Issue Tracker: Report

The Group
Dhimant Adhikari
Richard Duncan
Aswin Lohani
Andrea Soteldo

The Client
Professor Vladan Vuckovic

Project Plan

Scope

The project is to develop a **Bug Tracking System** software to allow the user to submit **problem tickets** and be responded to. The system will allow the admin to respond and designate priorities on tickets and close out tickets.
This project will have 3 main parts: developing a system to hold and catalog incoming data and form tickets, construct, and **administrative user interface that** allows the ability to change and categorize the incoming data from tickets, creating a **UI system** that a user can monitor a response from the admin.

Vision Plan

Currently users struggle to troubleshoot their technical issues that require further diagnosis and real time responses. Which can cause delays in their daily activities.

1. This proposed Bug Tracking system would **compile incoming "Tickets"** sent in by users seeking resolutions to their technical problems.
2. This proposed Bug Tracking would provide a listed **system framework** for users to submit and monitor their tickets that are submitted, **admin control interface** to monitor and prioritize submissions; thus, providing adequate and organized **troubleshooting solutions** to those seeking it.

Initial Project Glossary

Scope

Bug Tracking System: Also called as a defect tracking system in a software application
Problem Tickets: Electronic or written document created by help desk to record a problem call and track the resolution of the problem.
Administrative User Interface: an application that allows performing administrative tasks such as diagnosis, monitoring, scripting, and configuration.
UI system: user interface (UI) system; the point of human-computer interaction; also known as a pathway of communication between user and an application

Vision Plan
Compile Incoming "Tickets": produce a report by assembling information gathered from the tickets.
System Framework: the foundation or infrastructure of the system indicating what kind of programs can or should be built.
Admin Control Interface: Also referred to as the back end, it is the control panel of the project.
Troubleshooting Solutions: Process of diagnosing and often fixing the root cause of the problems.

Initial Risk Assessment
Collaboration Tools: apps or software programs that help team members work together more effectively
Projects Roadmap: a planning tool to visualize project goals, activities, and assignments on a timeline from start to deadline.
Resource Allocation: assigning and managing resources within various users
Unanticipated Shift: an unpredicted change in team responsibilities
Operational Risk: the uncertainties a team might face while developing the software project
Scheduling Tools: tools used to assign responsibilities to each team member and stay connected and updated about the project
System Integration: The process of creating a complex information system that may include designing or building an application.
Integrating: an act of bringing together all the components of the software project that complete the application
Barriers: obstacles preventing further progress in software development
Software Configurations: the components of a software that composes together to build the complete program.
Feasible: To carry out easily or conveniently.

Initial Project Plan & Schedule
Web Wireframe: It is a two-dimensional illustration of a page's interface that specifically focuses on space allocation and prioritization of content, functionalities available, and intended behaviors.
Preliminary Database Tables: Priority in preparing database tables for the software.
Screen Designs: The design of graphical user interface.

Requirements Analysis
UI/UX: User Interface and User Experience
Documentation: Evidence that serves as a record.
Table Interface: Interface tables are intermediate tables into which the data is inserted first

Override: To reject or cancel an action.
Concurrent Users: Number of existing and active users.

<u>*Initial Risk Assessment*</u>

1. Communication Issues:

Risk: Having poor communication among the group members poses a huge and unnecessary risk that can result in the project failing.
Solution: Choosing good **collaboration tools** and being clear in meetings about the processes that's being done by each member can help mitigate this risk.

2. Unclear Requirements:

Risk: Not having a clear view of the requirements from the start can result in inefficiency. It can cause a great loss of time if the members have already spent a huge amount of time on a project only to realize they misunderstood what was being asked.
Solution: Good listening skills go a long way toward mitigating this risk. Ask as many questions and do as much research as it takes to get a clear picture of the desired final product and its purposes. A clear, shared vision can prevent problems and provide inspiration for the team.

3. Poor Scheduling:

Risk: Poor scheduling can cause the team to not be able to complete required tasks by the deadline and can introduce all sorts of other conflicts in the project.
Solution: The project schedule must be accessible to every team member. Detailed planning is essential to creating a project schedule. There are many **project roadmap** tools that can prove helpful in staying on track.

