# 

# 

# 

System Design Document

*Project Planner* [Working Title]

**Client**

Doug Craig

**Team Hailstorm**

Hemang Bhatt

Aidan Gray

Cameron Hensel

Jonathan Love

Jacob Lutz

Phuoc Nguyen

Nirav Pancholi

25 October, 2017

|  |  |
| --- | --- |
| **Table of Contents** | Page |
| 1. Introduction | 2 |
| 1.1 Purpose of this Document | 2 |
| 1.2 References | 2 |
| 2. System Architecture | 3 |
| 2.1 Architectural Design | 3 |
| 2.2 Decomposition Description | 4 |
| 3. Persistent Data Design | 5 |
| 3.1 Database Descriptions | 5 |
| 4. Requirements Matrix | 7 |
| Appendix A - Agreement between Customer and Contractor | 14 |
| Appendix B - Team Review Sign-Off | 16 |
| Appendix C - Document Contributions | 18 |

## 

## **1. Introduction**

### 1.1 Purpose of This Document

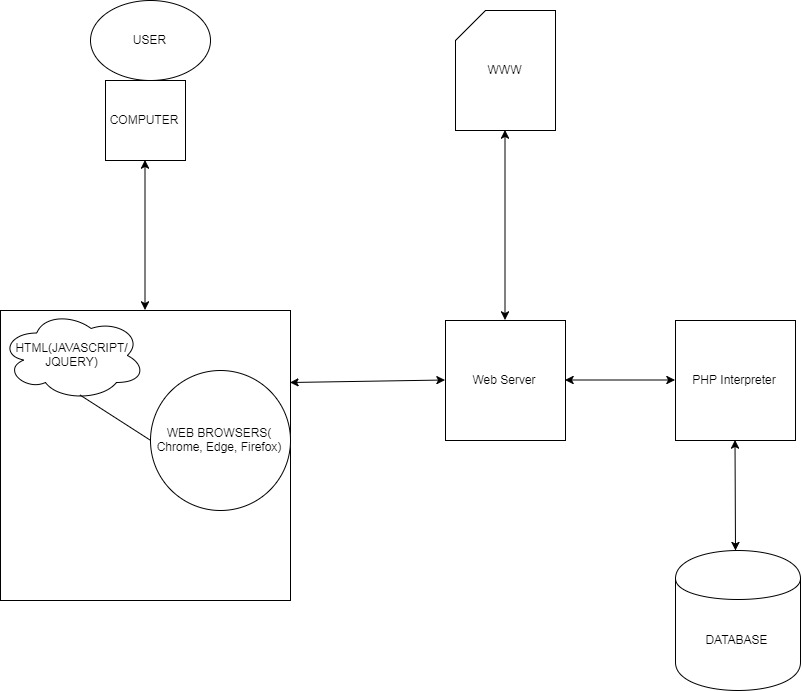
The purpose of this document is to describe the design and structure of the *Project Planner* application. This document covers topics such as the system architecture, the application file structure, and the database schema. It is intended for readership by members of Team Hailstorm as well as the client, Doug Craig.

### 1.2 References

This document makes references to the *Project Planner* System Requirements Specification document as well as the User Interface Design document.

