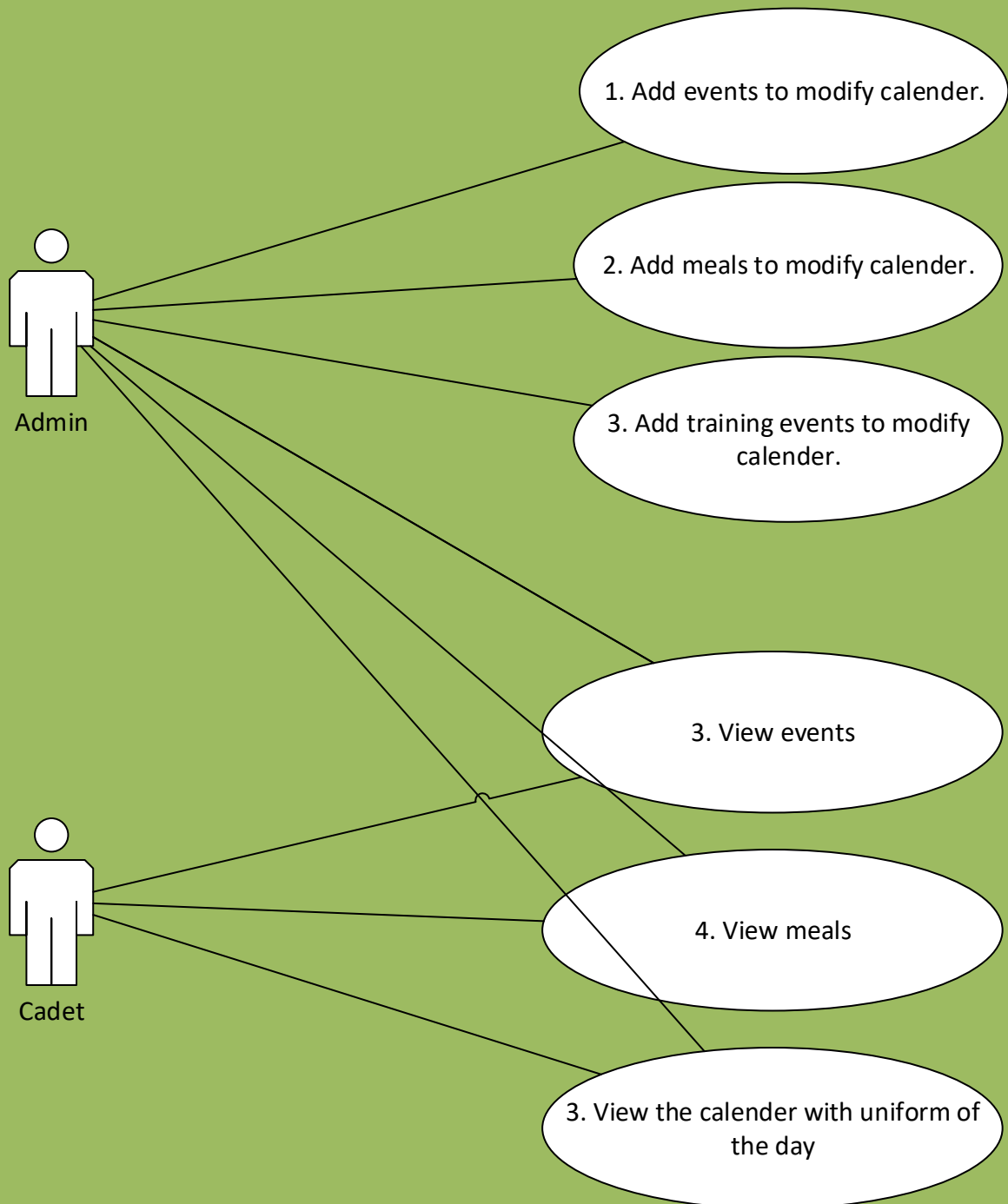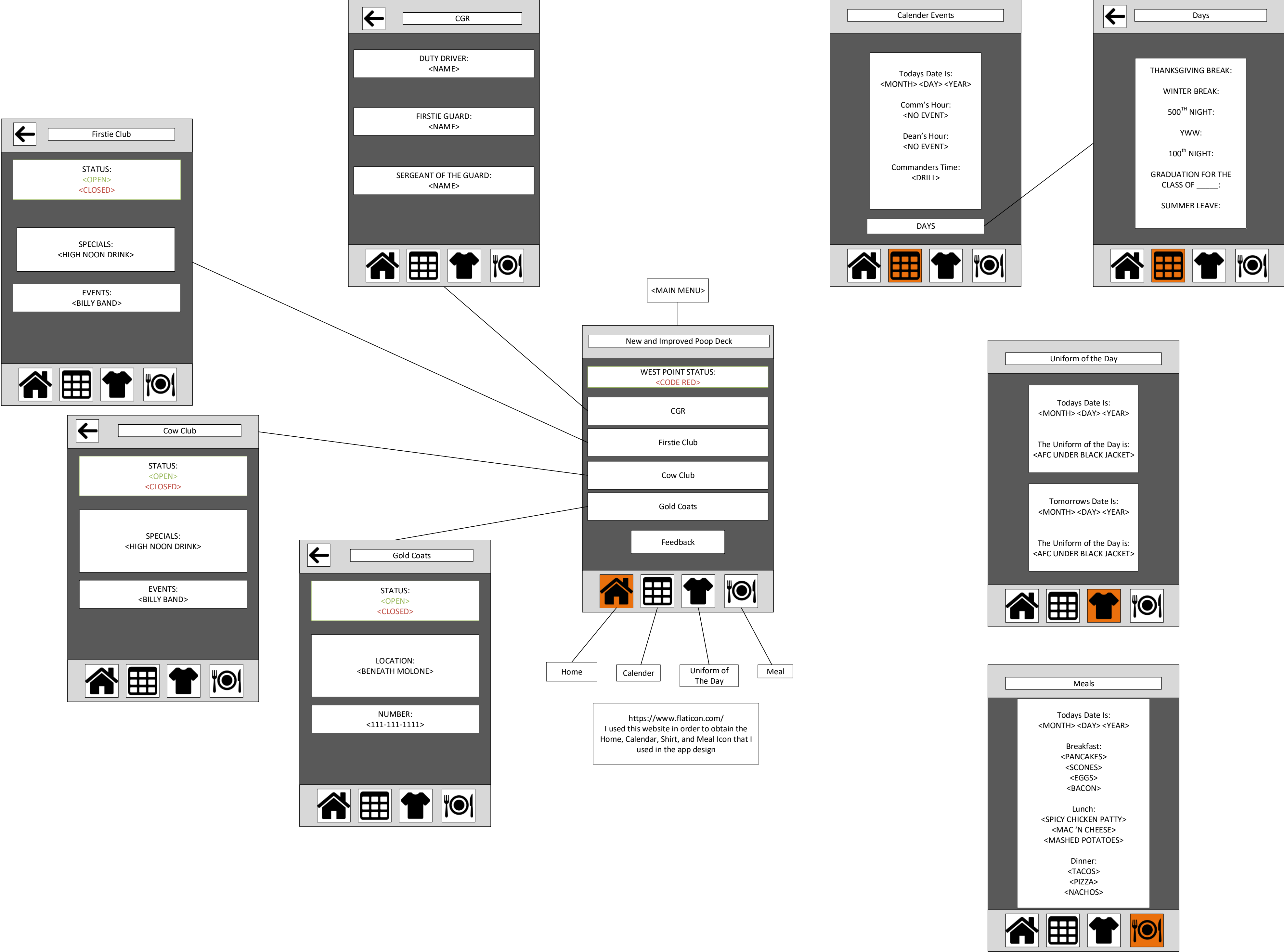Problem Description: Cadets at the United States Military Academy are required to complete numerous tasks with a narrow window of free time to complete these events. However, there is no source or application that compiles all the information necessary onto one page in order to enable these cadets to schedule events and plan ahead. The Poop Deck seeks to resolve this problem. There are so many things to keep track of at the academy, and thus this application lets you see everything you need at the tip of your fingers.