4. Stretched Resources:

Risk: Resources include time, skills, money, tools and manpower. Lacking any one of these creates risk factors. For example, since the group size is limited to a maximum of four members, one member might have to take on many roles. Also, not all members are extremely skilled when it comes to all the tools required for the project.
Solution: The best way to mitigate this is to create a **resource allocation** plan. A resource allocation plan makes the best use of team resources while maximizing resource impact and supporting team goals.

5. Operational Changes:

Risk: Changes in company or team procedures, such as an **unanticipated shift** in team responsibilities, management changes, or new processes that your team must adjust to, are examples of **operational risk**. These factors can cause distractions, need process changes, and affect project deadlines.
Solution: The group needs to make sure the team is prepared for the change and has time to adjust through team meetings, **scheduling tools**, or additional training.

6. System Integration:

Risk: It may be necessary for the Group to work on the system offline before **integrating** it with the production system after it is complete and well tested. Unpredictable **barriers** may arise as a result of differing **software configurations**.
Solution: To guarantee seamless **system integration**, the Group must have as much information on the configuration as **feasible** as soon as possible.

<u>*Initial Project Plan & Schedule*</u>

- (Feb 23, 2022) Requirements and Analysis, Plan, and Schedule
- (March 16, 2022) Architecture Baseline - **web wireframe**
- (March 21, 2022) **Preliminary Database Tables**
- (March 28, 2022) **Screen Designs**, initial code
- (March 28, 2022) User Manual (update)
- (March 30, 2022) Design, Test Plan, Initial Code due date
- (March 30, 2022) Progress report presentation due date – one member presents
- (April 14, 2022) Final Code
- (April 18, 2022) User Manual (final)
- (April 20, 2022) MS PowerPoint (final)
- (April 25-27, 2022) Final Presentation – with User Manual, PowerPoint and completed system to "test drive"

Requirements Analysis

The system needs to meet the following functional requirements:

1. Web Interface
    a. Admin
        i. Edit status of bug (submitted, pending, approved, resolved, close) with date and time stamp
        ii. Allow admin to resolve or close bugs
    b. User
        i. Create ticket/bug
        ii. Space to share description of bug
        iii. Assign priority to bug
        iv. Feature to categorize bugs in areas of concern (database, **UI/UX**, **documentation**, etc.)
    c. Public Side
        i. Display status of bug in **table interface**
2. Database to store the tickets
    d. Must allow admin to modify, delete or add

The system may have the following functional requirements:
Undecided

1. User may attach files/images to ticket requests
2. Admin can **override** the priority of bug and give a reason if possible
3. Admin has the ability to reopen bugs, if needed
4. An interface for admin to search and sort bugs by date, priority, user, etc.