## **2. System Architecture**

### 2.1 Architectural Design



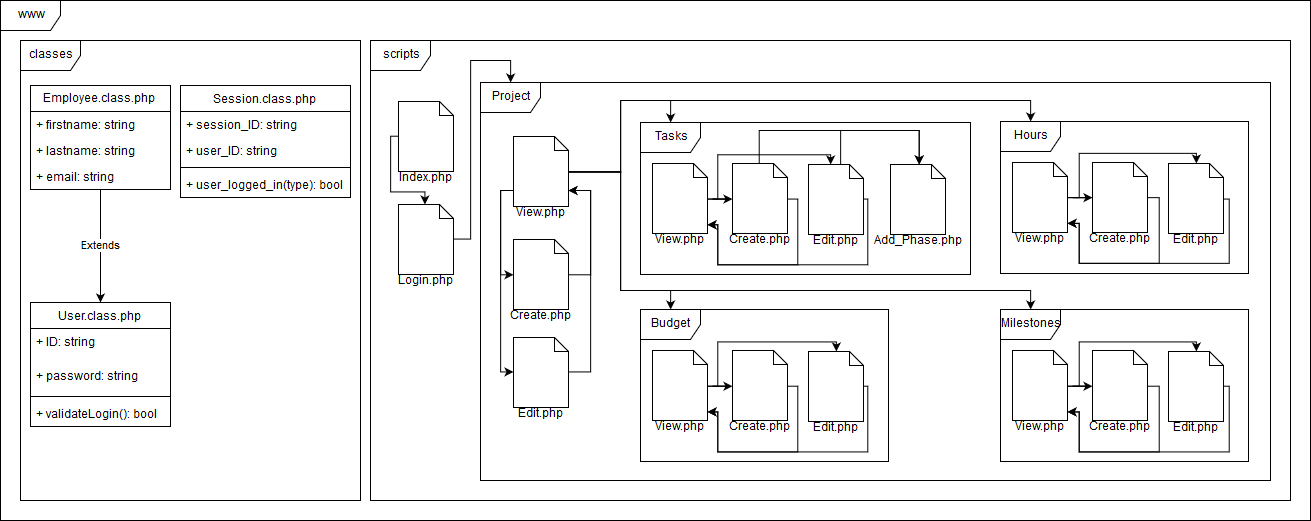
*Figure 1:* Project Planner *component diagram*

The *Project Planner* application will be built using PHP and hosted on a web server. This means that there are only three primary components to the server side of the application: the web server itself, the database, and the PHP scripts.

The web server will be hosted within the client’s company and will be accessible within the company’s network. PHP is flexible with its hosting as long as the server can run a PHP interpreter, so the structure of the web server need not be finalized at this stage of development. The database will contain all of the actual data for the application, with only the user and session information requiring classes. Since most of the data is only sent to the database or received by the browser for viewing, it will not need to be stored locally. The remainder of the application, the PHP scripts contained within the “www” server directory, will handle viewing and managing projects and attributes. PHP scripts are executed on the server itself; only the generated output of this, in the form of HTML, JavaScript, and JQuery, reaches the internet browser.

The database and its contents will be explained in further detail in the Persistent Data description in Section 3.1 of this document. Further information on the user interface and the HTML output of the application can be found in the *Project Planner* User Interface Description document. The PHP classes and scripts, and their file structure, will be explained in further detail below, in Section 2.2.

