**3musketeers**

# SplitAndGo Design

## Revision History

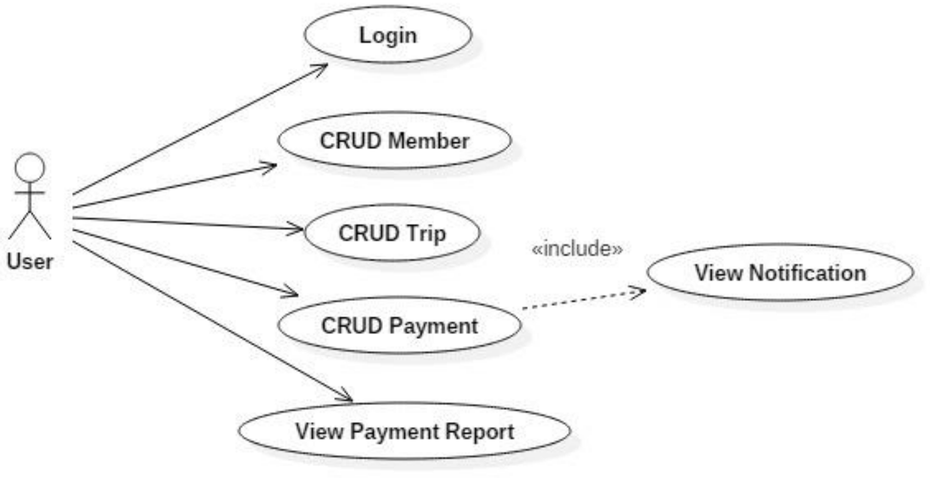
| Date | Issue | Description | Author |
|---|---|---|---|
| 25-May-17 | 0.1 | Add Architectural High Level Design<br>Add Use case, Sequence Diagram for Notification message | |
| 26-May-17 | 0.3 | Add Logical View of Notification<br>Add Sequence Diagram of Notification | |
| 26-May-17 | 0.5 | Add Use Case model and MVC Service Layer Diagrams | |
| 27-May-17 | 0.6 | Add Diagram for Daily Batch Job and issues | |
| 27-May-17 | 0.7 | Add Login and Payment Sequence Diagrams, integration issue between MVC and REST | |
| 27-May-17 | 0.8 | Add Use Case, sequence diagram and class diagram for REST services | |
| 27-May-17 | 1.0 | Correct Figure # , future consideration and baseline document | |

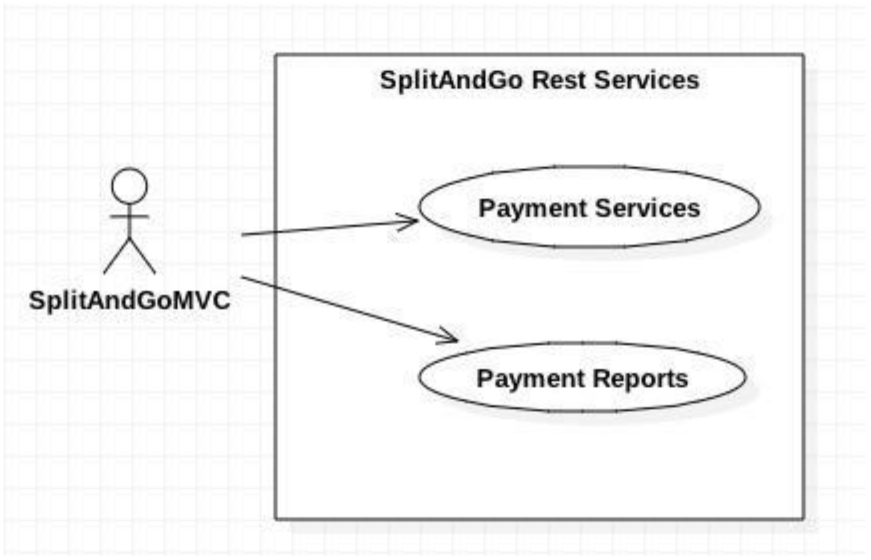
## Table of Contents

## 1. Introduction

SplitAndGo is a project that allow friends and travel mates to manage the expenses for a trip.

### 2. Requirements - Use-Case – Usage Scenarios





### 2.1 Login

In order to use the system, user must log into the system with a valid username and password.

### 2.2 CRUD Member

Only user with ROLE_ADMIN role can create, read, update and delete members of the system.

### 2.3 CRUD Trip

Only user with ROLE_ADMIN role can create, update and delete trips of the system. A fund and

members are added to the trip during this use case.
User with other roles can view the trips.

### 2.4 CRUD Payment

Only user with ROLE_ADMIN role can create, update and delete payments for a selected trip of the system.
User with other roles can only view the payments.

#### 2.4.1 View Notification

Whenever a new payment of a trip is created or an existing payment of a trip is updated, all members of that trip will receive a notification telling them about that payment (which trip, who does what action) when they log into the system.

### 2.5 View Payment Report

A login user can view the list of payment reports, then he is able to download a report for more detail which includes payments made during a trip.
System reachs out to Payment Rest service to get the reports from the Report storage which are generated by Batch job

#### 2.5.1 Batch Job

Asynchronous system to generate payment reports daily or immediately.

