<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Design Team A</td>
<td>21</td>
</tr>
<tr>
<td>6.2</td>
<td>Design Team B</td>
<td>23</td>
</tr>
<tr>
<td>6.3</td>
<td>Design Team C</td>
<td>24</td>
</tr>
<tr>
<td>6.4</td>
<td>Design Team D</td>
<td>25</td>
</tr>
</tbody>
</table>

6 System Architecture Analysis
1 Introduction

Although each team managed to produce specifications and documentation that looked wildly different on the surface, the core tenants of functionality and design of the system were consistent across all designs. In previous meetings to discuss integration of design, attempts were staved off as the general opinion was that of an intractable difference due to fundamental problems between each design. This is not truly the case, as each system describes a similar problem and solves it in a similar manner, providing the same set of tools for users to interact with.

In this document, the problem description, system requirements, use-cases, glossaries, and system architecture for each design are discussed and similarities and unique properties mentioned. In each category, there was minimal difference from a high level.

Moving forward, the next iteration of design will attempt to bring in the best of all four worlds as well as considerations outside of what had been thought of before.
2 Problem Description Analysis

Only Teams A and B provided an actual description of the problem, with C and D dipping into a description of a solution to a problem they already have specified. At a high level, the problem descriptions of Team A and B are identical. Both constitute an adequate description of the problem.

2.1 Design Team A

Many factors, such as dangerous weather and resource availability, can cause safety concerns and delays when scheduling work-tasks. These factors cause inefficiencies when schedules cannot be adapted rapidly enough to changing conditions to fully utilize available resources. Currently, Schlumberger does not have a cohesive solution to this issue. The goal of SSA is to pull data from a variety of sources into a central system that can be used to recognize and alert interested users about existing or potential scheduling issues.

2.2 Design Team B

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.

2.3 Design Team C

Note This was taken from slides as specification document did not detail a more involved version of the problem description. This constitutes a description of the solution than the problem.
• Schedule oil field workers
• Coordinate access to data
• Maximize efficiency and cost-effectiveness
• Must be scalable

2.4 Design Team D

Note This was taken from slides. This is more of a solution description than a problem description.

Automating job scheduling to efficiently allocate and resolve dependencies between resources by making predictions based on real-time data in order to avoid jobs failure.

A description beneath the slide reads:

The problem presented to us is the scheduling of oil field workers to maximize efficiency by avoiding weather delays. To solve this problem, we must coordinate access from a variety of users to data coming from diverse sources. This data will use various factors to create schedules to increase efficiency and cost-effectiveness. It must be scalable to support Schlumberger’s large number of clients and high volume of data.
3 Solution Statement Analysis

The differences in the approaches each team took are highlighted by the solution statements that they provide. Similar points raised in the solution statements are:

- Security, Scalability, and Extensibility
- Make Predictions based on input Data
- Manage Data Sources
- Aide Job Scheduling
- Provide User interface to modify data
- Notify Users on changing state
- Auditing Capabilities

Other points raised in the solution statements were:

- Maintain record of data for 5 years
- Estimate cost of a Job

3.1 Design Team A

Schlumberger Scheduling Assistant (henceforth referred to as SSA) is a software system that is intended to facilitate the scheduling of work by providing real-time alerts when issues arise that impact schedule work plans. The full system includes support for the creation and modification of route plans, flexibility to accept and process new data sources, coordination between route schedulers and field workers, and auditing capabilities. The system is designed to be secure, scalable, and extensible.

3.2 Design Team B

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.

3.3 Design Team C

The Product will provide interfaces for the input of data regarding job requests and for assigning resources to be used to fulfill these job requests. The Product shall provide a means of estimating the cost of a job using specified resources during specified time intervals of use. The Product shall provide notifications to relevant parties concerning changes in job status such as delays or cancellations which may be caused by status updates concerning resources or evaluation of other data sources such as weather data obtained by sensors. These notifications may include updated cost estimates to relevant parties.