Optional

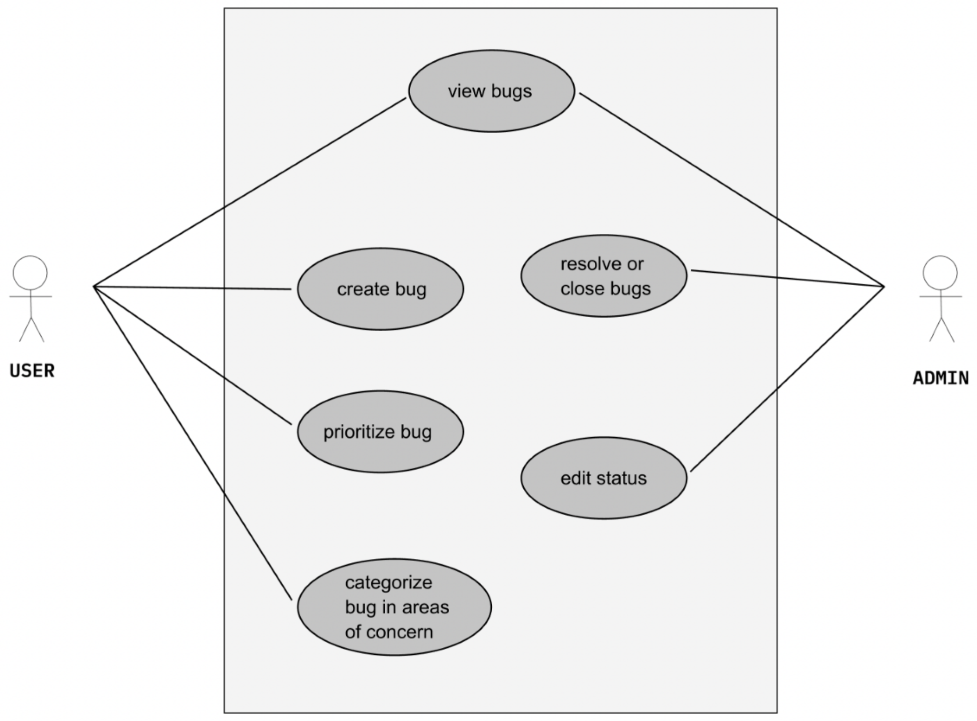
5. Login and logout from system
6. Admin side with the ability to assign tasks to a developer
7. The admin interface may not need to be web-based

Non-functional requirements are unspecified at this point. Foreseeable non-functional requirements may include up-time, reliability, number of **concurrent users** supported, response time of the system, etc.

Questions

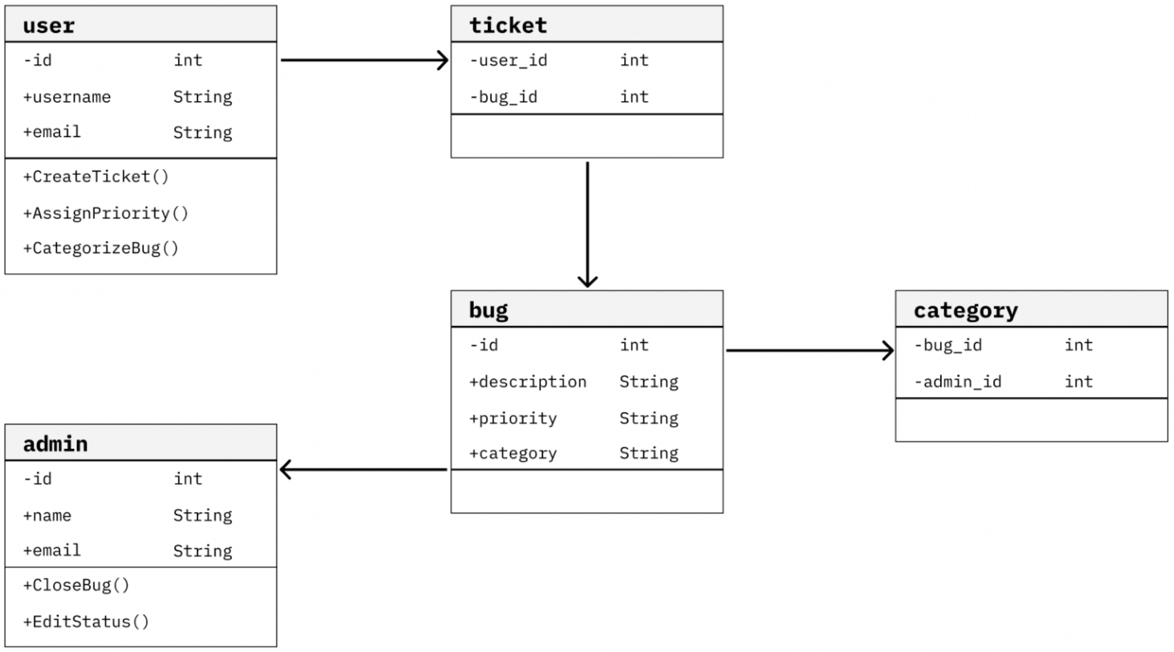
- What are the main tasks or functions that are performed by the actor?
  - Create ticket
- What system information will the actor acquire, produce or change?
  - Dashboard with tickets submitted, alongside the following ticket information: issue #, issue title, date created, ticket status, and priority. The actor will also be able to create a ticket with issue title, description and priority.
- Will the actor have to inform the system about changes in the external environment?
  - Yes, however, this isn’t required.
- What information does the actor desire from the system?
  - Ticket status
- Does the actor wish to be informed about unexpected changes?
  - Yes, the admin and developer will be able to comment under each ticket.

