**HANA FLIGHT PREDICTOR**

Version 1.0

**Architectural Design Document**

**Carnegie Mellon University – Silicon Valley**

*Jia Zhang*

*Edward Akoto*

*Lydian Lee*

*Madhok Shivaratre*

*Rashmi Devarahalli*

*Vidya Pissaye*

*Fall, 2012*

Table of Contents

Executive Summary 3

Project Requirements 4

Scope 4

Concerns 4

Principles 4

Constraints 4

Related Work 6

Architectural Design 7

Functional View 7

Information View 9

Data Flow 10

Deployment View 11

Architectural Justification 12

Background 12

Quality Attributes 12

Utility Tree 12

Architectural Options 14

ATAM Justification 16

Prototype 17

Implementation 18

Webservices 18

Client App 18

Glossary 20

References 21

# Executive Summary

As people are conducting increasingly more travel in modern society, the quality of flight services plays a critical role in people’s life. The flight delay prediction provides information about an unexpected delay helping passengers to save time and make consequent changes to their schedules. Hence, predicting the potential flight delay and providing passengers with corresponding information in advance has become a leading subject among the quality of flight services.

Researchers have developed many machine learning algorithms towards this prediction problem. Each have its own hypothesis and are able to achieve accuracy to certain extent. A recent research directed in combining the results from different models to provide a better accuracy. In this project, we propose a flexible and extensible architectural design that allows easy switching between the prediction models and dynamic additions of new models. In order to address the problem in real-world, this architecture enables using large-scale data sets and perform prediction in real-time.

# Project Requirements

## Scope

The objective of the system is to provide real-time prediction information about possible flight delays based on historical information and real-time data, e.g., weather information. The user can get the status of flights by providing the airlines, flight number and departure and arrival airport and the departure and arrival date. The system currently predicts for domestic flights within United States.

## Concerns

The key architectural concern is performance. We aim at designing an architecture that leverages the use of SAP HANA’s real-time analytics capabilities, while eliminating all possible causes of delays that can be experienced when processing a cloud-based flight investigation system.

## Principles

The following are the principles that are considered when designing the architecture for this project:

*Separation of concerns* - The application is divided into distinct features and functions with very little overlapping of functionalities. Since the functions identified are distinct with minimum interaction points, low cohesion and high coupling between the functionalities can be achieved.

*Single Responsibility* - Each module identified has a separate and distinct function to perform. Each module is responsible for achieving a specific feature. For example, the Travel Investigator module will only predict delays and the Data collector is responsible for collecting the past weather and flight information.

*Least Knowledge* - Each module is aware of only the functionality that it is responsible for. No module knows the internal details of other modules. For example, the Alternate Recommender module will only be aware of the function that it is responsible for. It will have no knowledge about what the Travel Investigator module does.

*Don’t repeat yourself* - a designated module achieves each feature/function. There is no feature overlapping between modules. No two modules will perform the same function.

*Keep it simple* - The architecture is designed to be simple so that it is easy to implement and understand.

## Constraints

1. The precision (quality) of prediction is based on the mathematical model that will be used. Since the mathematical model is not designed and developed by us, we do not have much control on the precision of the prediction results.
2. Public data is used as the data sources for the air time delay and past weather details and the accuracy of the prediction results depends on the correctness of this public data.
3. Weather is the only factor considered to investigate the delay in air travel. There might be other factors like airport construction, air flight repair, airline strikes etc., that might affect the travel.

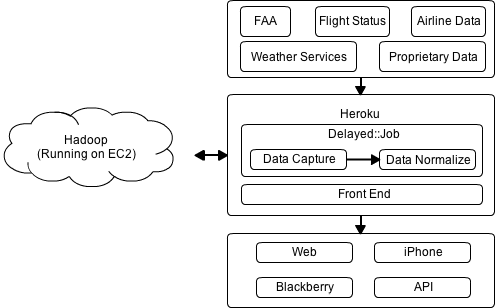
# Related Work

Currently, there are few existing services related to our project. Among them are Flight Stats, All things Analytics and Flight Caster.