The Product aims to enable schedulers of the Customer to track resource usage across job requests so that they may be efficiently rescheduled in a timely and cost-efficient manner as circumstances change.

3.4 Design Team D

The product is a cloud-based job management system which will be available through a web interface. It functions as a tool for securely aiding and automating job scheduling to efficiently allocate resources by making predictions based on real-time data in order to avoid job execution failures.

The system is designed as a collection of microservices that includes five basic services: real-time data management service, prediction service, job scheduling service, system monitor service, and database access service.
4 Use-Case Analysis

Overall, the teams had a lot of similarities in their use-case enumerations. There are minor differences that stem from the underlying model of the world that the system uses, such as a concrete schedule / route-plan object that Team A and D use. Most diagrams were the hybrid use-case / system-architecture diagrams that showed depth to the system and how each of the components would interact with each other, which will be treated separately in Section 6. The common groupings of use-cases are as follows:

- User / Permission Modification
- Data Source Modification
- Send Data / Process Data by a Data Source
- Data Source Data Processing Algorithm Modification
- Job Modification
- Resource Modification
- Location Modification
- Time-Series Data retrieval based on arbitrary Filter / Data Export
- Notify User based on some Alert criteria / Create Alert Criteria
- Job / Schedule Acceptance, Modification, and Rejection

Note that *Modification* in this instance refers to the general set of capabilities involving administration and manipulation of the object, such as create, update, modify metadata, delete, and view.

Unique use-cases from the design documents that fit a general case are:

- Edit User Views (UX Modification)
- Create Types of Job / Resource
- View System Performance / Debug views

Following is the extracted lists of use-cases for each team found through their specification documents and use-case diagrams. As stated above, these are the user-facing use-cases, not internal system components.
4.1 Design Team A

- Add User
- Update User Metadata
- Remove User
- View User

- Authenticate User
- Deauthenticate User

- Add Data Source
- Modify Data Source
- Delete Data Source

- Send Data
- Process Data

- Add Processing Algorithm
- Modify Processing Algorithm
- Delete Processing Algorithm
- Retrieve Processing Algorithm

- Publish Route-Alert
- Create Route-Notifier
- Modify Route-Notifier
- Remove Route-Notifier
- Review Route-Alert

- Create a new work-task (Job)
- Modify Job
- Delete Job
- View Job

- Modify Route-Plan
- Create Route-Plan
- View Route-Plan
- Delete Route-Plan
• View Route-Plan Analytics

• Add Resource
• Update Resource Status (Availability)
• Remove Resource
• View Resource
• View Resource Analytics

• Add new Location
• Update Location
• Remove Location
• View Locations

• View System Performance
• Debug System
• Restart Database

• Retrieve Log of Raw/Processed Data
• Retrieve Raw Data resulting in Processed Data
• Retrieve Actions taken by User related to Route-Plan
• Retrieve Data available at a time period
• Retrieve Route-Plans/Resource Analytics

4.2 Design Team B

• Add User
• Add Group
• Add/Remove User Permission
• Add/Remove Group Permission
• Add/Remove User to/from Group
• Update User Metadata
• Update Group Metadata

• Authenticate User
• Deauthenticate User

• Export Data
• Modify Data Point
• View Time-Series Data at a Location
• View Metadata of Data Source
• View Status of Data Source
• Update Metadata of Data Source
• Update Status of Data Source

• Send Data
• Process Data

• Create a new Condition Type
• Create a new Resource Type
• Create a new Job Type
• Delete a Condition Type
• Delete a Resource Type
• Delete a Job Type

• Create a new Condition
• Create a new Resource
• Create a new Job

• View Job
• View Resource
• View Condition

• View Resource Status
• View Job Status

• Update Resource Status
• Update Job Status

• Accept Job
• Decline Job
• Modify Job

• Add Processing Algorithm
• Modify Processing Algorithm
• Delete Processing Algorithm
• Retrieve Processing Algorithm

