Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

At the Heart of South Leinster

Utility Watch

Design Document

Student Name: Ray Shannon
Student ID: C00079959
Student e-mail: c00079959@itcarlow.ie
Supervisor: Paul Barry

BSc (Honours) in Software Development (4th Year)
Institute Of Technology Carlow,
Kilkenny Road,
Carlow.
Date: 17/04/2015

Department of Computing and Networking,
Institute Of Technology Carlow

Abstract

The purpose of this document is to show the design specifications for the Utility Watch application. As this is an agile development process, each iteration builds on the previous one until the project is complete. The design manual should enable another developer to implement the project in full.

Table of Contents

1. Introduction

Before any development begins, it is important to think about the architecture of how your software will be arranged. How you structure the code will determine the ease of maintenance on the code, if the project is easy to extend and troubleshooting the project. If your project handles data, that will need to be structured also. This document will take all iterations individually and cover both the high and low level designs.

2. High Level Detail

The overall project model is based on the Model-View-Controller (MVC) design pattern. In this design, the user sends a request to the controller, the controller then uses the model to retrieve and organise the requested data and passes it to the view. The view renders the requested webpage to the user.

2.1. Overview Diagram

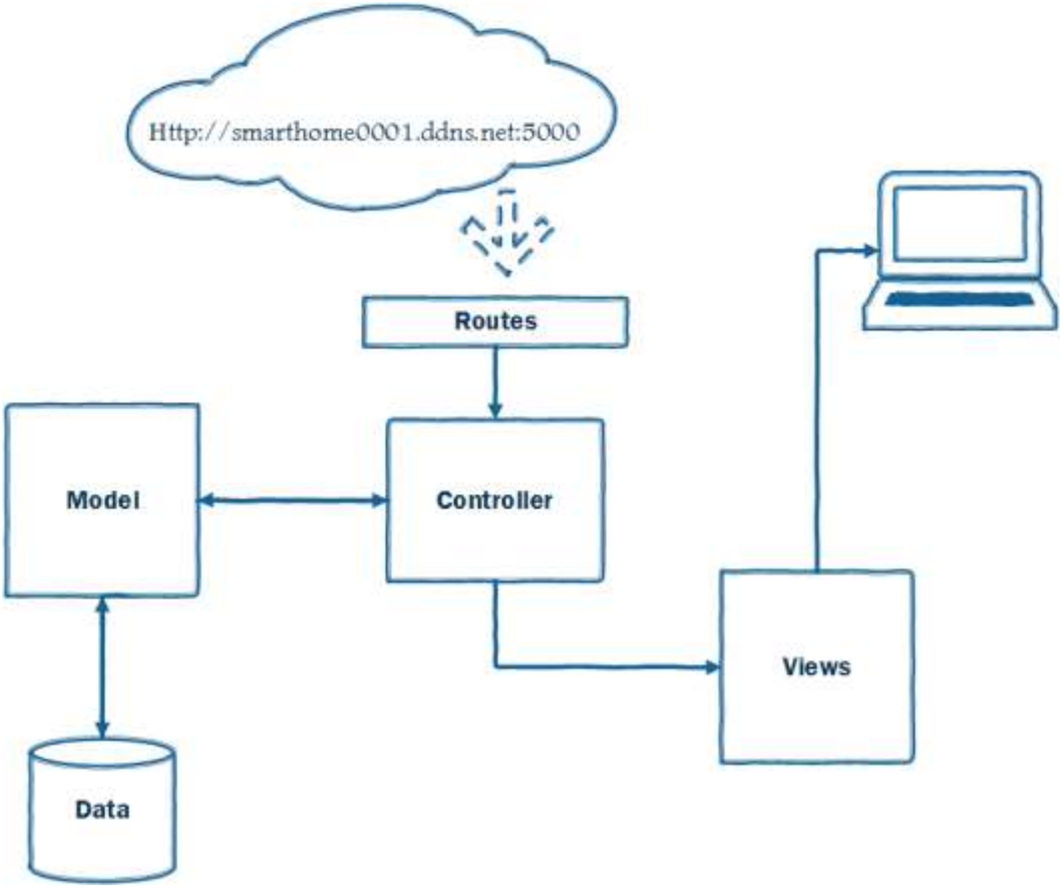


Fig. 1 Overview Diagram

2.2. System Architecture

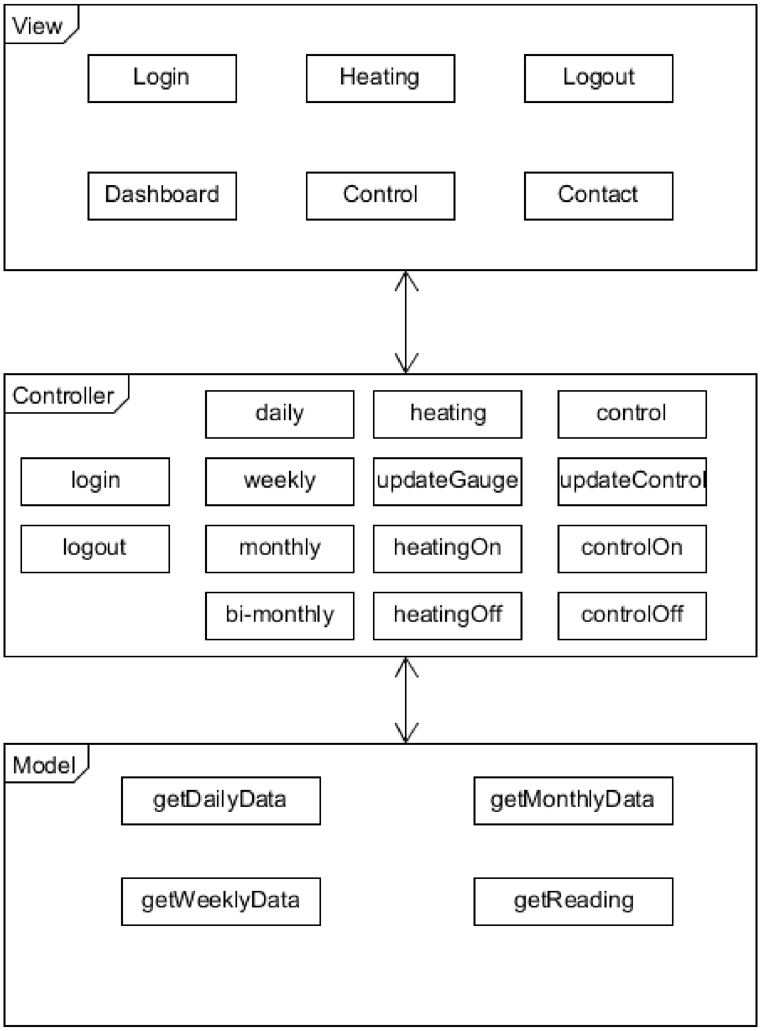


Fig. 2 Layered Architecture

3. Low Level Detail

As shown in the project timeline in the Functional specification, the **1st iteration** had very little code development. The backend data structure was created and some test scripts developed to test that architecture. As mentioned in the Project Report, the restrictions on the SQLite forces the creation of multiple databases each containing a table.

