Software Requirements Specification

COMP410 - Design Team B

February 2016
## Contents

1 Problem Statement .......................................................... 4

2 Requirements .................................................................. 4

3 Architecture .................................................................. 4
   3.1 Overview .................................................................. 4
   3.2 Data Source Front End (DSFE) .................................. 5
   3.3 User Interface Front End (UIFE) .............................. 5
   3.4 Database Service (DBS) ........................................... 5
   3.5 Compute Engine Back End (CEBE) ........................... 5

4 Solutions .................................................................... 5
   4.1 Entity Modeling ....................................................... 5
      4.1.1 Data Sources .................................................. 5
      4.1.2 Conditions ..................................................... 5
      4.1.3 Resources ...................................................... 6
      4.1.4 Locations ...................................................... 6
      4.1.5 Jobs ............................................................... 6
      4.1.6 Users ............................................................. 6
   4.2 Data Source Communication ...................................... 6
   4.3 Prediction Engine ..................................................... 6
   4.4 Scheduling Engine ................................................... 7
   4.5 User Interface ........................................................ 7
   4.6 Scalability ............................................................... 7
   4.7 Extensibility ............................................................. 7
   4.8 Security .................................................................. 7
      4.8.1 Connection Security ....................................... 7
      4.8.2 User Authentication ....................................... 7
      4.8.3 Access Control .............................................. 7
   4.9 System Analytics ..................................................... 8
   4.10 System Auditing ..................................................... 8
   4.11 Data Record ............................................................. 8

5 Features ................................................................... 9
   5.1 Data Source Front End (DSFE) ............................... 9
   5.2 User Interface Front End (UIFE) ............................... 9
   5.3 Database Service (DBS) ........................................... 9
   5.4 Compute Engine Back End (CEBE) .......................... 9
      5.4.1 Prediction Engine .......................................... 9
      5.4.2 Scheduling Engine ........................................ 10

Appendices .................................................................. 11

Appendix A Glossary ....................................................... 11
### Appendix B  Timeline

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Timeline and Justification</td>
<td>12</td>
</tr>
<tr>
<td>B.2 Core Functionality</td>
<td>12</td>
</tr>
<tr>
<td>B.2.1 Class Library</td>
<td>12</td>
</tr>
<tr>
<td>B.2.2 Cloud deployment</td>
<td>12</td>
</tr>
<tr>
<td>B.2.3 Security</td>
<td>12</td>
</tr>
<tr>
<td>B.2.4 Test Framework</td>
<td>12</td>
</tr>
<tr>
<td>B.3 Schedule Engine</td>
<td>13</td>
</tr>
<tr>
<td>B.3.1 Blackbox</td>
<td>13</td>
</tr>
<tr>
<td>B.3.2 Feasibility</td>
<td>13</td>
</tr>
<tr>
<td>B.4 Database System</td>
<td>13</td>
</tr>
<tr>
<td>B.4.1 Blackbox</td>
<td>13</td>
</tr>
<tr>
<td>B.4.2 Schema</td>
<td>13</td>
</tr>
<tr>
<td>B.4.3 Data Entry and Retrieval</td>
<td>13</td>
</tr>
<tr>
<td>B.4.4 Filter</td>
<td>13</td>
</tr>
<tr>
<td>B.5 Prediction Engine</td>
<td>14</td>
</tr>
<tr>
<td>B.5.1 Blackbox</td>
<td>14</td>
</tr>
<tr>
<td>B.5.2 Making predictions</td>
<td>14</td>
</tr>
<tr>
<td>B.6 Data Source Front End</td>
<td>14</td>
</tr>
<tr>
<td>B.6.1 Endpoints</td>
<td>14</td>
</tr>
<tr>
<td>B.6.2 Put Info</td>
<td>14</td>
</tr>
<tr>
<td>B.7 UI</td>
<td>15</td>
</tr>
<tr>
<td>B.7.1 Design</td>
<td>15</td>
</tr>
<tr>
<td>B.7.2 Endpoint</td>
<td>15</td>
</tr>
<tr>
<td>B.7.3 Job Entry</td>
<td>15</td>
</tr>
<tr>
<td>B.7.4 User Management</td>
<td>15</td>
</tr>
<tr>
<td>B.7.5 Audit</td>
<td>15</td>
</tr>
<tr>
<td>B.8 Integration and Testing</td>
<td>15</td>
</tr>
<tr>
<td>B.9 Prediction Engine</td>
<td>16</td>
</tr>
<tr>
<td>B.9.1 Generate More Accurate Predictions</td>
<td>16</td>
</tr>
<tr>
<td>B.10 Schedule Engine</td>
<td>16</td>
</tr>
<tr>
<td>B.10.1 Black Box</td>
<td>16</td>
</tr>
<tr>
<td>B.10.2 Generate High Quality Schedule</td>
<td>16</td>
</tr>
<tr>
<td>B.11 Data Source Front End</td>
<td>16</td>
</tr>
<tr>
<td>B.11.1 Add/Modify Algorithm</td>
<td>16</td>
</tr>
<tr>
<td>B.11.2 Dynamically Load Algorithms</td>
<td>16</td>
</tr>
<tr>
<td>B.12 Database Optimization</td>
<td>17</td>
</tr>
<tr>
<td>B.12.1 Optimization</td>
<td>17</td>
</tr>
<tr>
<td>B.13 UI</td>
<td>17</td>
</tr>
<tr>
<td>B.13.1 Data Source Management View</td>
<td>17</td>
</tr>
<tr>
<td>B.13.2 Analytics View</td>
<td>17</td>
</tr>
<tr>
<td>B.13.3 Condition Entry</td>
<td>17</td>
</tr>
</tbody>
</table>

