Übermind / Deloitte Digital

The Energenie Project

Andrew Shaw, Mark Shivers, Santiago Gonzalez, Zach Fleischman

June 19, 2012

Abstract:

Our team was tasked with the creation of an iPhone app and an accompanying web app that empowers the consumer with information on his or her home appliance power usage. The app was developed with the Model View Controller design pattern in mind. Our model is a RESTful database made with Ruby on Rails. Our controllers receive data from a singleton DataFetcher class within the Objective-C based iPhone app and our views are subclassed Cocoa UIViews.

I. Introduction:

Übermind, recently acquired by Deloitte Digital, is a software contracting company that specializes in developing mobile and web apps for large corporations. They have worked with Apple on the Apple Online Store, iTunes Store, and the Apple Store app for iPhone. Their list of clients goes on to include many other large corporations involucrated in computer technologies such as Intel and Adobe, as well as large department stores including Target, REI and Office Max, and some more far reaching clients like the American Heart Association and even the Australian Army.

Benjamin Dunton, a software engineer at Übermind, has always been a proponent of sustainable practices, green technologies, and informed energy usage. Our team was tasked with the creation of an iPhone app and a Web app that would empower the consumer with detailed information on the energy use of his or her home appliances. This app graphically displays to the consumer statistics on energy use of appliances relative to each other, and the ability to compare, by category, their appliances' wattage to the world-wide average wattage. Consumers will be able to see where energy is being wasted due to leaving appliances that have vampiric draw plugged in while off, or leaving computers in sleep mode. Dunton felt that this app would be an excellent tool for other like-minded energy conscious consumers to have better awareness and control over their energy usage.

II. Requirements:

The iOS application is designed to receive data regarding the consumer's energy usage. The app then takes that data and arranges it into various graphical representations for an overview of usage statistics as well as a comparison to other consumers.

Functional Requirements:

1. The application must be able to input device information from the consumer.
2. The application must be able to input consumer usage information from the consumer.
3. The application must be able to edit their usage information.
4. The application must be able to edit their appliance information.
5. The application must display an overview of the consumer's personal statistics.
6. The application must display consumer stats compared to the average world's stats
7. The consumer must be able to log into their respective account for the application
8. The consumer should be able to group their appliances into rooms

Non-Functional Requirements:

1. The program must be written in Objective-C
2. The program must be written within the Xcode Integrated Development Environment
3. The program must be written on an Apple due to the above two requirements
4. The program must use the advanced user interface storyboarding functionality from Xcode

III. Software Architecture:

Refer to Figure 2.1:
2.1.1: The database is the main holder of information in the system. It delivers consumer information as well as the averages and the worst instances over the scope of all consumers.

2.1.2: The Appliance Views are the main source of input for the system. It is where the consumer input's information about devices and usages and where that information gets sent to the database

2.1.3: The MyStats View is the main way for a consumer to view a complete overview of their own information. It groups all the the data into a graphical representation.

2.1.4: The World Stats View is the only way for the consumer to view how their information compared to that of the average from all other consumers through a graphical display.

2.1.5: The consumer adds new appliances and usage data through the Appliance View.

2.1.6: New information that was input by the consumer gets sent to the database after all necessary components are added to the Appliance View.

2.1.7: The Appliance View displays all the appliances that a consumer has created, so the consumer can edit or delete them as needed.

2.1.8: The Appliance View displays information that has been pulled from the database in an easy to read list for the consumer to view and edit.

2.1.9: The MyStats View needs the consumer's data in order to display to them.

2.1.10: The MyStats View displays a graphical representation of all the consumer's data in a set of Pie Graphs that can be understood.

2.1.11 & 2.1.12: The WorldStats View needs both the consumer's information as well as the average over the entire database in order to make accurate comparisons.

2.1.13: The WorldStats View displays the consumer's information up against that of the average of all other consumers back to the current consumer in the form of a head to head graph of one set of data versus the other.

Refer to Figure 2.2:
The user has a one to many relationship with rooms, and usage profiles. Rooms and Appliances also have a one to many relationship with usage profiles. Having the connection between a user and an appliance go through a usage profile for that appliance allows us to keep a single growing database of all appliances that all users can share. This will also make the database more efficient because it will theoretically avoid duplicate appliances being created (it all relies on the consumers being smart, which is, admittedly, a dangerous pitfall).

IV. Technical Design:

Tab view controller:

After the consumer logs in, a tab view controller serves as a nexus between the four primary views of the app: My Appliances, My Stats, World Stats, and Profile. At the primary views of these four sections, the tab controller is visible at the bottom of the screen for easy navigation between them. See Figures 3.4 and 3.5 for the layout of the tab view controller in the storyboard.

My Appliances:

This is a table view where consumers can see and delete each of their appliance profiles sorted by room. Selecting an appliance profile goes to a detailed view that describes the wattage and daily use of the appliance. From there, they can edit the appliance profile's room and daily usage.

The plus button from the appliances view launches a modal set of views where a new appliance profile is created. The consumer selects which appliance, which room, and sets up daily usage. If the consumer would like to add an appliance that they do not see they can very easily add a custom appliance as long as they are sure to include its on, off and sleeping wattage. The consumer can also add new rooms as they desire. The My Appliances view can be seen in Figure 1.1. A layout of the entire storyboard for the My Appliances view can be seen in Figures 3.2 and 3.3.