3.1. Data Structure

Raw Data Table

Database Name: UtilityWatchDB

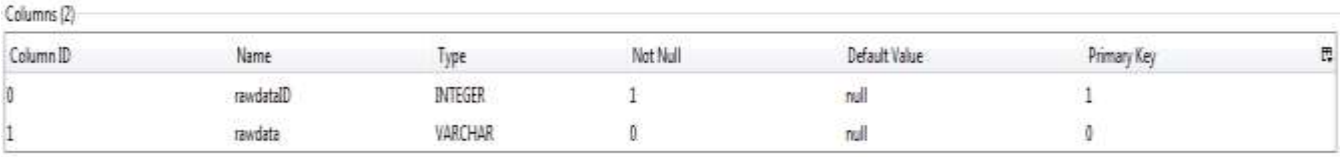
Table Name: rawdata

Operation: To store the rawdata generated from the request to external hardware.

Code:

```
    CREATE TABLE "rawdata" ("rawdataID" INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , "rawdata" VARCHAR)
```

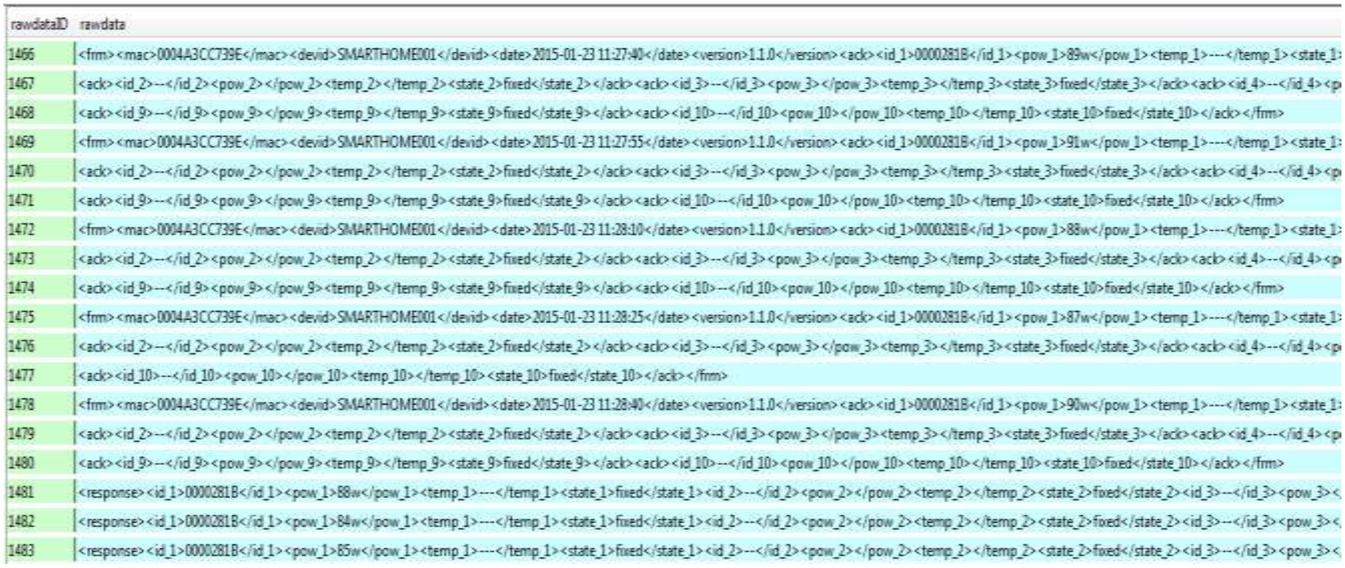
Table Structure



Columns (2)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	rawdataID	INTEGER	1	null	1
1	rawdata	VARCHAR	0	null	0

Sample Data from rawdataTbl



rawdataID	rawdata
1466	<frm><mac>0004A3CC739E</mac><devid>SMARTHOME001</devid><date>2015-01-23 11:27:40</date><version>1.1.0</version><ack><id_1>0000281B</id_1><pow_1>89w</pow_1><temp_1>---</temp_1><state_1
1467	<ack><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2></ack><ack><id_3>--</id_3><pow_3></pow_3><temp_3></temp_3><state_3>fixed</state_3></ack><ack><id_4>--</id_4><p
1468	<ack><id_9>--</id_9><pow_9></pow_9><temp_9></temp_9><state_9>fixed</state_9></ack><ack><id_10>--</id_10><pow_10></pow_10><temp_10></temp_10><state_10>fixed</state_10></ack></frm>
1469	<frm><mac>0004A3CC739E</mac><devid>SMARTHOME001</devid><date>2015-01-23 11:27:55</date><version>1.1.0</version><ack><id_1>0000281B</id_1><pow_1>91w</pow_1><temp_1>---</temp_1><state_1
1470	<ack><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2></ack><ack><id_3>--</id_3><pow_3></pow_3><temp_3></temp_3><state_3>fixed</state_3></ack><ack><id_4>--</id_4><p
1471	<ack><id_9>--</id_9><pow_9></pow_9><temp_9></temp_9><state_9>fixed</state_9></ack><ack><id_10>--</id_10><pow_10></pow_10><temp_10></temp_10><state_10>fixed</state_10></ack></frm>
1472	<frm><mac>0004A3CC739E</mac><devid>SMARTHOME001</devid><date>2015-01-23 11:28:10</date><version>1.1.0</version><ack><id_1>0000281B</id_1><pow_1>88w</pow_1><temp_1>---</temp_1><state_1
1473	<ack><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2></ack><ack><id_3>--</id_3><pow_3></pow_3><temp_3></temp_3><state_3>fixed</state_3></ack><ack><id_4>--</id_4><p
1474	<ack><id_9>--</id_9><pow_9></pow_9><temp_9></temp_9><state_9>fixed</state_9></ack><ack><id_10>--</id_10><pow_10></pow_10><temp_10></temp_10><state_10>fixed</state_10></ack></frm>
1475	<frm><mac>0004A3CC739E</mac><devid>SMARTHOME001</devid><date>2015-01-23 11:28:25</date><version>1.1.0</version><ack><id_1>0000281B</id_1><pow_1>87w</pow_1><temp_1>---</temp_1><state_1
1476	<ack><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2></ack><ack><id_3>--</id_3><pow_3></pow_3><temp_3></temp_3><state_3>fixed</state_3></ack><ack><id_4>--</id_4><p
1477	<ack><id_10>--</id_10><pow_10></pow_10><temp_10></temp_10><state_10>fixed</state_10></ack></frm>
1478	<frm><mac>0004A3CC739E</mac><devid>SMARTHOME001</devid><date>2015-01-23 11:28:40</date><version>1.1.0</version><ack><id_1>0000281B</id_1><pow_1>90w</pow_1><temp_1>---</temp_1><state_1
1479	<ack><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2></ack><ack><id_3>--</id_3><pow_3></pow_3><temp_3></temp_3><state_3>fixed</state_3></ack><ack><id_4>--</id_4><p
1480	<ack><id_9>--</id_9><pow_9></pow_9><temp_9></temp_9><state_9>fixed</state_9></ack><ack><id_10>--</id_10><pow_10></pow_10><temp_10></temp_10><state_10>fixed</state_10></ack></frm>
1481	<response><id_1>0000281B</id_1><pow_1>88w</pow_1><temp_1>---</temp_1><state_1>fixed</state_1><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2><id_3>--</id_3><pow_3><
1482	<response><id_1>0000281B</id_1><pow_1>84w</pow_1><temp_1>---</temp_1><state_1>fixed</state_1><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2><id_3>--</id_3><pow_3><
1483	<response><id_1>0000281B</id_1><pow_1>85w</pow_1><temp_1>---</temp_1><state_1>fixed</state_1><id_2>--</id_2><pow_2></pow_2><temp_2></temp_2><state_2>fixed</state_2><id_3>--</id_3><pow_3><

