**CST-361 - Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | *Milestone 1 Weather Collector App using Raspberry Pi* | |
| **Date:** | *12-2-2018* | |
| **Revision:** | *1.99* | |
| **Team:** | 1. Jerran Fredricks | |
| 1. Ramon Leon | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *Factory Design Patter* | *Jerran/Joe* | *4* | *0* | | *Login and Registration Revision* | *Joe* | *3* | *0* | | *DTO.* | *Jerran\Joe* | *0* | *0* | | *Datagrid* | *Jerran/Joe* | *2* | *0* | | *Documentation* | *Jerran/Joe* | *1* | *0* | | *Visual cleanup.* | *Jerran/Joe* | *1* | *0* | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | https://Skie@bitbucket.org/Skie/weathervane.git | |
| **Loom URL:** | https://www.useloom.com/share/d1468e1bd636474bba2a96d681720d2f | |
| **Peer Review:** | *Y* | We acknowledge that our team has reviewed this report and we agree to the approach we are all taking. |

**Planning Documentation**

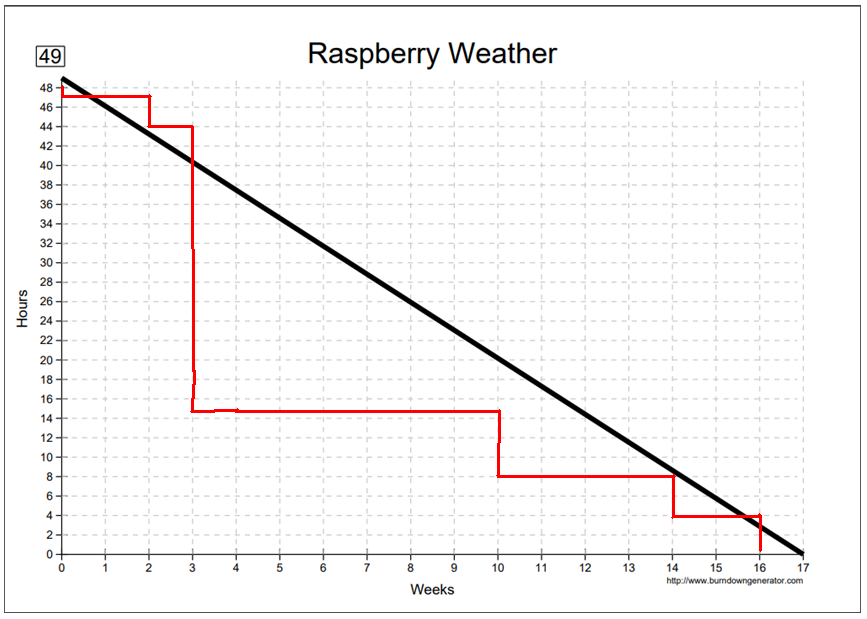
**Agile Scrum Product Backlog:**

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Type | Status | Estimate Hours |
| Login/Registration | Requirement | Finished | 2-4 |
| Database | Requirement | Finished | 6-8 |
| Raspberry Setup | Requirement | Finished | 2-4 |
| Rest API | Requirement | Finished | 6 |
| Bus API | Requirement | Finished | 6 |
| User Interface | Requirement | In Progress | 2-4 |
| Admin Functions | Requirement | Out of Scope | 4-5 |
| Cleanup/Optimization | Requirement | Finished | 12 |
| Iot Application | Requirement | Finished | 2 |

**Agile Scrum Sprint Backlog:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tasks | Milestone1 | Milestone2 | MileStone3 | MileStone4 | MileStone5 | MileStone6 |
| Login/Registration | **1** | **0** | **0** | **0** |  | **0** |
| Database | **0** | **0** | **1** | **0** | **0** | **0** |
| Raspberry Setup | **0** | **0** | **2** | **25** | **0** | **0** |
| Rest API | **0** | **0** | **0** | **1** | **7** | **2** |
| Bus API | **0** | **0** | **0** | **0** | **0** | **0** |
| User Interface | **0** | **0** | **0** | **0** | **12** | **0** |
| Admin Functions | **0** | **0** | **0** | **0** | **0** | **0** |
| Cleanup/Optimization | **0** | **0** | **0** | **4** | **0** | **3** |

**Agile Scrum Burn Down Chart:**

****

**Agile Retrospective Results:**

|  |
| --- |
| **What Went Well** |
| We had more done than we originally thought. |
|  |

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
| There were portions of our requirements that were not labeled correctly. | Talked to Mark Reha to clarify. | 12-9-18 |
|  |  |  |
|  |  |  |

**Design Documentation**

**Install Instructions:**

By cloning the provided GIT repository, all of the classes and files needed will be located within the Java Resource/src file under the appropriate packages.

