Job Management

Proto type

TABLE OF CONTENTS

Document Revision History			
Revision #	***Date***	***Author***	***Brief Revision Description***
1	2/06/2019	Sachin katarnaware	Initial Draft
2			

I. Functional Area

1. **GUI :**

   There will be one gui screen to schedule the job. This screen is developed using decorator design patter so that we enhance the design in future.The fields inside the Screen include.

   - Complete Class Path
   - Upload Job (button to upload file to execute)
   - Calender to select the schedule date and time.
   - Repeat Check Box(to execute the repeatedly after 24h interval)
   - Register (button to execute job on scheduled time)
   - Execute Now(button to excecute the job now)
   - Check Status (Text box which takes job Name)
   - Check Status (button to check the status of the job)
   - List of Queued Jobs
   - List of Success Jobs
   - List of Failed jobs
   - List of Running jobs

2. **Validation of the fields for the new Screen:**
   - Class Path – mandatory
   - Upload Job (mandatory to upload)

3. **Service**

   This is responsible for scheduling the jobs passed or uploaded to system.This will contain the Wrapper,Management and service layer. ()

4. **Service API**
   - This job implement the service api so that job management system can handle it.

II. Specifications

Functional Design Reference	Functional_Design_Job_Scheduling.doc
Wireframe Reference	
Technical Design Name	Functional_Design_Job_Scheduling.doc
Short Description	The purpose of is to develop the prototype for the job management system. The system will unaware of the type of the job, it will just execute the job at scheduled time.
Type (Required)	J2SE
Category (Required)	*Submission*

III. Modules

1. **UI**
2. **Service**
3. **Jobs**

IV. Design Overview

1. **Screen for Job Registration or Scheduling**.
   - **Config**
     - **Registration_cofig.config**
       - Entry to upload the job.
       - Entry to load the job.
   - **Java**
     - **ScheduleJobFormImpl.java**
       - Complete Class Path
       - Upload Job (button to upload file to execute)
       - Calender to select the schedule date and time.
       - Repeat Check Box(to execute the repeatedly after 24h interval)
       - Register (button to execute job on scheduled time)
       - Execute Now(button to excecute the job now)

- **CheckJobStatusFormImpl.java**
  - List of Queued Jobs
  - List of Success Jobs
  - List of Failed jobs
  - List of Running jobsList of Queued Jobs
  - List of Success Jobs
  - List of Failed jobs
  - List of Running jobs

2. Core Service

- **JobManagementServiceImpl.java**
  - Responsible for handing the jobs and calling the service.