Flight Stats [1] provides detailed information about flight status. The FlightStats Flex APIs provide APIs to access flight data including the position, upline/downline information, history, time-zone offsets and flight plans. However these APIs are not free.

All Things Analytics [2] visualize the analysis of flight service quality based on the historical flight service data. To give a faster visualization, this project benefits from SAP HANA database. However, the above project does not provide the prediction of flight’s service quality.

FlightCaster[3] is currently the only service that provides the flight service prediction. It uses Hadoop structure to provide a scalable prediction service. As our goal is to build a flexible architectural structure and utilize the power of in-memory database, FlightCaster is a good project we could reference. Figure 1 shows the overview of the architecture structure of FlightCaster.

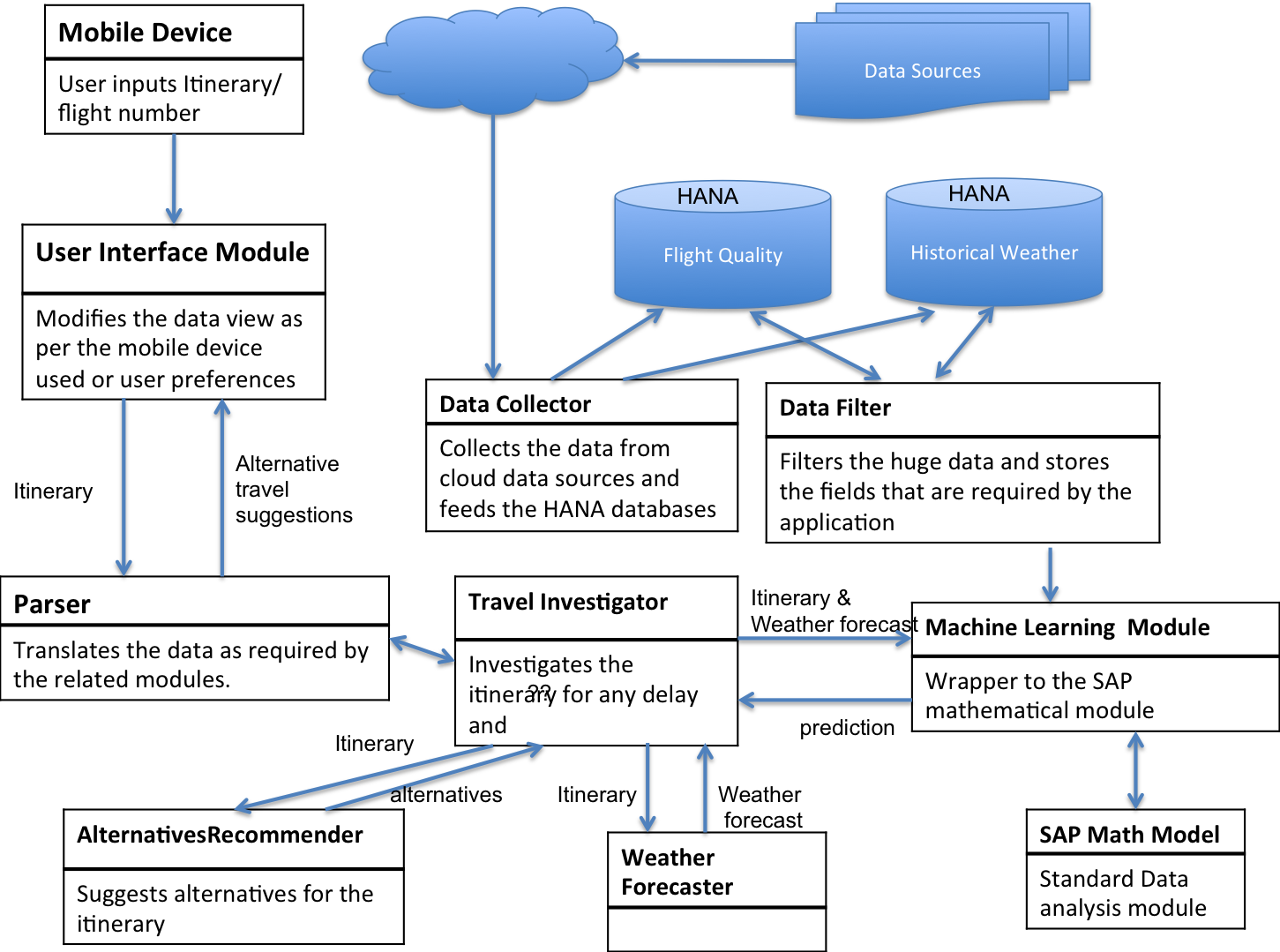


**Figure 1 - FlightCaster Architecture Overview**

# Architectural Design

## Functional View

The functional view captures the modules in the application as per their functionality and the interaction between them.



**Figure 2 - Functional View**

The different functional entities in the application are explained below:

1. User interface module

This module is responsible for receiving the user input and presenting the prediction data back to the user. It also takes care of rendering the view as per the mobile device used and display preferences of the user.

2. Parser

This module acts as a translator between the user interface module and the travel investigator. User can input either the flight number or the detailed itinerary. This module parses the different types of user inputs and provides the itinerary fields as required by the travel investigator module. It also translates the prediction result data of the travel investigator to the format understood by the user interface module.

3. Data Collector

This module collects the required data through the cloud from external data sources like ‘air time delay data source’ and ‘past weather data source’ and stores it in the SAP HANA database used by the application.

4. Data filter

This module filters the large volume of data and updates the HANA database to store the relevant data as required by the application. This module provides the historical data to the machine-learning module.

5. Travel investigator