• View Job History
• View Resource History
• View User Action History

4.3 Design Team C

• Authenticate User
• Deauthenticate User

• View Unfulfilled Jobs
• View Job Constraint Status
• Calculate Cost of Job Assignment
• Receive Constraint or Job Update
• Schedule a Job

• Create a Job
• View Job

• Receive Notifications based on Alert
• Update Job Status
• Update Resource Status

• Create a new Resource
• Update Resource Status

• Send Data
• Process Data

• Add User
• Remove User
• Update User Permissions

• View Filtered Data
4.4 Design Team D

- Authenticate User
- Deauthenticate User

- Edit User
- View User
- Delete User
- Add user

- Create a new Condition
- Create a new Resource
- Create a new Job

- Modify Condition
- Modify Resource
- Modify Job

- Delete a Condition
- Delete a Resource
- Delete a Job

- Add Metadata to Data-Source
- Update Metadata of Data-Source

- Export Data
- View Time-Series / Streaming Data

- View Schedule
- View Schedule Changes
- Accept Schedule
- Modify Schedule

- Edit User Views

- Audit User Information
- Audit Job Information
- Audit Raw Data
- Audit System Prediction
- Audit Resources
5 Glossary Analysis

Here distilled are some common definitions from every team’s glossaries to create a more consistent notion of what the meaning of these entities are.

- **Location** Some physical location, may exist in a hierarchy. Can be viewed and modified by users.
- **Status** A notion of a current state of an entity. Can be modified and changed as the state changes. Resources, Jobs, and Data Sources all have a status.
- **Resource** People, objects, and entities that will be required to complete a Job. A resource might exist at a location. Resources might have a hierarchy. Resources might have dependencies or conditions in which they cannot operate. Can be viewed and modified by users.
- **Job / Work-Task** A Job to complete at a Location that uses resources. Is issued by a user and can be updated or viewed by many users. Might have other dependencies, such as resource status and availability. Can be viewed and modified by users.
  - **Job Dependency** A Job that needs to be completed before this Job can execute.
- **Data** Data from Data Sources. Can come in a variety of forms.
  - **Raw Data** Data as it comes into the system.
  - **Processed Data / Condition** Data that is the result of being processed.
- **Data Source** An entity that submits data to the system through some means, representing a spatio-temporal series that can be analyzed. Generally represents a field sensor, but can be more abstract. Should be uniquely identified. Can be viewed and modified by users.
- **Route-Plan / Schedule** A mapping of a Job to the time it will be executed. Needs to be approved by a User, and can be modified if necessary.
- **User / Client** A person or entity that accesses the system. A User has permissions.

Despite this relative consistency in content of the glossary, there are still points of contention arising once again from the underlying model and priorities of the solution.
• **User Roles** See Team A’s documentation. They detail the set of permissions that each User (a semantic grouping of responsibilities) can have.

• **Processing Algorithms / Predictions** The algorithms that convert input any form of Data into Processed Data. The algorithms may or may not have predictive elements. Can potentially be user defined and modified.

• **Route Alert** A change in status about a scheduled Job. Sent to a Route-Notifier.

• **Route-Notifier** Subscriber that accepts and modifies data based on Route-Alerts. Alerts User to changing status.

• **Job Cost** The cost of a job assignment.

Following in the next sections is the full glossary of each team.

### 5.1 Design Team A

**User Role Definitions**

• **Analyst** Person who creates algorithms and route-notifiers to process raw-data and create route-alerts.

• **Auditor** Person who reviews past actions taken by users in the system and the root causes of those actions. They can views all types of data in the system, although they cannot modify any of the data.

• **Field-Worker** Person who uses the route-plan to complete work-tasks at locations-of-interest. They also update work-tasks as they are completed.

• **Manager** Person who monitors the efforts of a route-scheduler and resource-manager and adds locations-of-interest to the system.

• **Raw-Data-Source** A source of stream of Raw-Data into the system.