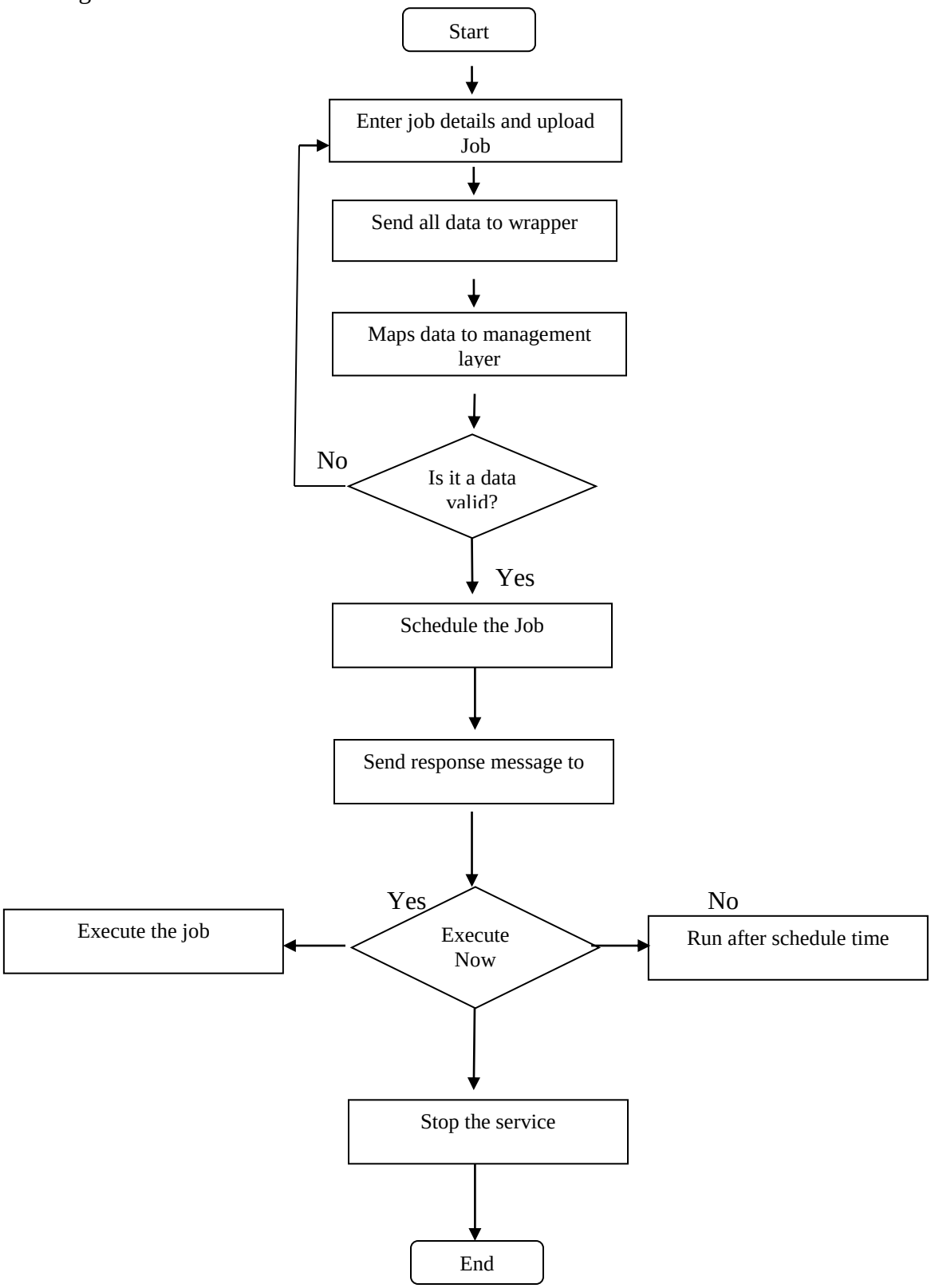
- **SchedulerServiceImpl.java**
  - Responsible for scheduling the jobs and executing it on scheduled time**.**