#### 2.5.2 Payment Services

Rest services (resided in REST server)  provided for all CRUD operations on Payment. It take the request from the SplitAndGoMVC and perform the real DB operations

#### 2.5.3 Payment Reports

A Rest service provided to fetch the Report from report storage and send back to SplitAndGoMVC. The Reports are generated from the Batch job which are located at the same server with the Rest

## 3. High Level Design

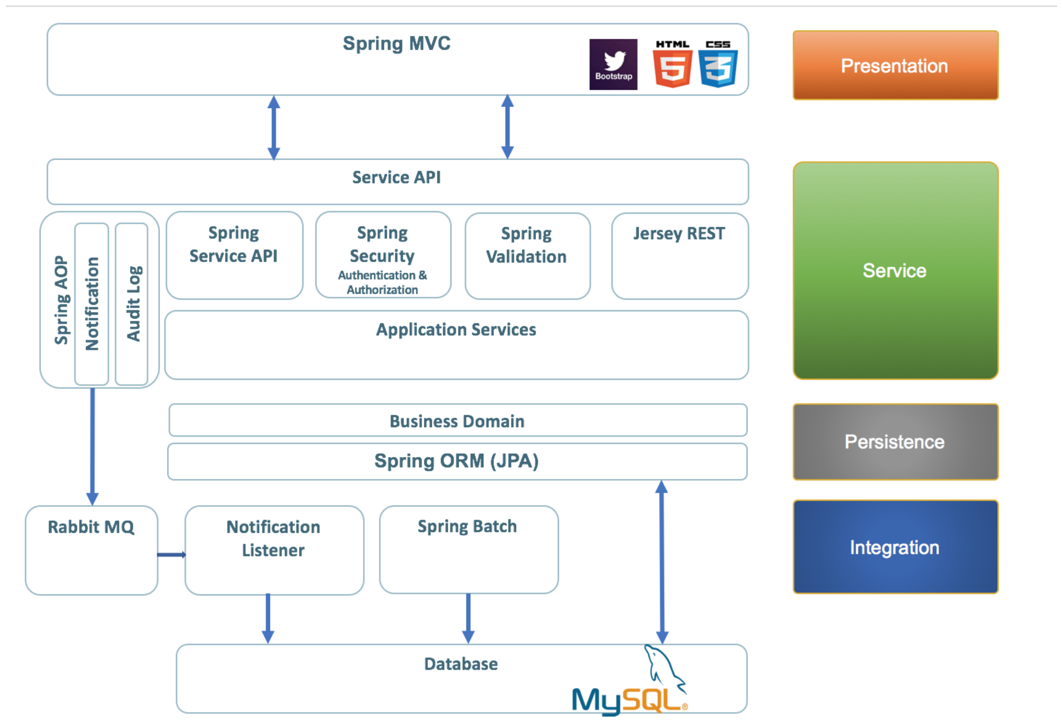


Figure 1 < High Level Design Diagram >

## 4. Detailed Design

## 4.1 Class Diagram

### 4.1.1 Web-Based application

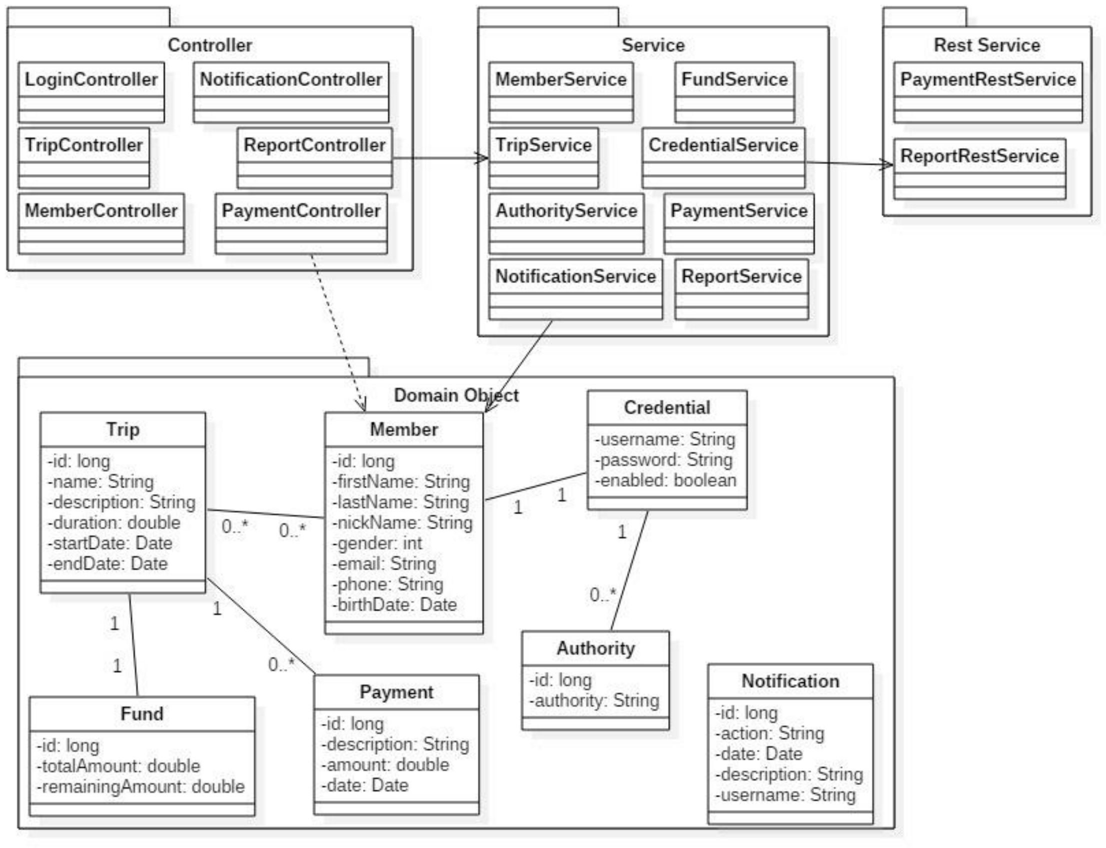


**Figure 2 < SplitAndGo MVC Class Diagram >**

**The client request first goes to Controller, and the sent data will be validated based on Domain Object. If data is valid, it's fowarded to the Service to handle the business logic.