Edit/Create Appliance Profile:

The three sliders for minutes on, asleep, and off each are linked to two events, one that triggers on value changed and one on the "touch up" event, which occurs when the user's finger leaves the screen. Each of these events link to a balancing function to ensure that the total of all three sliders' values is 1440 minutes (24 hours). First, the event function enters a while loop that handles the case that the sliders add up to more than 1440 minutes, then to a similar loop for less. Each slider has a designated "primary" and "secondary" slider. Both while loops balance the three bars by either adding or subtracting from first the primary slider, until it hits an upper or lower limit, and then the secondary slider. The resulting interaction is very smooth and intuitive user interaction with the slider bars. The Edit Appliance Profile view can be seen in Figure 1.2.

My Stats:

The My Stats view displays the user's own data back to them in the form of several pie charts. The initial pie chart displayed is an overview of the user's total usage, and divides the chart into representations of how much each room uses. The view also displays two critical pieces of information above and below the large pie chart, displaying the cost the current pie chart represents, and the worst device within the current chart. Within the initial pie chart that overviews all of the user's information, the extra information will show the user's total cost as well as the worst device over all. At the bottom of the view is a slider full of multiple mini pie charts, the first of these mini charts is a representation of to total view and is located at the far left, from there the slider can scroll through each room's mini representation. The user can pull up the specific data on a room which will then display the chart sectioned by the values each device in the room. When one of the room charts is selected the extra values will break down their cost and worst devices to represent only what is contained within the selected room. The My Stats view can be seen in Figure 1.3.

World Stats:

World Stats compares the user's usage of appliances against the world usage of that same category of appliance. This view is made up of what will be called a 'bar comparison graph' that consists of one large bar representing the scale of the perfect device that uses no power to the worst device we know of from the database. Two smaller indicators along the length of the bar represent the values of both the user's data and the average from the database, both labeled respectively. The large bar will change color based on how they compare to the average usage in that category, going from a green to yellow to red depending on the comparison. The bar will be yellow if they are within 10% of the average. If the consumer is more than 10% over the average the bar will turn red and if they are past 10% under the average their bar will turn green, representing that they are a very "green" consumer. The World Stats view can be seen in Figure 1.4.

V. Design and Implementation Decisions:

The very first decision made was to develop on the iOS platform, as opposed to Android, because objective C interfaces very well with the database platform. Our team was divided into two groups, one would work on the database model and how it interfaces with the controllers, and one to work on the GUI and how that interfaces with the controllers. This was done because of the specialization of our members and need to have both backend and frontend developed at the same time.

On the UI side, one of the most early decisions was to use a tab view controller to switch between the different views of the app. See Figure 3.4 for the a storyboard view of the tab controller. We decided to use mostly standard iOS UI elements to keep the look and feel of the app streamlined and clean. Most of the views were fairly straightforward to put together, using labels, table views, buttons etc. See Figure 3.1 for an overview of the entire storyboard.

The view for creating editing appliance profiles was fairly complex. Our goal was to give the consumer a graphical representation of what states the appliance was in over 24 hours that could be easily manipulated, but also let them have precise control over the exact times. After a number of iterations, we settled on a slider, lock, and two text areas for each state (on, asleep, and off). The sliders could be quickly and intuitively moved and locked, and the text areas doubled as a display of the exact hours and minutes of the slider, and a way to precisely set them. A logic controller actively adjusts all the sliders so they would always sum to 24 hours. See Figure 1.2 and 3.3 for detailed views of the usage profile editor.

The another main element of the UI side of the application is displaying data back to the consumer in ways that are easily interpretable. In order to display the consumer’s own information back to them in a more compiled format, pie charts were added. These pie charts start with an overview of the consumer’s entire profile, divided into each pie slice by room. From that point multiple thumbnail pie charts that represent each room can be selected, displaying a large pie chart and the room’s stats, each slice representing the ratio of each device in the room (as opposed to the rooms themselves). This was changed from the initial design in which there was a list of large pie charts that the consumer could scroll through to see each one.

The second way in which the consumer is displayed data is as a comparison between their own data against other data. Both the consumer’s data and the average representation of that device, which is obtained by averaging that category from the database, are compared on a bar in which their position is a graphical representation of their values. This is displayed once for every device and each device is compared to the statistical average device of the same type.

To make each consumer have the ability to log in, we used a library called ASIHTTPRequest. This allowed us to easily make user sessions with HTTP proxy authentication logic. The way we log in is by applying the username and password to each HTTP request. We pull the username and password from the login screen where we prompt the consumer for the information. These credentials are securely stored in an encrypted keychain to allow the user to not have to enter the credentials every time they would like to use the app. The password is of course obscured in the user view and in the database, it is a completely secure salted hash that would take billions of years to break if there ever was an hacker in the database. The ASIHTTPRequest library also includes many ways to fetch data when given a URL. With our database this data was returned as JSON (JavaScript Object Notation).

JSON is a very compact text based, standard representation used for data. Once the data was fetched we had to parse it because the JSON is not a human readable format initially. We chose to use a library called SBJSON to parse the data returned from the database. SBJSON has many tools to extract the data we need into arrays which can then be parsed into NSDictionaries. NSDictionaries are a wonderful data structure very similar to hash tables. NSDictionaries use keys and objects to keep track of all the data we are storing. This makes it very easy to populate the views of our application with the data from each appliance and appliance profiles. Each appliance has different values for on watts, sleep watts, and off watts.