3. Job Interface

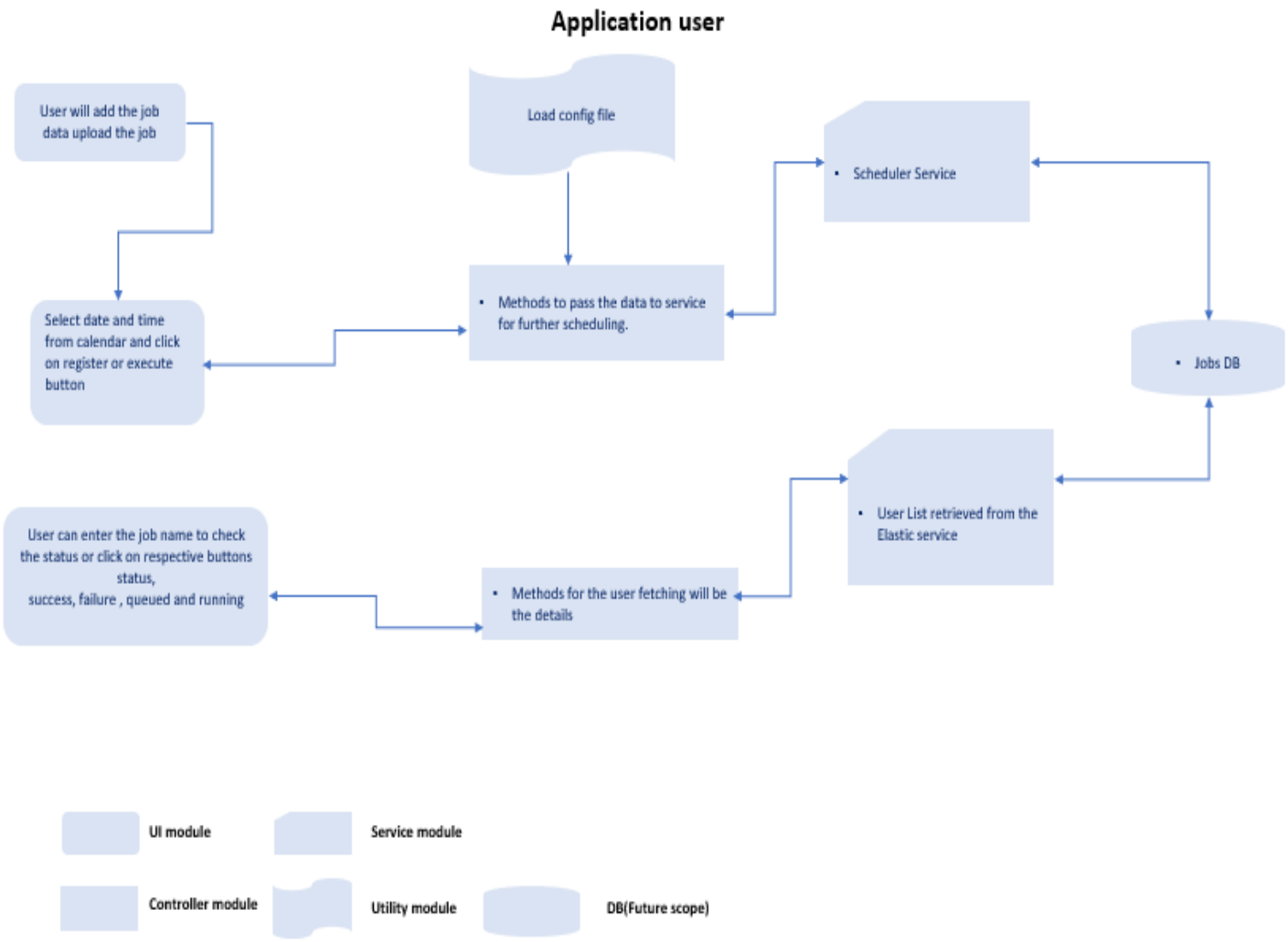
- **Job.java**
  - This interface is exposed to client which will implement the method**.**

V. Program Flow

High level diagram



VI. Design Diagram



VII. Model Objects

1. JobInformation
2. JobStatus

7) Required Grants and Synonyms

VIII. Design Assumptions

Reference #	*Description*
1	It will be an desktop application to schedule jobs in an organization.
2	In an organization different departments will implement the job management system interface.
3	The department or client will not know anyting it will just import the jar and implement it.
4	The client will send us the implement jar will upload it into the system to execute it On scheduled time.
5	Jobs will be in the form of jars.(In future scope we can allow to add zip folder)
6	The user will upload the jar into the system with the configuration file if needed.
7	Types of jobs will be like sending mails, auditing or archiving last 6 months data etc
8	We are not using any database for now(In future it may needed to audit the jobs)

IX. Outstanding Issues Future scope

- **Cancellation** – User often want to kill a long running job, or prevent one from running.
- **Priority** - User often want high priority jobs to run in preference to low priority jobs. But implementing this in a way that low priority jobs don't wait forever in system where lots of jobs are generated is "non-trivial"
- **Resources** - some jobs may only be schedulable on systems which have certain resources. E.g. some will require large amounts of memory, or fast local disk, or fast network access. Allocating these efficiently is tricky.
- **Dependencies** - some jobs may only be runable once other jobs have completed, and thus can not be scheduled before a given time.
- **Deadlines** - some jobs need to be completed by a given time. (or at least be started by a given time.)
- **Permissions** - some users may only be able to submit jobs to certain resource groups, or with certain properties, or a certain number of jobs, etc.
- **Quotas** - some systems give users a specified amount of system time, and running a job subtracts from that. This could make a significant difference to the numbers in your example.
- **Suspension** - some systems allow jobs to be check-pointed and suspended and the resumed later.
  :

X. Appendix

XI. Sign-off