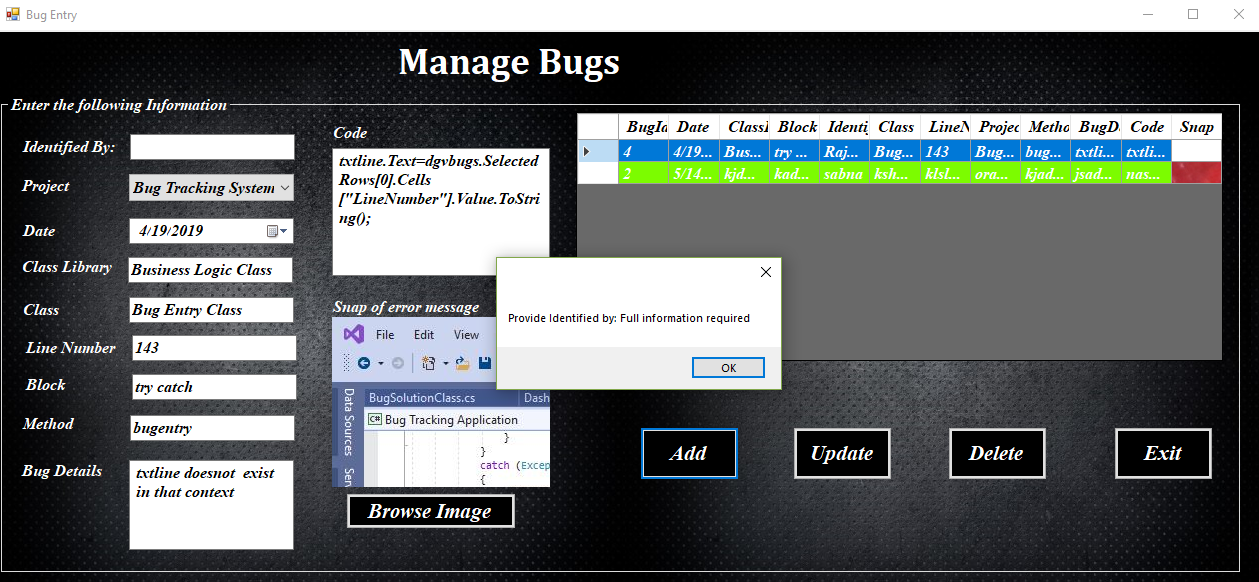
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Test  Description | Intended Result | Actual Output | Action/  Remarks |
| 1 | Null space validation | When any input field is left blank, adding user becomes unsuccessful. | Creating User was unsuccessful when any of the fields in the User form was left blank. | The null  validation  test was successful. |
| When all the fields are typed, the creating user was successful. | The creating user was successful when all the input fields were filled. |
| 3 | Duplicate  User Test | When same user name was entered then Error message should be shown. | The unsuccessful message was shown as there was already a same username. | The test was successful. |