### Appendix C  Use Case Diagrams

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.8 Integration and Testing</td>
<td>15</td>
</tr>
<tr>
<td>B.9 Prediction Engine</td>
<td>16</td>
</tr>
<tr>
<td>B.9.1 Generate More Accurate Predictions</td>
<td>16</td>
</tr>
<tr>
<td>B.10 Schedule Engine</td>
<td>16</td>
</tr>
<tr>
<td>B.10.1 Black Box</td>
<td>16</td>
</tr>
<tr>
<td>B.10.2 Generate High Quality Schedule</td>
<td>16</td>
</tr>
<tr>
<td>B.11 Data Source Front End</td>
<td>16</td>
</tr>
<tr>
<td>B.11.1 Add/Modify Algorithm</td>
<td>16</td>
</tr>
<tr>
<td>B.11.2 Dynamically Load Algorithms</td>
<td>16</td>
</tr>
<tr>
<td>B.12 Database Optimization</td>
<td>17</td>
</tr>
<tr>
<td>B.12.1 Optimization</td>
<td>17</td>
</tr>
<tr>
<td>B.13 UI</td>
<td>17</td>
</tr>
<tr>
<td>B.13.1 Data Source Management View</td>
<td>17</td>
</tr>
<tr>
<td>B.13.2 Analytics View</td>
<td>17</td>
</tr>
<tr>
<td>B.13.3 Condition Entry</td>
<td>17</td>
</tr>
</tbody>
</table>

Appendix C  Use Case Diagrams                                           | 20   |
1 Problem Statement

The customer operates in an oilfield logistics role, and has a limited set of high-value resources with which to complete many jobs, some of which occur concurrently. Currently, safety concerns due to weather conditions at a job site or en route to a job site may delay a resource’s availability for its assigned job. In general, these delays are simply “waited out.”

This results in valuable resources sitting idle, waiting for conditions to improve, when those resources might be applied to some other job where the prohibitive conditions do not apply. This is inefficient: when resources are standing idle, less work gets done per calendar period, and that work is relatively more expensive.

The customer wants to use past, current, and predicted weather data to allocate and route resources to job sites while minimizing delays and maximizing resource utilization.

2 Requirements

The system shall:

1. Model the entities of interest.
2. Accept high-volume submission of data from many Data Sources concurrently.
3. Predict future Conditions.
4. Determine feasibility of a Job schedule based on Conditions.
5. Provide an interface for user interaction.
7. Be extensible.
8. Be secure.
9. Offer system analytics.
10. Offer system auditing.
11. Maintain a 5-year record of data.

3 Architecture

3.1 Overview

The system will be written in C#, targeting the .NET framework. It will be deployed to the Microsoft Azure cloud computing platform. It will address the
system requirements through the use of four primary components. The system will additionally make use of the Microsoft Azure Active Directory service for user authentication.

3.2 Data Source Front End (DSFE)

The DSFE is a public-facing interface that Data Sources use to submit data to the system.

3.3 User Interface Front End (UIFE)

The UIFE is a public-facing interface that users use to interact with the system.

3.4 Database Service (DBS)

The DBS is a private service used internally by the system for data storage and querying.