### 2.2 Decomposition Description



*Figure 2:* Project Planner *decomposition diagram*

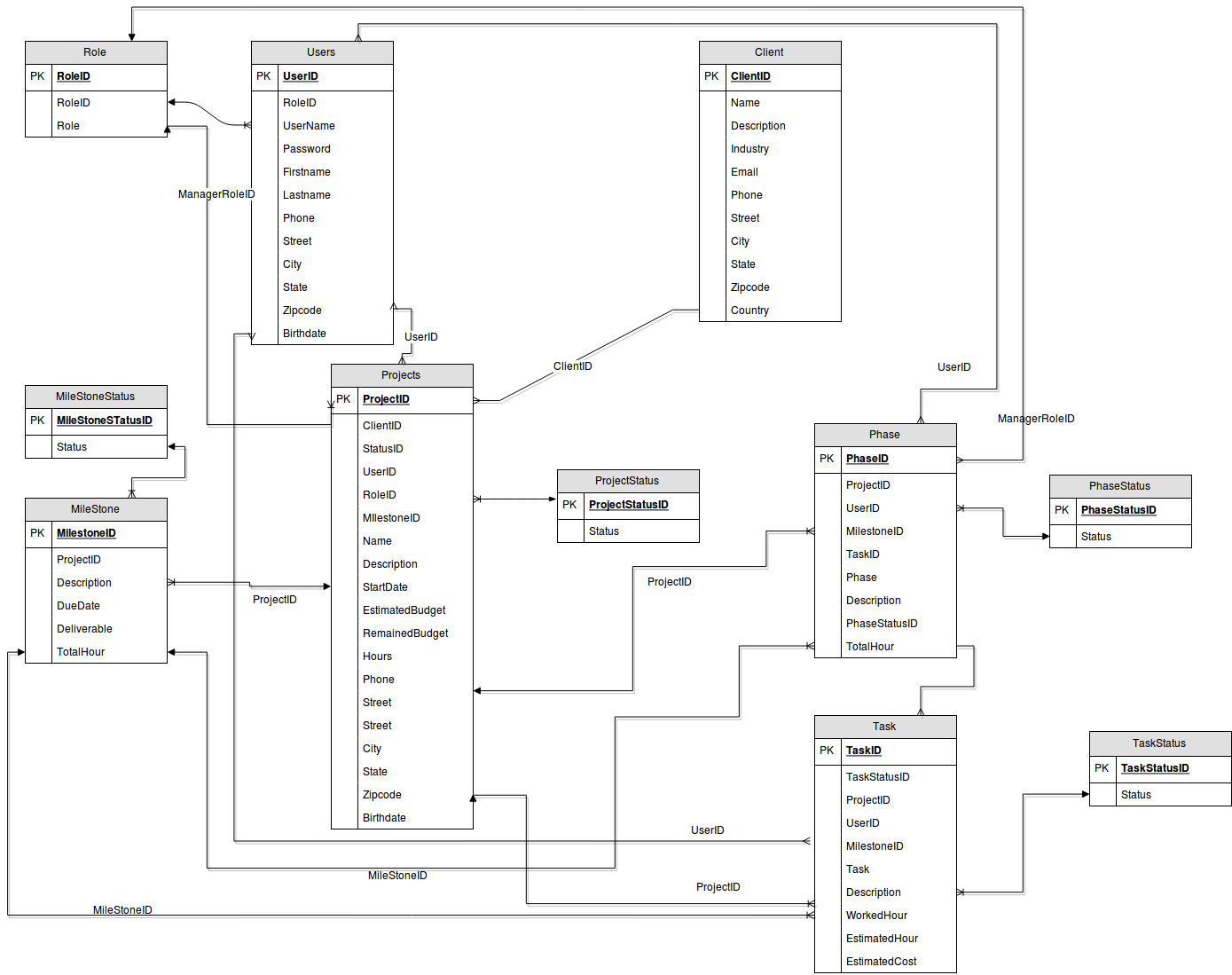
The project’s code consists of classes and scripts. Since most of the data will be stored in and retrieved from the database when needed, only three classes are needed for persistent data. The Employee class receives information from the Login screen and uses that information to retrieve the user logging in as well as establish a session. The User class remains constant throughout the session, containing data gathered by the Employee class. The User class also handles querying the database to authenticate the login. The Session class simply stores a unique identifier for the session.

The scripts consist of PHP files separated into directories for which the files are associated. The directories represent the project level itself and each of the project’s attributes - tasks and phases, budget, worked hours, and milestones. At the highest level, the user connects to *Login.php* (*Index.php* merely hides the system’s file structure from the user machine). After logging in, the user is directed to View.php in the Project directory. Managers receive full navigation rights throughout the system, whereas employees can access only *Project / Tasks / View.php* and *Project / Hours / View.php* from *Project / View.php*. To ensure this difference in privileges between managers and employees, *Project / View.php* will appear different depending on the role of the user logged in.

Managers can create, edit, and delete projects; the system contains a separate PHP file for handling each process excluding deletion. Managers navigated from the level’s *View.php* to *Create.php* or *Edit.php* depending on the necessary action. Managers can also access the *View.php* files of all subdirectories, each representing one of the project’s attributes. These attributes’ file structures mimic that of the *Project* directory structure, and the process for adding or editing attributes is the same for each. No separate page is necessary for deletion because *View.php of* each attribute handles that process. Additionally, the *Task* subdirectory contains an additional *Add\_phase.php* for adding task phases in the event the phase does not already exist.