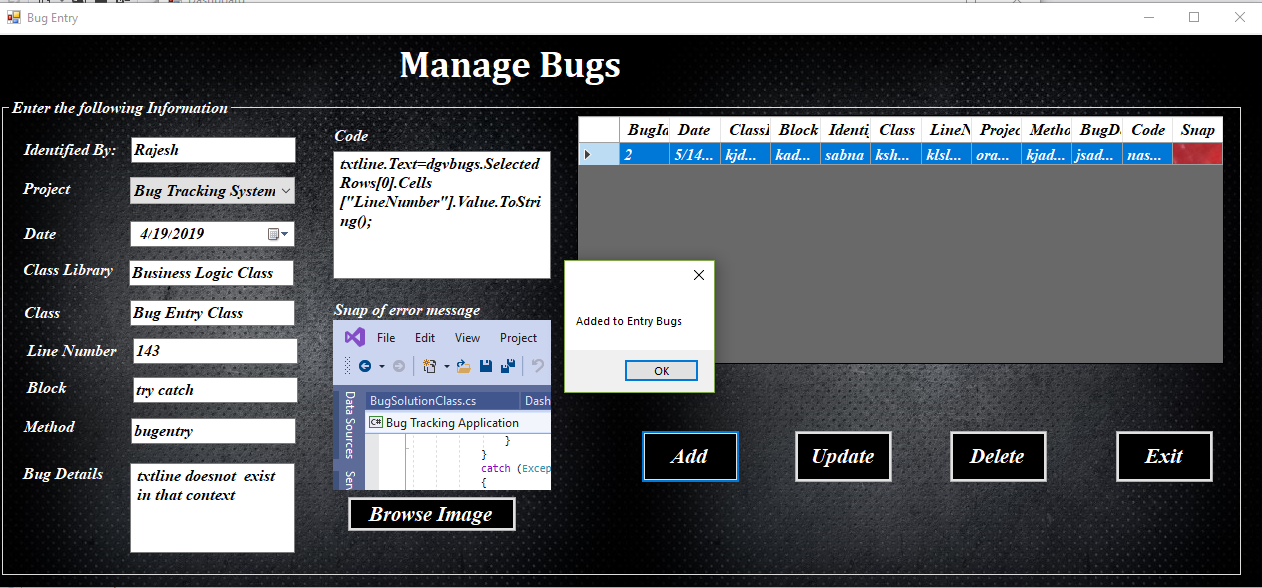
**Manage Project:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Test**  **Description** | **Intended output** | **Actual Output** | **Remarks /**  **Action** |
| 1 | Null space validation | When any input field is left blank, the process becomes unsuccessful. | The process was unsuccessful when any of the fields in the Category Form was left blank. | The null validation  test was successful. |
| When all the fields are filled, the process successes. | The process was successful when all the fields were filled. |
| 2 | Duplicate category testing | Repetition of Same Category Name is not allowed. | The process was unsuccessful when same category name was added. | The test result was  Successful. |

**Manage Bug:**

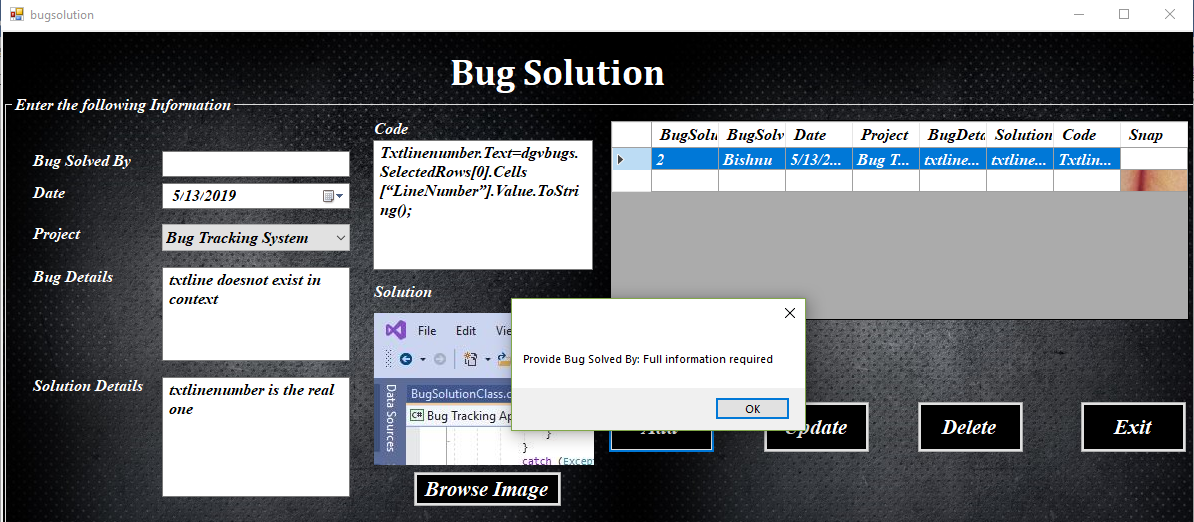
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Test**  **Description** | **Intended output** | **Actual Output** | **Remarks /**  **Action** |
| **1** | Null Space  Validation | When any input field is left blank, the  Process becomes unsuccessful. | The process was unsuccessful when fields were left blank. | The null  Validation  test was successful. |
| **2** | Null Space  Validation | When all information was added then the process got success. | The process got success when all the input fields were filled. | The null  Validation  test was successful. |