Unit Reading Table

Database Name: UtilityWatchDB

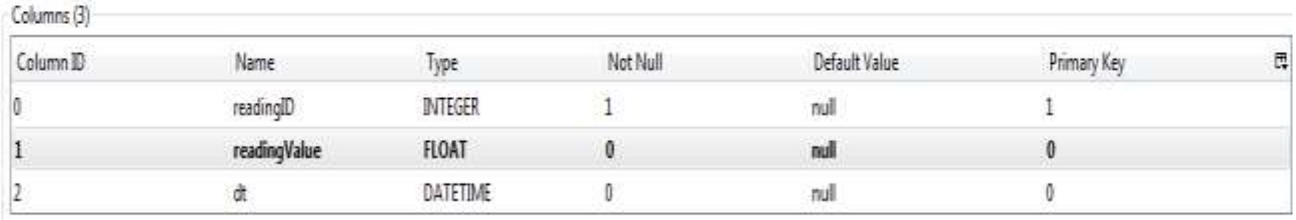
Table Name: readingTbl

Operation: To store the electricity meter reading value and timestamp.

Code:

```
   CREATE TABLE “readingTbl” (“readingID” INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL , “readingValue” FLOAT, “dt” DATETIME)
```

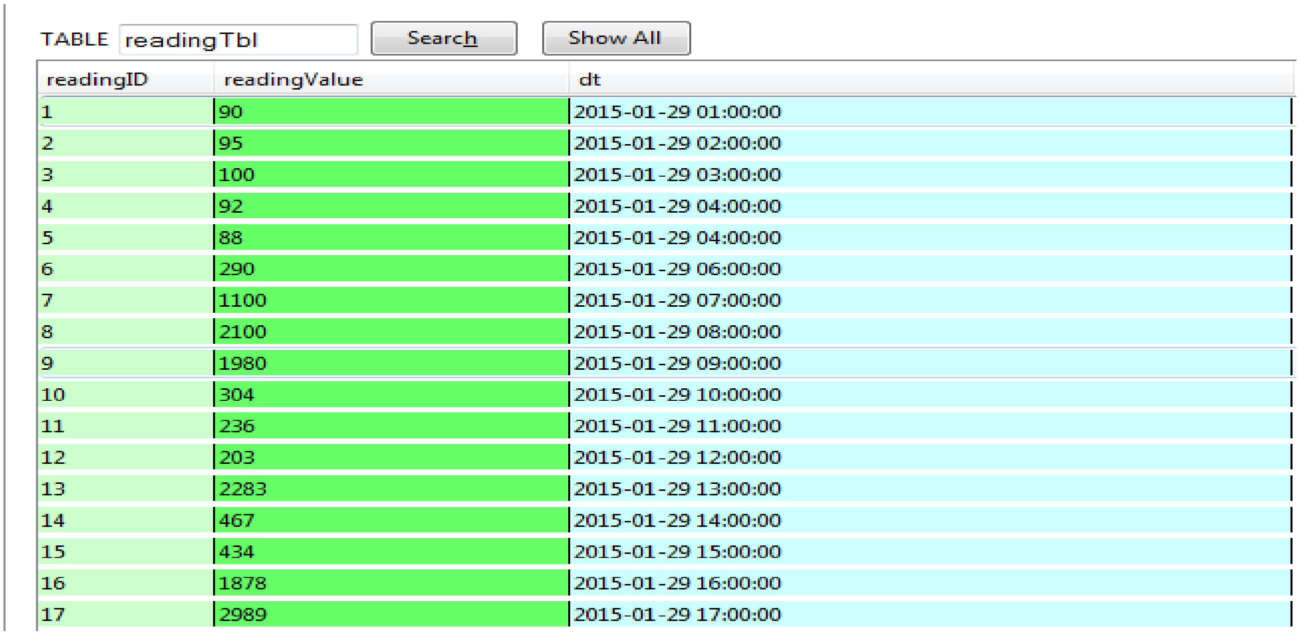
Table Structure



Columns (3)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	readingID	INTEGER	1	null	1
1	readingValue	FLOAT	0	null	0
2	dt	DATETIME	0	null	0

Sample Data for readingTbl



readingID	readingValue	dt
1	90	2015-01-29 01:00:00
2	95	2015-01-29 02:00:00
3	100	2015-01-29 03:00:00
4	92	2015-01-29 04:00:00
5	88	2015-01-29 04:00:00
6	290	2015-01-29 06:00:00
7	1100	2015-01-29 07:00:00
8	2100	2015-01-29 08:00:00
9	1980	2015-01-29 09:00:00
10	304	2015-01-29 10:00:00
11	236	2015-01-29 11:00:00
12	203	2015-01-29 12:00:00
13	2283	2015-01-29 13:00:00
14	467	2015-01-29 14:00:00
15	434	2015-01-29 15:00:00
16	1878	2015-01-29 16:00:00
17	2989	2015-01-29 17:00:00

Daily data Table

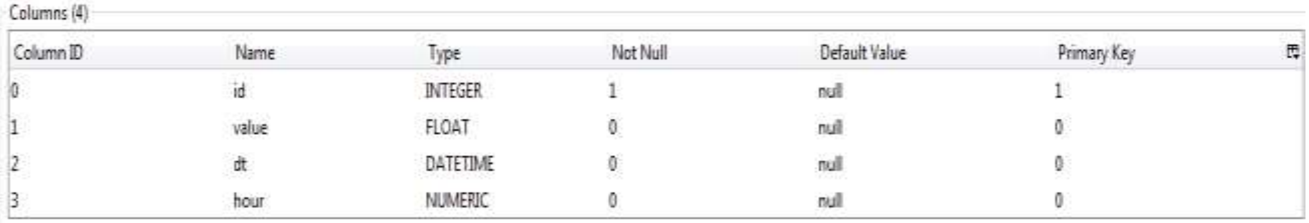
Database Name: ChartDataDB

Table Name: dailyDataTbl

Operation: To store the average hourly usage from the readingTbl.

