Code Inspection Report

*Project Planner*

**Client**

Doug Craig

**Team Hailstorm**

Hemang Bhatt

Aidan Gray

Cameron Hensel

Jacob Lutz

Phuoc Nguyen

Nirav Pancholi

29 November, 2017

|  |  |
| --- | --- |
| **Table Of Contents** | Page |
| 1. Introduction | 2 |
| 1.1 Purpose of this Document | 2 |
| 1.2 References | 2 |
| 1.3 Coding and Commenting Conventions | 2 |
| 1.4 Defect Checklist | 2 |
| 2. Code Inspection Process | 3 |
| 2.1 Description | 3 |
| 2.2 Impressions of the Process | 4 |
| 2.3 Inspection Meetings | 4 |
| 3. Modules Inspected | 5 |
| 4. Defects | 6 |
| Appendix A - Peer Review Sign-off | 8 |
| Appendix B - Document Contributions | 10 |

#### 

### **1. Introduction**

1.1 Purpose of This Document

The purpose of this document is to give an outline of the coding standards used throughout the implementation process of the software, as well as keep track of meetings in which we reviewed our code. Additionally, any defects or flaws that were found are documented.

1.2 References

This document makes references to the *Project Planner* System Requirements Specification Document, Software Design Document, and User Interface Design Document.

1.3 Coding and Commenting Conventions

The coding and commenting conventions that we adhered to were decided by the majority of the implementation team. We used standard conventions, such as camel-case for variables and capital first letters for functions. For the HTML CSS code, the standard formatting was used which is automatically done by the IDE. For the MySQL database, attributes of each entity have capitalized first letter and underscores as spaces.

1.4 Defect Checklist

Defect Categories

1. Coding Conventions

* These types of defects refer to code that does not follow the agreed upon style and conventions for that particular programming language.

1. Logic Errors

* These types of defects refer to errors in the logic of the code, such as incorrect passing of variables, improper array manipulation, etc.

1. Security Oversights

* These types of defects refer to compromises in security aspects of the system, such as being vulnerable to SQL injections.

1. Commenting

* These types of defects are purely documentation-based: whether the code has enough comments, has useful comments, and does not have too many comments.

Defect Checklist

|  |  |
| --- | --- |
| **Category** | **Defect** |
| Coding Conventions | Database variables not properly capitalized |
| Security Oversights | SQL injection susceptibility was not handled in our login page |
| Logic Errors | Data fields were susceptible to errors from NULL data |
| Logic Errors | When entering data into the data fields, edge cases were not properly implemented. |
| Logic Errors | Improper CSS code obscured information on webpage |
| Logic Errors | Creation of account redirects to wrong page |
| Logic Errors | Employee cannot delete account |
| Logic Errors | Resizing web browser causes UI defects |
| Coding Conventions | Some of the functions created were not generic or abstract |
| Commenting | Some of the PHP modules had terse commenting |
| Logic Errors | Assigning a user to a phase does not update the assigned users list |

**2. Code Inspection Process**

2.1 Description

Our code was reviewed using a fairly non-standard process. The majority of code inspection was performed by the team member who coded that particular module, with other various team members looking over the code quickly during meetings or over GitHub and Slack. The reason for this was that our team member schedules did not line up well, making in-person meetings throughout the development cycle quite difficult. Having the ability to discuss and share code using GitHub and Slack was very helpful, although certainly not ideal.

In addition to simply discussing the code inspections through Slack, we also created separate channels solely for this purpose. This allowed us to keep our conversations about implementation and documentation separate from fixing defects and debugging. Finally, we all came together during our few meetings to discuss overall code progress and general coding conventions.

2.2 Impressions of the Process

We believe that, while not ideal, our process did allow us to complete the inspection at a satisfactory level of confidence. Also, we inspected the code by going through the modules as we worked on them, not in any particular order, which could have been performed more methodically.

Given more time, and more flexible schedules, we would have performed the code inspection process by breaking off into teams that include the member who coded the module and another member who did not. This would allow us to have one person who knows the code well looking for minute details, while another person who does not know the code well can see the defects “hiding in plain sight.” Our code that was reviewed by the most people likely have the least number of defects, while our code that was done by one person and not altered again will likely have more defects.

