Architecture design document for Tracking and Mining Flight Prices (TAMFLIP)

1. Overview

    a. System Overview: Tracking and Mining Flight prices is a software to help the user find out the cheapest and most convenient flights and also track them or predict future prices for flights.

    b. System Context: The system context is defined clearly in the SRS. Basically, the user is the main sink of the information. The user is also a major source of information/data. The other source of data is a website from where data on current prices is obtained.

    c. Stakeholders of TAMFLIP: The main stakeholders for the system are the individual users who might use the system and the system designer/developer who will build TAMFLIP. The main concerns of the two stakeholders are :

        For Users: The usability of the system (providing search, displaying statistics etc), response time of the results and getting notifications everyday for their tracked flights.

        For designer/developer: The system needs to be easy to modify, scalable, secure and handle future extensions which are mentioned in the SRS.

    d. Scope of this document: In this document, we describe two possible architectures for TAMFLIP. We compare these two architectures and select the most appropriate one.

    e. Acronyms and Abbreviations:
        - API: Application Programming Interface.
        - TAMFLIP: Tracking and Mining Flight Prices.
        - SA: Software Architecture.
        - UI: User Interface

Definitions:

- API: A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service (freely available online flight info APIs in this case)
- Flight: Informal term for an aeroplane trip.

2. Architecture Design

   a. Architecture 1:

   Components are the computational elements or data stores.

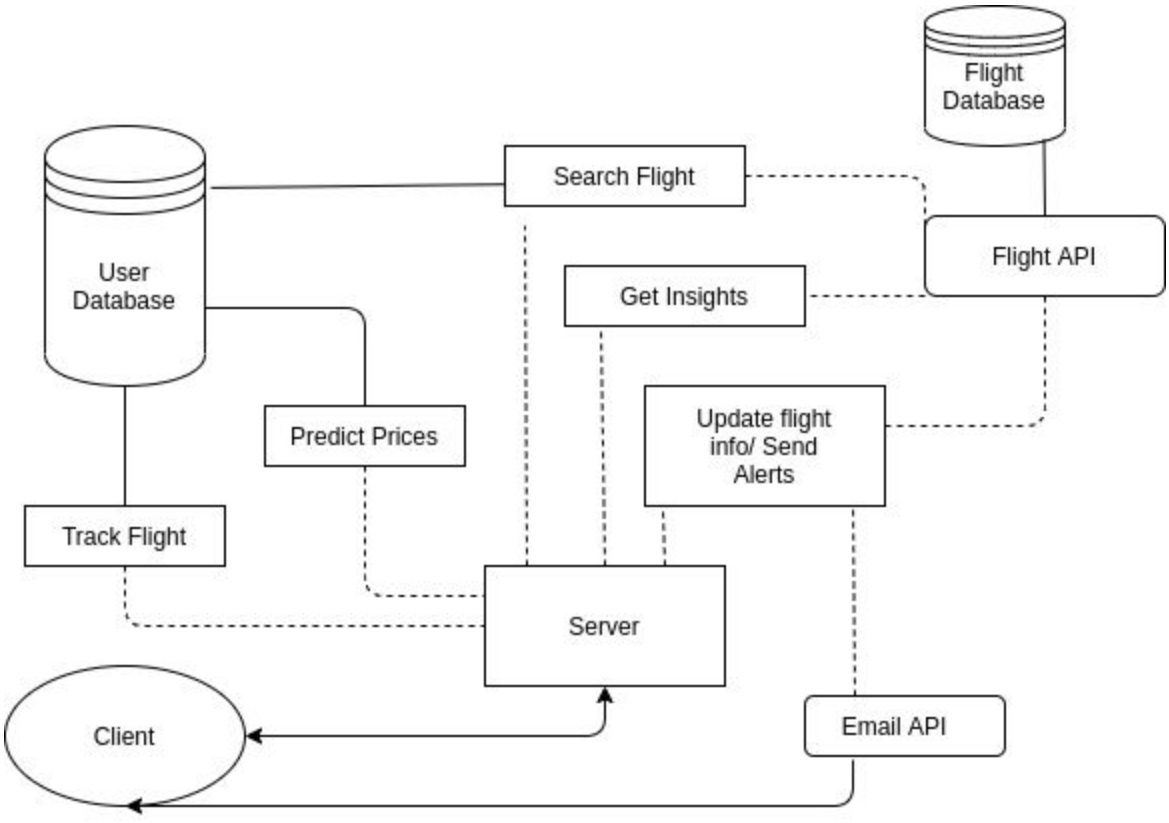
#	Component	Component Type	Description
1	Flight data repositories	Database	This module represents all the databases containing information about flights
2	User data repository	Database	This module is a database containing the user information (email and flights to be tracked)
3	Flight APIs	Database (Websites)	This module represents the different flight APIs which are used.
4	Email API	Application	This module represents the email API which is used to send emails regarding the tracked flights to the users.
5	Master Controller	Processing (interface module)	This is the graphical interface module which interacts with the user on one side and makes appropriate calls to the other modules, discussed below to serve the user requests.