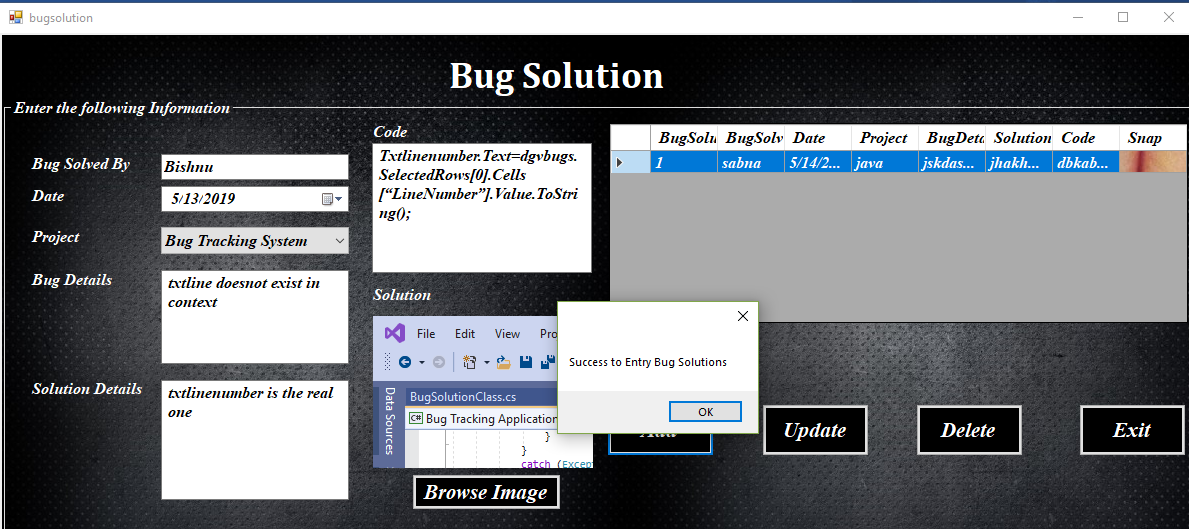




**Manage Bug Solution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Test**  **Description** | **Intended output** | **Actual Output** | **Remarks /**  **Action** |
| **1** | Null Space  Validation | When any input field is left blank, the  Process becomes unsuccessful. | The process was successful when fields were left blank. | The null  Validation test was not successful. |
| **2** | Null Space  Validation | When all information was added then the process got success. | The process got success when all the input fields were filled. | The null  Validation  test was successful. |