Code: `CREATE TABLE “dailyDataTbl” (“id” INTEGER PRIMARY KEY AUTOINCREMENT  NOT NULL , “value” FLOAT, “dt” DATETIME, “hour” NUMERIC)`

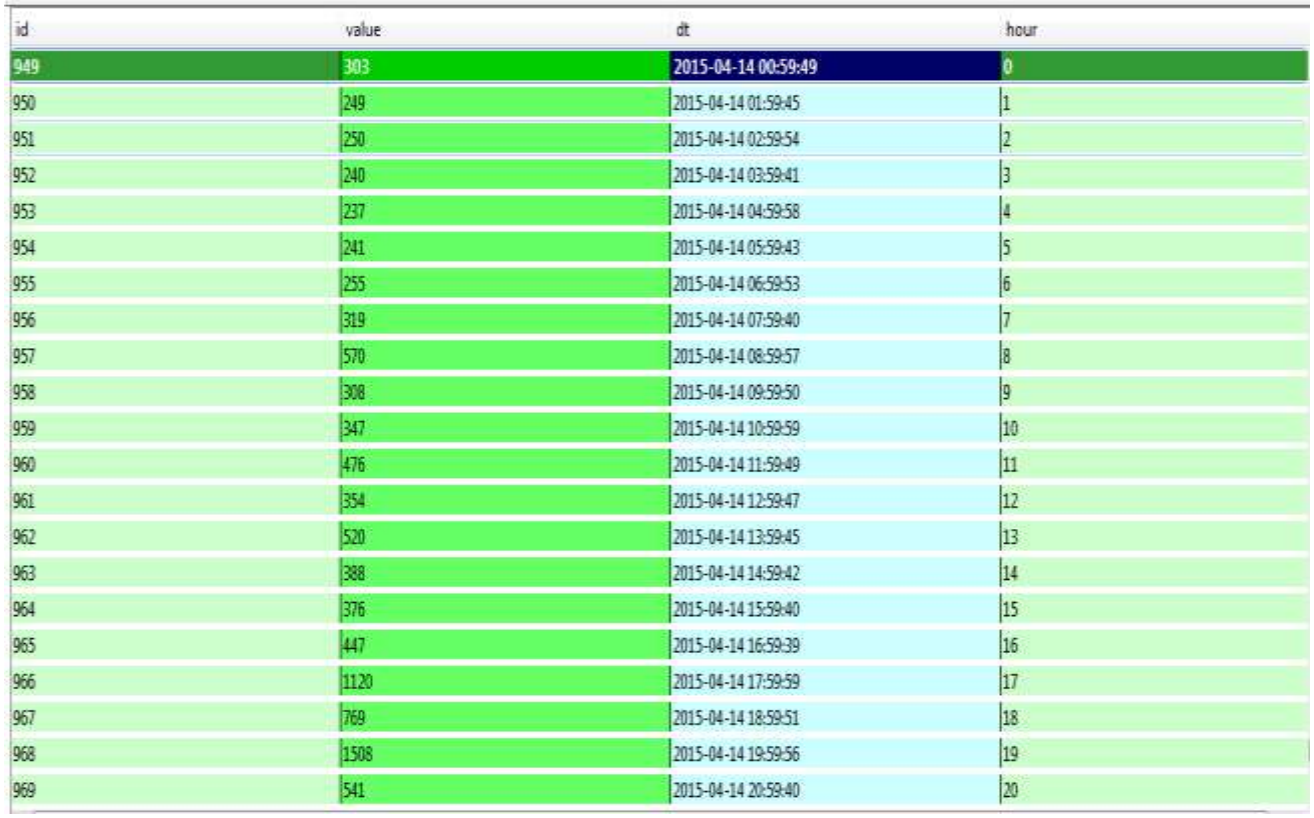
Table Structure



Columns (4)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	id	INTEGER	1	null	1
1	value	FLOAT	0	null	0
2	dt	DATETIME	0	null	0
3	hour	NUMERIC	0	null	0

Sample Data for dailyDataTbl



id	value	dt	hour
949	303	2015-04-14 00:59:49	0
950	249	2015-04-14 01:59:45	1
951	250	2015-04-14 02:59:54	2
952	240	2015-04-14 03:59:41	3
953	237	2015-04-14 04:59:58	4
954	241	2015-04-14 05:59:43	5
955	255	2015-04-14 06:59:53	6
956	319	2015-04-14 07:59:40	7
957	570	2015-04-14 08:59:57	8
958	308	2015-04-14 09:59:50	9
959	347	2015-04-14 10:59:59	10
960	476	2015-04-14 11:59:49	11
961	354	2015-04-14 12:59:47	12
962	520	2015-04-14 13:59:45	13
963	388	2015-04-14 14:59:42	14
964	376	2015-04-14 15:59:40	15
965	447	2015-04-14 16:59:39	16
966	1120	2015-04-14 17:59:59	17
967	769	2015-04-14 18:59:51	18
968	1508	2015-04-14 19:59:56	19
969	541	2015-04-14 20:59:40	20

Weekly data Table

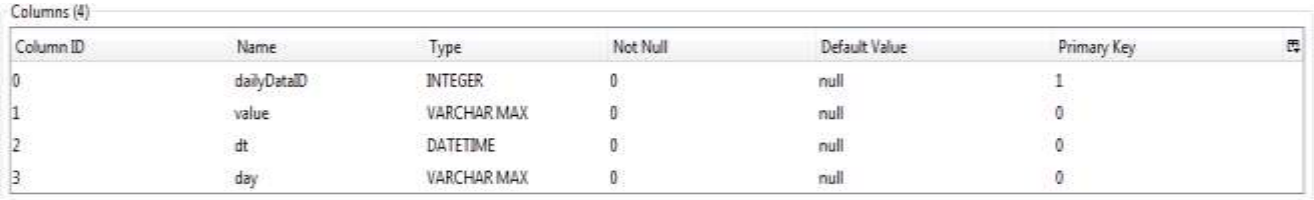
Database Name: weeklyChartData

Table Name: weeklyDataTbl

Operation: To store the average daily usage from the readingTbl.

Code: `CREATE TABLE weeklyDataTbl(dailyDataID INTEGER PRIMARY KEY AUTOINCREMENT,value VARCHAR MAX NULL,dt DATETIME NULL,day VARCHAR MAX NULL)`