2.3 Inspection Meetings

**Meeting #1**

Date: 11/22/17

Location: CMSC 447 Classroom

Start Time: 1pm

End Time: 3pm

Participation: Full group participation

Units Covered/Description: Phase creation implementation, employee database implementation

**Meeting #2**

Date: 11/27/17

Location: CMSC 447 Classroom

Start Time: 1:30pm

End Time: 2:30pm

Participation: Full group participation

Units Covered/Description: Database coding conventions, CSS formatting of the webpage

**Meeting #3**

Date: 11/28/17

Location: CMSC 447 Classroom

Start Time: 1pm

End Time: 5:15pm

Participation: Full group participation

Units Covered/Description: Code inspection report

**Meeting #4**

Date: 11/29/17

Location: CMSC 447 Classroom

Start Time: 1:30 pm

End Time: 8pm

Participation: Full group participation

Units Covered/Description: Code inspection report, thorough testing of all modules and user interface, testing report, SQL injection testing, edge case handling

**3. Modules Inspected**

The following modules have been inspected by various members of the team, as well as brief descriptions of each.

Code Description Table

|  |  |  |
| --- | --- | --- |
| **#** | **File** | **Brief Description** |
| 1 | classes/Session.class.php | PHP file for the Session class |
| 2 | classes/User.class.php | PHP file for the User class |
| 3 | scripts/dbconfig.ini | Database configuration file |
| 4 | scripts/home.php | Home screen script |
| 5 | scripts/index.php | Initial script to mask file structure; also checks database connection and redirects to first-time setup if necessary |
| 6 | scripts/login.php | Login and Session Start script |
| 7 | scripts/logout.php | Logout and Session Close script |
| 8 | scripts/style.css | CSS script for project GUI |
| 9 | scripts/Project/Add\_Client.php | Script to add new client to a project |
| 10 | scripts/Project/Create.php | Script to create a new project |
| 11 | scripts/Project/Delete.php | Script to delete a project |
| 12 | scripts/Project/Edit.php | Script to edit a project |
| 13 | scripts/Project/Style.css | CSS script for the project page |
| 14 | scripts/Project/View.php | Script to view a project description and outline |
| 15 | scripts/Project/Phases/Create.php | Script to create a phase in a project |
| 16 | scripts/Project/Phases/Delete.php | Script to delete a phase from a project |
| 17 | scripts/Project/Phases/Edit.php | Script to edit a phase |
| 18 | scripts/Project/Phases/Edit\_User\_Assignment.php | Script to edit which users are assigned to a phase |
| 19 | scripts/Project/Phases/User\_Search.php | Script to search the database of users, to add to a phase |
| 20 | scripts/Project/Phases/View.php | Script to view a phase description |
| 21 | scripts/Project/Tasks/Create.php | Script to create a new task |
| 22 | scripts/Project/Tasks/Delete.php | Script to delete a task |
| 23 | scripts/Project/Tasks/Edit.php | Script to edit a task |
| 24 | scripts/Project/Tasks/View.php | Script to view a task |
| 25 | scripts/Users/Change\_Role.php | Script to change a user’s role |
| 26 | scripts/Users/Create.php | Script to create a new user |
| 27 | scripts/Users/Delete.php | Script to delete a user |
| 28 | scripts/Users/Edit.php | Script to edit a user account |
| 29 | scripts/Users/View.php | Script to view a user account |
| 30 | scripts/FTS/FTS1.php | Script to setup the database connection |
| 31 | scripts/FTS/FTS2.php | Script for admin creation |

