Java Home Project

1. Following is a Java project, to be develop independently.
2. The completed project should be completed and delivered within 5 days.
3. For any issue or clarification, feel free to ask and send mail to

**Tomer.Winer@rbbn.com**

1. The completed project should include the following:
   1. Detailed design Word document.
   2. Detailed user guide Word document for installing & running the project.
   3. All compiled classes \ jars etc.
   4. All Java source code
2. Develop the following:
   1. Simple standalone HTTP client server ( **See Hint #2 and #3** *for uploading server with Vertx in rxjava configuration* ) of UI for getting the following inputs:
      1. Number of scheduled tasks to run in parallel (between 1 to 20 tasks)
      2. Time to run tasks (within number of seconds between 1 to 20 seconds)
      3. Prefix input
   2. BE server performing the following:
      1. Create at least 20 CSV files in folder named: In, containing at least 250 lines of names. Each line will have at least 10 names.
      2. Reading **in parallel** ( **See Hint #2** ) the CSV files from above In folder, **line by line**, scheduled to be performed according to client input (Section a.)
      3. Writing **in parallel** above CSV files, line by line, to folder named: out.
      4. Once each file was completely read and written successfully, the file will be deleted from the In folder.
      5. Once the In folder is empty - Execute one new service, doing the following:
         1. Reading all the files lines into **cached data** ( **See Hint #1** )
         2. After you completed to read all – Go over all your cached data and write to final result log file, called autoCompleteSuggestion.log - All the relevant suggestions for a given prefix (Section 5.a.iii) e.g.:

If the names from the files where To,Avi,Tom,Avner,Amin,Aviv,Yoni,Tomato, Yoyo

Then the result for To should be:

To->To,Tom,Tomato

Then the result for A should be:

A->Avi,Avner,Amin,Aviv

Then the result for Av should be:

Av->Avi,Avner,Aviv

Then the result for Avi should be:

Avi->Avi,Aviv

.

.

.

1. Following, hints:
   1. Hint #1: Consider your **best efficient** **data structure** for holding & manipulating your **cached data** (Section 5.b.v.)
   2. Hint #2: Use vertx-rx jar (**Vertx**) for reading stream for each data from files.
   3. Hint #3: Use rxjava jar (The **Single** ) for HTTP Server subscription.