• **Resource-Manager** Person who manages and updates the work-resources listed in the system.

• **Route-Scheduler** Person who is responsible for creating and editing route-plans, and modifying them when issues arise.

• **System-Admin** Person who monitors and runs the overall system, including managing system users and their permissions, as well as checking system analytics.
**Item Definitions**

- **Automated-Scheduler** An algorithm that suggests a route-plan.
- **Locations-of-Interest** A location involved in a work-task or route-plan. Examples include well sites, work-resources storehouses, route stops, etc.
- **Processed-Data** The result of processing raw-data using processing-algorithms.
- **Processing-Algorithms** Algorithms that convert inputs into processed-data. These inputs to these algorithms may include raw-data, processed-data, and/or historical (previously computed and stored) data. These algorithms may have predictive elements. Examples include the impact of weather on locations-of-interest or the availability of work-resources.
- **Raw-Data** External Data added to the system. Examples include weather data, military activity, etc. Processed using Processing-Algorithms to generate Processed-Data.
- **Route-Alert** Indications of a change in a status within a route plan. These are added by route-notifiers and are forwarded from route-plans to users. They includes a timestamp and a reference to the processed data that caused it to be generated.
- **Route-Analytics** Statistics and summaries of the effectiveness and accuracy of route-plans
- **Route-Notifier** Stream processor that modifies route plans by adding route alerts. Subscribes to processed-data associated with specific work-resources or locations-of-interest. Updates the work-resources and/or locations-of-interests to reflect the existing or predicted change of status according to the processed-data and sends route-alerts to work-tasks to notify them of these changes.
- **Route-Plan** A sequence of scheduled work-tasks and the locations-of-interest traveled to complete these tasks.
- **System-Analytics** Data about how well the system is performing. Examples include query times, db capacity, debug info, etc.
- **Work-Resources** People & objects allocated to complete work-tasks. These may or may not be consumed by the work-task. Examples include vehicles, raw materials, field-workers.
- **Work-Resource-Analytics** Statistics and summaries of the usage and utilization of work-resources.
- **Work-Task** A job to be completed at a location-of-interest along with the necessary associated work-resources. These tasks can be assigned to field-workers.

5.2 Design Team B

- **Condition** 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.

- **Data Source** 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.

- **Endpoint** Exposed interface for communication with a system.

- **Filter** A constrained predicate which operates over some data and returns only instances of data that satisfy its condition.

- **Job** 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.

- **Location** 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).

- **Resource** 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 Resources has a set
of Conditions in which it may not be used or operated. Resources may be viewed or modified by Users

- **Time frame** An interval of time over which something is relevant.
- **User** 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.

5.3 Design Team C

- **Customer** Schlumberger and its associates.
- **Product** The name of the product this document specifies.
- **Client** The program, which usually has a GUI, which is used by an employee of the Customer to interact with cloud services in well-defined ways.
- **Job** A task which requires resources to have certain statuses (such as to be at certain locations a given time) and other job dependencies to be satisfied.
- **Job Request** A message sent in order to schedule a job. Includes where the job is and what resources it requires as well as any job dependencies.
- **Scheduled Job** The time and resources allocated to the job. Fulfills the requirements of a job request.
- **Job Status** Includes requested, scheduled, in progress, canceled, and completed.
- **Job Dependency** A job that must be performed before another job. Dependencies cannot be cyclic.
- **Job Cost** A function determining the cost of a job assignment given scheduled resources and scheduled dependent jobs.
- **Resource** Anything a job needs to be completed that is managed by our system. Must be identified by type.
- **Resource Status** Includes location of a resource, information on the resource’s condition, and a time interval for which that information applies.
- **Data Source** Sends environmental data to the cloud to inform analytics (ex: weather sensor, updates in political status of regions).
- **Constraint** A factor with a location and a time that affects when/if/at what cost a job can be scheduled.
Under these two definitions, Data Sources and Resources are both Constraints.
Location and time connect the Constraints to the job requests and assignments.
Can also consider job dependencies to be Constraints.