VI. Results:

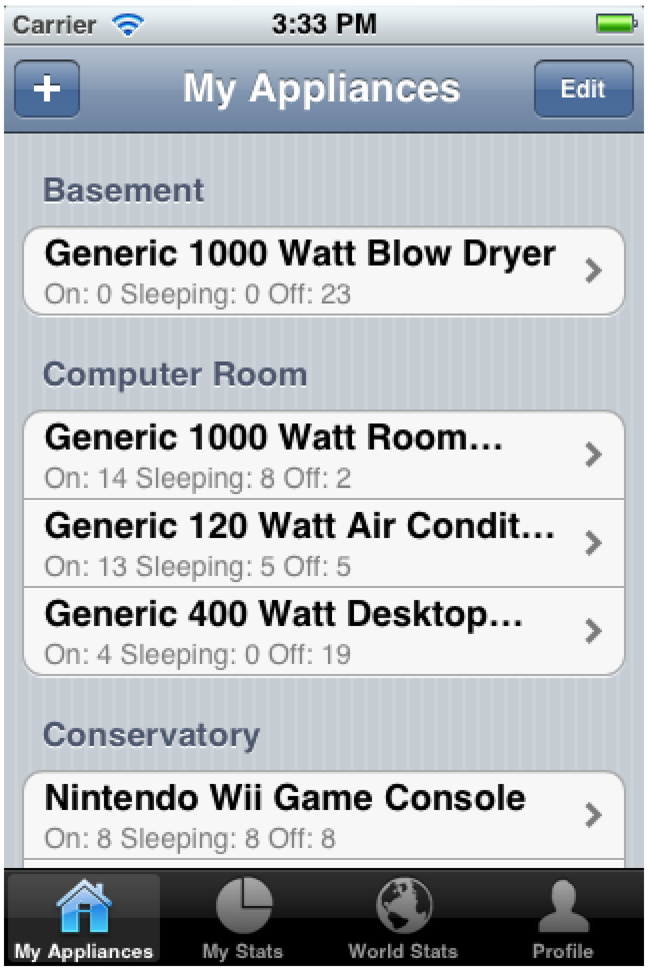
We were able to complete all of the functional and nonfunctional requirements needed to complete the project. Several of the additional features were left out due to time constraints. Some of these features include a Map tab able to show regional statistics and be able to compare that to the consumer's data. Another future feature would be a social function where a consumer could add friends and directly compare energy usage stats stats with them.

We have been able to do some limited testing with the application with persons outside of the team. With the ease of registration, the consumers were able to easily use the application. Some consumers noticed that the app slowed down a little in certain sections, but otherwise the consumers were able to understand each feature.

Due to this testing future work is planned to store some of the data on the device and pull it from the database a little less often. Specifically in the WorldStats view the device noticeably slows down as it finds the average for each section.

We would also like to get a larger sample size for testing. This larger group would be more likely to catch a more infrequent error that could be problematic enough that we would want to fix before going to market.

Images



My Appliances

Basement

Generic 1000 Watt Blow Dryer
On: 0 Sleeping: 0 Off: 23

Computer Room

Generic 1000 Watt Room...
On: 14 Sleeping: 8 Off: 2

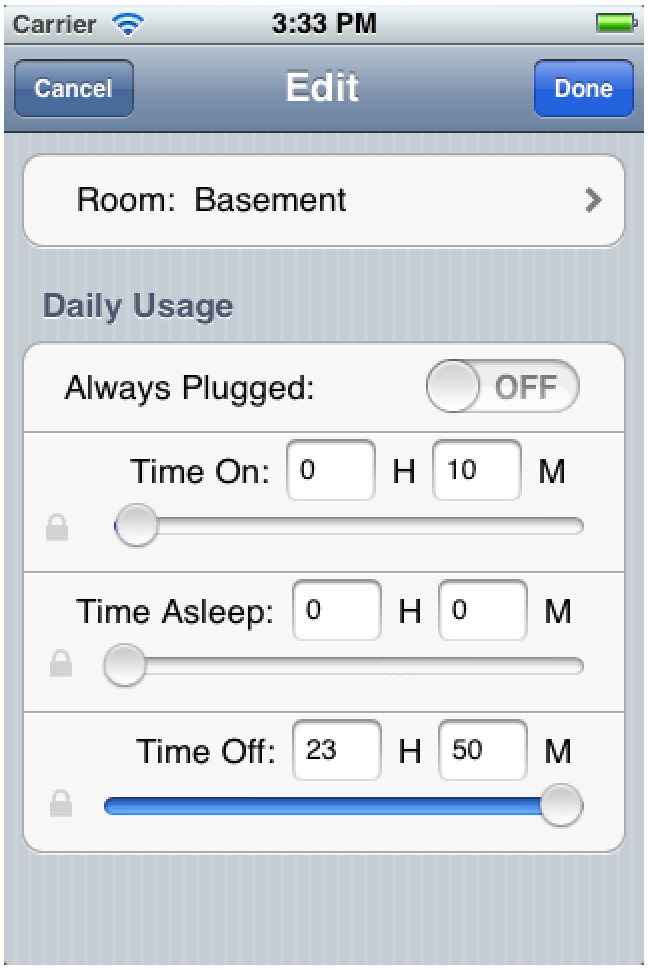
Generic 120 Watt Air Condit...
On: 13 Sleeping: 5 Off: 5