**

**Two special services of Payment and Report that they will be forwarded to the external Rest Service for handling the business logic.**

**Basic methods of Controller where (*) is Trip/Member/Payment:**

- **public String get*(Model model)**
- **public String add*(*)**
- **public String edit*(*)**
- **public String delete(Long id)**

**Basic methods of Service where (*) is Trip/Member/Payment:**

- **public void save(*)**
- **public * update(*)**
- **public List<*> findAll()**
- **public * findOne(Long id)**
- **public void delete(Long id)**

### 4.1.2 Notification

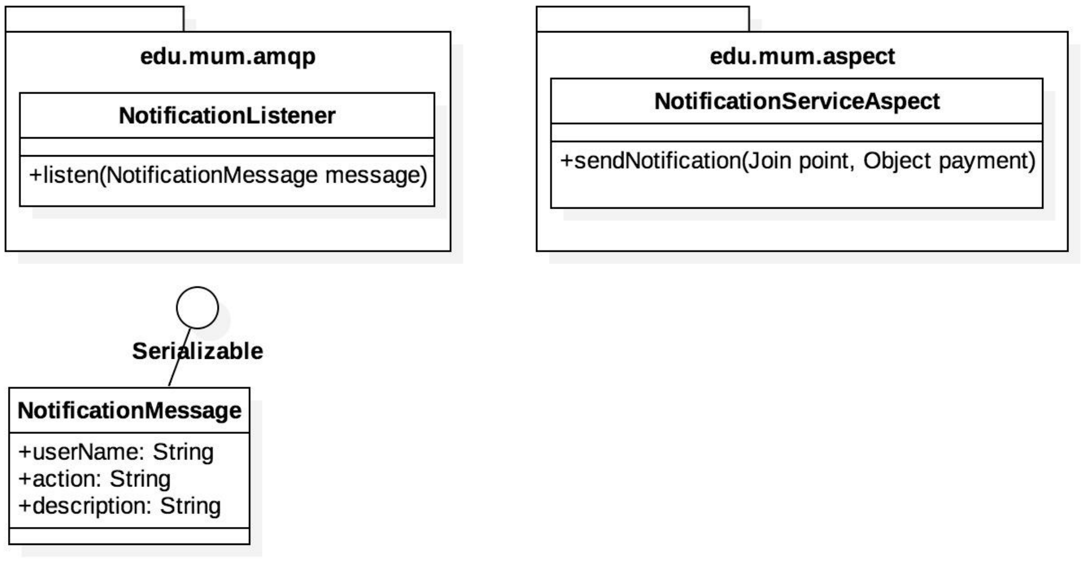


**Figure 3** **< Notification Class Diagram >**

- All Payments services are marked with @Notification annotation that will trigger **sendNotification()** method of **NotificationServiceAspect instance**