Use Case Diagram



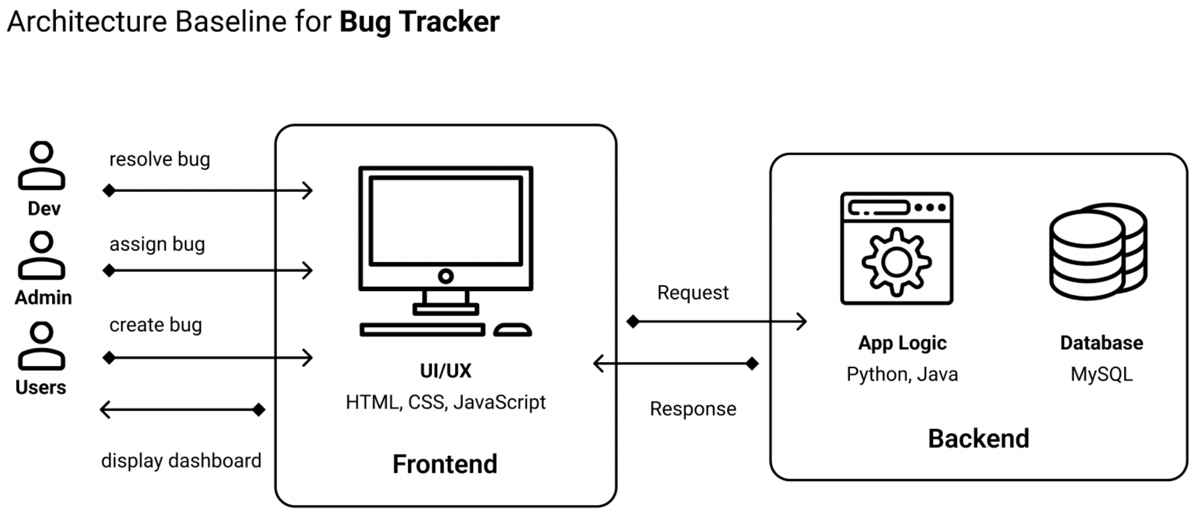
Actor: user
1. The user logs onto the Bug Tracker website.
2. The user enters his or her user ID.
3. The user enters their password.
4. The system displays the dashboard with all major function buttons.
5. The user selects the "create ticket" button.
6. The system displays the "create ticket" GUI.
7. The user inputs issue title, description and priority.
9. The user selects the "submit" button.
10. The system displays a confirmation window with the Issue #.
11. The system sends a ticket to the admin to assign.

Class Diagram



Architecture Baseline

General description of the application from an architectural view (skeleton of the application).



Design

The system consists of three functional requirements: a database to hold a library catalog of all the tickets submitted, an administrative interface that would allow for managing and changing the data, and a User Interface (UI) to expose this data to the user. Therefore, the programming languages being explored to develop the system and to achieve these requirements are Python, Java, JavaScript, HTML, CSS and MySQL. The front-end and back-end of the system has been distributed amongst the group to address each feature respectively.

Test Plan

User's Manual

System Overview:
Bug tracker is an application which allows the user to submit problem tickets which will be designated to the staff members to handle, according to the respective priorities and then allows the admin to edit the status of the bug. The bug tracking application works on Windows, and its operational status is under development.