Generic 400 Watt Desktop...
On: 4 Sleeping: 0 Off: 19

Conservatory

Nintendo Wii Game Console
On: 8 Sleeping: 8 Off: 8

Figure 1.1



Room: Basement

Daily Usage

Always Plugged: OFF

Time On: 0 H 10 M

Time Asleep: 0 H 0 M

Time Off: 23 H 50 M

Figure 1.2

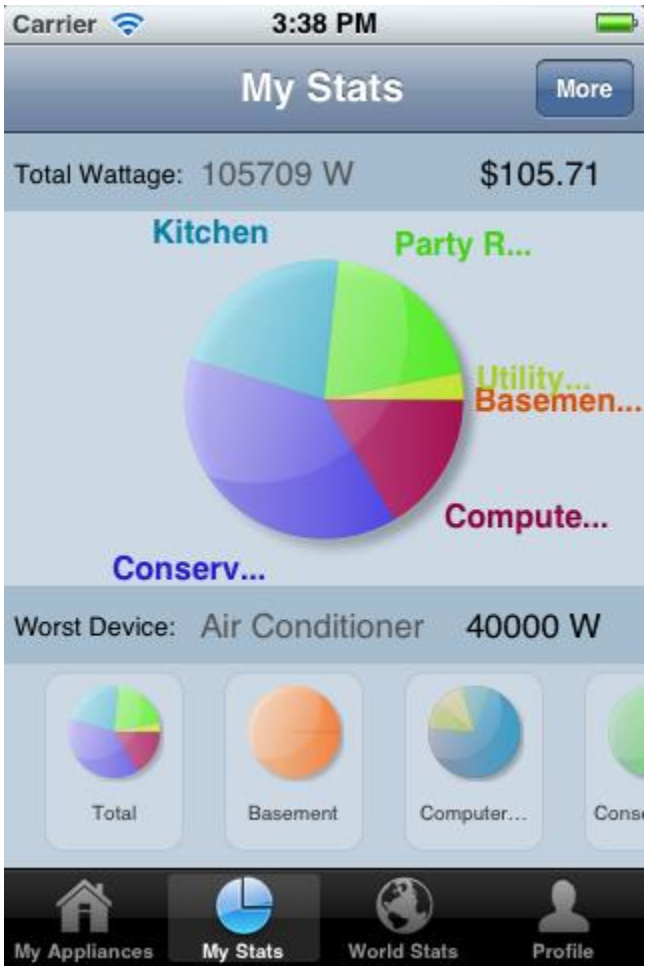

Carrier
3:38 PM
My Stats
More
Total Wattage: 105709 W
$105.71
Kitchen
Party R...
Utility...
Basemen...
Compute...
Conserv...
Worst Device: Air Conditioner
40000 W
Total
Basement
Computer...
Cons
My Appliances
My Stats
World Stats
Profile

Figure 1.3

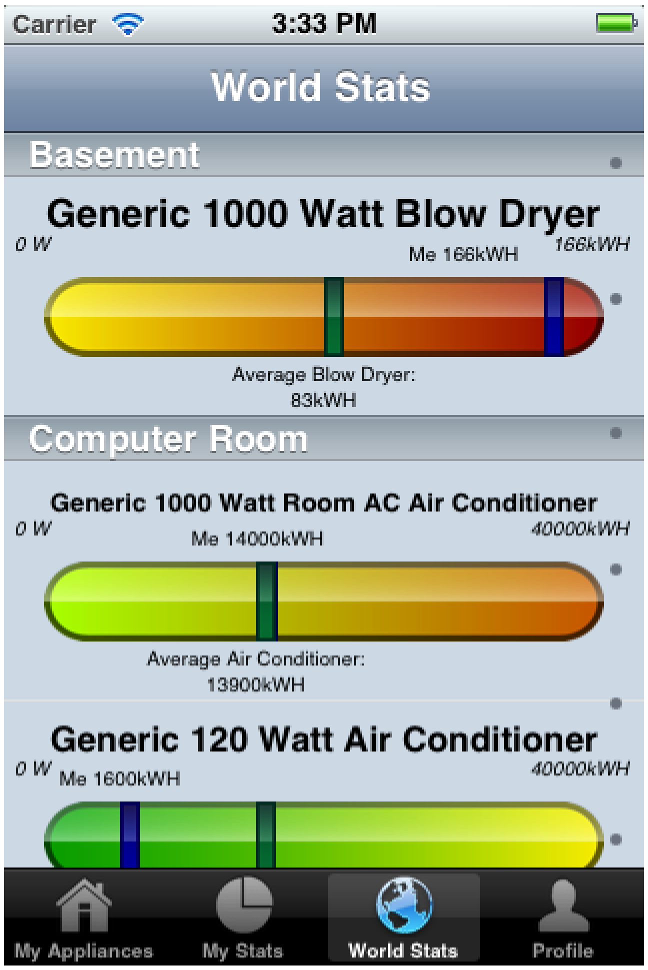

Carrier
3:33 PM
World Stats
Basement
Generic 1000 Watt Blow Dryer
0 W
Me 166kWH
166kWH
Average Blow Dryer:
83kWH
Computer Room
Generic 1000 Watt Room AC Air Conditioner
0 W
Me 14000kWH
40000kWH
Average Air Conditioner:
13900kWH
Generic 120 Watt Air Conditioner
0 W
Me 1600kWH
40000kWH
My Appliances
My Stats
World Stats
Profile

