C# Project

GOALS

This project is divided into three main sections of development, and each of them is targeting different level of skills, experience and coding patterns.

1. **Dynamic Link Library (DLL)** *(1h-3h)*

   In the first part, we are going to create a library project. In this library, we will consume a third party (external) free API, which provides information about IP addresses. This library will expose a couple of public classes, through which the rest of the code will communicate with the library and therefore with the API.

2. **WebApi** *(3h-6h)*

   In the second part, we are going to create a WebApi project. This project will use the library we previously created, and expose some functionality to web. For example, it may serve a request for details for a specific IP, use the library to get the necessary information, and send them back to client.

3. **WebApi: Batch Request Job** *(4h-6h)*

   In the last part, we are going to add some extra functionality to WebApi. It needs to support a batch operation for updating IP details. First, it must expose a method where a post request can be made providing an array of items of IP details that should be updated. The caller will get a GUID as response, with which they can call a second method of the WebApi later to get information about the progress of the job. Posted items, will be put into a buffer, where they will be processed in batches of 10 items at a time.

SPECIFICATIONS

Dynamic Link Library (DLL)

A C# .Net library must be created, which will encapsulate all the necessary logic and code for the communication with IPStack.

IPStack is the external service we are going to consume.

The final user of this library, must be able to instantiate a new IIPInfoProvider, and use its GetDetails(ip) method. The library, behind the scenes, will make a request to the api to get information about the requested IP, parse the response, compose the corresponding object, and return it back to the caller.

The library should comply with the following interface:

```
interface IIPInfoProvider
{
    IPDetails GetDetails(string ip);
}

interface IPDetails
{
    string City { get; set; }
    string Country { get; set; }
    string Continent { get; set; }
    double Latitude { get; set; }
    double Longitude { get; set; }
}
```

*The library must handle any possible exceptions that may be thrown during the request to the api, and if any, it must throw a custom exception named “IPServiceNotAvailableException” to the caller.

WebApi

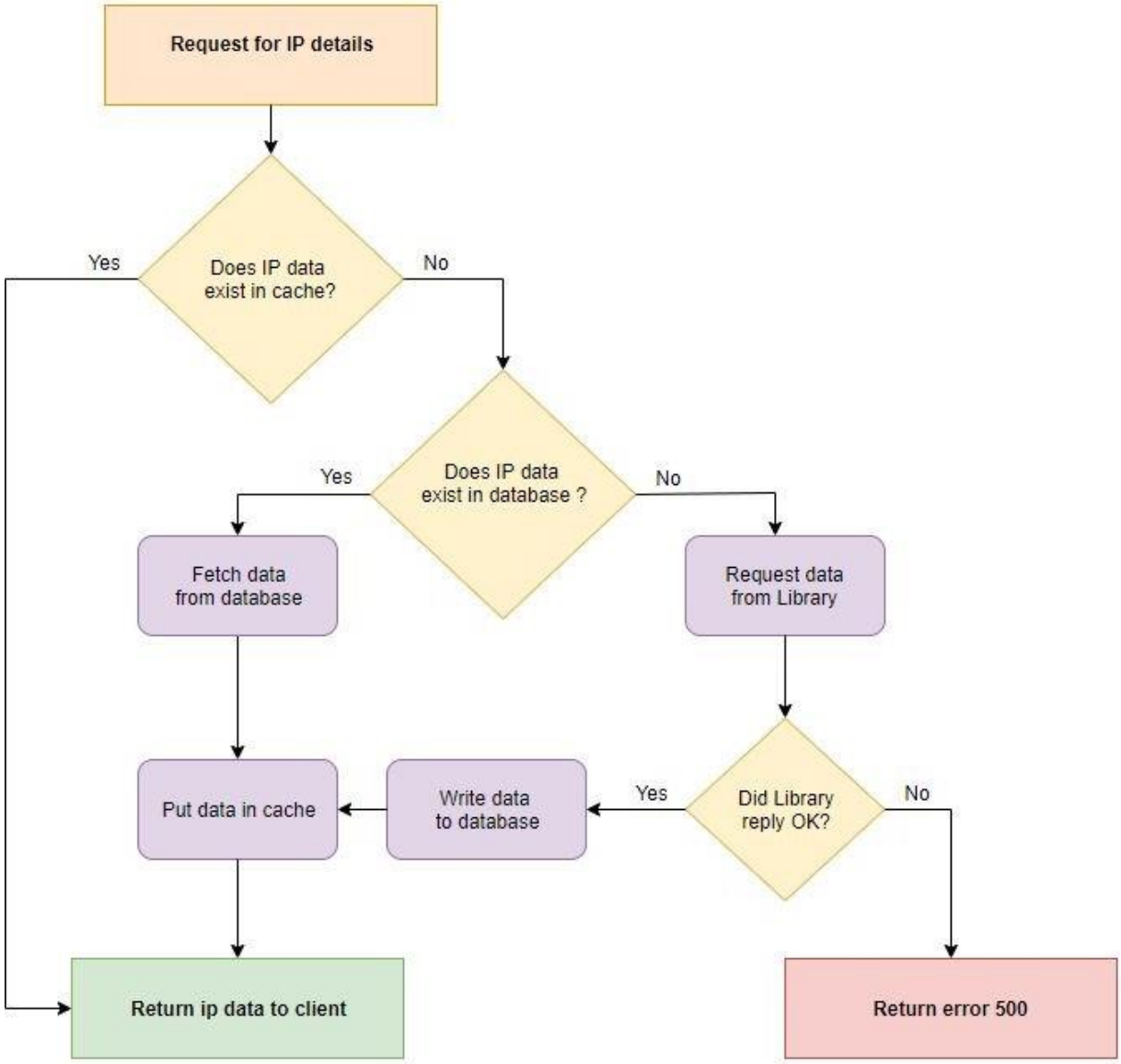
A C# .Net WebApi must be created, which must comply with the following specifications.