## **3. Persistent Data Design**

### 3.1 Database Descriptions



*Figure 3:* Project Planner *database schema*

The *Project Planner* database consists of different data tables that collect information about each entity in the application. Further information on each table is as follows:

1. Role: (Primary key: RoleID)

The Role table holds the information about the user role according to their privilege. RoleID is used to divide functionalities between manager and employee. Many users can have one role, no user can have more than one role.

2. Users: (Primary Key: UserID, Foreign key(s): RoleID)

The Users table holds the basic information about the user, including first name, last name, RoleID, username, password, etc. It is connected to the Role table via a many-to-one relation.

3. Client: (Primary Key: ClientID)

The Client table holds all the basic information about the client. Additionally, it contains a

description of the client’s requirements or additional comments on a project(s).

4. Projects: (Primary Key: ProjectID, Foreign key(s): RoleID, UserID, ClientID, TaskID,

PhaseID)

The Projects table holds all the necessary information of a project. It is one of the most critical tables in the database, since it is connected with the majority of all other tables. It has a many-to-one relationship with Role, a many-to-many relationships with Users, and a many-to-one relationship with Client.

5. ProjectStatus: (Primary Key: ProjectStatusID, Foreign key(s): ProjectID)

ProjectStatus shows the current status of the project. It has a one-to-many relationship with the Projects table.

6. Milestone: (Primary Key: MilestoneID, Foreign key(s): ProjectID)

Milestone shows the milestone(s) associated with a particular project. It contains a brief description of what to achieve in that milestone and the deadline of the milestone. The Milestone table has a many-to-one relationship with Projects.

7. MileStoneStatus: (Primary Key: MilestoneStatusID, Foreign key(s): MileStoneID)

MileStoneStatus shows the current status of the milestone. It has a one-to-many relationship with MileStone table.

8. Phase: (Primary Key: PhaseID, Foreign key(s): ProjectID, UserID, MilestoneID, TaskID)

The Phase table holds the necessary information about the phases of the current project. For example, if Project A is in the “Implementation” phase then It would have all necessary information about that phase. Phase has a many-to-many relationship with Users, many-to-one with Project, one-to-many with Tasks, and many-to-one with MileStone.

9. PhaseStatus: (Primary Key: PhaseStatusID, Foreign key(s): PhaseID)

PhaseStatus shows the current status of a particular phase. It has a one-to-many relationship with Phase.