- **Query** Direct request for data from the cloud. Receives a single direct response from the Communication Hub.
- **Communication Hub** Takes data and the clients to which it is to be sent, and relays the data to those clients.
- **User Permissions** Permissions determine what actions a user can perform. Includes the permissions to change user permissions.

### 5.4 Design Team D

- **Requested Job** A requested job is entered by user and includes a set of resources and a time frame to complete a job.
- **Scheduled Job** A scheduled job has an expected start time, an expected end time and allocated resources. When all of a Requested Job’s resources are available, it can be scheduled. The allocated resources will be marked busy at the scheduled timeframe.
- **Resource** Each resource maintains a set of attributes that reflect its state during a set of time frames. These attributes will be used to determine whether they are available to perform a certain job at a certain time.
- **Prediction** A prediction is a set of attributes that defines the state of a resource object at a given time frame, with a confidence level.
- **User triggered event** A user-triggered event is a change of resource state or job state requested by the user, it can include but is not limited to: addition of a resource, job delay, job cancellation, job priority change, etc.
- **Real-time data** Real-time data is time series data input from field sensors. Different types of data can have different forms. These data includes but is not limited to: wind speed, temperature, atmospheric pressure, NWS warnings, etc.
Overall, each Design Team produced a system architecture that had the same high level semantic components, which were each encapsulated for abstraction and extensibility purposes. The main component types were:

- **Generic database wrapper.** Provides high level abstract access to insert arbitrary data and then retrieve it based on some filter that is extensible enough to model full relational queries.
- **User interface.** A separate system element that makes queries to other components of the system on behalf of a user. Handles most of the data manipulation done.
- **Data Source endpoint.** A scalable endpoint that receives data from data sources and then performs some processing to it, storing it eventually into the database.
- **Notification service.** A subscriber to events that detects when a state has changed and then notifies the relevant users.
- **Prediction service.** Makes predictions based on data received from data sources and stores these predictions in the database.
- **Scheduling service.** Takes in all data relevant to a set of jobs that need to be scheduled and verifies the execution potential of the jobs based on predictions or other relevant factors.

The most unique features from a system architecture were the treatment of how information was communicated between the disparate components. All systems were based practically around a publish/subscribe model of communication, with services providing new information that consumers who were subscribed to the service can then process or update state based on.

The following section is an extraction of each team’s system architecture from their specification document.

### 6.1 Design Team A

Team A’s architecture can be viewed at a high level as a pipeline that takes incoming data from Data Sources and pushes it to the user who is at a faucet, and can either retrieve or store data back into the system. They provide a set of views that encapsulate their use-cases to interact with and modify the underlying data storage which then activates relevant subscribers. Here is their description of their system:
At a high level, SSA is based on a four-tiered publish-subscribe model. In some tiers it may be possible to bypass the full complexities of the publish-subscribe model. For example, each producer in Tier 4 could easily know the subscribers to which it publishes, effectively eliminating the need for a more-complicated publish-subscribe framework in that location.

- **Tier 1** Processed-Data Creation
  - Content-based model based on the format and content of the raw-data.
  - Published: Raw-Data from Raw-Data-Sources
  - Subscribers: Processing-algorithms

- **Tier 2** Route-Alert Creation
  - Filtering based on locations-of-interest and work-resources.
  - Published: Processed-Data from publishing-algorithms
  - Subscribers: Route-Notifiers

- **Tier 3** Route-Alert Distribution
  - Topic-based model where locations-of-interest and work-resources are the topics
  - Published: Route alerts from Route-Notifiers
  - Subscribers: Work-tasks

Route alerts are sent from route notifiers to the route plans containing affected work-tasks. Route plans can then be updated accordingly by the system. Work-tasks are notified of alerts pertaining to associated work-resources and locations-of-interest.