Create a new Dynamic Web Project and then create two new packages: model and controller. Take the corresponding files from the repository and place them within the new packages.

* model : User.java
* controller: UserController.java

Within the WEBContent folder, place the following files:

* index.xhtml
* Login.xhtml
* Regester.xhtml
* MainMenu.xhtml

To assemble the data tables for the Pi, open up the SQL server in either MySQLWorkbench, or PHPMyAdmin and run the DLL code provided within this document.

For the code for the pi, take the included temp.py file and then install it to your pi. After booting it up, it should run without further issue.

**General Technical Approach:**

The raspberry pi will collect data such as the pressure, temperatures, and humidity of the room and then store that information into a SQL database table called weatherdata. A user can login to a webapp that will bring them to a page that will display that information in a visual form that was collected over the course of time.

**Key Technical Design Decisions:**

To ensure that the Pi has a stable connection to the database to upload information, Postman was used as a middle ground for processing data into the database. The chart API chosen was Google Charts, which can present data from an array as any major form of charts. We chose a line chart design as that would be the best way to represent the changes in the weather over the course of time.

For the logger, we chose JLogger which is a surprisingly very powerful logger that keeps tracks of all major events within the application. The English page does not provide any instructions and is mainly incomplete, however, the French page has everything written out to install and utilize the logger. What is also very impressive is that all the code is committed so well, that while documentation is limited, there is more than enough instructions within the code itself to fully understand how it works and how to implement it.

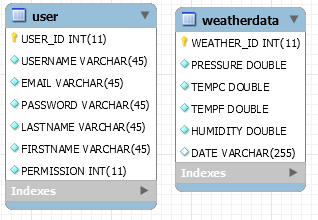
**Known Issues:**

The datatable only displays information that was collected the day it was running.

**Risks:**

No known or documented risk.

**ER Diagram:**

**

**DDL Scripts:** **-- MySQL Workbench Forward Engineering**

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema weathervane

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema weathervane

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `weathervane` DEFAULT CHARACTER SET latin1 ;

USE `weathervane` ;

-- -----------------------------------------------------

-- Table `weathervane`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `weathervane`.`user` (

`USER\_ID` INT(11) NOT NULL AUTO\_INCREMENT,

`USERNAME` VARCHAR(45) NOT NULL,

`EMAIL` VARCHAR(45) NOT NULL,

`PASSWORD` VARCHAR(45) NOT NULL,

`LASTNAME` VARCHAR(45) NOT NULL,

`FIRSTNAME` VARCHAR(45) NOT NULL,

`PERMISSION` INT(11) NOT NULL,

PRIMARY KEY (`USER\_ID`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `weathervane`.`weatherdata`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `weathervane`.`weatherdata` (

`WEATHER\_ID` INT(11) NOT NULL AUTO\_INCREMENT,

`PRESSURE` DOUBLE NOT NULL,

`TEMPC` DOUBLE NOT NULL,

`TEMPF` DOUBLE NOT NULL,

`HUMIDITY` DOUBLE NOT NULL,

`DATE` VARCHAR(255) NULL,

PRIMARY KEY (`WEATHER\_ID`))