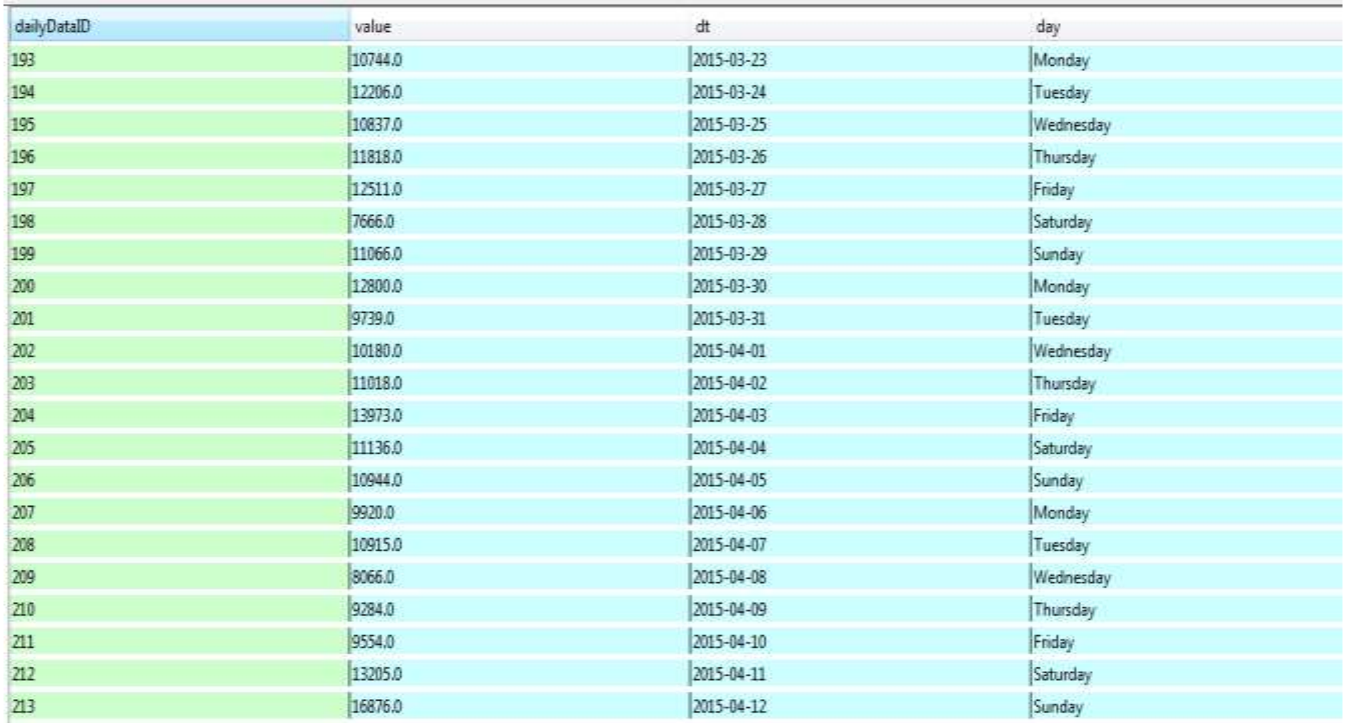
Table Structure



Columns (4)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	dailyDataID	INTEGER	0	null	1
1	value	VARCHAR MAX	0	null	0
2	dt	DATETIME	0	null	0
3	day	VARCHAR MAX	0	null	0

Sample Data for weeklyDataTbl



dailyDataID	value	dt	day
193	10744.0	2015-03-23	Monday
194	12206.0	2015-03-24	Tuesday
195	10837.0	2015-03-25	Wednesday
196	11818.0	2015-03-26	Thursday
197	12511.0	2015-03-27	Friday
198	7666.0	2015-03-28	Saturday
199	11066.0	2015-03-29	Sunday
200	12800.0	2015-03-30	Monday
201	9739.0	2015-03-31	Tuesday
202	10180.0	2015-04-01	Wednesday
203	11018.0	2015-04-02	Thursday
204	13973.0	2015-04-03	Friday
205	11136.0	2015-04-04	Saturday
206	10944.0	2015-04-05	Sunday
207	9920.0	2015-04-06	Monday
208	10915.0	2015-04-07	Tuesday
209	8066.0	2015-04-08	Wednesday
210	9284.0	2015-04-09	Thursday
211	9554.0	2015-04-10	Friday
212	13205.0	2015-04-11	Saturday
213	16876.0	2015-04-12	Sunday

Monthly data Table

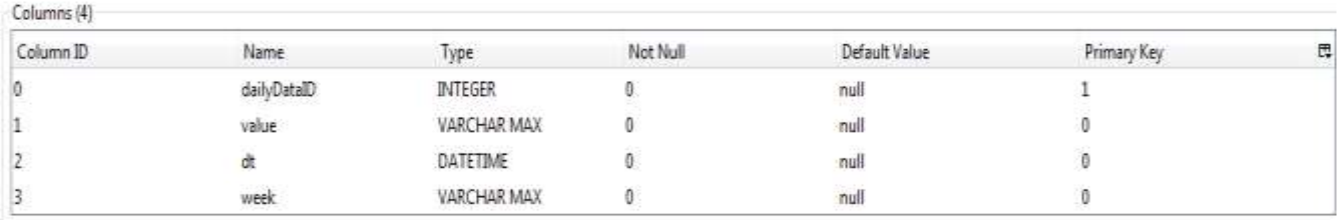
Database Name: monthlyChartData

Table Name: monthlyDataTbl

Operation: To store the average weekly usage from the readingTbl.

Code: `CREATE TABLE monthlyDataTbl(dailyDataID INTEGER PRIMARY KEY AUTOINCREMENT,value VARCHAR MAX NULL,dt DATETIME NULL, week VARCHAR MAX NULL)`

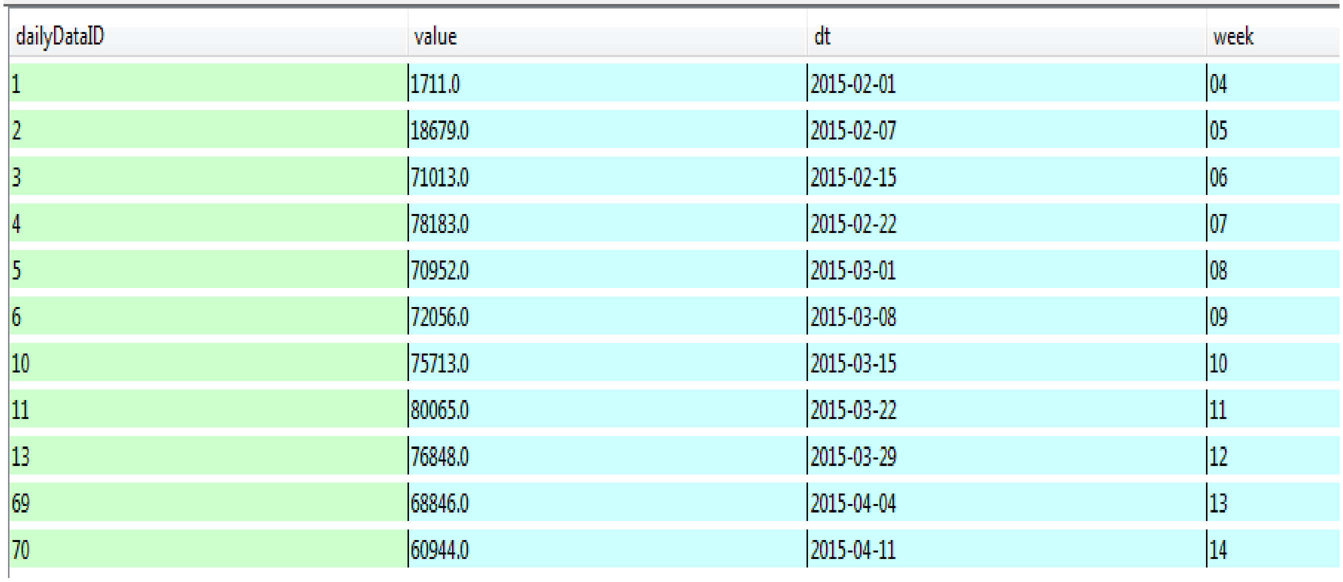
Table Structure



Columns (4)

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	dailyDataID	INTEGER	0	null	1
1	value	VARCHAR MAX	0	null	0
2	dt	DATETIME	0	null	0
3	week	VARCHAR MAX	0	null	0