3.5 Compute Engine Back End (CEBE)

The CEBE is a private service used internally by the system to perform data processing, data analysis, resource availability predictions, job schedule validation/generation, and other computationally-intensive tasks.

4 Solutions

4.1 Entity Modeling

The system shall provide a 1-to-1 isomorphism for each of these entities.

4.1.1 Data Sources

A Data Source is an entity that submits data to the system, which is eventually processed to generate Condition entities. Examples include weather sensors and weather forecasting services. A Data Source is uniquely identified by itself so it may be managed in the system. A Data Source is associated with one or many Location(s) that it is located at. The data submitted by Data Sources represents a spatio-temporal series.

4.1.2 Conditions

A Condition represents the state of one or many Location(s) over some time interval. A Condition may represent either an observation or a prediction. For example, heavy rain in Dallas from 5 PM, 2/5/2016 to 6 AM, 2/7/2016.
4.1.3 Resources

A Resource represents some entity that is necessary to the completion of a job, and includes both material, service, and personnel resources. A Resource includes information about the Resource’s current Location, if it has one. Examples include a truck, an electrician, a shipment of drill pipe, etc. Resources in general are hierarchical: for example, a truck depot Resource might “own” several truck resources, a mechanic resource, some tool resources, etc. A Resource has a set of Conditions in which it may not be used or operated. Resources may be viewed or modified by Users.

4.1.4 Locations

A Location represents some physical location in the world. Locations exist as a hierarchy, such that Conditions that apply to a Location representing a region also apply to Locations representing some area or point inside that region, and such that a Resource or Job associated with a specific Location is also considered to be in any enclosing Location(s).

4.1.5 Jobs

A Job is some task which requires the use and/or consumption of Resources to complete, and is issued by some User of the system. A Job may have a Location that it needs to be executed at. A Job has a set of Conditions in which it may not be executed. A Job may be viewed or modified by Users.

4.1.6 Users

A User is a person who has access to the system. Because it is anticipated that most Users of the system will work on Jobs, a User is also a Resource.

4.2 Data Source Communication

The system will expose a communication Endpoint for Data Sources to submit data to the system that is provided by an automatically scaling load-balanced collection of public-facing computing nodes.

4.3 Prediction Engine

The Prediction Engine will use spatio-temporal series stored in the database and generate a predicted list of Conditions for a given Location and Time Frame. Predictive algorithms will be able to be added to the system in an extensible way.
4.4 Scheduling Engine
Given a schedule, the Scheduling Engine will be able to determine if the schedule is feasible based on conditions predicted by the Prediction Engine 4.3 and provide information on where an infeasibility occurred.

4.5 User Interface
The User Interface will prevent the users from directly talking with the Database Service. Given a user input, it will verify the validity of the input and deliver the data to the Database service. When the user requests data, it will create a corresponding Filter, deliver it to the database system, receive data, and show the data in a readable, user-friendly way.

4.6 Scalability
Scalability will be achieved by implementing the system such that the UIFE, DSFE, and CEBE are provided by automatically scaling load-balanced collections of computing nodes.

4.7 Extensibility
The system is designed with future growth in mind. The isomorphisms for data representation are specified as to accommodate changing needs of the client and model more elements of the physical world that wish to be included in the system. Entities are described as being semantic objects that might affect other entities, not necessary modeling any one instance of an object.

4.8 Security
4.8.1 Connection Security
Users' connection to the system will be accomplished over TLS.

4.8.2 User Authentication
User authentication is provided by Azure Active Directory. System can be configured to use a customer’s existing Azure AD instance, or an Azure AD instance can be provided as part of the system. If desired, the Azure AD instance can be periodically synchronized with a local Active Directory domain controller or the Azure AD instance can delegate user authentication to a local Active Directory domain controller.

4.8.3 Access Control
The system will make use of Permissions and User Groups.

A Permission represents a specific User’s right to interact with some object in the system in a specific way. Permissions are stored in a central Permission
Store, and both the Permission owner and the entity it applies to, if any, maintain a reference to any live Permissions they are associated with. A truck driver User might maintain a reference to a Permission in the Permission Store representing his right to view/update the Resource associated with his truck (which will also maintain a reference to the Permission), and references to Permissions representing his rights to view/update any Jobs he is assigned to (which will also maintain references to the Permissions).

This concept of Permissions makes use of the hierarchical nature of Resources. To continue the example given above, a truck depot manager User might maintain a reference to a Permission in the Permission Store that represents his right to view/update the Resource representing the truck depot he manages. Since the truck depot Resource “owns” its associated truck Resources, the truck depot manager User also has view/update permission to those truck Resources.