Figure 1.4

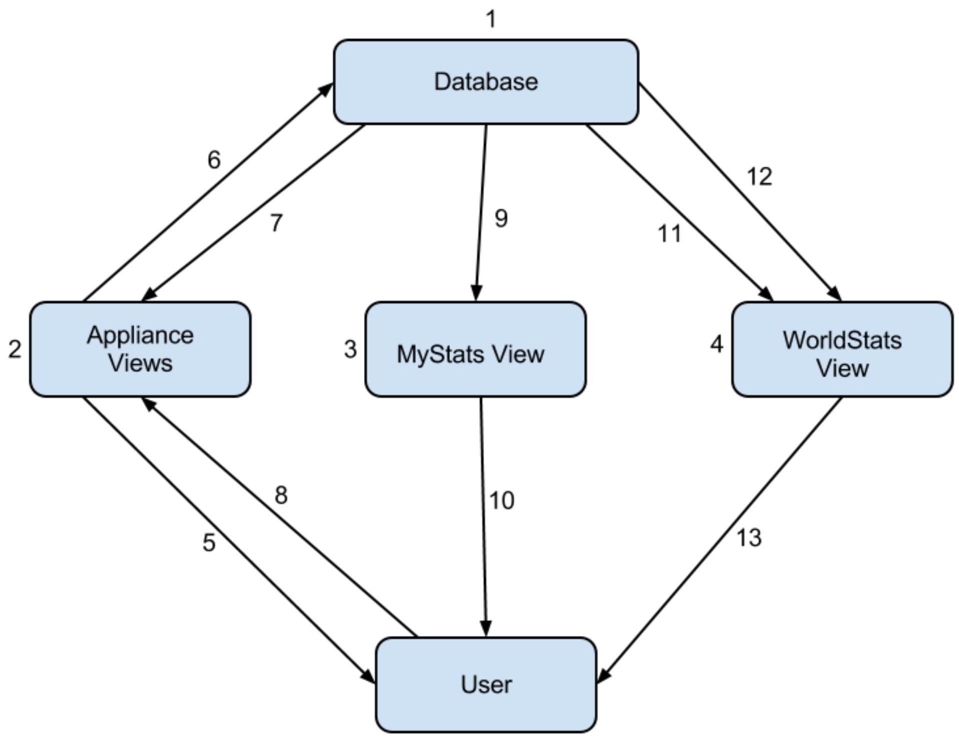


Figure 2.1

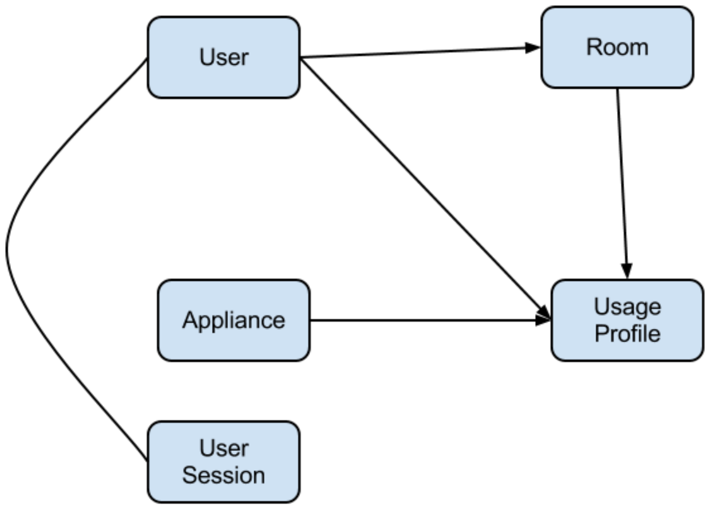


Figure 2.2

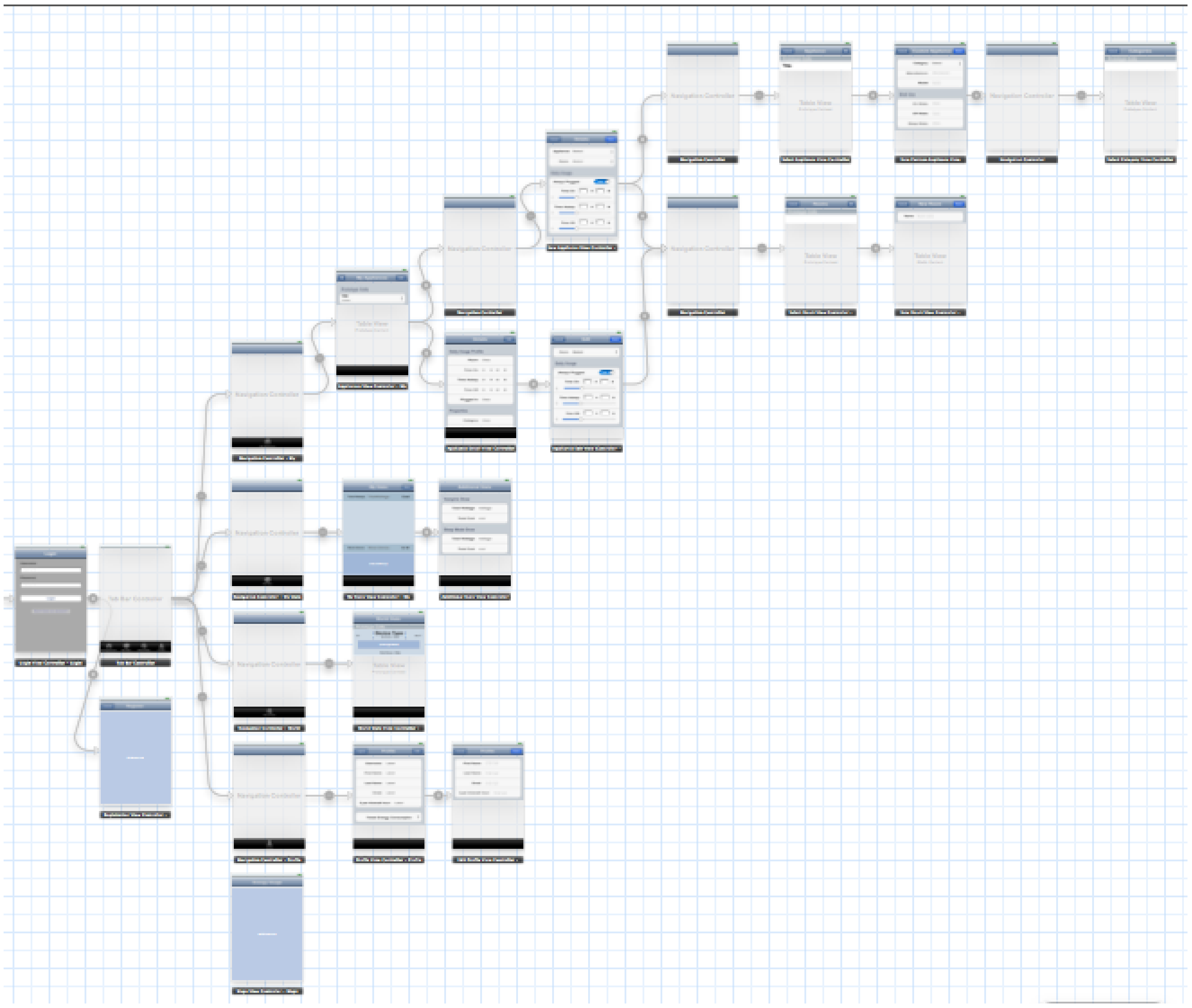


Figure 3.1

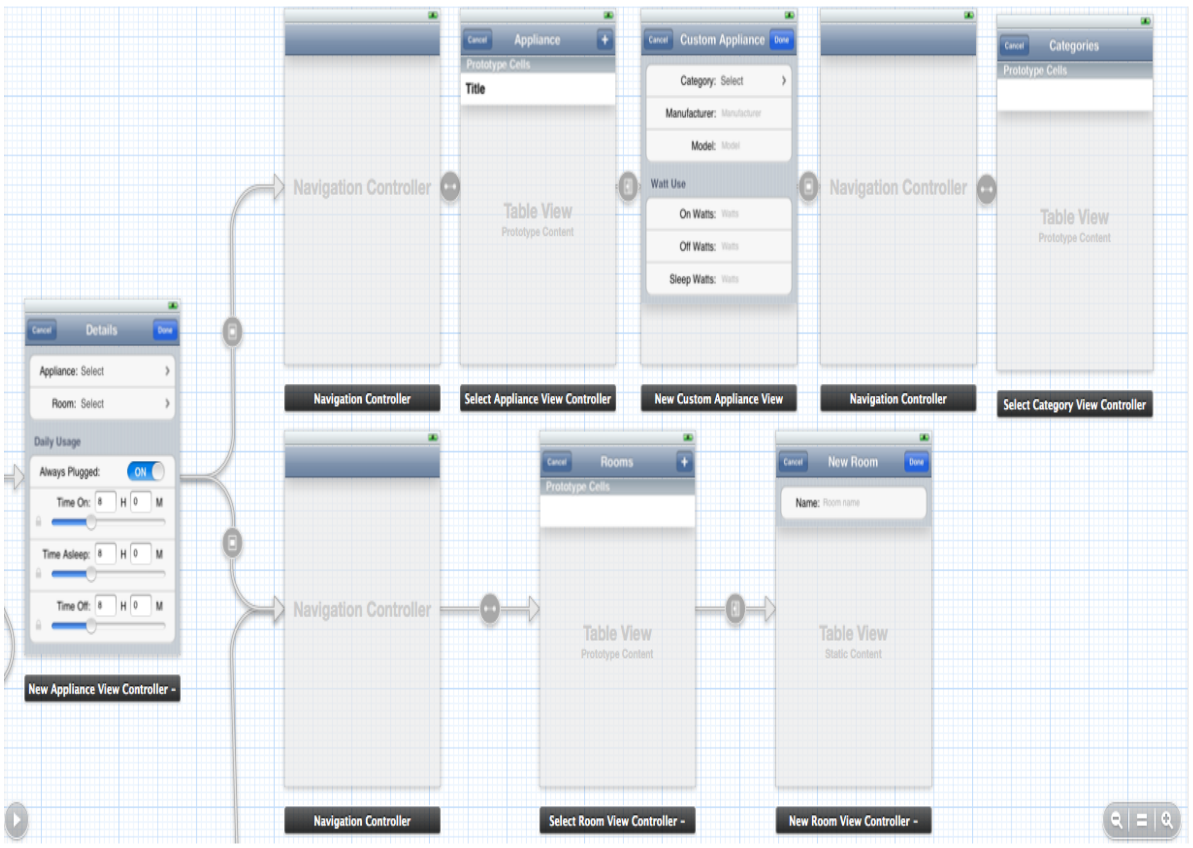

Navigation Controller
Appliance
Prototype Cells
Title
Table View
Prototype Content
Select Appliance View Controller
Cancel
Custom Appliance
Done
Category: Select
Manufacturer: Manufacturer
Model: Model
Watt Use
On Watts: Watts
Off Watts: Watts
Sleep Watts: Watts
New Custom Appliance View
Navigation Controller
Categories
Prototype Cells
Table View
Prototype Content
Select Category View Controller
Details
Appliance: Select
Room: Select
Daily Usage
Always Plugged: ON
Time On: 8 H 0 M
Time Asleep: 8 H 0 M
Time Off: 8 H 0 M
New Appliance View Controller -
Navigation Controller
Rooms
Prototype Cells
Table View
Prototype Content
Select Room View Controller -
New Room
Name: Room name
Table View
Static Content
New Room View Controller -