- **Tier 4** Route-Alert Notification
  - Topic-based model where route-plans are the topics.
  - Published: Route-alerts associated with specific route-plans
  - Subscribers: Users interested in specific route-plans

Underneath this system there is a Database Wrapper (DBW) that contains a set of methods for handling incoming requests to modify data (add / delete / update) as well as a set of processing algorithms to process stored data from Data Sources. Methods are segregated based upon the entity that they are meant to handle.
6.2 Design Team B

The system architecture is composed of four main components. These are the Data Source Front End (DSFE), the User Interface Front End (UIFE), The Database Service (DBS), and the Compute Engine Back End (CEBE).

- **Data Source Front End** The DSFE 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.

- **User Interface Front End** The UIFE 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.

- **Database Service** The Database Service will provide storage, long-term retention, and querying for all system data. 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.

- **Compute Engine Back End**

  The CEBE will encapsulate the Prediction Engine (PE) and the Scheduling Engine (SE).

  - **Prediction Engine** The PE 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 PE will call algorithms that generate raw predictions and then translate these predictions into a list of Conditions. The PE 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 DBS to input
the predictions into persistent storage.

- **Scheduling Engine** The SE 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 SE will examine the sche-
  dule and the predictions from the PE 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 of the the SE found,
  and this information will be supplied back to the user as well.

### 6.3 Design Team C

They identify three main paths that data flows through their system that use
different encapsulated modules that communicate within the context of the
cloud service. Queries are directed through the same endpoint and then to a
routing service that identifies which modules to call upon. Results returned
to a user are done through a subscription model with a communication hub,
that publishes the results of queries or changes in state to listening Users.
The pathways are:

- **New Data** Any new information that is about user-defined entities
  in the system. This goes through two database operations, first enter-
  ing the data in the database and then finding dependencies based on
  relations stored in the entity.

- **New Data from Data Source** New data of this form is processed to
  change the status of anything that it might affect (e.g. rain postponing
  a Job). The affected entities then have their dependencies resolved and
  their statuses updated, which is all stored in the database. Finally,
  changed status are then pushed to Users who are set to be notified on
  these status changes.

- **Queries** Arbitrary queries on the system are directed to the database,
  and the result is then sent to the communication hub to be sent to the
  User.
They also have a generic database wrapper.

6.4 Design Team D

The system architecture is composed of five main components. These are the System Monitoring Layer (SML), Data Management Layer (DML), Prediction Layer (PL), and Job Scheduling Layer (JSL). These components all lie on top of an overarching Database Access Service (DBAS).

- **System Monitoring Layer** Host of all user views into the system. Provides authentication gate that prevents unauthorized access into the system as well as views to handle administrative use-cases. Provides a subscription mechanism to receive incoming data from data-sources.

- **Data Management Layer** The real-time data management service hosts an endpoint that accepts data from client applications, and stores raw data for up to 5 years. It allows subscription to incoming new data through customized filter.

  Event-based message-passing layer in order to provide access into the system. Data Sources can submit data into the system through here to be processed. Users can subscribe to incoming data based on some filter.

- **Prediction Layer** The prediction service hosts all prediction engines, and is redeployable for if the prediction algorithm needs to be changed. The prediction engine subscribes to the real-time data management service through a customized filter, and produces predictions based on selected data input. Prediction results is maintained within the service, and it allows subscription to updated prediction results through customized filter.

  The prediction layer is subscribed to incoming data from the DML and makes predictions based on some set of customizable Processing Algorithms. These are stored into a prediction database. Services can subscribe to new predictions as they occur.

- **Job Scheduling Layer** The prediction service hosts all prediction engines, and is redeployable for if the prediction algorithm needs to be changed. The prediction engine subscribes to the real-time data management service through a customized filter, and produces predictions based on selected data input. Prediction results is maintained within
the service, and it allows subscription to updated prediction results through customized filter.

Takes Requested Jobs and uses a Scheduler that takes into account dependencies on resources and predictions to generate a set of Scheduled Jobs.

- **Database Access Service** Generic database access wrapper.