**@Notification**
public void save(PaymentDto payment) {}

**@Notification**
public PaymentDto update(PaymentDto payment) {}

### 4.1.3 Daily Batch Job

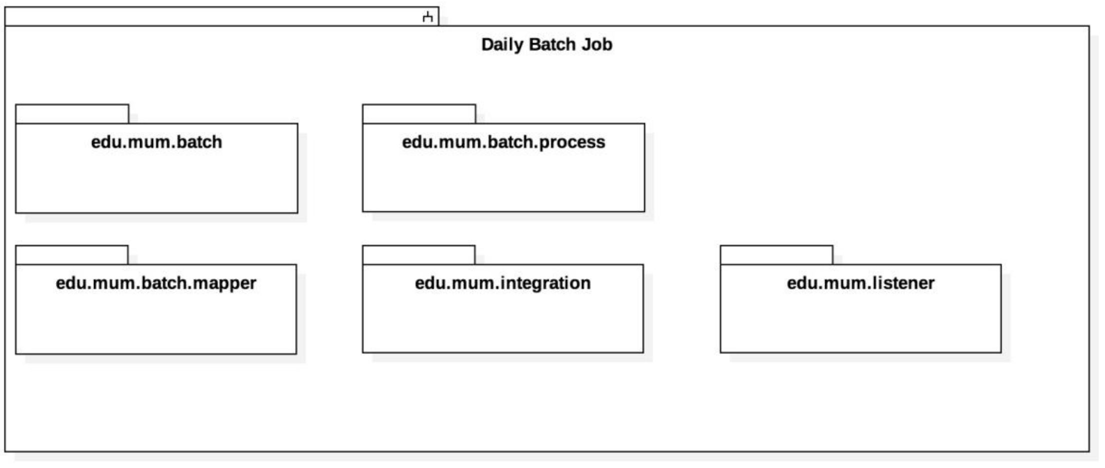


**Figure 4** < **Batch Job Package Diagrams** >

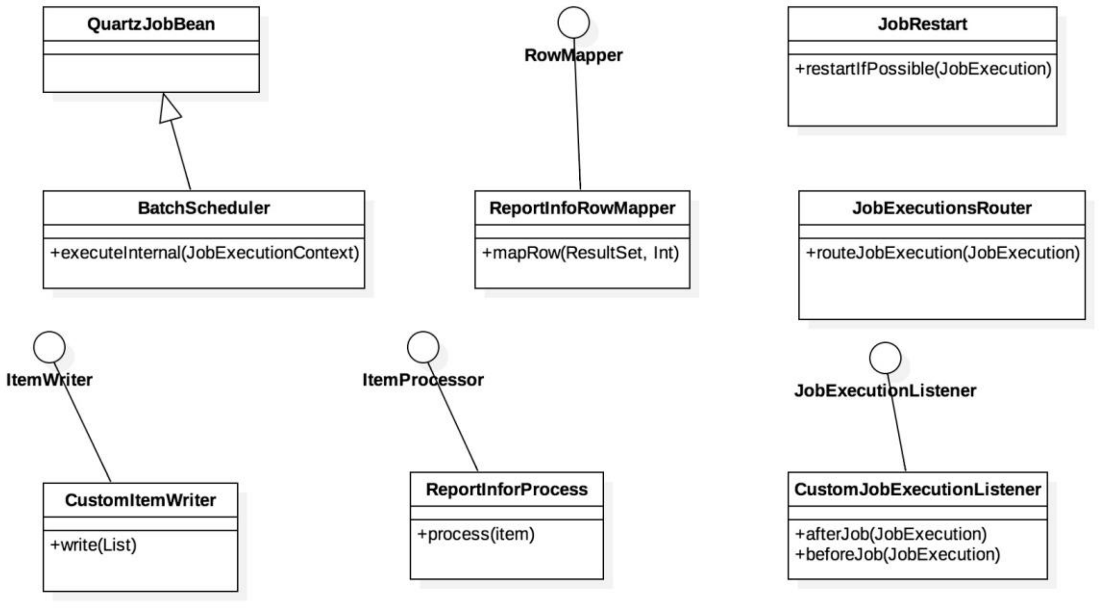


**Figure 5** < **Batch Job Class Diagrams** >

- **ReportInfoRowMapper** will map data fetched from database into report object - **TripPayment**
- **BatchScheduler** instance will run as background thread and invoke job at cycling time point.
- **JobRestart** will be invoked by Service Activator to restart Job
- **JobExecutionRouter** is message endpoint that will base on Job Execution status to route message to Job restart or email service
- **CustomItemWriter** generate report
- **ReportInforProcess** is used as filter item that is eligible in report or not
- **CustomJobExecutionListener** is gateway to forward message to JobExecutionRouter

### 4.1.4 REST Services

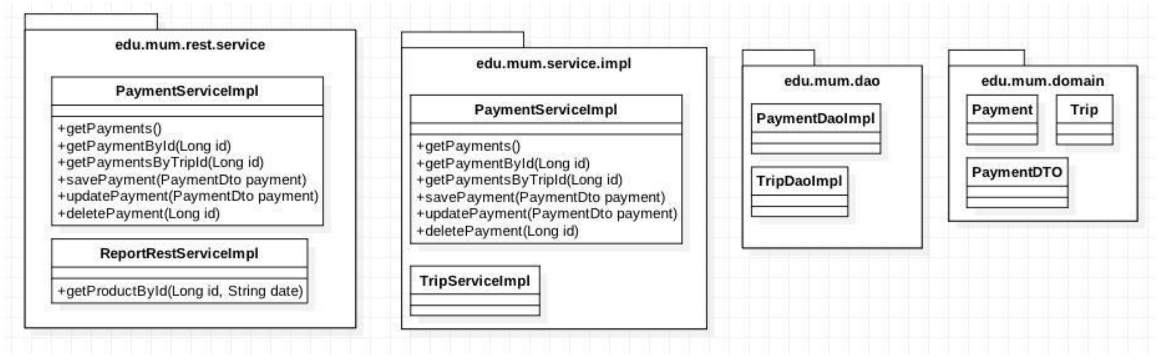


**Figure 5** < REST Sercices Class Diagrams >

## 4.2 Interaction Diagram

For the web-based application, I list out here important sequence diagrams only, others are quite similar.