This module acts as the main coordinator module that communicates with other modules like the parser, weather forecaster, machine-learning module and alternatives-recommender to provide the information needed by the user.

6. Weather forecaster

This module fetches the weather forecast for the itinerary of the user.

7. Machine learning module

This module acts as a wrapper to the SAP mathematical module. It receives the itinerary and weather forecast from the travel-investigator module and historical data from the Data filter module and feeds this information to the SAP mathematical module. It then provides the prediction results as analyzed by the SAP mathematical module back to the travel-investigator.

8. SAP mathematical model

This is the standard mathematical model from SAP that does the data analysis.

9. Alternatives Recommender

This module takes in the itinerary and suggests alternative travel arrangements, by referring to external sources like flight schedules, car rentals and hotels.

## Information View

The information view is comprised of the data schema (Figure 3) and data flows (Figure 4)



Figure 3 - Data Schema

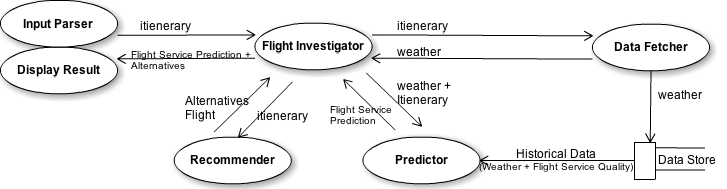
Data Schema

The data model consists the following models:

1. **User:** basic user information
2. **Itinerary**: The traveling plan, including expected itinerary date time(departure/arrival) and city(departure/arrival)
3. **Airport:** Airport code and the city it belongs to
4. **City:** The city name and average weather
5. **Flight**: A flight information, including airlines, departure and arrival city and time.
6. **Weather**: Represent a weather of a city at a given time
7. **Prediction Model:** The service quality prediction based on weather and flight
8. **Historical Flight Service Data**: the historical service quality data of the airline
9. **Flight Service Report**: the predict result of the flight quality
10. **Alternative Flights**: the recommendation of alternative flights
11. **Data Stores**: The external information
    1. Historical weather
    2. Weather forecast
    3. Flight service data

A **user** based on his/her expectations of specific **itinerary**, could have multiple **flights**, according to user’s preference. For each **city**, it has multiple **airports** and multiple **weather** record( **Forecast** and **historical data**). For each **airport**, it contains many flight. For each flight, we also have several **historical service data**. Based on these information, we can then gather a prediction model. For each flight, the **prediction model** generates the **service report** as well as the **alternative flights**.