Sample Data for weeklyDataTbl



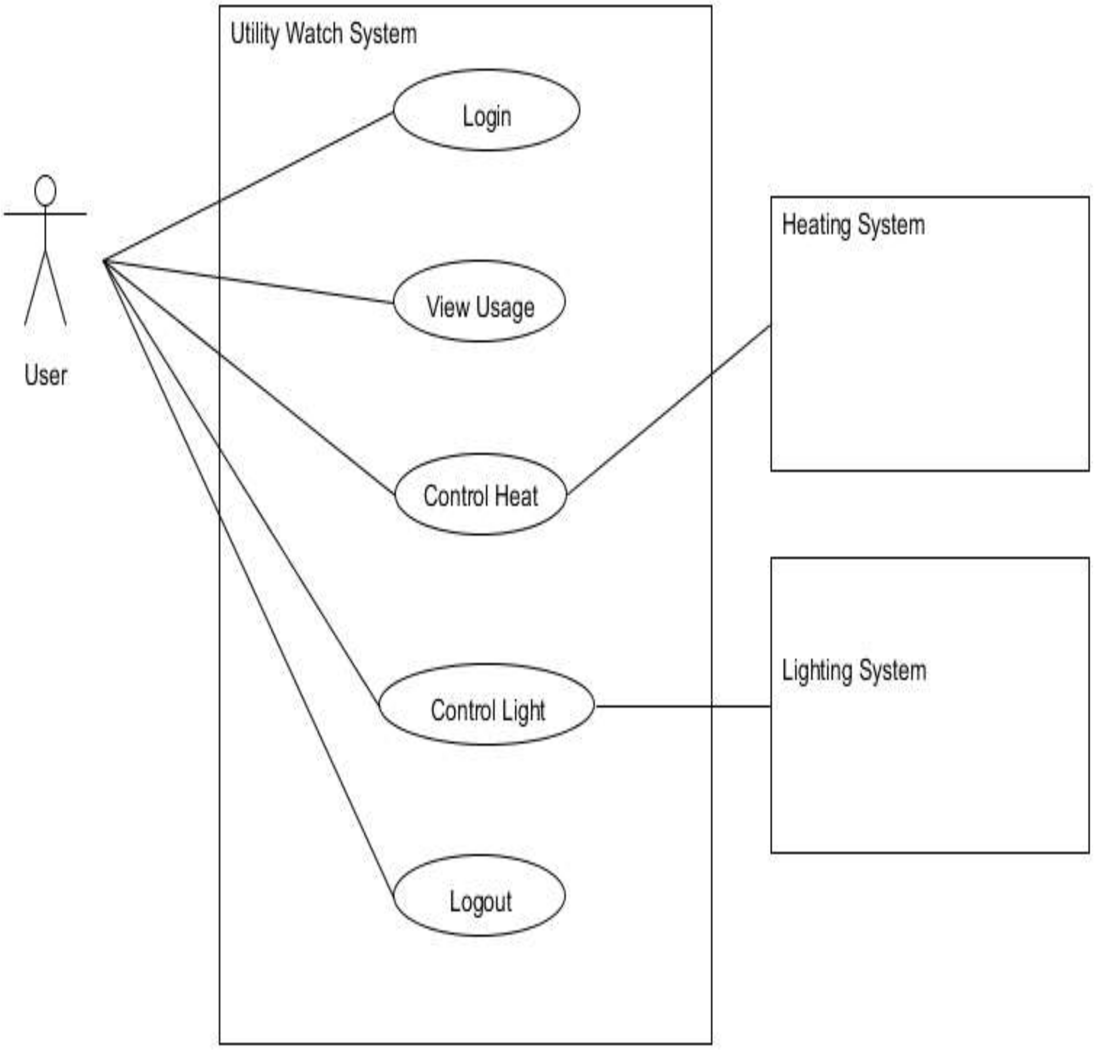
dailyDataID	value	dt	week
1	1711.0	2015-02-01	04
2	18679.0	2015-02-07	05
3	71013.0	2015-02-15	06
4	78183.0	2015-02-22	07
5	70952.0	2015-03-01	08
6	72056.0	2015-03-08	09
10	75713.0	2015-03-15	10
11	80065.0	2015-03-22	11
13	76848.0	2015-03-29	12
69	68846.0	2015-04-04	13
70	60944.0	2015-04-11	14

4. MVC

4.1.Use Cases

Use Case Diagram

This use case diagram has been discussed in the specification document. This is just to assist with the understanding of the detailed use case which will follow.



4.2. Detailed Use Cases

Name: Login
Actors: User, System (model, view, controller)
Description: This use case begins when a user wishes to login to the application. The user must provide a username and password. The system checks if details are valid.
Main Success Scenario:

1. The user selects Login.
2. The view sends a request to the controller.
3. The controller handles request and renders the login view.
4. The user enters a username and password and presses submit.
5. The view sends a request to the controller.
6. The controller passes the details to the model.
7. The model returns a reply.
8. The controller handles the reply and renders the dashboard view.

Alternatives:

4. a The controller detects that one or more of the fields are empty/incorrect
   i) The view displays error message and invites user to re-insert data
   ii) The user re-inserts the data

Name: View Usage
Actors: User, System
Description: This use case begins when a user wishes to view the energy consumption.
Main Success Scenario:

1. The user selects Electricity on the dashboard view.
2. The view sends a request to the controller.
3. The controller asks the model for the required data.
4. The model replies with the requested data from the database.
5. The controller renders the usage page with the populated chart for viewing.
6. **Alternatives:**

4. a The controller detects that session has expired
   i) The view displays error message and invites user to re-insert login data
   ii) The user re-inserts the data

Name: Control Heating
Actors: User, System
Description: This use case begins when a user wishes to turn on the heating.
Main Success Scenario:

1. The user selects Heating on the dashboard view.
2. The view sends a request to the controller.
3. The controller retrieves the request and renders the heat view.
4. The user selects the Heating On option.
5. The view sends a request to the controller.
6. The controller carries out the request and updates the heat view.
7. **Alternatives:**

4. a The controller detects that session has expired
 i) The view displays error message and invites user to re-insert login data
 ii) The user re-inserts the data

Name: Control Lighting
Actors: User, System
Description: This use case begins when a user wishes to turn on the lighting.
Main Success Scenario:

1. The user selects Lighting on the dashboard view.
2. The view sends a request to the controller.
3. The controller retrieves the request and renders the light view.
4. The user selects the Lighting On option.
5. The view sends a request to the controller.
6. The controller carries out the request and updates the light view.
7. **Alternatives:**

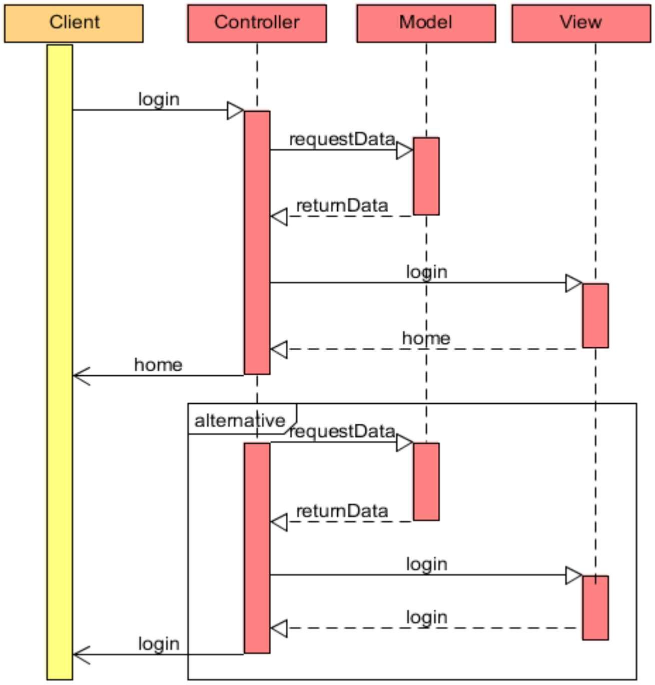
4. a The controller detects that session has expired
 i) The view displays error message and invites user to re-insert login data
 ii) The user re-inserts the data