Poop Deck Use Case Diagram

Admin

Cadet

1. Add events to modify calender.

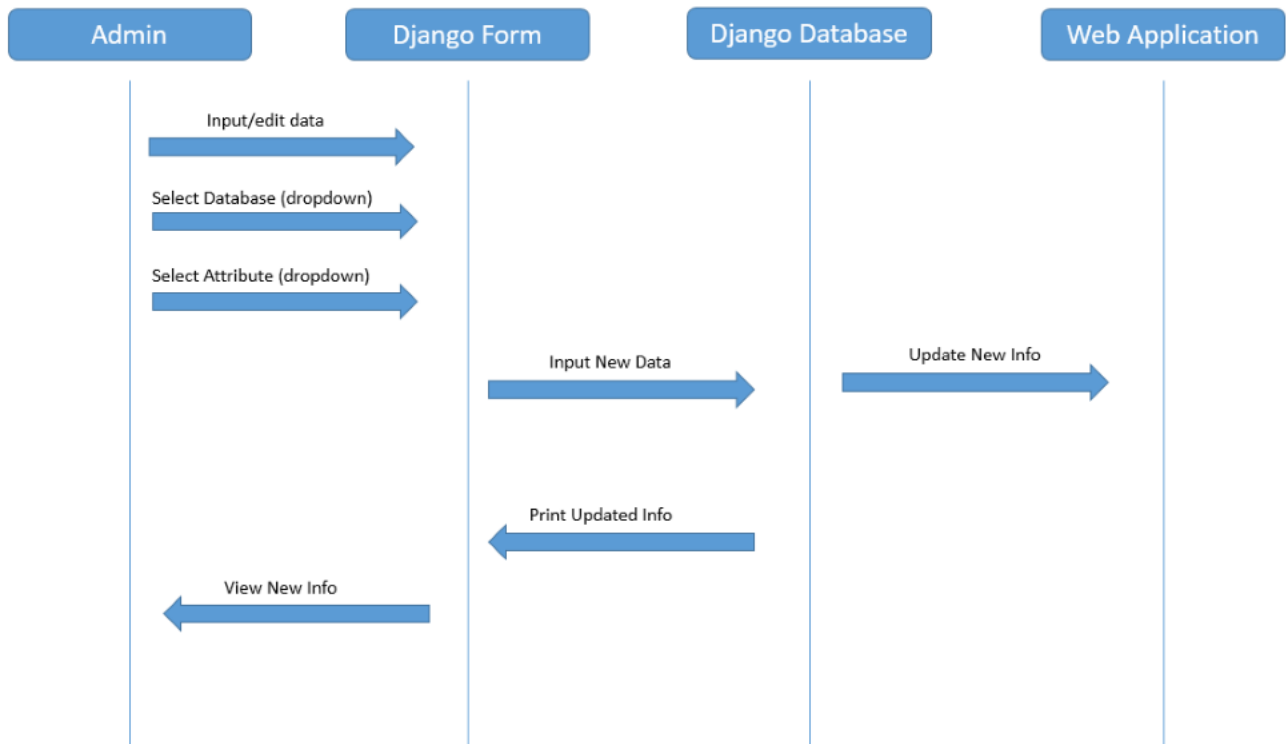2. Add meals to modify calender.

3. Add training events to modify calender.

3. View events

4. View meals

3. View the calender with uniform of the day

CGR

DUTY DRIVER:
<NAME>

FIRSTIE GUARD:
<NAME>

SERGEANT OF THE GUARD:
<NAME>

Firstie Club

STATUS:
<OPEN>
<CLOSED>

SPECIALS:
<HIGH NOON DRINK>

EVENTS:
<BILLY BAND>

Cow Club

STATUS:
<OPEN>
<CLOSED>

SPECIALS:
<HIGH NOON DRINK>

EVENTS:
<BILLY BAND>

Gold Coats

STATUS:
<OPEN>
<CLOSED>

LOCATION:
<BENEATH MOLONE>

NUMBER:
<111-111-1111>

Calender Events

Todays Date Is:
<MONTH> <DAY> <YEAR>

Comm's Hour:
<NO EVENT>

Dean's Hour:
<NO EVENT>

Commanders Time:
<DRILL>

DAYS

Days

THANKSGIVING BREAK:

WINTER BREAK:

500TH NIGHT:

YWW:

100th NIGHT:

GRADUATION FOR THE CLASS OF _____:

SUMMER LEAVE:

<MAIN MENU>

New and Improved Poop Deck

WEST POINT STATUS:
<CODE RED>

CGR

Firstie Club

Cow Club

Gold Coats

Feedback

Home

Calender

Uniform of The Day

Meal

https://www.flaticon.com/
I used this website in order to obtain the Home, Calendar, Shirt, and Meal Icon that I used in the app design

Uniform of the Day

Todays Date Is:
<MONTH> <DAY> <YEAR>

The Uniform of the Day is:
<AFC UNDER BLACK JACKET>

Tomorrows Date Is:
<MONTH> <DAY> <YEAR>

The Uniform of the Day is:
<AFC UNDER BLACK JACKET>

Meals

Todays Date Is:
<MONTH> <DAY> <YEAR>

Breakfast:
<PANCAKES>
<SCONES>
<EGGS>
<BACON>

Lunch:
<SPICY CHICKEN PATTY>
<MAC 'N CHEESE>
<MASHED POTATOES>

Dinner:
<TACOS>
<PIZZA>
<NACHOS>

Meals

PK	MealID INT

TypeofMeal CHAR(20)

SpecialOccasion Char(30)

Location Char(30)

DurationRange CHAR(30)

AnnoucementsMade Char(200)

FK	Day INT

Events

PK	IDnumber INT

Description Char(100)

DurationRange CHAR(30)

ImportanceStatus Char(30)

Location Char(30)

CompletionStatus Char(3)

FK	Day INT

CGR

PK	IDnumber INT

Company Char(100)

DurationRange CHAR(30)

DutyDriver Char(30)

SergeantOfGuard Char(30)

FK	Day INT

Day

PK	CalID INT

CalenderDate DATE

ServicesID INT

Weather Char(50)

Uniform Char(20)

TAPS INT

DescriptionOfDay(30)

Connect Day to Services

PK FK	CalID INT
FK	IDnumber INT

Services

PK	IDnumber INT

Description Char(100)

DurationRange CHAR(30)

Location Char(30)

Status Char(30)