### 4.2.1 Login

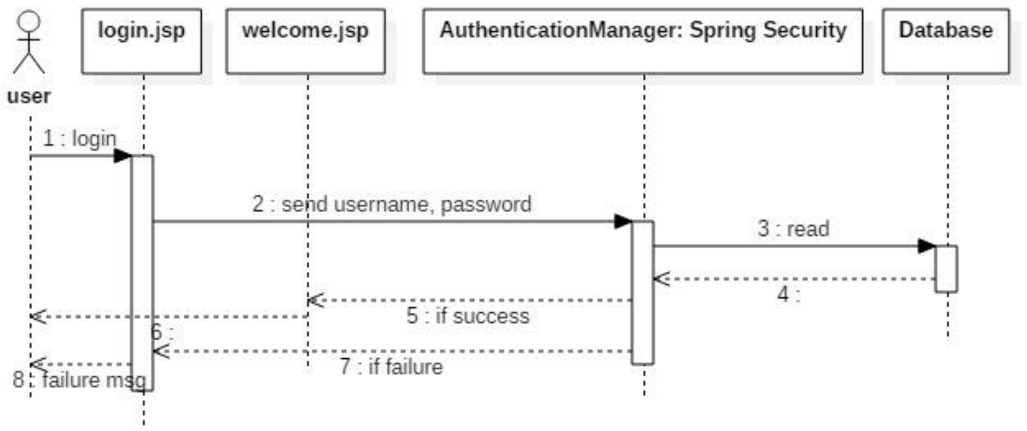


**Figure 6** < Login Sequence Diagram >

### 4.2.2 Add Payment

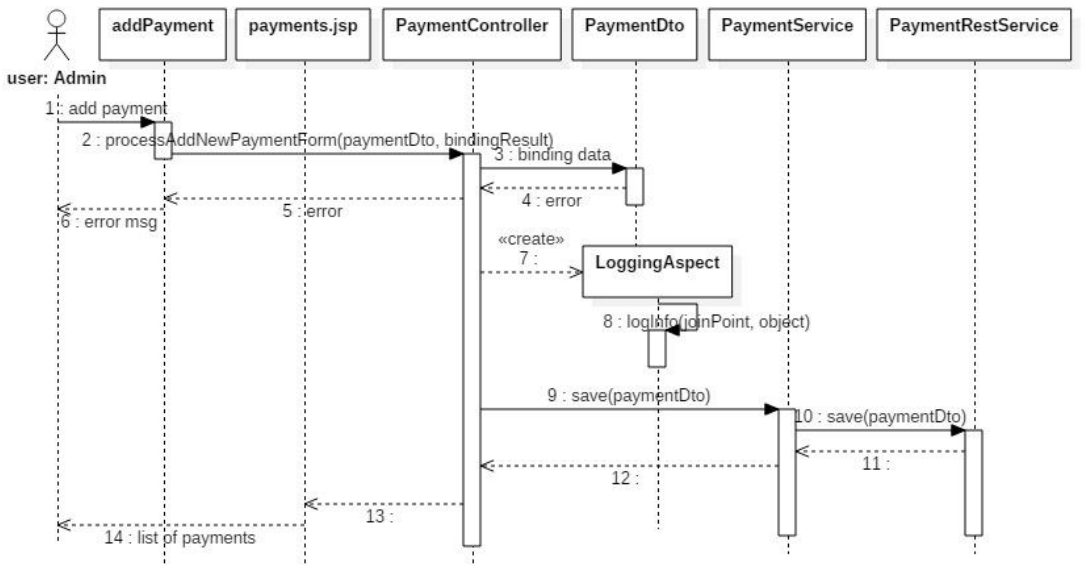


Figure 7 < Add Payment Sequence Diagram >

### 4.2.3 REST Payment services

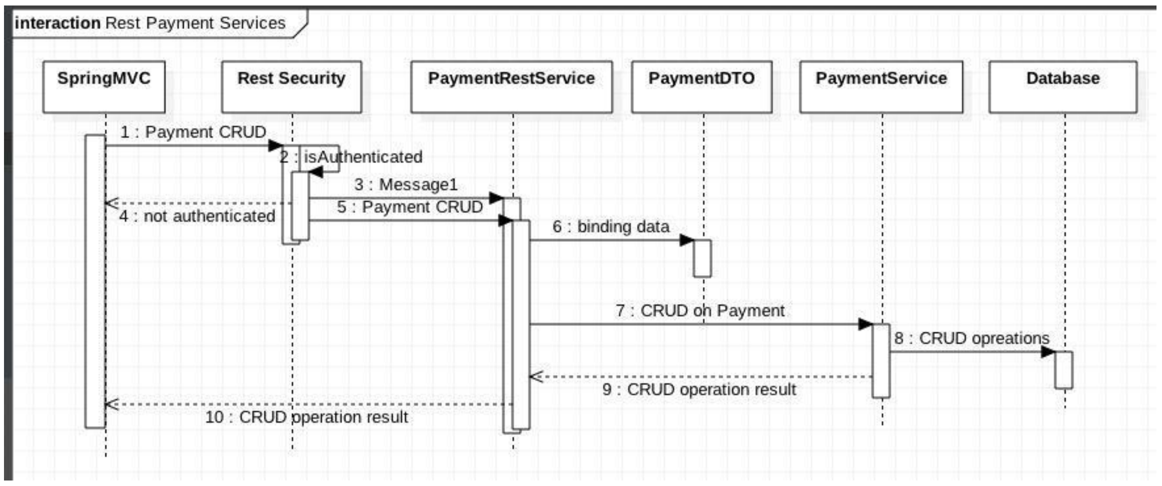


Figure 8 < REST Payment service Sequence Diagram >

### 4.2.4 REST Report service

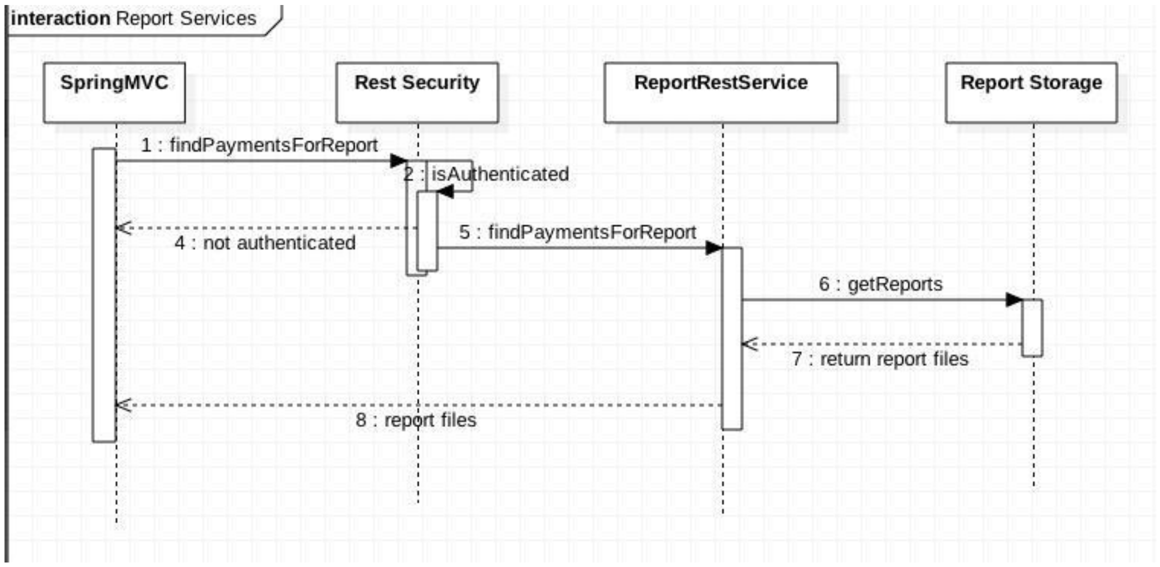