Figure 3.2

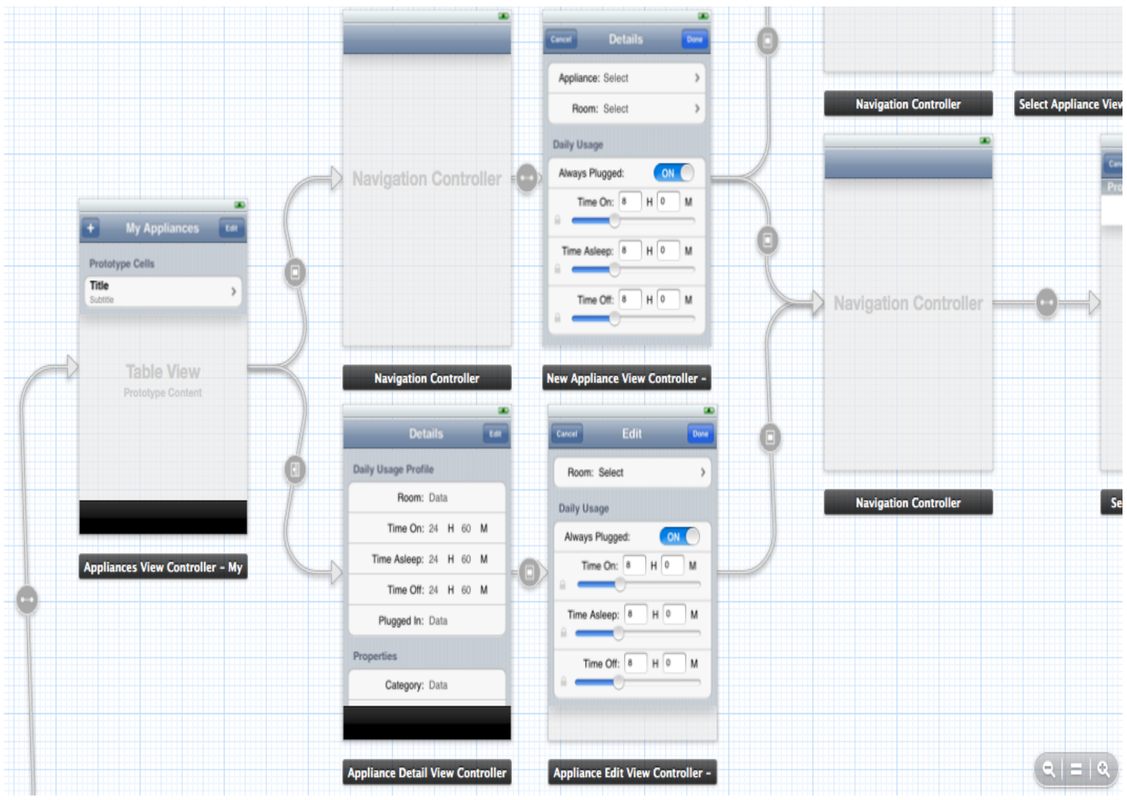

My Appliances
Edit
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Appliances View Controller - My
Navigation Controller
Navigation Controller
Cancel
Details
Done
Appliance: Select
Room: Select
Daily Usage
Always Plugged: ON
Time On: 8 H 0 M
Time Asleep: 8 H 0 M
Time Off: 8 H 0 M
New Appliance View Controller -
Details
Edit
Daily Usage Profile
Room: Data
Time On: 24 H 60 M
Time Asleep: 24 H 60 M
Time Off: 24 H 60 M
Plugged In: Data
Properties
Category: Data
Appliance Detail View Controller
Cancel
Edit
Done
Room: Select
Daily Usage
Always Plugged: ON
Time On: 8 H 0 M
Time Asleep: 8 H 0 M
Time Off: 8 H 0 M
Appliance Edit View Controller -
Navigation Controller
Select Appliance View
Navigation Controller
Navigation Controller