10. Task: (Primary Key: TaskID, Foreign key(s): ProjectID, PhaseID, UserID, MileStoneID)

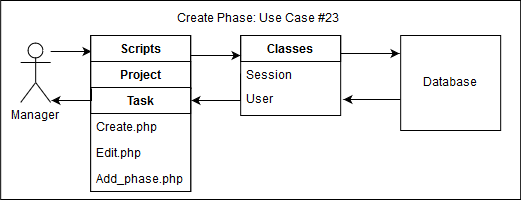
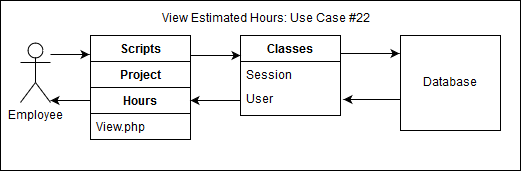
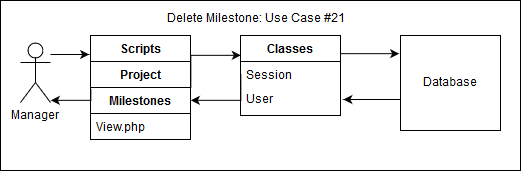
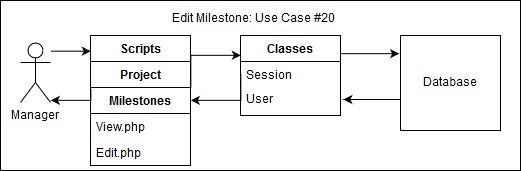
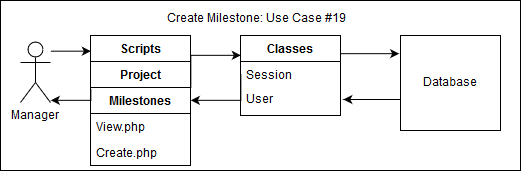
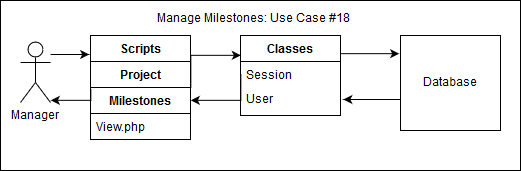
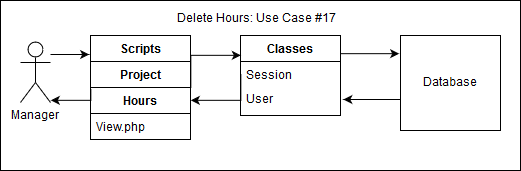
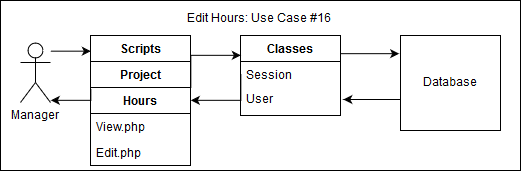
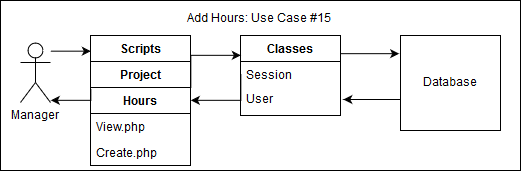
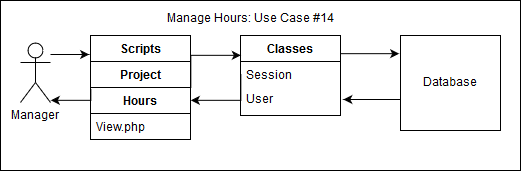
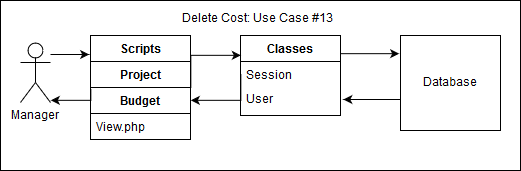
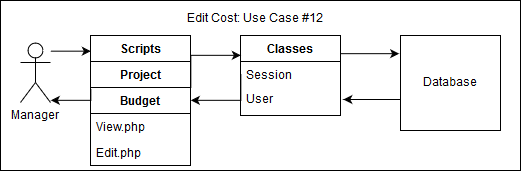
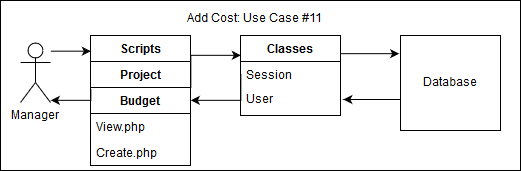