A User Group represents a group of Users with rights to interact with the system in a general way. The system stores the set of User Groups, and for each User Group stores references to the set of Users that belong to it. Each User also maintains a list of references to the User Groups (if any) it belongs to. For example, an analyst User might maintain a reference to an Analytics User Group representing the right to access the system analytics facilities, and the Analytics User Group will also maintain a reference to the User.

### 4.9 System Analytics

The system will provide a means to query arbitrary spatio-temporal series of data from the database. From these series, arbitrary algorithms can be run to generate analytic information to display.

### 4.10 System Auditing

The system will provide a means to audit the data in the system. All events that occur in the system that are executed by a user, as well as all data that is entered into the system will be able to be queried based on arbitrary Filters. These user events are connected to the user that executed them, as well as the time they were executed.

### 4.11 Data Record

The system will maintain a Data Record. This Data Record maintains storage of at least the past five years of data received from Data Sources, user interactions with the system, and important system events.
5 Features

5.1 Data Source Front End (DSFE)

The Data Source Front End provides a means for Data Sources to submit data into the system. Data sources that use the DSFE Endpoint must identify themselves with a unique identification key as to have the information they submit be tagged and searchable. All management aspects of data sources are done using the unique identification key they provide. When data is submitted to the system, it is processed by a set of algorithms that are user defined.

5.2 User Interface Front End (UIFE)

The User Interface Front End allows the users to manage the types of jobs, resources, or conditions, create new jobs, resources or conditions, view or update jobs or resources, approve or decline a job, audit the system, manage the data source, manage user accounts, and analyze the data. The accessibility of the view to the user will depend on permissions which he or she will be given.

5.3 Database Service (DBS)

The Database Service will provide storage, long-term retention, and querying for all system data. See section 4.1 for the system entities that will be supported by the Database Service.

A database driver will provide an abstraction for the rest of the system to interact with the Database Service. The driver will expose Endpoints that will allow the system to store and retrieve data from the database, with relations intact based on any arbitrary Filter.

The system will provide an extensible Filter object that will be used to query the database. A Filter can correspond to a single, simple query, or can be composed of other Filters in order to create arbitrarily complex queries.

5.4 Compute Engine Back End (CEBE)

The CEBE will encapsulate the Prediction Engine (4.3) and the Scheduling Engine (4.4).

5.4.1 Prediction Engine

The Prediction Engine will accept as input a Location and a Time frame. It will request a spatio-temporal series of a certain time radius and location radius sufficient to make a prediction for the given Location and Time Frame. The Prediction Engine will call algorithms that generate raw predictions and then translate these predictions into a list of Conditions. The Prediction Engine will be extensible enough to allow new prediction algorithms and prediction algorithms for new types of data. The predicted Conditions will be provided to
the requester, a background process, that will use the Database Service (5.3) to input the predictions into persistent storage.

5.4.2 Scheduling Engine

The Scheduling Engine will accept as input a schedule, in the form of a list of Jobs that encapsulate their currently scheduled Time frame, necessary Resources, Location, and list of Conditions under which they cannot operate. The Scheduling Engine will examine the schedule and the predictions from the Prediction Engine and determine, based on the predicted Conditions during the Time frames as well as the set of resources requested by the jobs, whether or not the given schedule is feasible. If a schedule is found to be infeasible, there would have had to have been at least one infeasibility point the the Scheduling Engine found, and this information will be supplied back to the user as well.
Appendix A  Glossary

Glossary

Condition  See section 4.1.2. 5

Data Source  See section 4.1.1. 4–6, 8, 9, 14

Endpoint  Exposed interface for communication with a system. 6, 9

Filter  A constrained predicate which operates over some data and returns only instances of data that satisfy its condition. 7–9

Job  See section 4.1.5. 6

Location  See section 4.1.4. 6

Resource  See section 4.1.3. 6

Time frame  An interval of time over which something is relevant. 9, 10

User  see section 4.1.6. 6
Appendix B  Timeline

B.1 Timeline and Justification

The bullets are of the form Topic: Week Numbers Worked On, Inclusive. The justifications for the topic and timeframe are given in the bullet below. The timeline for the MVP can be found in Figure 1 and Post-MVP timeline can be found in Figure 2. The justification covers elements by section, ordered by time started.