ENGINE = InnoDB

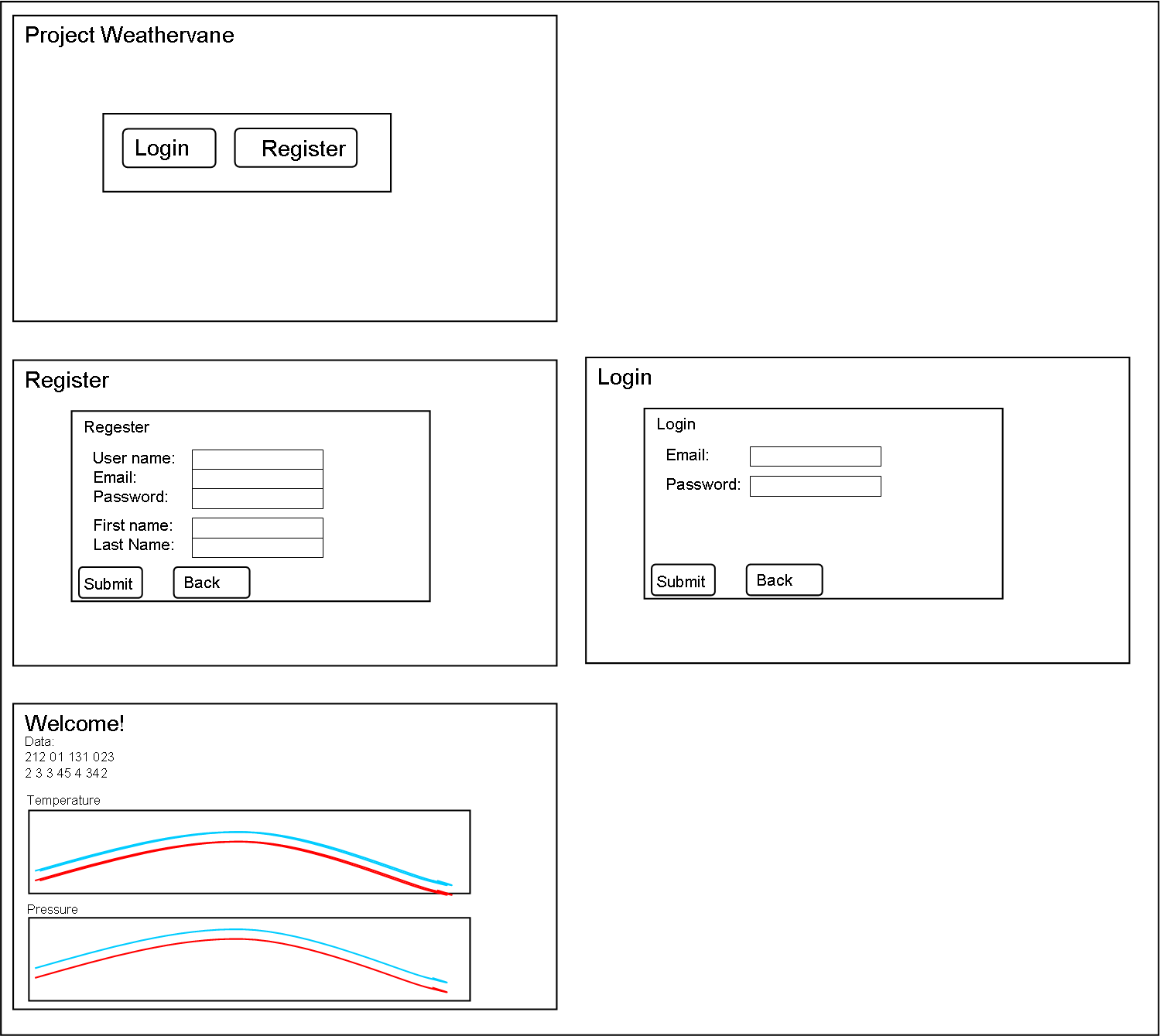
DEFAULT CHARACTER SET = latin1;

SET SQL\_MODE=@OLD\_SQL\_MODE;

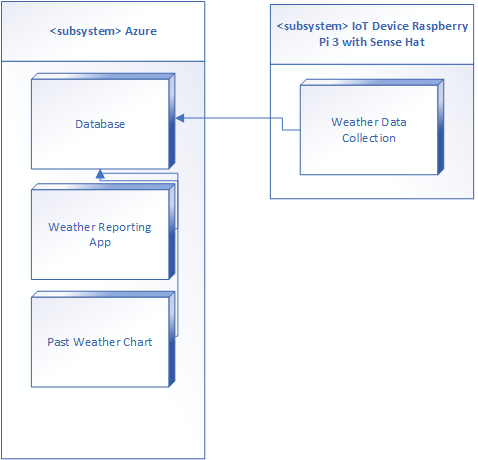
SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