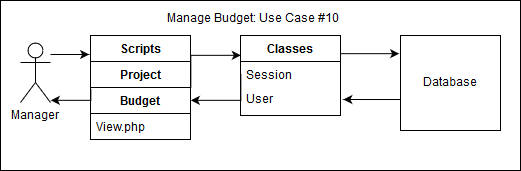
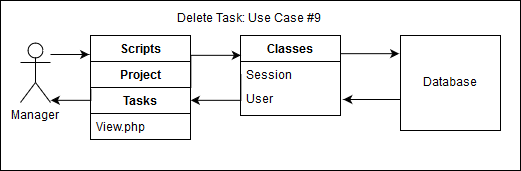
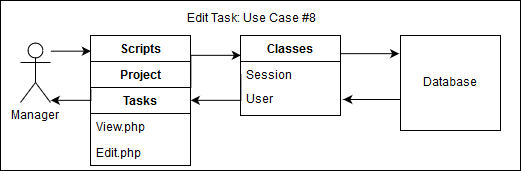
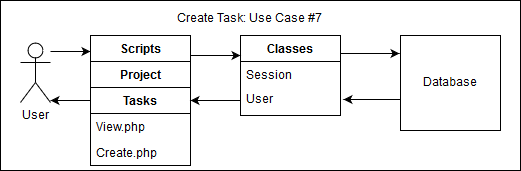
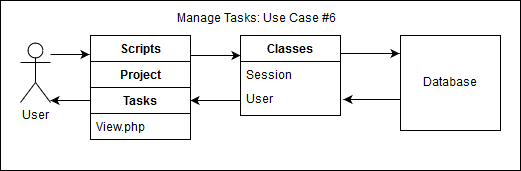
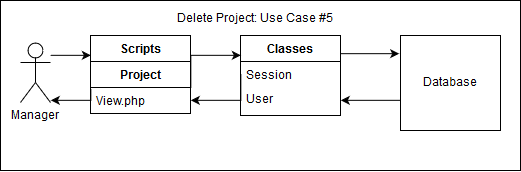
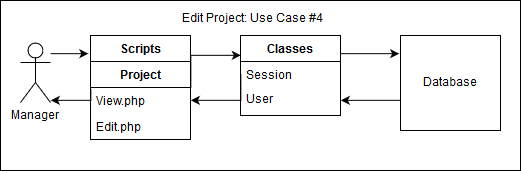
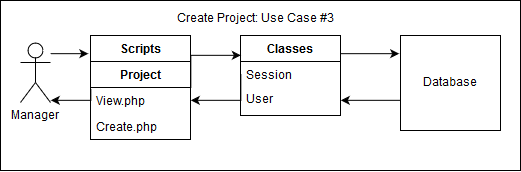
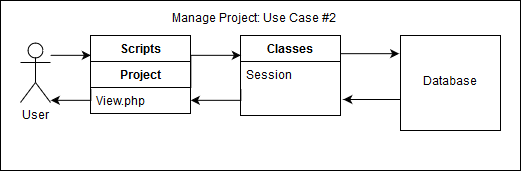
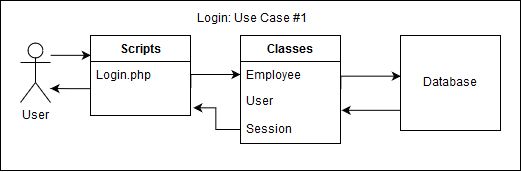
Task holds all the information about a particular task in a specific phase. It shows what user is associated with that task, shows the estimated hours, and etc. Task has a many-to-one relationship with Phase, Milestone, Projects, and Users.