B.2 Core Functionality

B.2.1 Class Library

Weekly: 1  The shared class library is the cornerstone of the project and provides a means to share information in a consistent way across all aspects of the system. The design and specification of these classes is required for any future work on the system. This is why it must be done immediately and quickly.

B.2.2 Cloud deployment

Weekly: 1-2  Having a testable platform available early is a must for integration testing and a shared understanding of the system by each component team. Deploying the software resources to a testing platform establishes the means for this to be done in the future as well as validates high level design. This is why this must be done immediately. Setting up of all platforms, selecting cloud infrastructure components, configuring automated deployment, and testing analytic and autoscale components of the cloud service will take considerable time to do right.

B.2.3 Security

Weekly: 3-5  As this is our customer's main concern but not a necessary feature of an absolutely minimal system, we can postpone the implementation until after critically essential components are complete. However, it is still a key concern, so it is done immediately afterwards and as quickly as possible. Providing a security black box to authenticate users will need time to integrate with all pieces of the system.

B.2.4 Test Framework

Weekly: 1-3  Setting up a unit testing and platform testing framework from the start allows all future development to have means of accessing these necessary tools. Making testing easy and available will improve the quality and number of tests in the system.
B.3 Schedule Engine

B.3.1 Blackbox

Weeks: 2  The schedule blackbox is necessary for the user interface, although not immediately. It needs to be started in week 2, however, because the feasibility portion will require several weeks to complete.

B.3.2 Feasibility

Weeks: 3-5  The feasibility portion of the Scheduler will have to query the prediction blackbox and then use that information to decide if a given schedule is feasible. This requires checking the condition lists of all of the jobs in the schedule against the predicted conditions and verifying that none of the timeframes or resources overlap. This could get complicated depending on the complexity of the jobs and resources in the schedule and needs to be done correctly and unit tested as it is supremely important. This should be able to be set up in a week and unit tested in the subsequent week.

B.4 Database System

B.4.1 Blackbox

Weeks: 2  The database blackbox setup is necessary for other parts of the system to be able to act as if there is a database that they are using. Setting up the API should be accomplished as early as possible, within one week. This will rely on the class library developed in week 1.

B.4.2 Schema

Weeks: 1-2  Start designing schema in week one, because the schema does not depend on the completion of the class library, but rather it just need to know what the class library will consist of (defined in spec). Will take two weeks to research the best storage methods for various parts of our system (the solution may consist of multiple types of databases) and to design an architecture that allows the DB system to access all data, regardless of specific storage location.

B.4.3 Data Entry and Retrieval

Weeks: 3  After database is deployed (along with rest of cloud), functionality to insert/store/retrieve all known data types in system should be completed within a week.

B.4.4 Filter

Weeks: 4-5  After we can put/retrieve data from database, spend rest of time working on receiving arbitrary filters, constructing corresponding queries, and returning proper result set. Major challenge will be “processing arbitrary
filters”, or understanding exactly what data is being requested and how to construct arbitrarily complex queries.

B.5 Prediction Engine

B.5.1 Blackbox

**Weeks:** 2  The prediction black box is necessary for the scheduling engine, which will start its work in week 3. Thus this black box needs to be completed by then.

B.5.2 Making predictions

**Weeks:** 3-5  3 weeks is budgeted for making predictions, which should be enough time to have an algorithm that generates predictions, but probably not good ones. There will be modeling/ML work needed that researches have spent years developing, so this portion needs the maximum possible amount of time, which is 3 weeks. The customer said that the Prediction Engine was a necessary and even key feature of the system, so we should try to devote as much time to it as possible.

B.6 Data Source Front End

B.6.1 Endpoints

**Weeks:** 2-3  The Data Source endpoint will serve as the API through which any kind of sensor/etc. will communicate with our application. This is a very key part of the project as it will be the only means by which we can begin to fill the database with real data. We plan to start in week 2 after shared class library is up and running. Starting early allows us to start putting real info into the database as early as possible. We are budgeting two weeks for dev/int/test which should be a timeframe consistent with the abilities of our class.

B.6.2 Put Info

**Weeks:** 1-4  Filling the database with data generated externally is an essential part of the MVP. This relies on the database entry capability that will be developed in week 3, which explains why this happens starting in week 4. We are budgeting two weeks for this portion of the project. Since this will require collaboration with the DB team and finding some way to test it by actually trying to input external data, we believe that two weeks is sufficient and necessary for dev/test/int.
B.7 UI

B.7.1 Design

**Weeks: 1-4** The UI Design does not depend on the core library design, so we can probably start it earlier to get a mockup of our UI. The whole process will be 3 weeks. The reason is that we see it as a very important part of our system, as the customers will directly interact with the UI and to a great extent they will judge our system according to our UI. We should definitely put more time in it so that we can have a more polished UI presented to the customers.