To begin with, it must expose an endpoint with a method, where a request can be made to get details for an IP. Internally, the WebApi will use the library previously created to get information about requested IP. In order to avoid repetitive calls for details of an IP, WebApi must implement caching and repository mechanisms, where details of an IP will be stored and retrieved for performance reasons.

Cache: .Net MemoryCache must be used for the implementation of the cache. It may have the IP as the key and its details as value. Each item in the cache must be expired and rejected from cache after one minute from its creation.

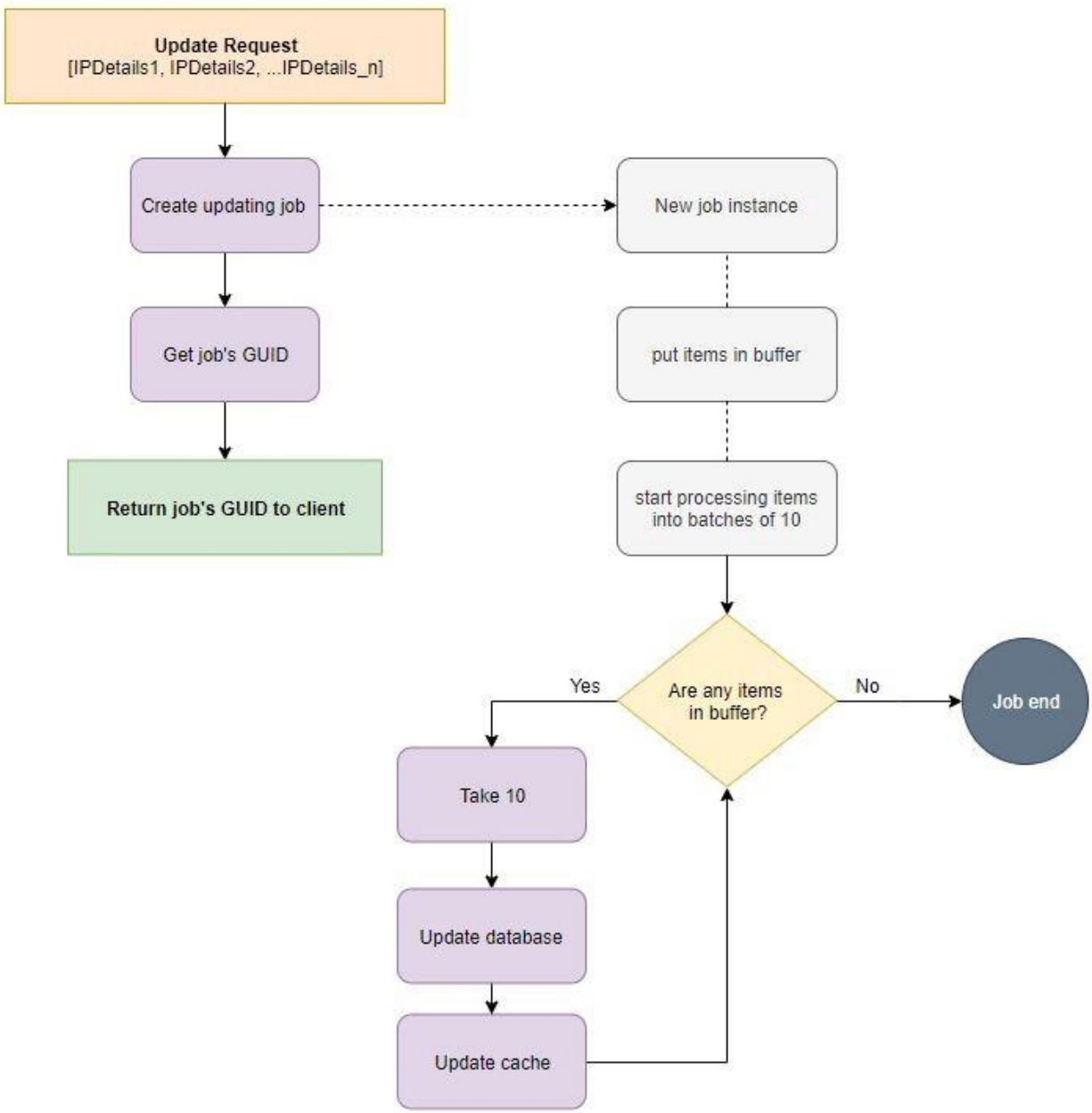
Repository: Entity Framework and Microsoft SQL Server must be used to accomplish the communication with the database, where IP details will be stored.