11. TaskStatus: (Primary Key: TaskStatusID, Foreign key(s): TaskID)

TaskStatus shows the current status of the given task. It has a one-to-many relationship with Task table.

## **4. Requirements Matrix**

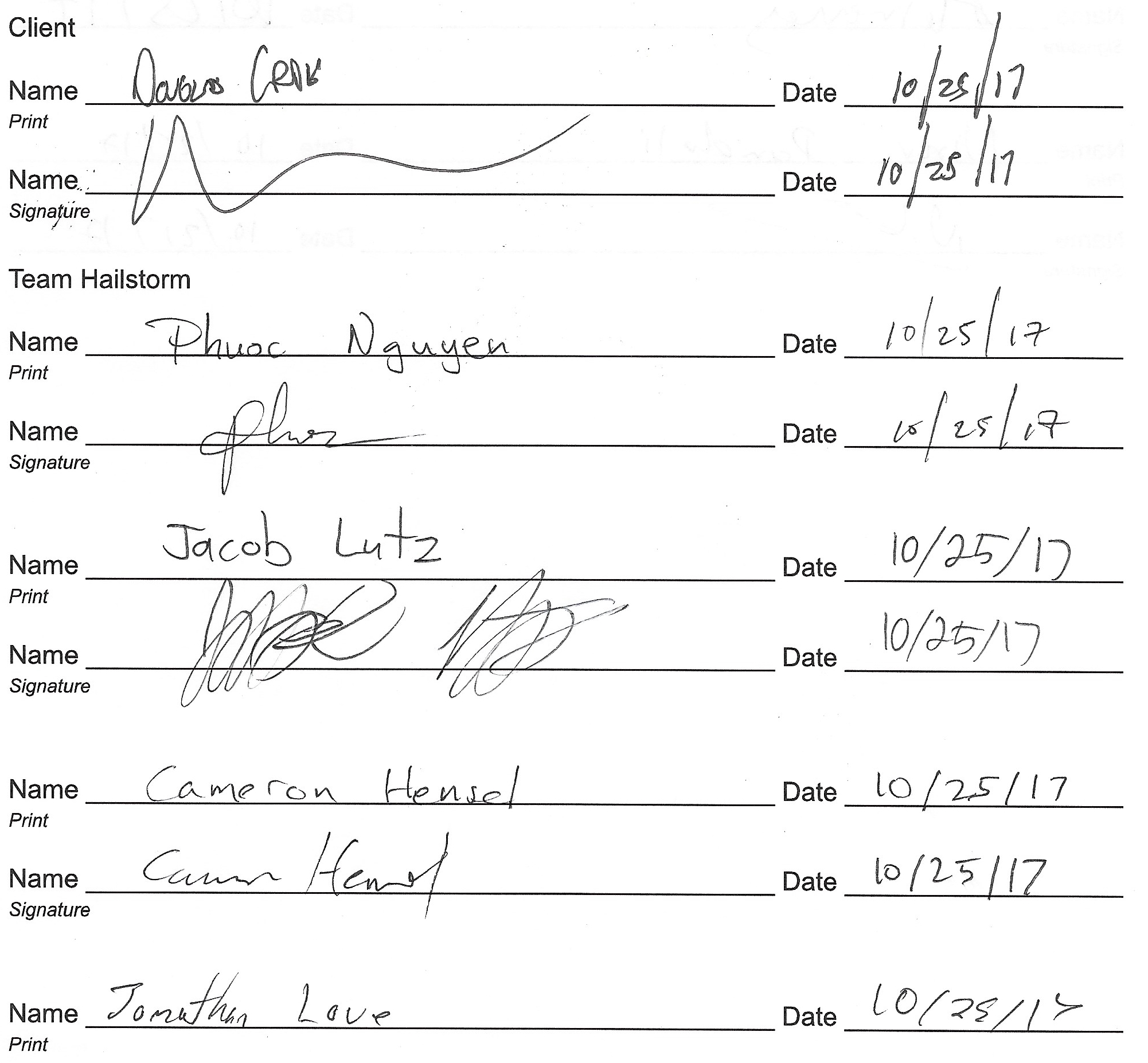
Please refer to the *Project Planner* system requirements specification document for detailed information on each of the corresponding use cases in the requirements matrix below.