Name: Logout
Actors: User, System
Description: This use case begins when a user wishes to logout.
Main Success Scenario:

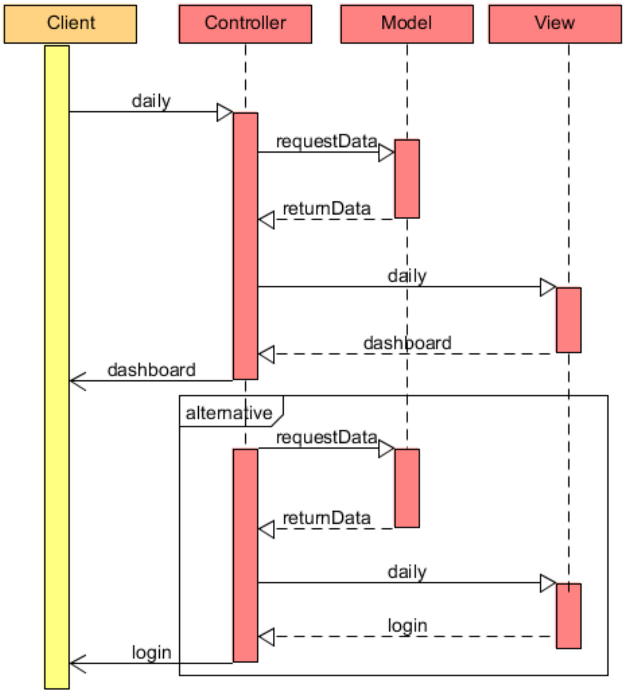
1. The user selects the logout option on the current view.
2. The view sends a request to the controller.
3. The controller retrieves the request and clears the session.
4. The controller renders the login view.