System Requirements:
Windows OS

System Features:
The login page allows the user to enter their username and password. Upon clicking the login button the user can sign in into their account. Once there, user can create a ticket by following these steps:

- Report Issue
- Description Box
- Set priority
- Select category and Submit
- View bugs/status

Tools:
HTML, CSS, Javascript

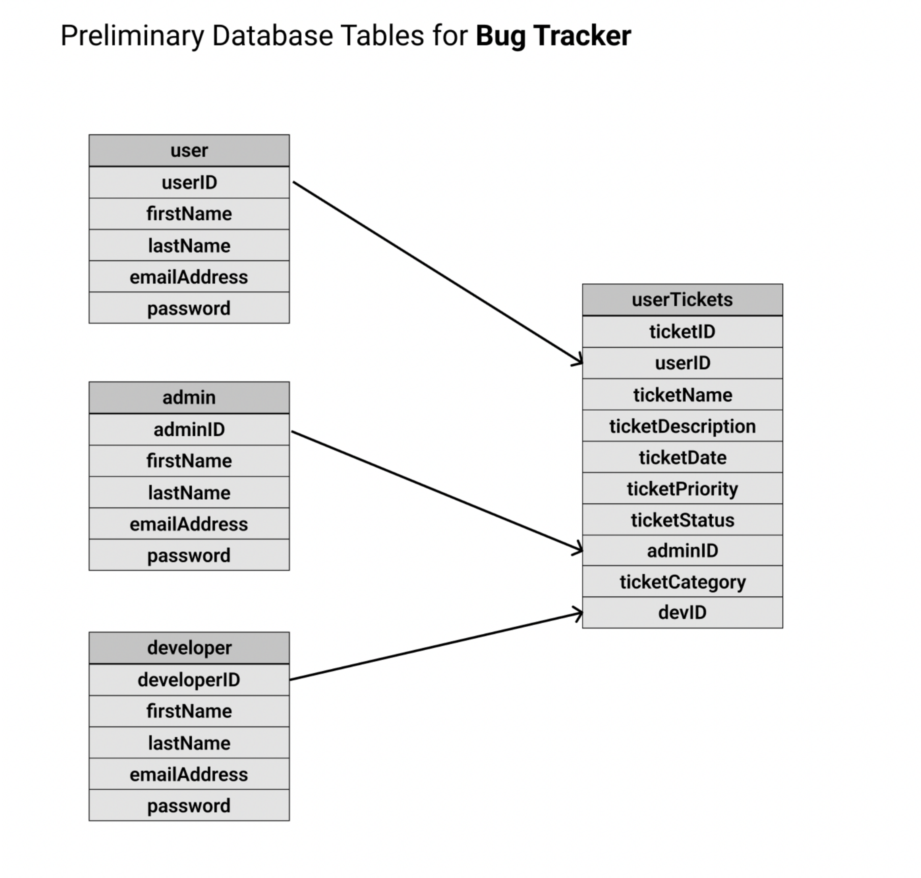
Testing:
Each member will be given a role for the testing. One of the members will report/create bugs and will test the end user side of the application. One of the members will take the role of an admin and will test the administrative part of the application that allows them to set the status of the bug and assign it to a staff. Other members will be the staff, who will receive and handle the bugs. This will be done in various iterations incrementally to find as many issues with the program and resolve them. Each iteration will result in an update in the program.

What is the expected input/output:
The expected input is the ticket/bug along with its description and priority. The expected output is then the status of the bug.