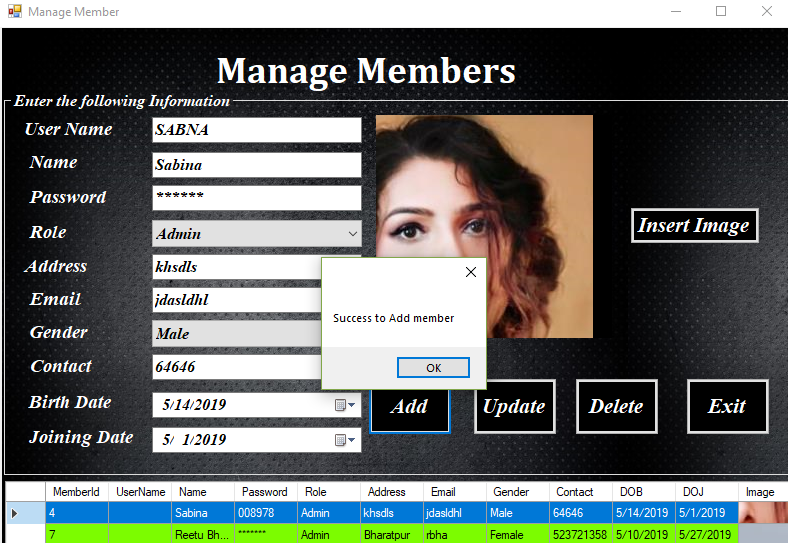


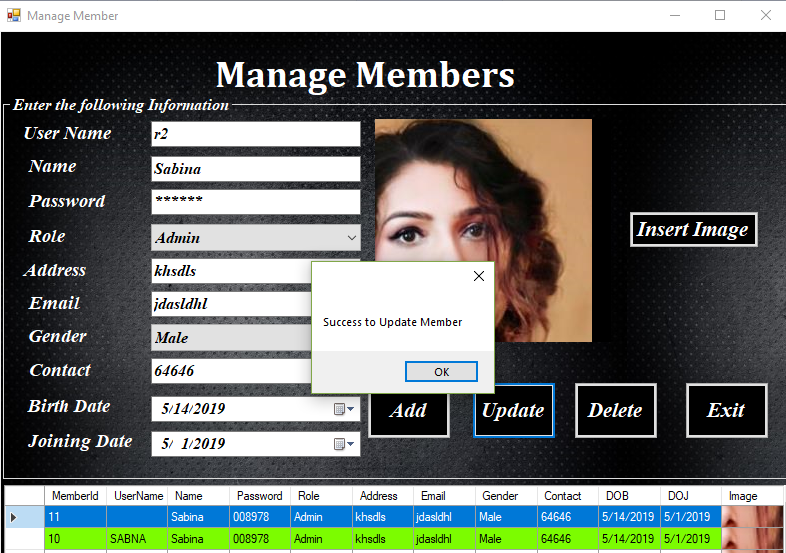
**Integrated Testing:**

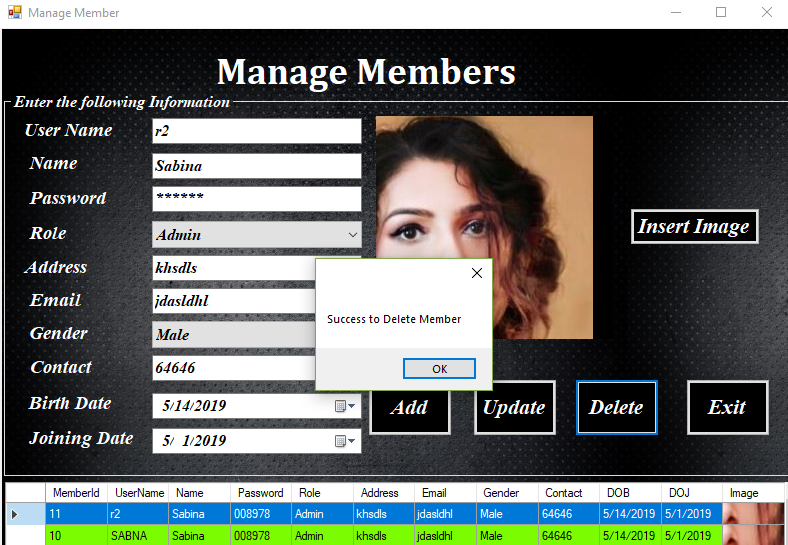
For checking the proper working of the save, update and delete buttons which is used in the form is done through integrated testing. The given table shows success or failure of the integration testing done from project.