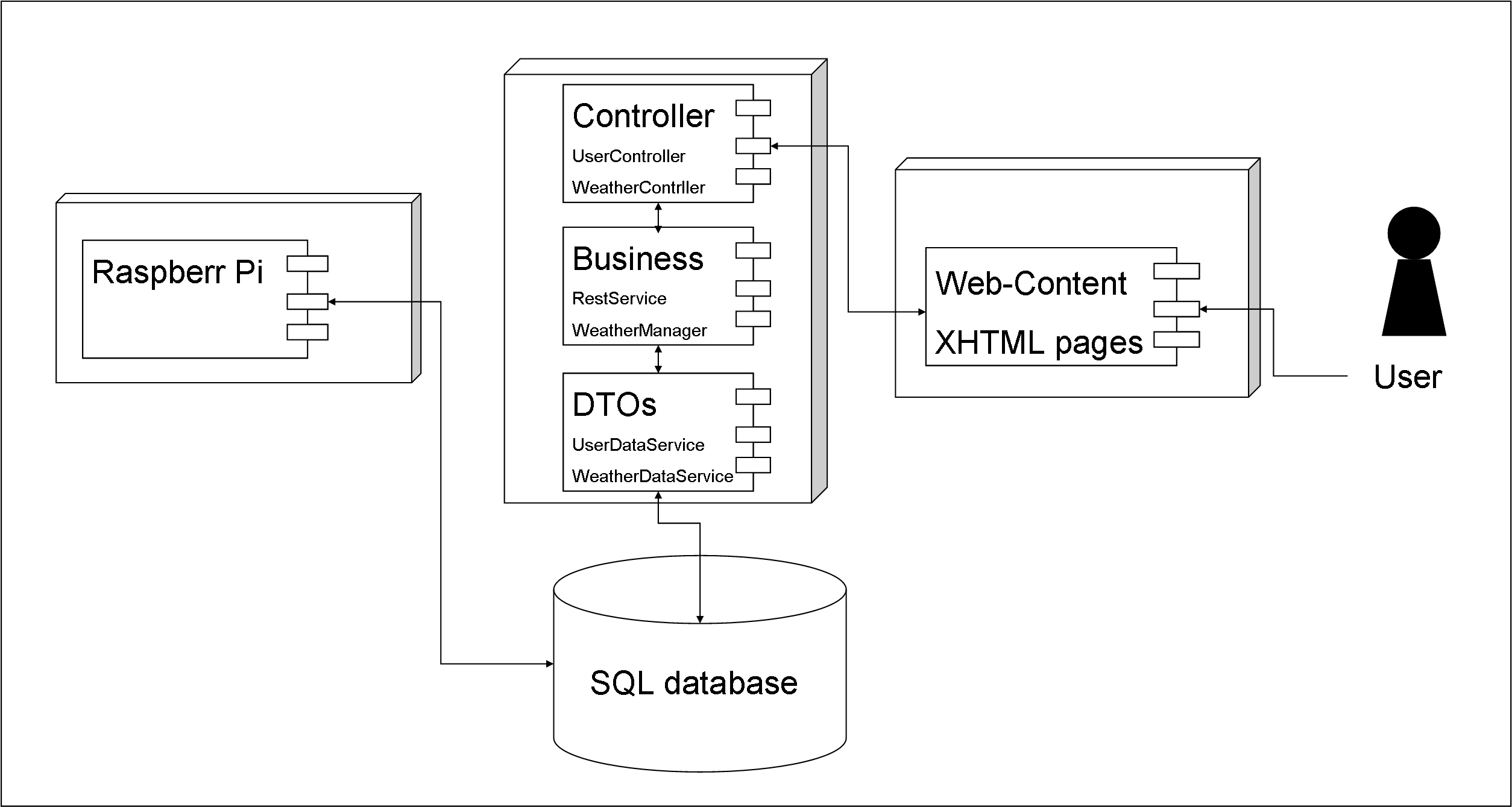
**Wireframes Diagrams:**Generated in PowerPoint:

****

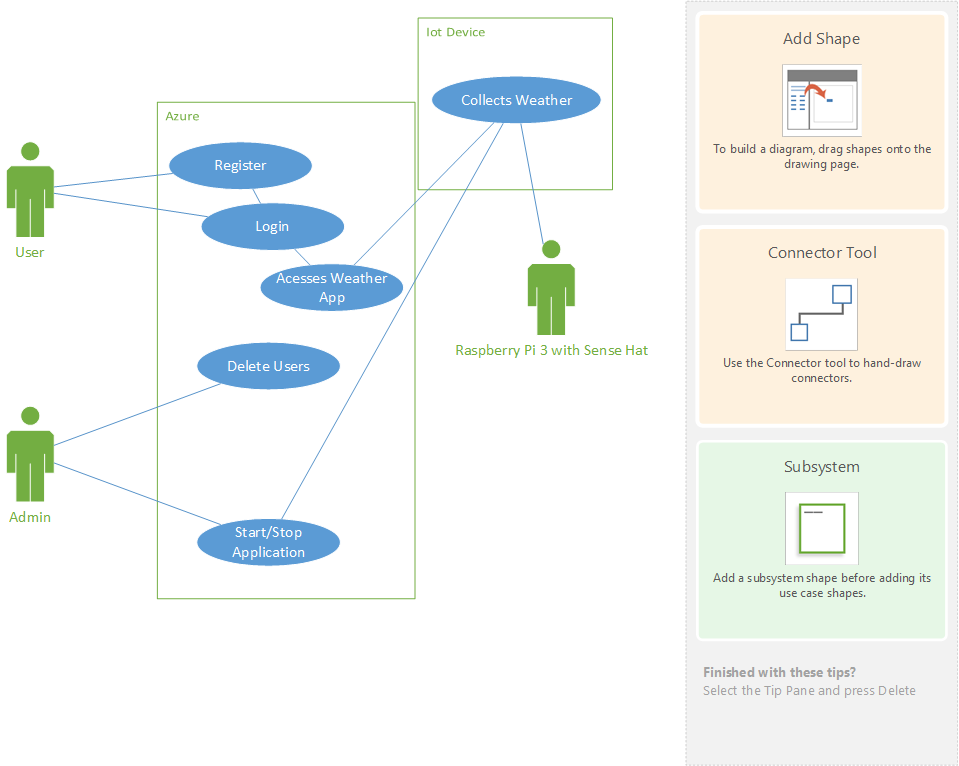
**Component Diagram:**

**

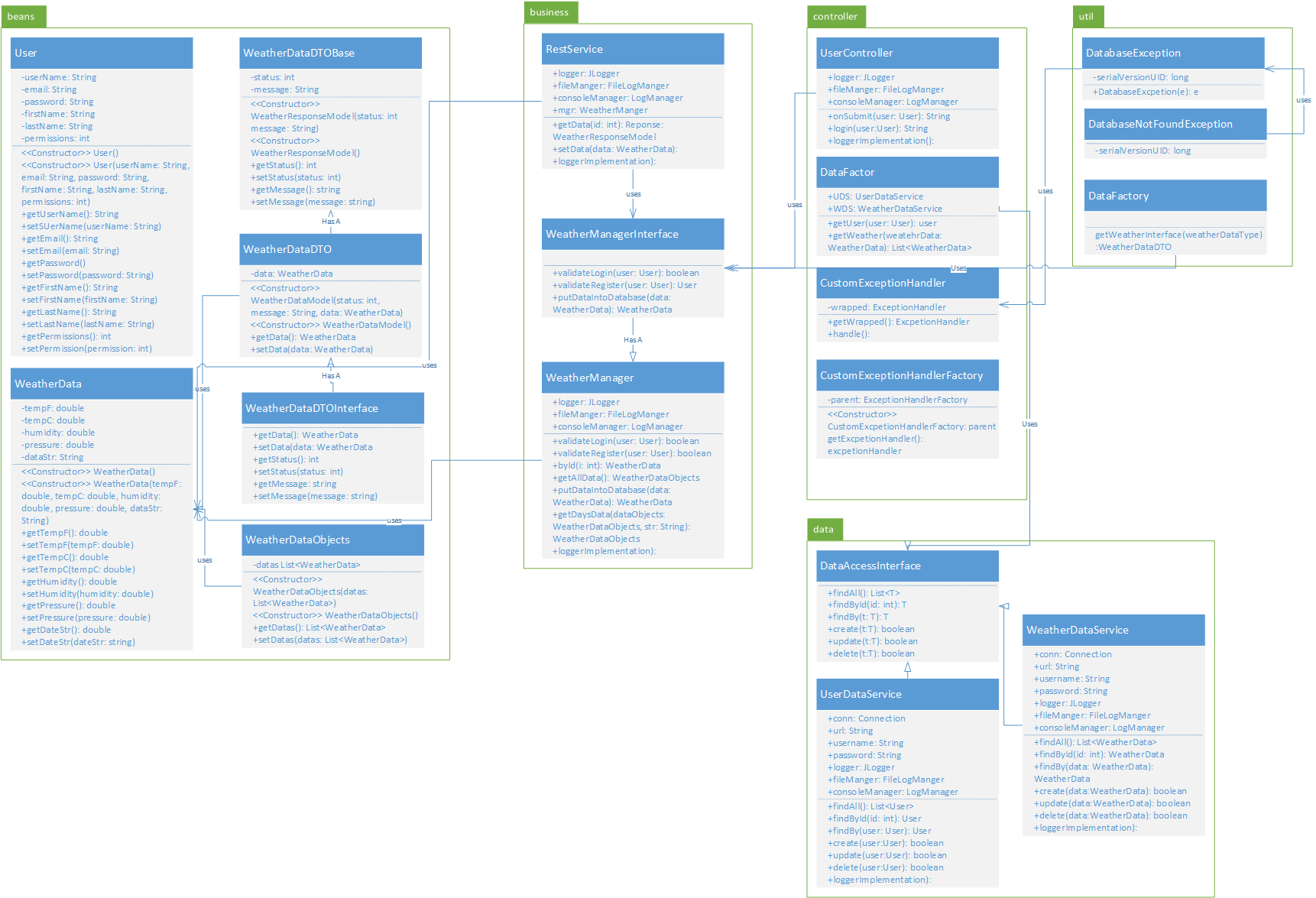