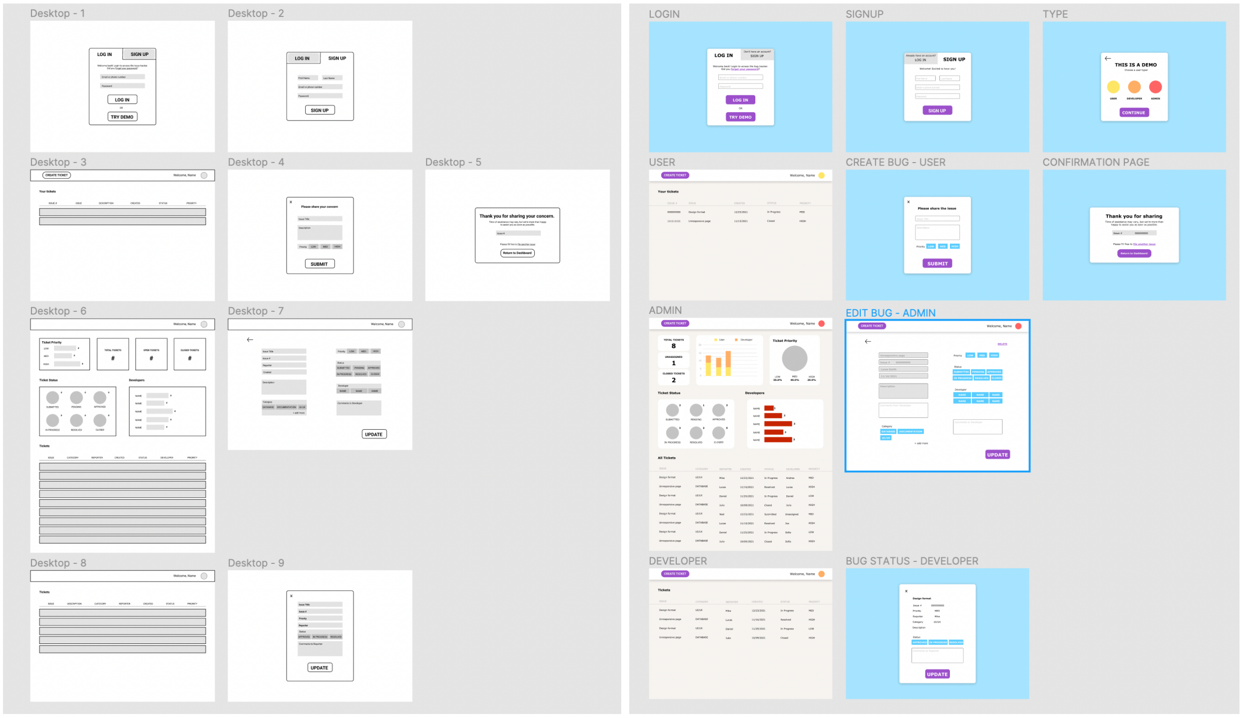
Successful criteria:
The program would be considered successful if the user is successfully able to report/create a bug/ticket and set its priority along with a description of the bug. And if it allows the admin to assign it to a staff. And if the staff is successfully able to see the ticket, its priority and description. And then once the problem is handled the admin should be able to set the status of the ticket successfully.