**Manage Member Form:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Test**  **Description** | **Intended output** | **Actual Output** | **Action /**  **Remarks** |
| 1 | The working of Add button | After clicked on save button, member need to be added and saved in the database. | As per expected result Member is added and saved in the database. | The test was successful. |
| 2 | The working of Update button | After clicked on update button, member need to be updated. | As per expected result member is updated. | The test result was successful. |
| 3 | The working of Delete button | After clicked on delete button, member need to be deleted. | As per expected result member deleted. | The test result was successful. |

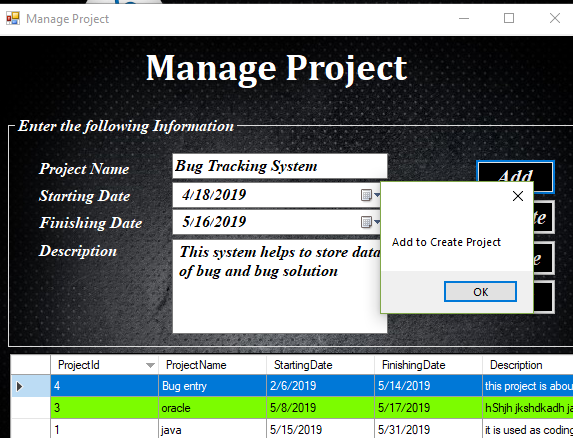


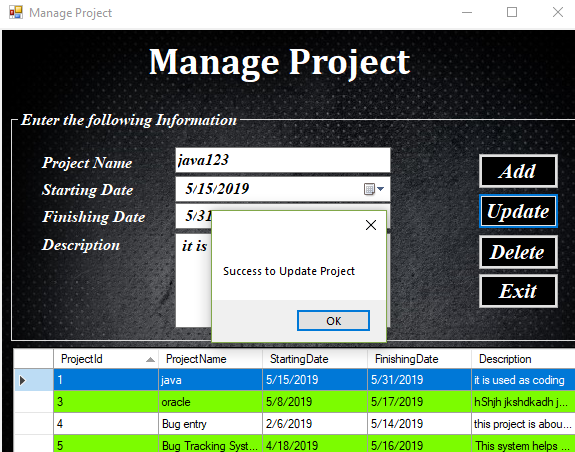


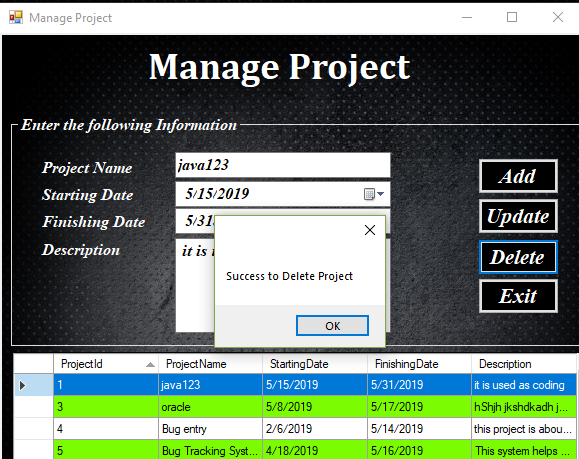


**Integrated Testing of Manage project Form:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.  N | Test Description | Intended  Output | Actual Output | Action  Remarks | / |
| 1. | The working of  Add button | After clicked on save button, project need to be added and saved in the database. | As per expected result  project is added and saved in the database. | Tested successfully. |  |
| 2. | The working of  Update button | After clicked on update button, project need to be updated. | As per expected result  project is updated. | Tested successfully. |  |
| 3. | The working of  Delete button | After clicked on delete button, project need to be deleted. | As per expected result project deleted. | Tested successfully. |  |