## Data Flow

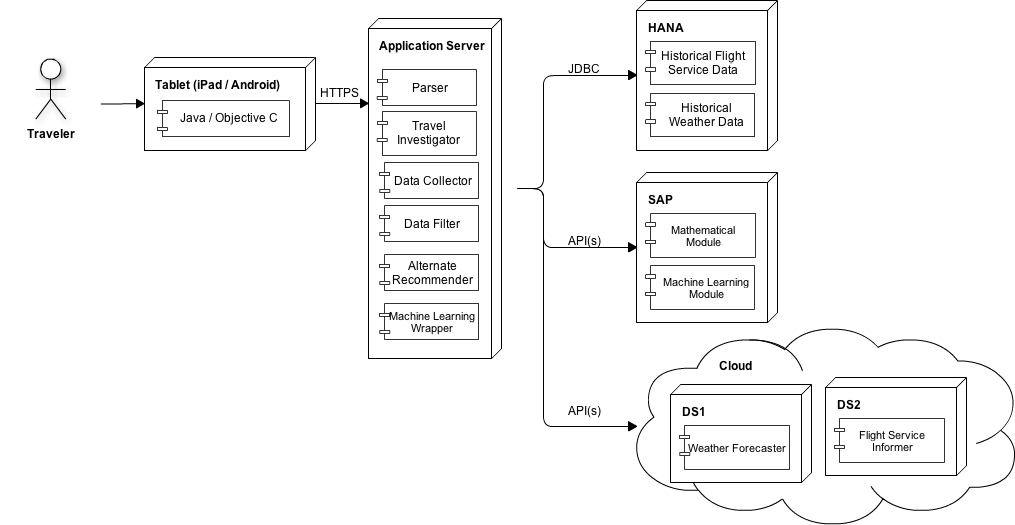


**Figure 4 - Data Flow**

The data flow shows that at first user given itinerary information. These given information will be parsed into **itinerary** object and send to **flight investigator**. Based on the information from itinerary, flight investigator will ask the **data fetcher** for fetching the **weather** forecast information. After the investigator gets the required information, it sends the **itinerary** and **weather** information and ask for the **flight prediction**. The predictor will based on the **historical data** stored in **data store** and give the prediction. The investigator uses the itinerary to ask the recommender for the **alternatives flights**. The investigator then integrates all the information and sends to the display object to show all the information to user.

## Deployment View

The deployment view can be seen in Figure 5. We have chosen to use Google App Engine as it lets us easily scale our application on-demand with load and only pay what we use. It also manages all of the instances, infrastructure, load balancing, replication, and failover for us automatically. App Engine also uses the BigTable-based Megastore that enables our databases to scale to a large number of simultaneous users. Note the JSP instances are our stateless frontend instances spun up on-demand by App Engine.



**Figure 5 - Deployment View**

# Architectural Justification

## Background

The goal of this project is to design a system for analyzing huge amounts of real-time data that affects flights and predict flights’ timings. The system is meant to be open source, scalable, accurate, reliable, modular, maintainable and deliverable as a service.

Business and Mission Drivers

The primary business drivers for this project are, to:

1. Leverage the available prediction mathematical models offered by SAP.

2. Leverage the availability of high performance analytical platforms such as HANA.

3. Create an open source platform to demonstrate the ability of the SAP HANA database.

4. Create a modular system that allows functional components, such as the database and the mathematical models, to be easily replaced or modified without affecting other parts of the system.

5. Design a system infrastructure that facilitates system maintenance and enhancement.

## Quality Attributes

The business drivers indicate high level requirements and contexts for the system. Additionally they help to identify the driving quality attributes for the architecture.

The important quality attributes derived for our application are as follows:

1. Scalability

2. Reliability

3. Maintainability

4. Interoperability

5. Performance

## Utility Tree

The following tables represent utility trees for the highest priority quality attributes.