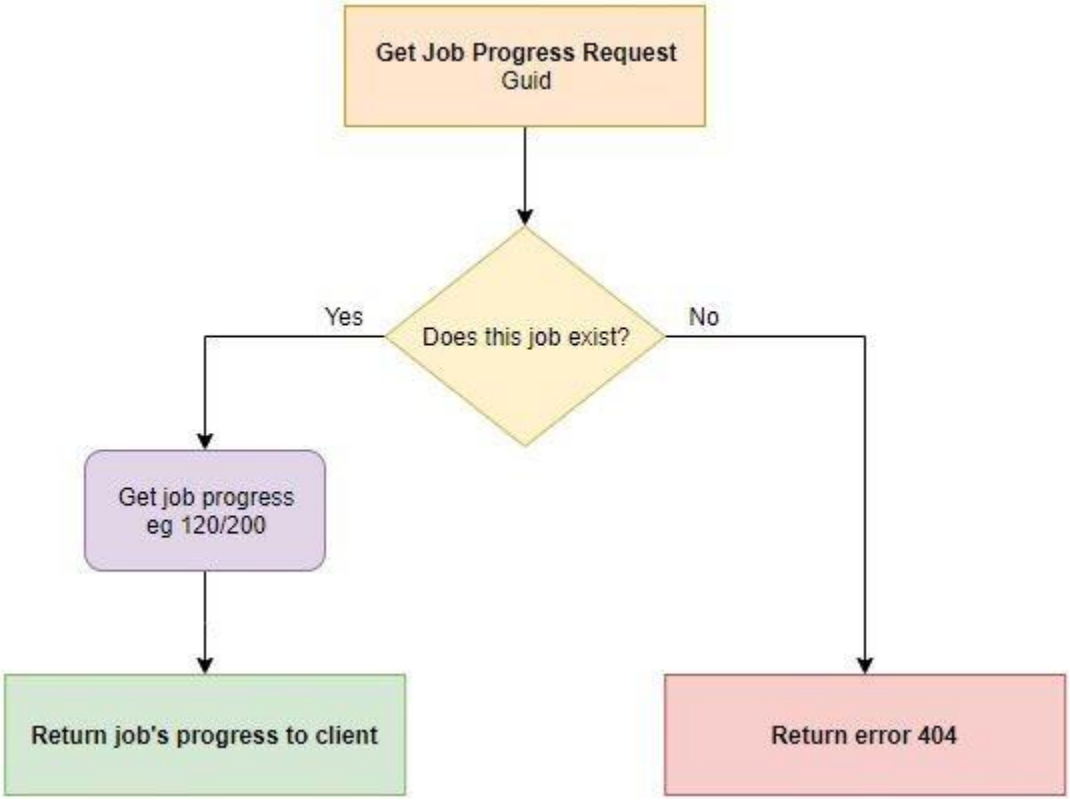
Below is the flowchart of a request for getting IP details:



WebApi: Batch Request Job

Feel free to implement the last task in any way you see fit. We include the following two flowcharts outlining what each of the two API methods (batch update & get progress) is expected to do.




Get Job Progress Request
Guid
Does this job exist?
Yes
No
Get job progress
eg 120/200
Return job's progress to client
Return error 404