Figure 3.3

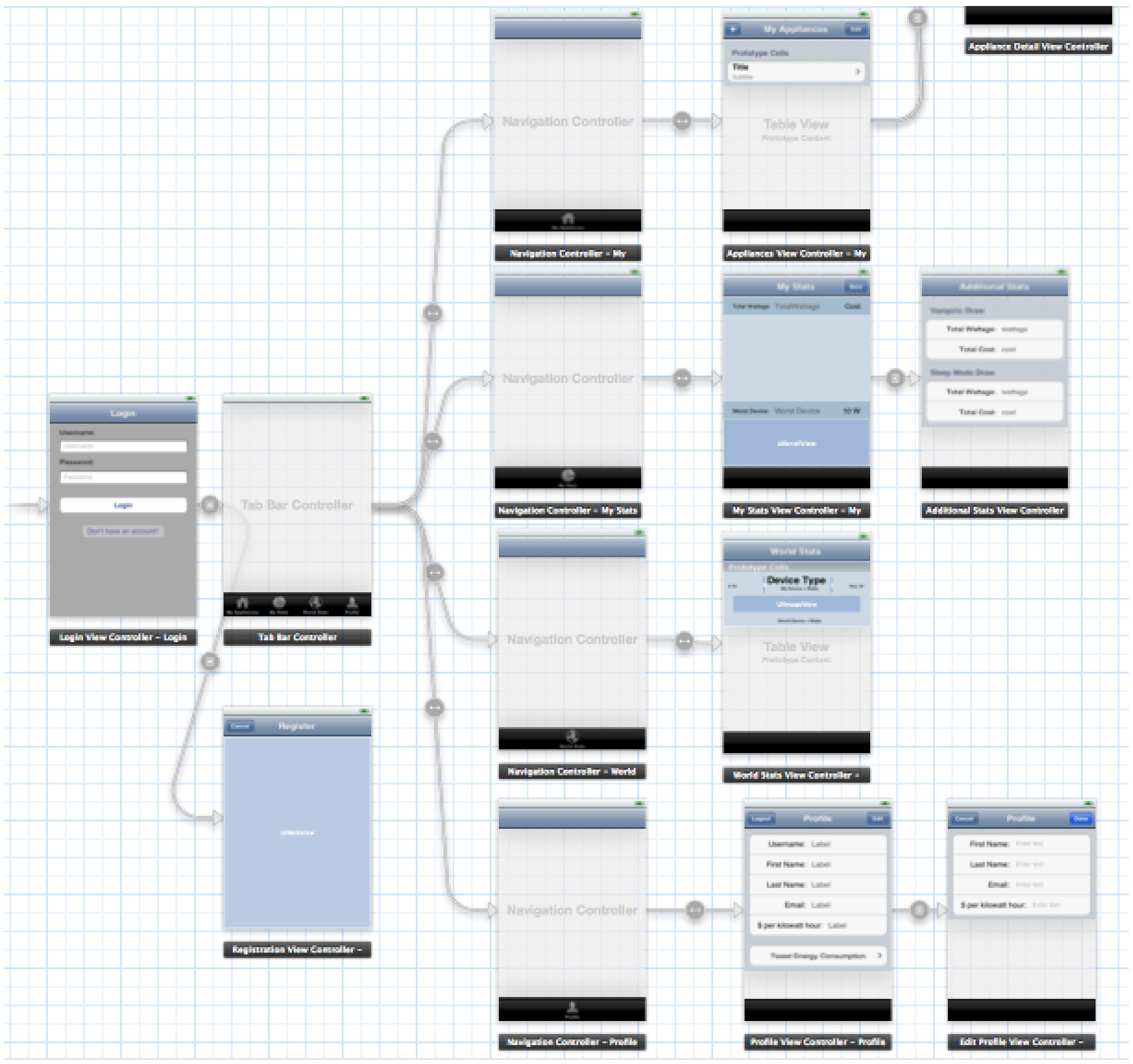

Login View Controller – Login
Tab Bar Controller
Registration View Controller –
Navigation Controller – My
Appliances View Controller – My
Appliance Detail View Controller
Navigation Controller – My Stats
My Stats View Controller – My
Additional Stats View Controller
Navigation Controller – World
World Stats View Controller –
Navigation Controller – Profile
Profile View Controller – Profile
Edit Profile View Controller –

Figure 3.4

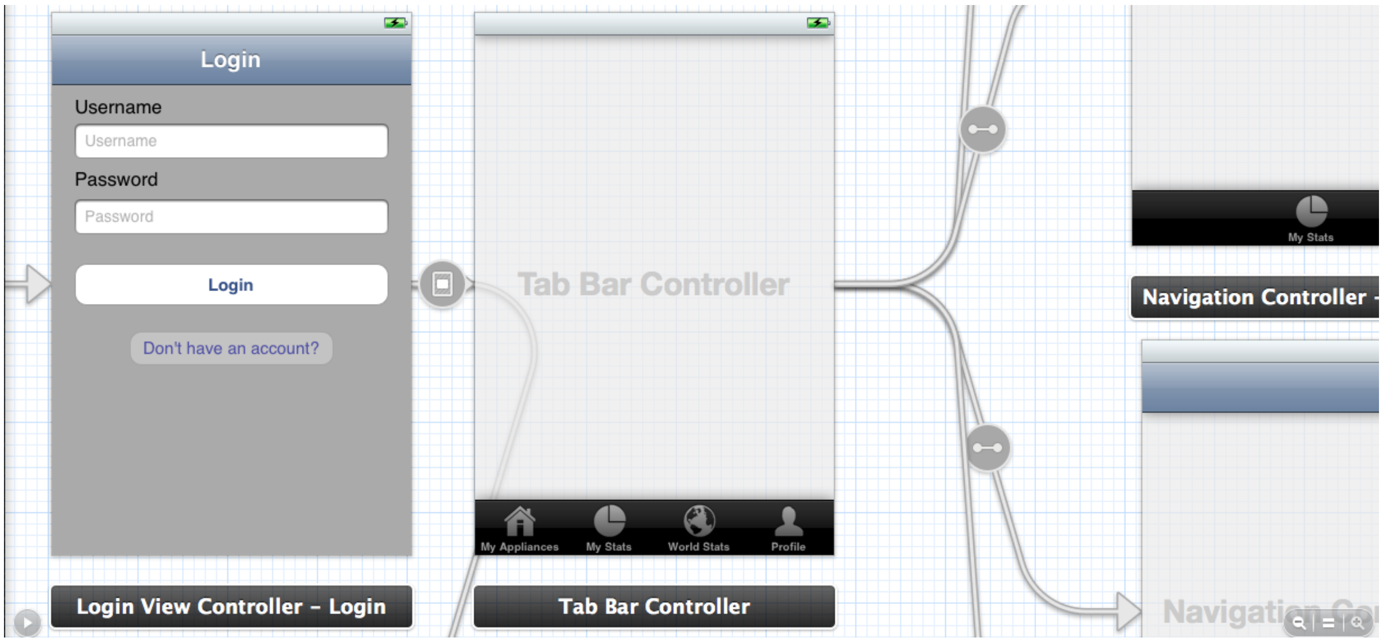

Login
Username
Username
Password
Password
Login
Don't have an account?
Login View Controller – Login
Tab Bar Controller
My Appliances
My Stats
World Stats
Profile
Tab Bar Controller
My Stats
Navigation Controller -
Navigation Co

Figure 3.5