B.7.2 Endpoint

**Weeks: 2-3** This includes the UI having the functionalities of communicating with the other parts of the system. This should be done immediately after the shared class library is created in order to integrate the system with other systems. Two weeks are allocated since the REST API has to be properly set up and the skeleton functions has to be created for all the views.

B.7.3 Job Entry

**Weeks: 4-5** The functionality related to jobs will start from week 4 till the end of week 6. We will start to work on this part as soon as we are able to insert/retrieve data into/from the database, which will not happen until the beginning of week 4. Work on this section will last for 3 weeks, because “job” is the core functionality of our system. In week 6, when most of the parts are finished, we will finally integrate it with our scheduler engine.

B.7.4 User Management

**Weeks: 4-6** User management includes allocating and removing permissions to the user, and adding and removing a user. This will be done in weeks 4-5, after certain functionalities of security are implemented, and after most of the views involving job events are implemented.

B.7.5 Audit

**Weeks: 4-6** Auditing functionality enables authenticated users to see all the activities that are done through our system. It will be implemented after the main functionalities that involves jobs which includes creating jobs / resources, job types and resource types, approving jobs, viewing jobs are done, and when the UI design is done since we need to test the workflow of the events.

B.8 Integration and Testing

**Weeks: 6** This is an umbrella component for system testing and integration of system components. This is left for the final week as it requires each individual
component to have been completed, and provides ample buffer time for potential problems or unforeseen circumstances to not affect the gross timeline.

B.9 Prediction Engine
B.9.1 Generate More Accurate Predictions

Weeks: 7-10 The generation of more accurate predictions is an iterative process that can be continually improved until the end of the project lifecycle.

B.10 Schedule Engine
B.10.1 Black Box

Weeks: 7 To add to the scheduling engine, the capability of actually modifying the scheduled jobs will be added. Adding this black box functionality to accomplish a minimal edit of the schedule is needed to move forward with UI elements that would access this procedure.

B.10.2 Generate High Quality Schedule

Weeks: 7-9 Schedule generation is a complex problem and will take a time and a concerted effort to implement. However, we need to find a good stopping point and make sure what we have is well integrated and tested. That is why we plan to stop development of this portion before the last week and save that to make sure it is properly integrated.

B.11 Data Source Front End
B.11.1 Add/Modify Algorithm

Weeks: 7-8 The ability to define and modify user created algorithms is a necessary step to improve the workflow of the addition of data sources in the system, as well as change parameters of processing as they evolve. This task will be complicated as it involves defining the mechanism for algorithm storage as well as an ability to load and modify them.

B.11.2 Dynamically Load Algorithms

Weeks: 8-9 After algorithms can be added by users through the UI, they need to be loaded by the DSFE to be used on incoming data. This process will require loading, dynamically adding algorithmic capabilities to the system, and caching these classes to get reasonable speed.
B.12 Database Optimization

B.12.1 Optimization

Weeks: 7-10  Database optimization is an iterative process that can be continually improved without change to the API that the rest of the system is using. Thus we plan to continue to work on it until the end of the project lifecycle.

B.13 UI

B.13.1 Data Source Management View

Weeks: 7-10  This view provides the capability to modify metadata associated with data sources and the ability to view the data associated with the data source in the system. This involves using the already built capabilities of the database service, and simply displaying access to this data in an intuitive format.

B.13.2 Analytics View

Weeks: 7-10  This view provides the capability to view spatio-temporal data for arbitrary data sources. The database already provides functionality to retrieve these series, a method of display needs to be developed. This will require tuning and constant iterative progress in order to develop an effective interface.

B.13.3 Condition Entry

Weeks: 7  The ability to enter new conditions is a critical part of the post-MVP product, but should only take a week to implement given the rigor of the MVP. All back-end functionality for this should be present, so it should be mostly a matter of creating the proper view.
Figure 1: Minimum Viable Product Development Timeline
Figure 2: Post Minimum Viable Product Development Timeline
Appendix C  Use Case Diagrams

These are the use case diagrams for Administrator Use Cases (Figure 3), Data User Use Cases (Figure 4), and Job User Use Cases (Figure 5).

Figure 3: Administrator User Use Case
Figure 4: Data User Use Case
Figure 5: Job User Use Case