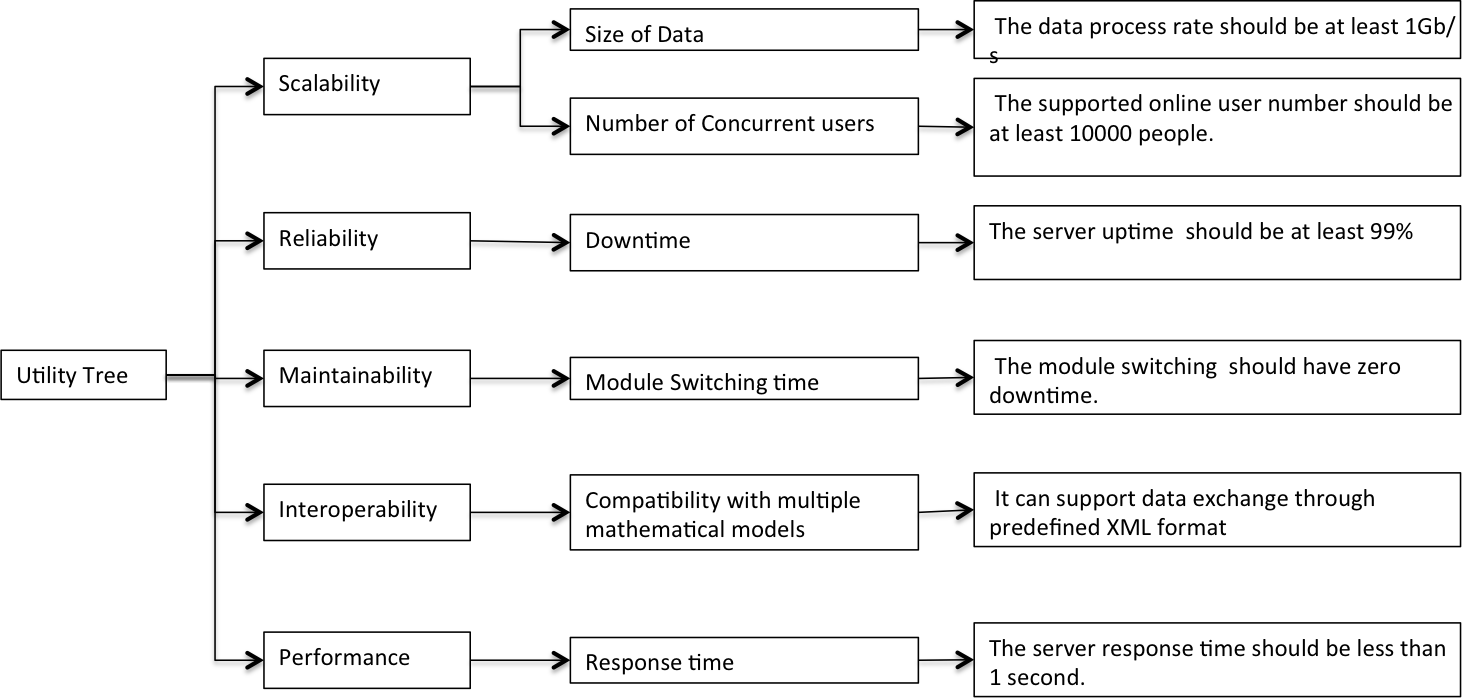
|  |  |
| --- | --- |
| **Quality Attribute** | Scalability |
| **Attribute Concerns** | Size of Data |
| **Scenarios** | The Data Process rate should be at least 1Gb/s |
| **Attribute Concerns** | Number of Concurrent Users |
| **Scenarios** | The supported online user number should be at least 10,000 people |

|  |  |
| --- | --- |
| **Quality Attribute** | Reliability |
| **Attribute Concerns** | Down time and ability to recover from crash |
| **Scenarios** | The server uptime should be at least 99% |

|  |  |
| --- | --- |
| **Quality Attribute** | Maintainability |
| **Attribute Concerns** | Ease to maintain the code |
| **Scenarios** | To make changes to the code and add new features |

|  |  |
| --- | --- |
| **Quality Attribute** | Interoperability |
| **Attribute Concerns** | Ability to work on different platforms and use different mathematical modules. |
| **Scenarios** | Compatibility with multiple mathematical module and multiple platforms like iOS, android or web applications. |

|  |  |
| --- | --- |
| **Quality Attribute** | Performance |
| **Attribute Concerns** | Response Time |
| **Scenarios** | The server response time should be less than 1 second |



**Figure 6 - Utility tree**

## Architectural Options

To attain the above stated quality attributes, various architectural approaches are employed in designing the system.

