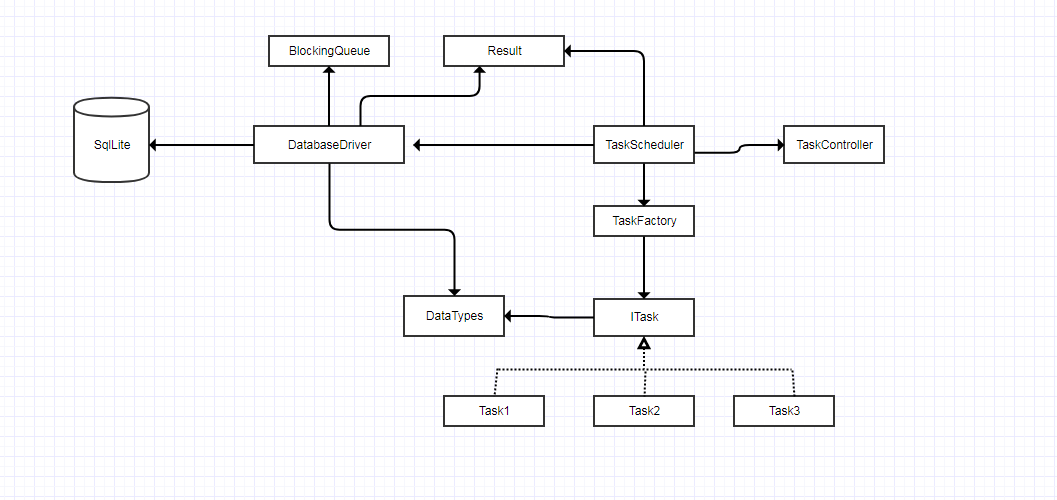
**Task Scheduler**

In this document I have talked in brief about the various packages and the project flow. The system has been developed using visual studio 2015 on windows 10(virtual machine on mac).

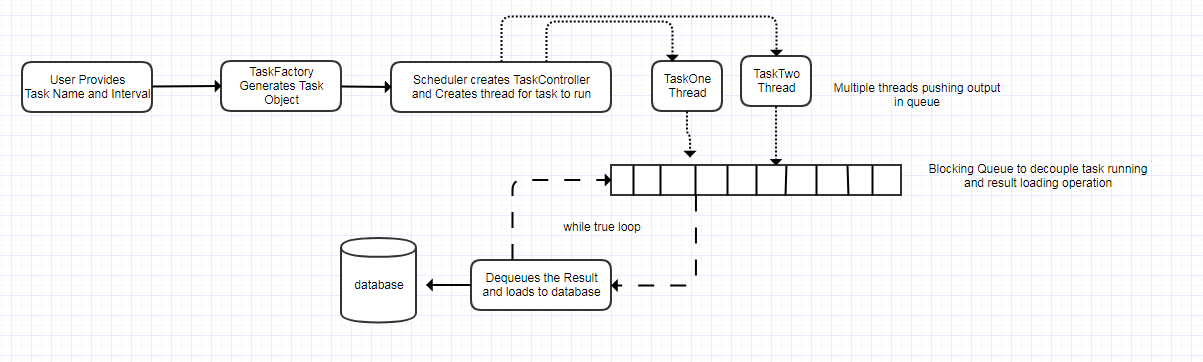
**Partitioning**



1. DataType: In current scenario we have return type as decimal for the task. But in future we can have different data types needed. In order to make or design more scalable to support different return types, I have created this class which for not can handle int, double and string.
2. ITask: This is the interface which has a pure virtual function doTask(), which needs to be implemented by all the task we want to schedule. The other way I could have think of making this work was use of different lambdas as task.
3. TaskFactory: These Class Acts like an object factory for creating objects of concrete task class based on the name provided by the user in schedule task function.
4. TaskController: The reason for creating this class is to keep the track about a task to be run. It helps in monitor and change task interval, delete task and saving previous task run details. This class is used by the TaskScheduler to schedule task.
5. Result: This class is designed to save the results created by running different task at different duration. This class helps in getting data in different format when required example xml or json.
6. BlockingQueue: This is a thread safe queue, which has been created to separate the process of task running and saving results in database. By making these two part decoupled, we are avoiding the condition when we have database connection issues and the thread is trying to connect to database. By this we have threads to just run task and threads to just save data in database.
7. Driver: This class is basically to load the data result dequeued from result queue into the database. For now I have started one thread, depending upon system we can have more threads for data load operation.
8. TaskScheduler: This is the main class of the whole system as it will be the center point for scheduling a task run and will also take care of loading the data into the database with the help of other classes.

\*\* For this project I have created the datatypes class which supports various data types. But in the project I have done processing for only decimal as this is just a prototype.

**Project Flow**



The flow of the project starts with user selecting a particular task and duration for it and giving it to the task scheduler which gets objects from task factory of those task. The task scheduler creates the object of task controller which helps in controlling the lifetime of the task given by the user and also creates a thread for the task execution inside a while true loop. Task scheduler provides a task id to the user which he can use to future make changes in interval as well as the cancel the task.

Once the task has executed it will give the result to the task scheduler which will put it in the blocking queue of the database driver. Results received on execution of the task are stores in for Result class which will come handy in case in future we want data in some other format example xml or json. Once the result has been enqueued into the blocking queue, job of scheduler is done in terms of the task it ran. Now the database driver will need to take the data and create insert statement to load data into the database. As the time was short, I gave more thought on design as installing sqlLite and connecting it with C++ would have taken time. So I have left the data with driver in Result object format as well as json format.