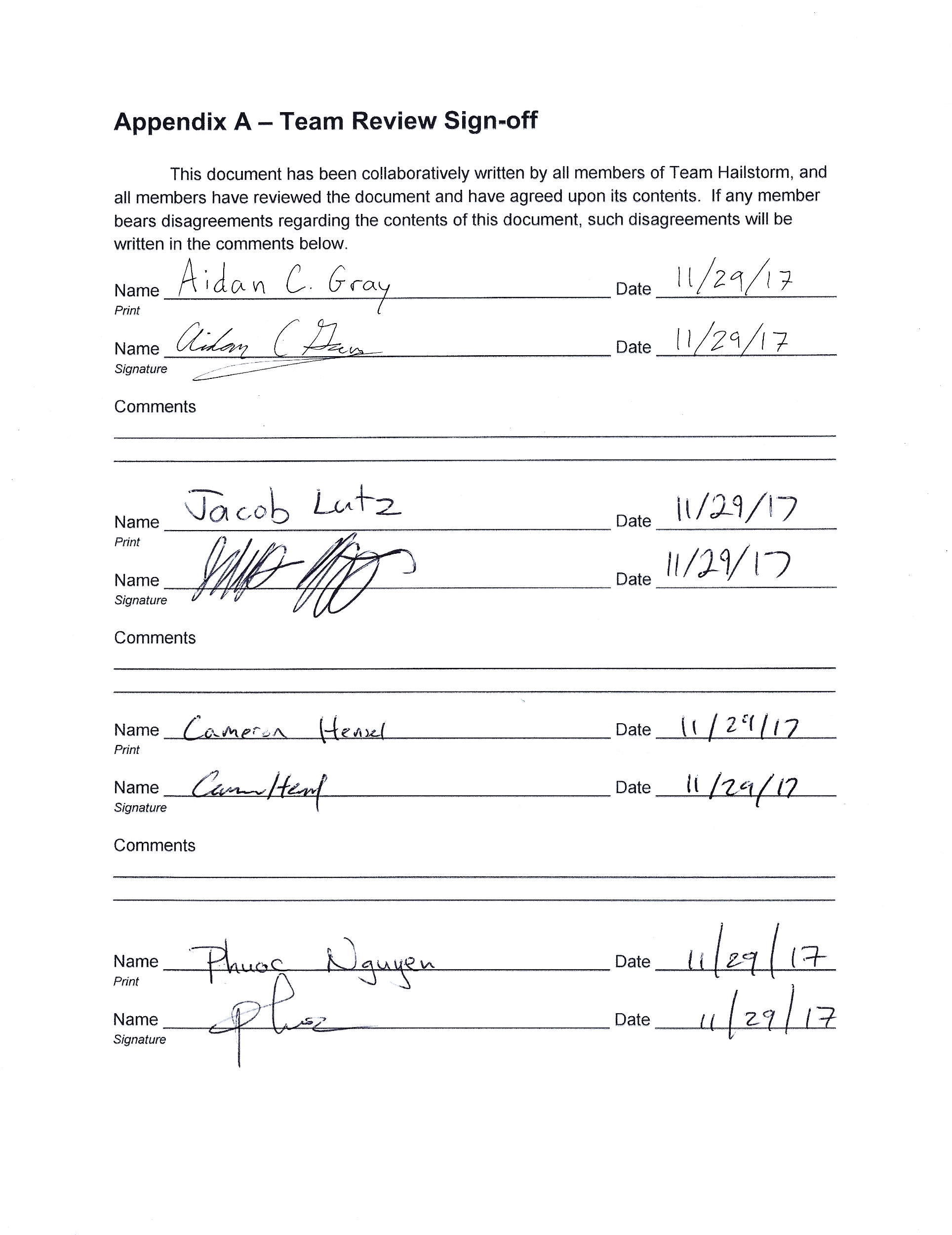
### **4. Defects**

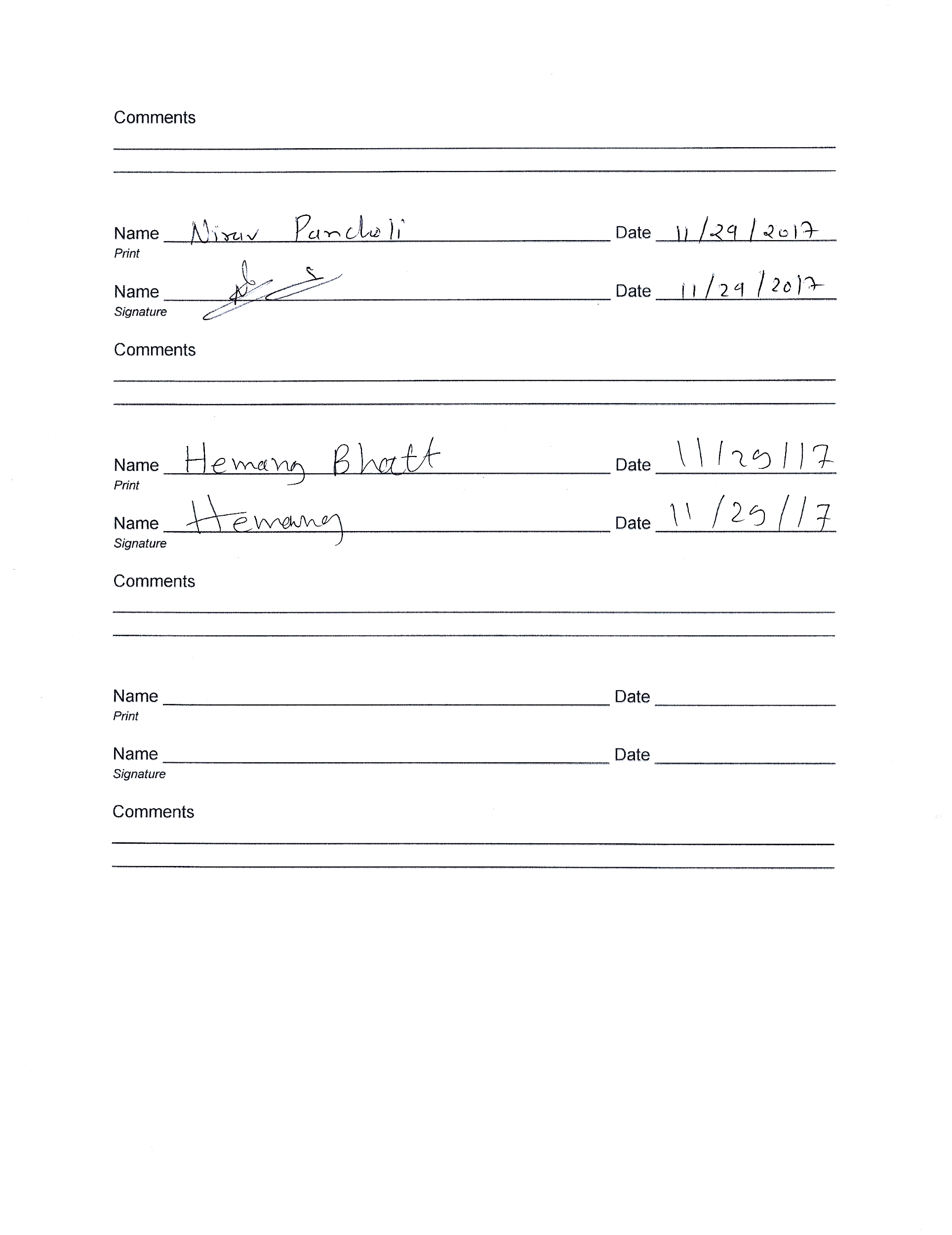
This table lists the defects found in our program.

|  |  |  |  |
| --- | --- | --- | --- |
| **Defect Category** | **Location** | **Comments** | **Fixed** |
| User Friendliness/ Correctness | scripts/Project/Phases/Create.php | The cancel button does not return the user to the previous screen. | No |
| Correctness | scripts/Project/Phases/Edit\_User\_Assignment.php | Assigning a user to a phase does not add them to the list of assigned users. | Yes |
| User Friendliness/ Correctness | scripts/style.css | Resizing the web browser causes resizing issues. | No |

**Appendix A – Team Review Sign-off**

This document has been collaboratively written by all members of Team Hailstorm, and all members have reviewed the document and have agreed upon its contents. If any member bears disagreements regarding the contents of this document, such disagreements will be written in the comments below.





**Appendix B – Document Contributions**

All team members contributed to the code inspections and review of the document. Additionally, Aidan Gray oversaw the primary creation of this document, with minor changes made by Jacob Lutz.