Occurs during => 1..1

Occurs In => 1..1

1..1

1..1

0..n

Occurs In =>

0..n

Occurs In => 1..1

Occurs In => 1..1

1..n

1..n

Adjustment Justifications:
Each event table had an attribute named 'TimeFrame'. This was changed to DurationRange entered in CHAR. The original had time as the data type which would not have worked.
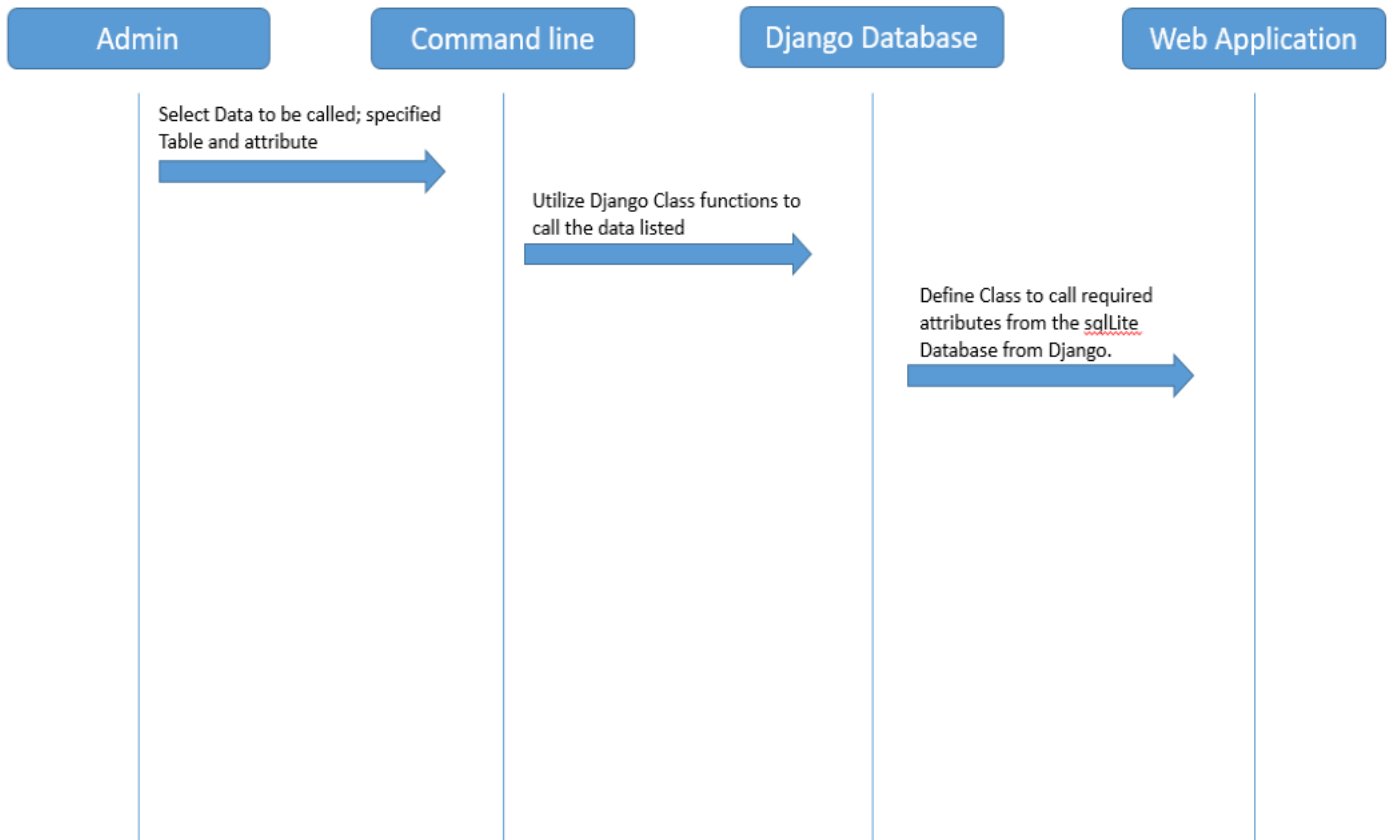
Another edit that was taken out was the type of Day (1 or 2). Day 1 or 2 has a large influence in determining the academic schedule, but little otherwise. After this realization, we saw that it was unnecessary to add a separate table for this information. Therefore, we decided to simply put it into the Description attribute of the Day table.

We also added a connecting table between Day and Services. This was to fix the many to many relationship and split it up into two one to many relationships.

Update Database Information

Admin   **Django Form**   **Django Database**   **Web Application**

Input/edit data →

Select Database (dropdown) →

Select Attribute (dropdown) →

Input New Data →

Update New Info →

← Print Updated Info

← View New Info

Call Information from database

Admin   **Command line**   **Django Database**   **Web Application**

Select Data to be called; specified Table and attribute →

Utilize Django Class functions to call the data listed →

Define Class to call required attributes from the sqlLite Database from Django. →

To Do List

ID	Story	Esimation	Priority
1	Startproject	1	1
2	Verify that server works	1	2
3	Edit Views	3	3
4	Create models	3	4
5	Create Admin user	1	5
6	Start Development Server	1	6
7	Write more Views	3	7
8	Error handling	4	8
9	Update templates	4	9
10	Write Forms	3	10
11	Write Automated Tests	4	11
12	Test and fix bugs	5	12

Initial Testing Plan

Models

- tests.py

o tests the validity of forms

o determines if the models are being created properly

o confirms that data can be retrieved

Views

- write tests to help determine if correct information is being displayed on views

- test forms to see if data is being handles properly

Templates

- Because templates are more about aesthetics, we will have people conduct test interactions with our app to create a more user friendly and appealing app.