6	Search flight	Processing (API access)	This module interacts with the APIs to get the flights matching the required search criteria
7	Update flight Info	Processing (API access)	Update information of tracked flights once every day.
8	Track Flight	Processing	Store user and his/her tracked flights info in database
9	Get Insights	Processing (API access and computation)	This module accesses the Flight API to get and display the statistics of a flight.
10	Predict prices	Processing (API access and computation)	This module accesses the Flight API to get the history of flight prices and then predicts the expected price accordingly.
11	Send alerts (email)	Processing (Database access and API access)	This module accesses the user database to get the required flight and then accesses the flight API to get the details and sends the details using the email API to the user.

Connectors: Means of interaction between components

Connector	Connector Type	Description
R/W connector	Database access/ modification	This connector is between the user data repository and the module 8 (track flight).
Read only connector	Database access	This connector is between 1. User data repository and the module 11 (send alerts). 2. Flight API and the

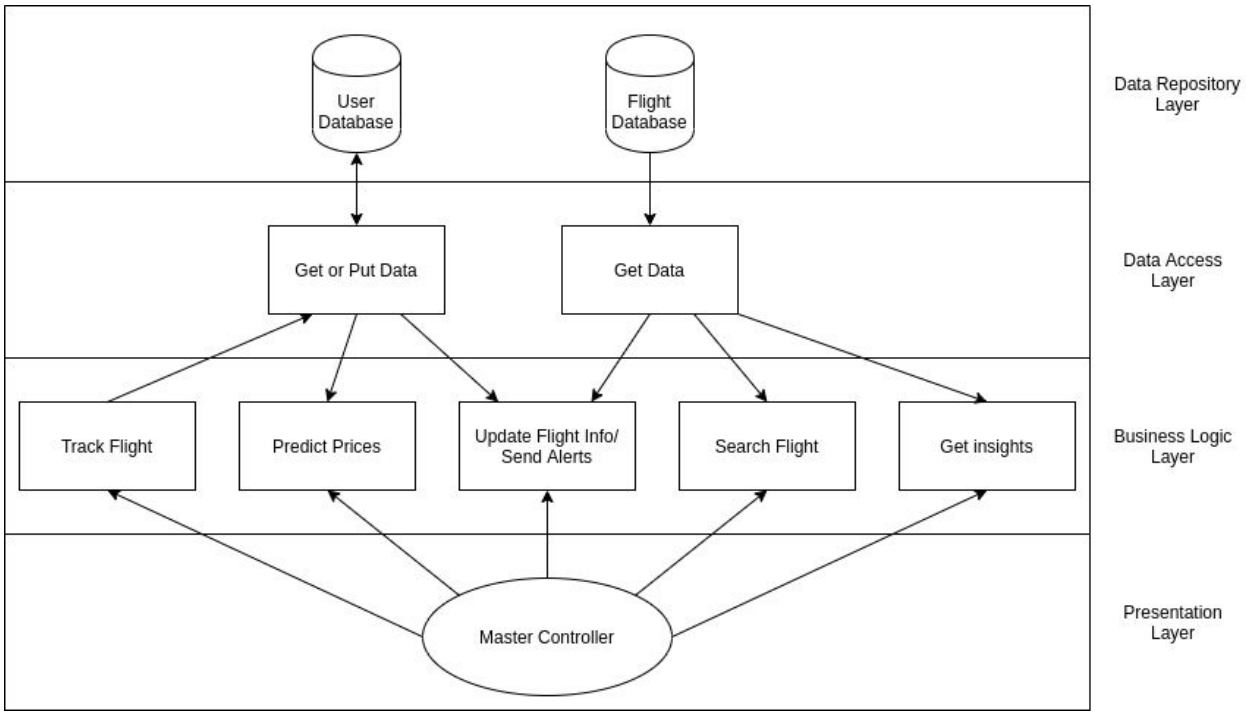
		modules 6, 8, 9 and 10 (search flight, track flight, get insights and predict prices).
URL connector	API request/ API response	These connectors are for making requests (and getting responses) between the server and the modules 3 and 4 (Flight APIs, Email API).
Control Connector	Module invocation or function calls	This connector is between module 5 (master controller) and the modules 6, 7, 8, 9, 10, 11



b. Architecture 2

This architecture is similar to the data repository architecture. The main difference lies in the fact that here, we have a data access layer separating the business logic and data repository. Data retrieval and modification is done via this data access layer, while all the processing of data or implementation of the business logic done in the business logic layer.

The 4th layer is the presentation layer (master control) which is responsible for interacting with the user and invoking modules of the business logic layer. The usefulness of the data access layer comes from the fact that if the type of database is changed then only the access layer needs to be modified while the processing logic remains the same.



c. Comparing the Architectures:

Criteria	Architecture 1	Architecture 2
Access to data restricted or not	No	Yes
Change in Data repositories (Flight and User)	Not easy	Easy
Addition in functionalities	Easy	Easy
Extension to multi-user	Difficult	Easier
Changing data structure used to store the data	Difficult	Easier
Performance	Faster than arch 2	Slower than arch 1

3. Final architecture of TAMFLIP

From the above table we see that architecture 2 is better than architecture 1 in many quality attributes. Both the models maintain independence between their components. It is a little easier to extend to a multi-user system in architecture 1 by keeping a load-balancer in between the client and server. And it is easier to replace the data repositories in the future or if we need to add more databases to the system.

But the performance of Architecture 2 is slightly worse than the first one, because a request has to pass many layers to get served. **And the usage of external APIs for email alerts breaks the hierarchy of Architecture 2.** Since email alerts are an integral part of TAMFLIP we decided to use Architecture 1.