The application is primarily designed based on the Client Server architecture. The tablet or the mobile device shall be the Client and the Server shall comprise of the business logic, the math model and the Database. The application uses the SAP math model and SAP HANA database. The application shall also be designed to be modular, so that the users can use custom math model and database.

A third party application can make use of the prediction model and the database as a service. Also, the application makes use of mathematical model by SAP as a service to predict the delay. To achieve this, the system shall make use of the Service Oriented Architecture(SOA).

The architectural approaches are listed below:

1. Client Server (maintainability)
2. SOA (reusability)
3. Layered architecture: The system shall be designed as a layered architecture making it modular and easy to maintain. (Interoperability, Maintainability, Reuse)
4. Event-based: The system shall update itself continuously with the real time data from the external sources.
5. Load balancer (Scalability): The system shall be able to support many users at a time without compromising on the performance. To achieve this, a load balancer will be used to distribute the load uniformly across the processing units when the application is used by thousands of users at a time.

## 

## ATAM Justification

|  |  |
| --- | --- |
| **Quality Attribute** | **Architectural approaches to address the quality attribute** |
| Scalability | •Load balancer that distributes the load across different processing units.  •Multiple instances of Predictor to cater to many users at a time  •Abstract factory that creates the predictor instances on demand. |
| Reliability | •Client server architecture provides data centralization and integrity.  •SAP Hana DB automatically backs up the data periodically. This makes sure that the system can recover quickly from a crash without any data loss. |
| Maintainability | •Layered architecture  •Component based – loosely coupled |
| Interoperability | •User Interface parser layer that provides the data as needed by the Server.  •Adapters for database and mathematical models that makes sure the system can use different database or mathematical models. |
| Performance | •HANA Database: in-memory database makes data access incredibly fast.  •Cache Memory: Prediction result for most repeated flights stored in cache to improve the performance. |

# Prototype

The prototype that we developed is a simplified version of our final application, containing only the most necessary modules. The prototype includes the following modules:

1. User interface

The prototype user interface is built for iPad. The interface is mainly composed of two screens; the first screen gets user itinerary information that includes airline, flight number, departure and arrival airports and dates from the user and the second screen displays the predicted flight delay information.

1. Prediction model simulator

The prototype prediction model is a placeholder for an actual prediction model that utilizes SAP’s mathematical model. This model uses the user flight itinerary from the *UI interface*, current weather information from the *Web Crawler,* and the historical flight delay information from the *MySql Database* to predict flight delays and returns a JSON object consisting of the predicted results. The model is built using Java.