**Deplyment Diagram:**

****

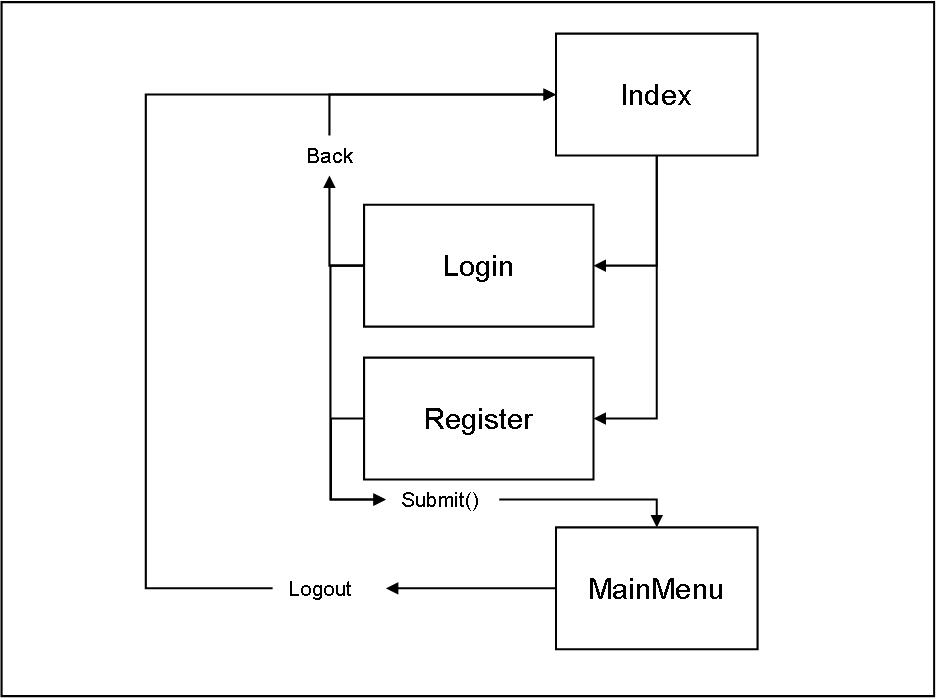
**User Interface Diagrams:**



**Class Diagrams:**



**Sitemap Diagram:**

****