4.3. Sequence Diagrams

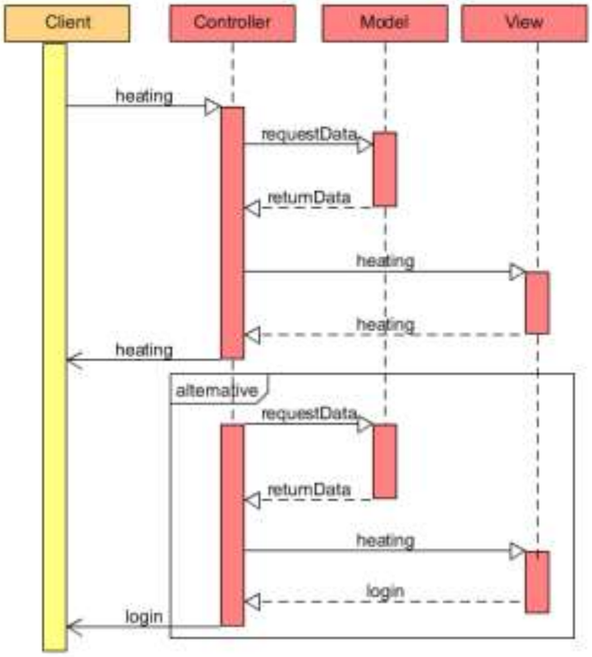
Login Sequence diagram



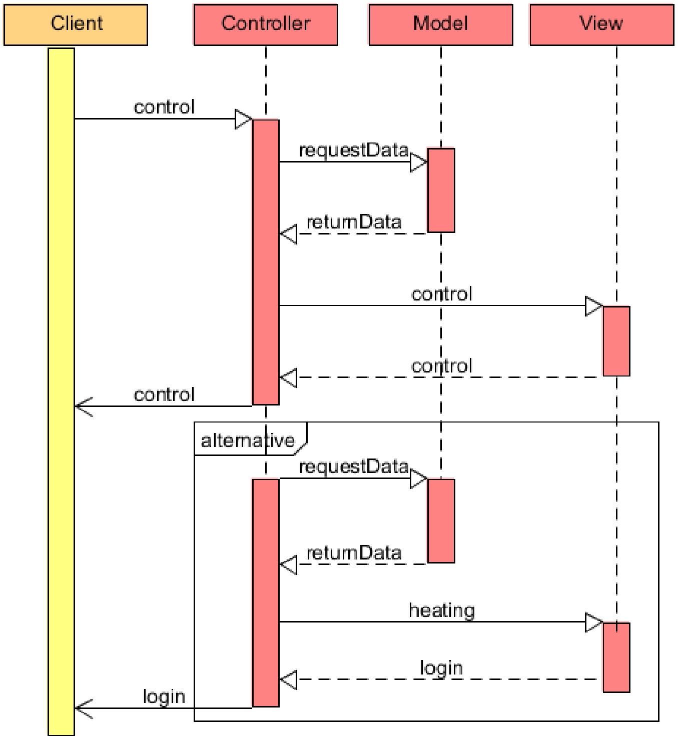
View Usage Sequence diagram



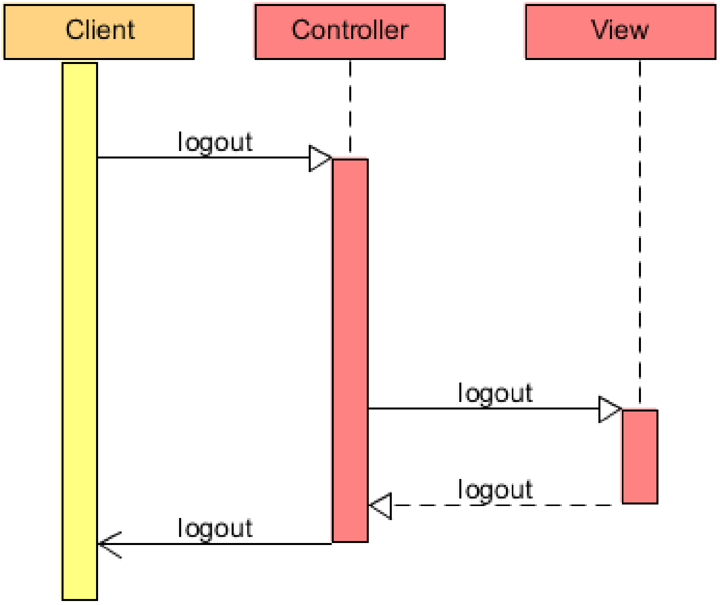
Control Heating Sequence diagram



Control Lighting Sequence diagram

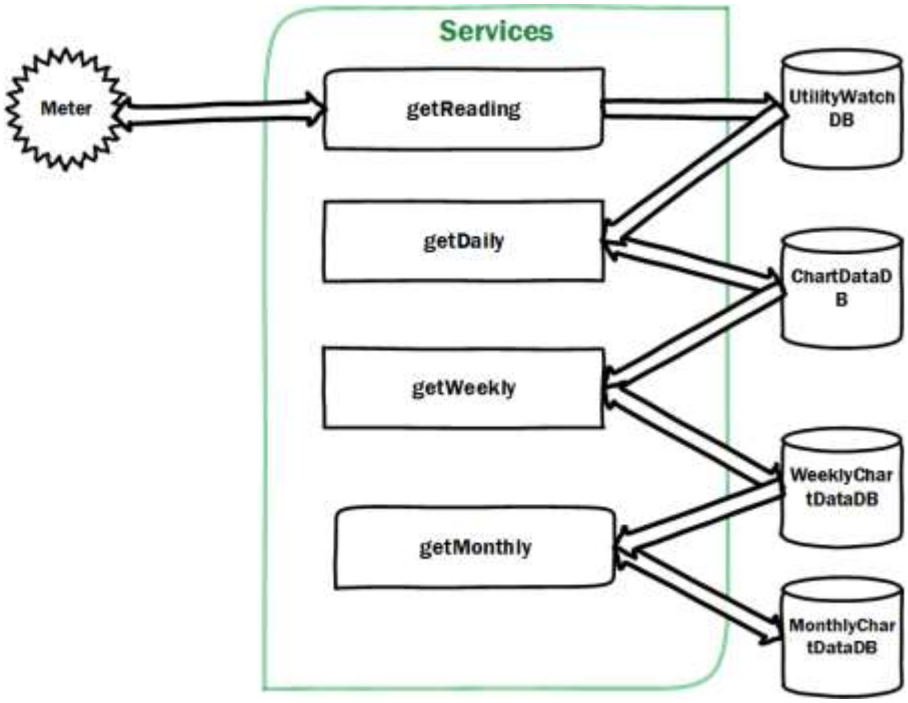


Logout Sequence Diagram

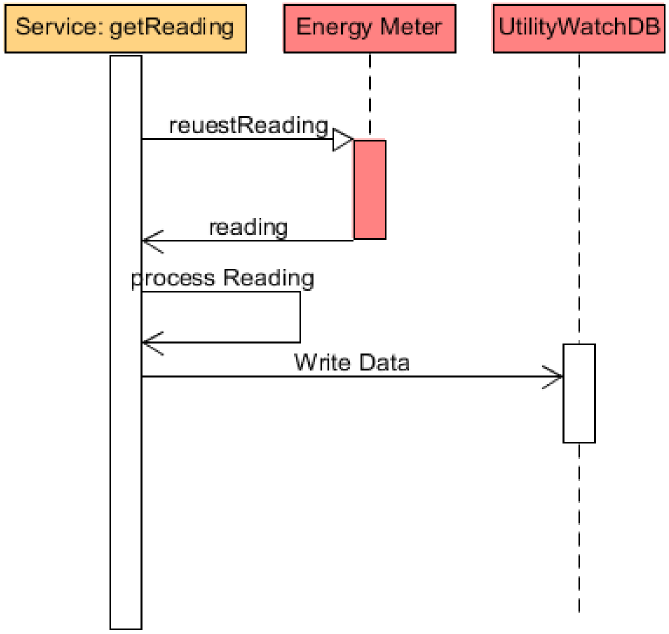


5. Services

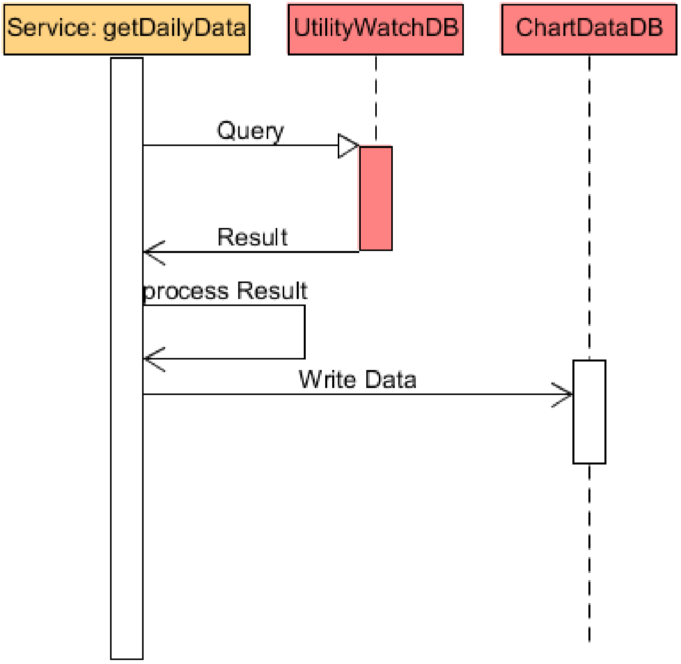
In order for the Utility Watch to function correctly and up to date, there are services that are working away in the background. These services cover routines for the data creation. There are also programs running to ensure the Flask server is active and allowing access to the application at all times.



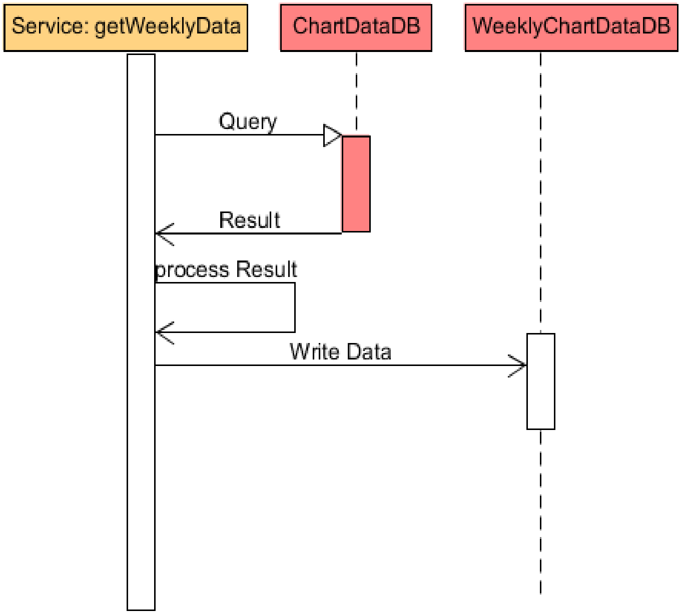
GetReading Sequence Diagram



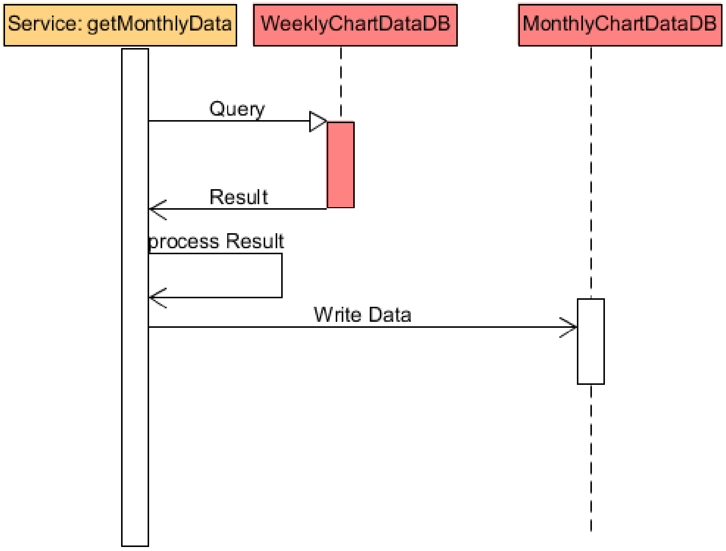
Get Daily Data Sequence Diagram



Get Weekly Data Sequence Diagram



Get Monthly Data Sequence Diagram


Service: getMonthlyData
WeeklyChartDataDB
MonthlyChartDataDB
Query
Result
process Result
Write Data