Initial Code

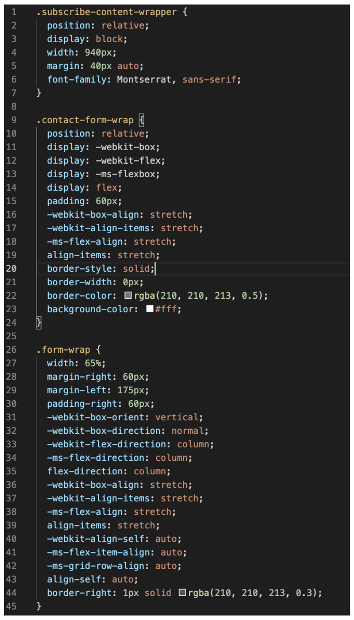
Preliminary Database Tables

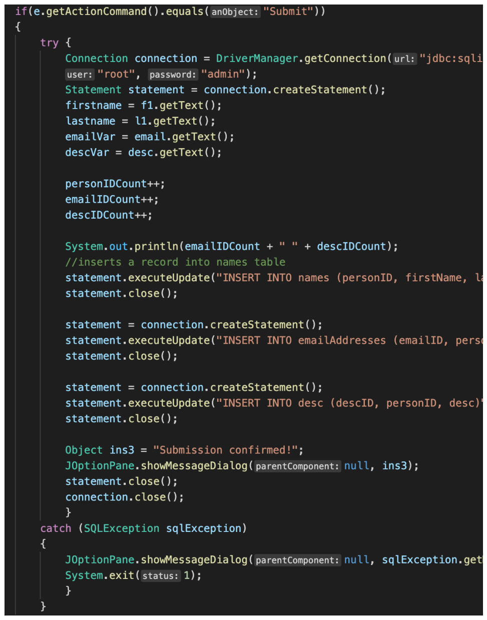


Screen Designs



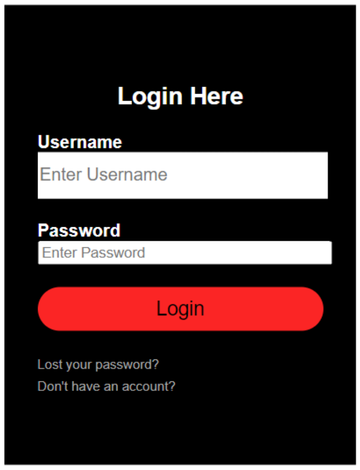
Code Sample


```
.subscribe-content-wrapper {
  position: relative;
  display: block;
  width: 940px;
  margin: 40px auto;
  font-family: Montserrat, sans-serif;
}

.contact-form-wrap {
  position: relative;
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
  padding: 60px;
  -webkit-box-align: stretch;
  -webkit-align-items: stretch;
  -ms-flex-align: stretch;
  align-items: stretch;
  border-style: solid;
  border-width: 0px;
  border-color: rgba(210, 210, 213, 0.5);
  background-color: #fff;
}

.form-wrap {
  width: 65%;
  margin-right: 60px;
  margin-left: 175px;
  padding-right: 60px;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  -webkit-flex-direction: column;
  -ms-flex-direction: column;
  flex-direction: column;
  -webkit-box-align: stretch;
  -webkit-align-items: stretch;
  -ms-flex-align: stretch;
  align-items: stretch;
  -webkit-align-self: auto;
  -ms-flex-item-align: auto;
  -ms-grid-row-align: auto;
  align-self: auto;
  border-right: 1px solid rgba(210, 210, 213, 0.3);
}
```


```
if(e.getActionCommand().equals(anObject: "Submit"))
{
    try {
        Connection connection = DriverManager.getConnection(url: "jdbc:sqli
        user: "root", password: "admin");
        Statement statement = connection.createStatement();
        firstname = f1.getText();
        lastname = l1.getText();
        emailVar = email.getText();
        descVar = desc.getText();

        personIDCount++;
        emailIDCount++;
        descIDCount++;

        System.out.println(emailIDCount + " " + descIDCount);
        //inserts a record into names table
        statement.executeUpdate("INSERT INTO names (personID, firstName, la
        statement.close();

        statement = connection.createStatement();
        statement.executeUpdate("INSERT INTO emailAddresses (emailID, perso
        statement.close();

        statement = connection.createStatement();
        statement.executeUpdate("INSERT INTO desc (descID, personID, desc)"
        statement.close();

        Object ins3 = "Submission confirmed!";
        JOptionPane.showMessageDialog(parentComponent: null, ins3);
        statement.close();
        connection.close();
        }
    catch (SQLException sqlException)
    {
        JOptionPane.showMessageDialog(parentComponent: null, sqlException.get
        System.exit(status: 1);
        }
    }
```

ECT.

Login page



Login Here

Username

Enter Username

Password

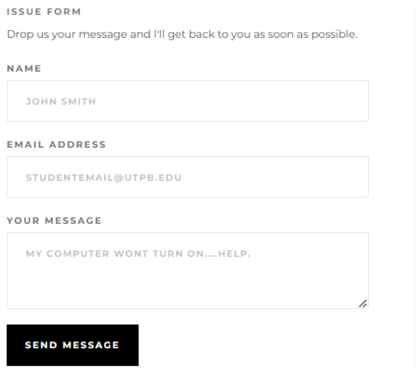
Enter Password

Login

Lost your password?

Don't have an account?

Bug Submission



ISSUE FORM

Drop us your message and I'll get back to you as soon as possible.

NAME

JOHN SMITH

EMAIL ADDRESS

STUDENTEMAIL@UTPB.EDU

YOUR MESSAGE

MY COMPUTER WONT TURN ON....HELP.

SEND MESSAGE