1. Web crawler

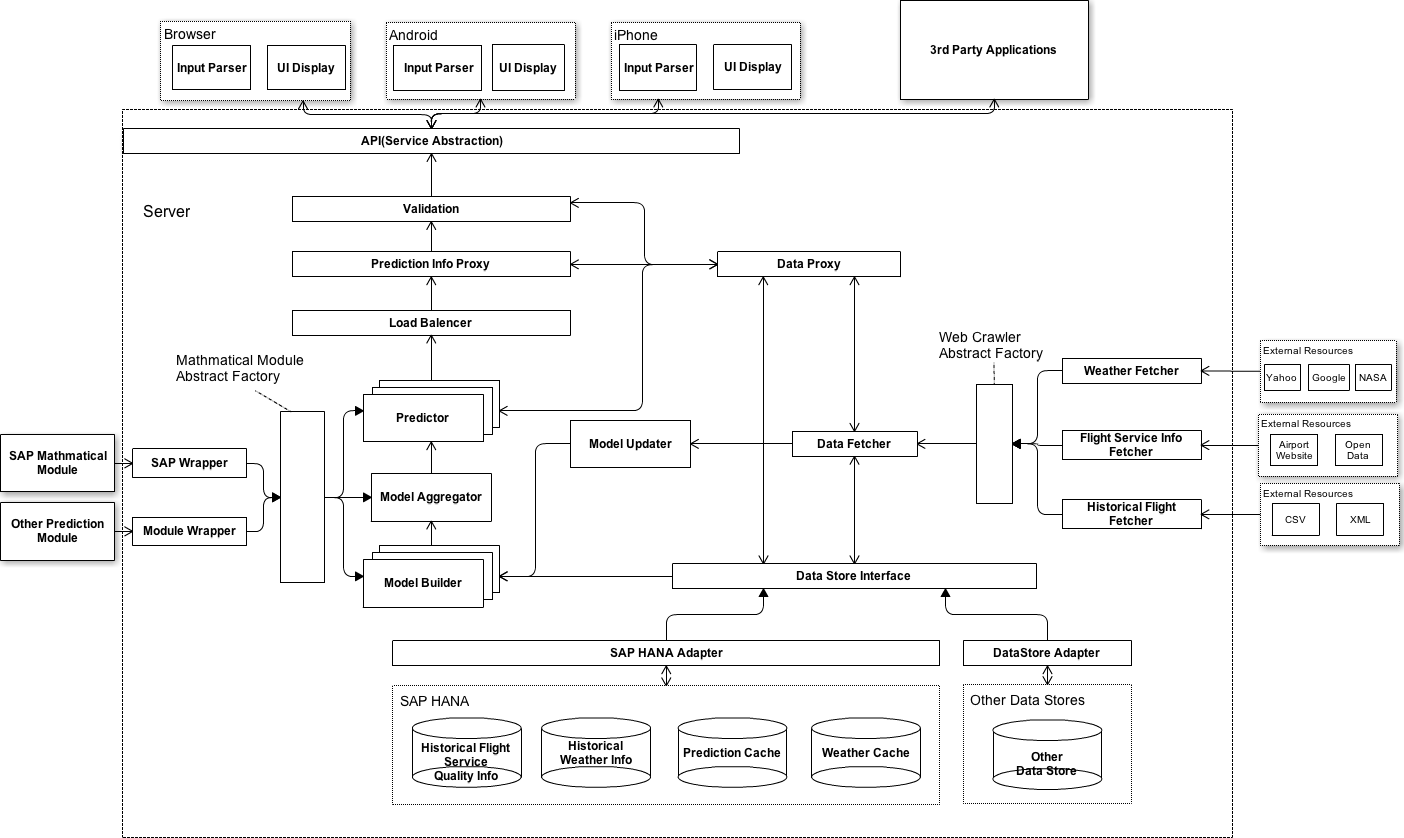
The web crawler fetches a five day weather forecast information from a free online API on the web (weather.com) and return data in JSON or CSV format as requested.

1. MySQL database

The MySQL database stores historical flight delay data needed to create the prediction model. The database contains airline, departure city, destination city, flight number, date and time.

## Implementation

Basically, the implementation could be separated into 2 parts: Webservice and Client App(iPad). Figure 7 shows our implementation architecture overview.



**Figure 7 - Architectural Overview**

## Webservices

We built the web service on Play MVC framework. This is a Java-based framework, and it provides a flexible and simple solution for building RESTful web service. Since currently we don’t have access permission to HANA database, we use the similar SQL-based database: MySQL for the first version.

## Client App

For client app, we design a thin client structure. Therefore, the client app is in charge of basic input parser, validator and final display. We built our first version demo on iPad, therefore Objective C is used for building this demo app.

# Glossary

ATAM: Architecture Tradeoff Analysis Method

HANA: High Performance ANalytic Appliance (name of SAP’s database, not an acronym)

SAP:A market and technology leader in business management software, solutions, and services for improving business processes.

Cloud:pertaining to or doing business on the Internet

# References

[1] Flight Stats <http://www.flightstats.com/>

[2] All Things Analytics <http://ata.blog.s3-website-us-east-1.amazonaws.com/>

[3] Flight Caster <http://www.flightcaster.com/>