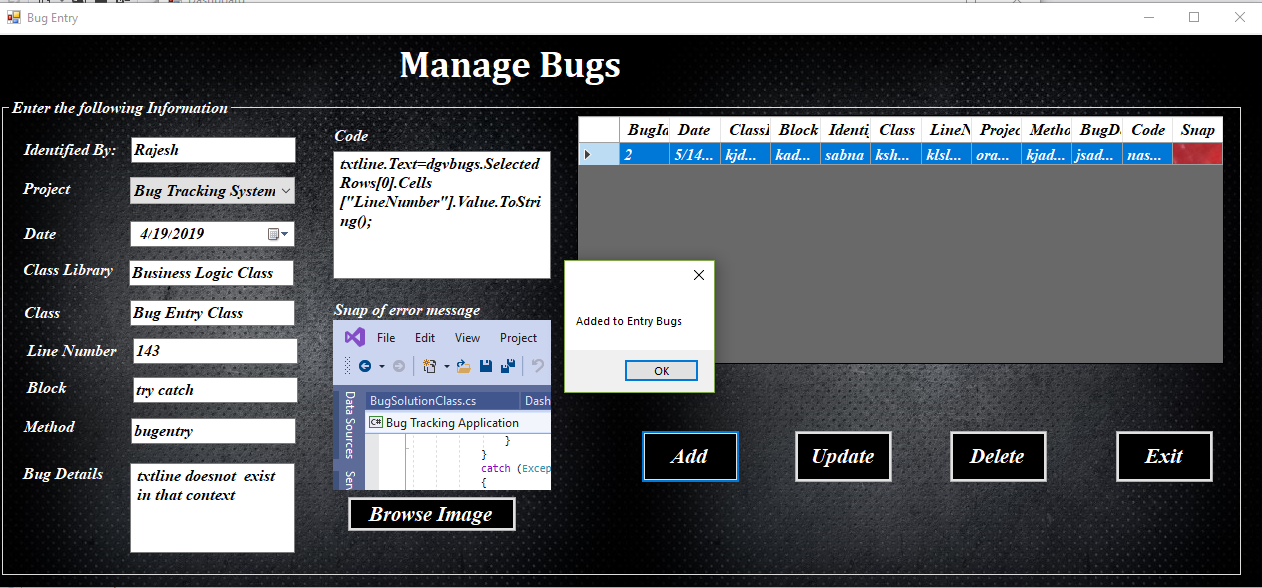


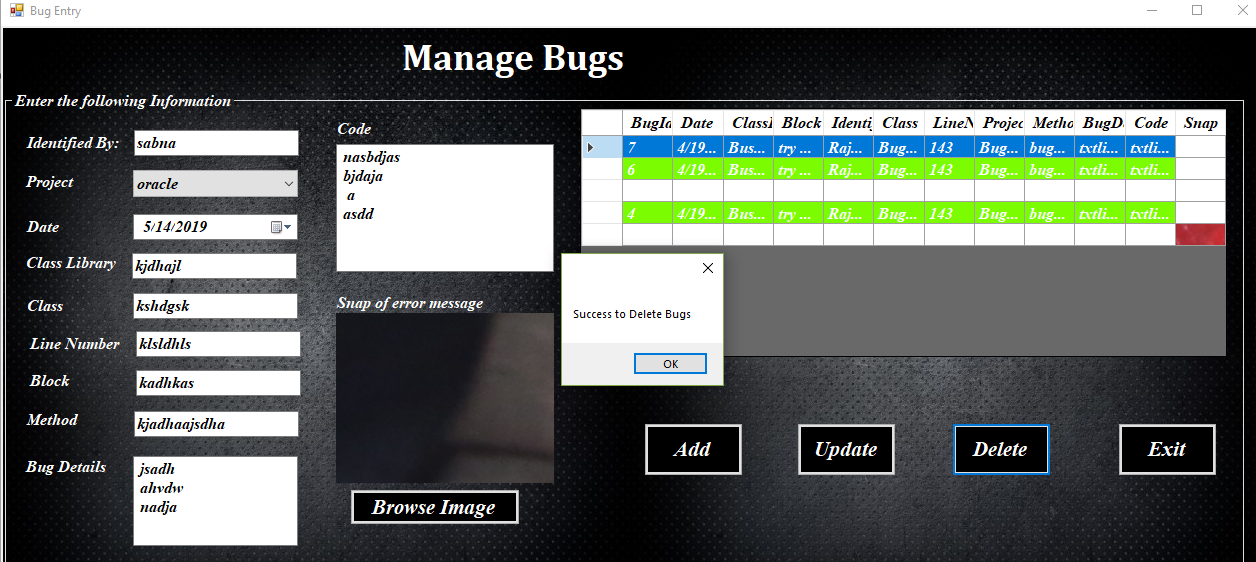


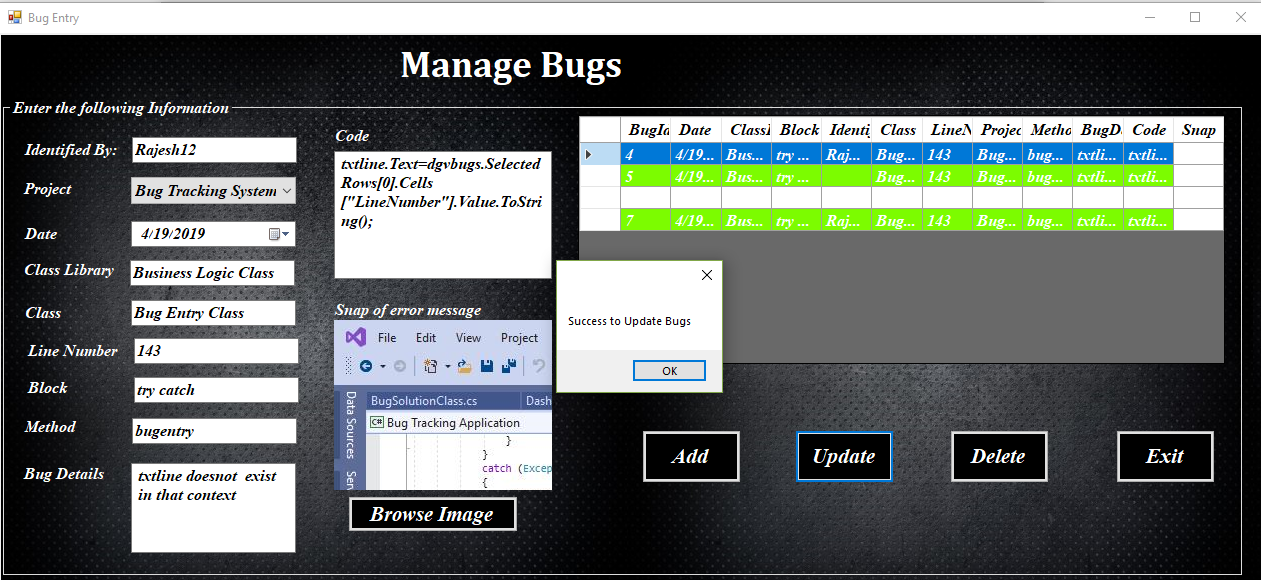


**Integrated Testing for Manage bug:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.  N | Test Description | Intended  Output | Actual Output | Action  Remarks | / |
| 1. | The working of  Add button | After clicked on save button, Bug need to be added and saved in the database. | As per expected result  Bug is added and saved in the database. | Tested successfully. |  |
| 2. | The working of  Update button | After clicked on update button, Bug need to be updated. | As per expected result  Bug is updated. | Tested successfully. | |
| 3. | The working of  Delete button | After clicked on delete button, Bug need to be deleted. | As per expected result Bug deleted. | Tested successfully. | |







**Integrated Testing for Manage bug solution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.  N | Test Description | Intended  Output | Actual Output | Action  Remarks | / |
| 1. | The working of  Add button | After clicked on save button, Bug solution need to be added and saved in the database. | As per expected result  Bug solution is added and saved in the database. | Tested successfully. |  |
| 2. | The working of  Update button | After clicked on update button, Bug solution need to be updated. | As per expected result  Bug solution is updated. | Tested successfully. | |
| 3. | The working of  Delete button | After clicked on delete button, Bug solution need to be deleted. | As per expected result Bug solution deleted. | Tested successfully. | |