*Figure 4:* Project Planner *requirements matrix*

## **Appendix A – Agreement Between Customer and Contractor**

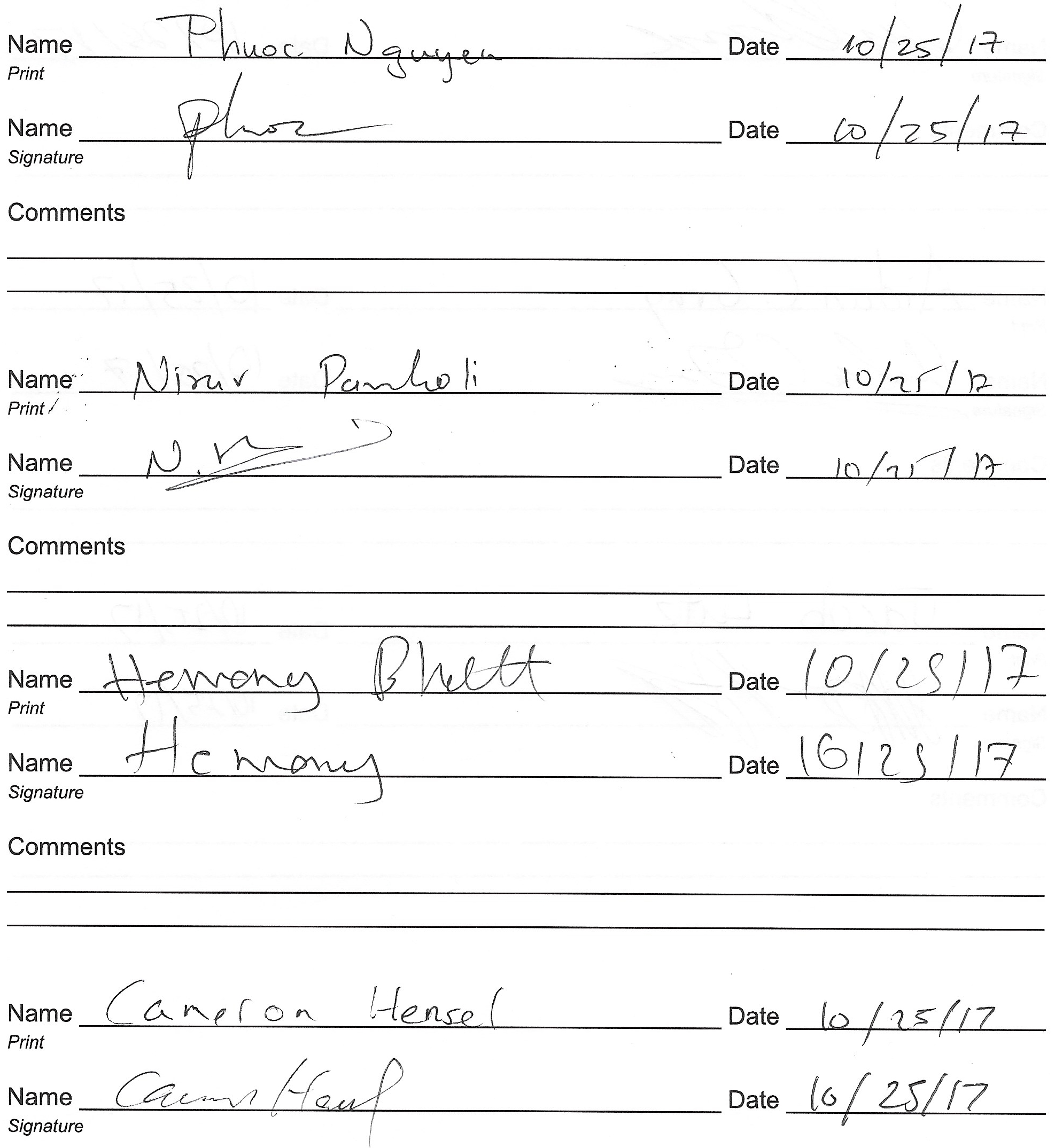
The client agrees to a *Project Planner* web application utilizing a database for project scope estimation, tracking projects and their attributes, and report generation. See the *Project Planner* system requirements specification document for more information. More features may be included in future iterations of the product.

In the event that requirements of the product are changed in the future, a new version of this document will be drafted and presented to the client for review. Upon approval, the draft will be signed off by the client and Team Hailstorm.

## 

## 

## **Appendix B – 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 C – Document Contributions**

All team members contributed to the text descriptions of this document. Additionally, Phuoc Nguyen designed the component architecture diagram and the database schema, and Jacob Lutz designed the decomposition diagram and the requirements matrix as well as edited the text descriptions for grammar consistency.