**Figure 9** **< REST Report service Sequence Diagram >**

### 4.2.5 Notification Message

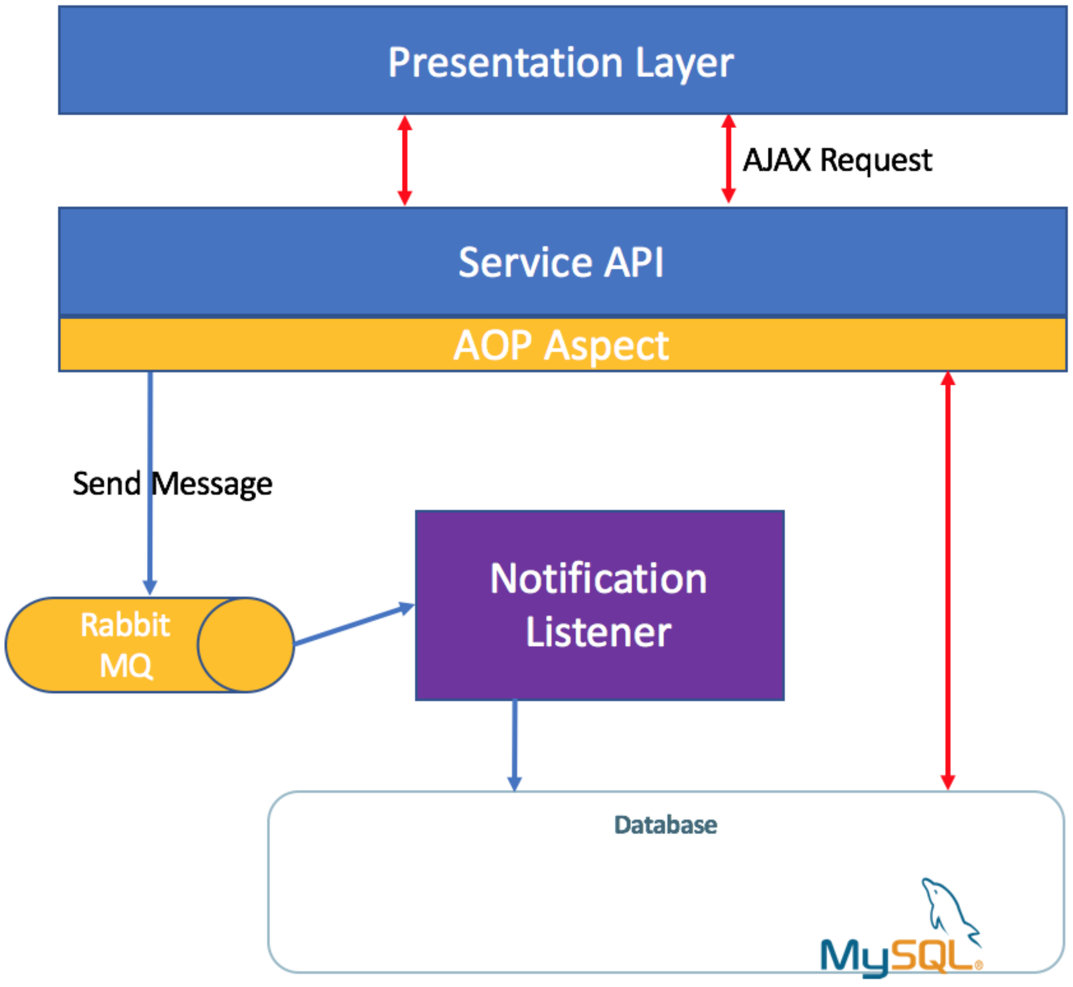


Figure 10 < Notification Message Logical View >

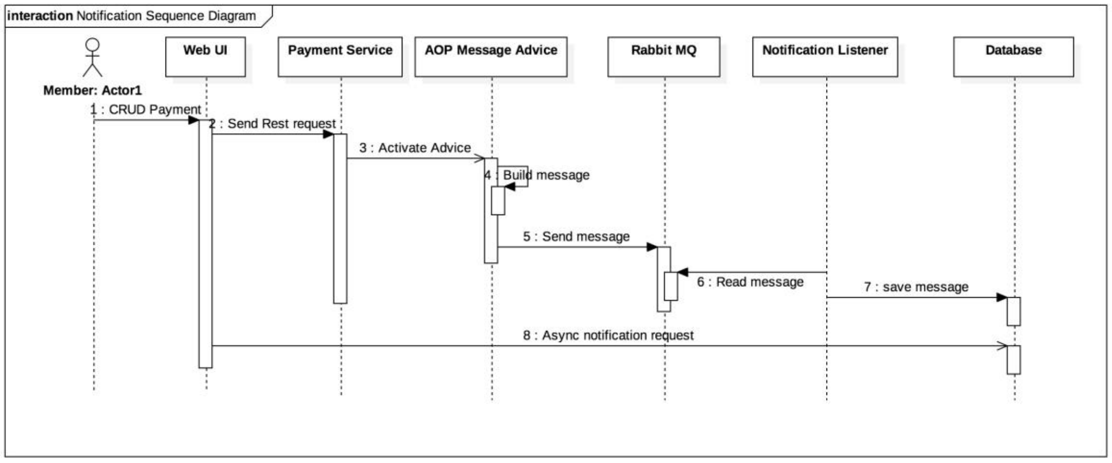


Figure 11 < Notification Message Sequence Diagram >

### 4.2.6 Daily Batch Job

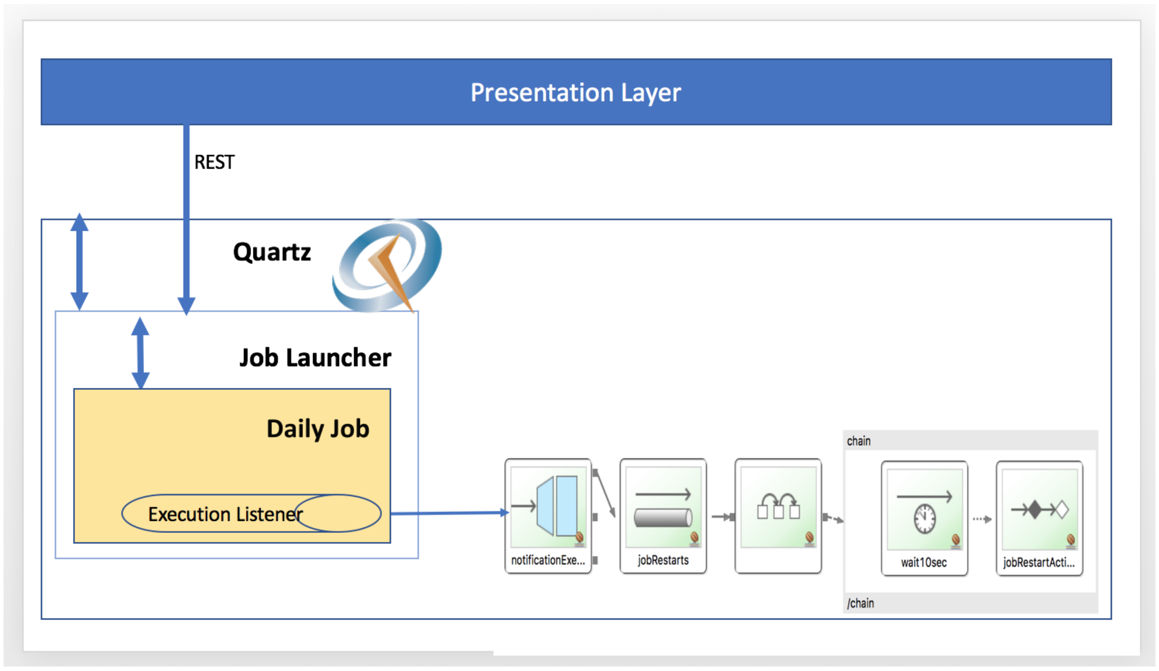


Figure 12 < Batch Job Logical View >

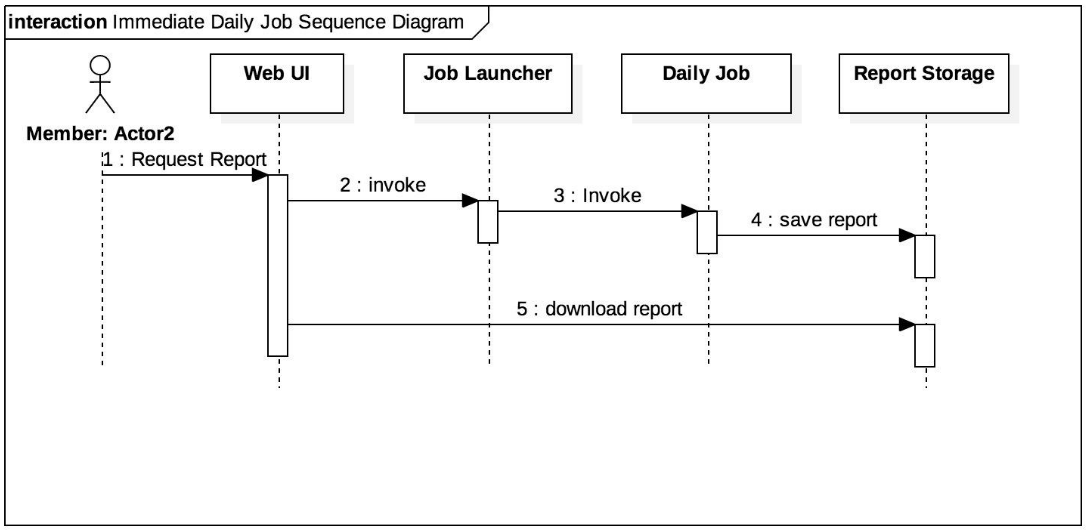


**Figure 13** < Immediate Batch Job Sequence Diagram >

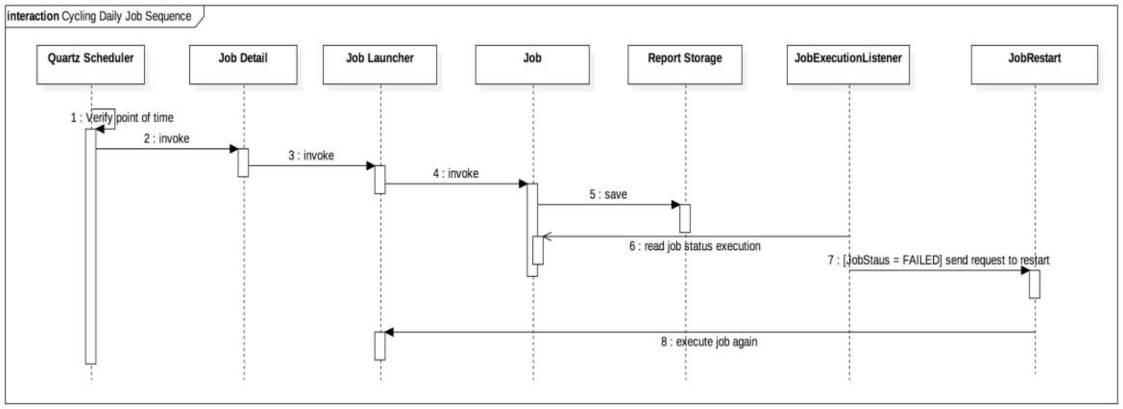


**Figure 14** < Cycling Batch Job Sequence Diagram >

## 5. Design Alternatives

Batch Job interacts with other third party payment gateway such as bank, paypal …. However it is not implemented due the time constraint and complexity of integration and testing.

## 6. Issues, Risk and Dependencies

Issues:

- We faced an issue when doing integration between MVC and REST when we try to return the whole domain object Payment to MVC from REST, it ran into StackOverflow exception due to the Bidirection relationship of Payment object and Trip/Fund object.
  Solution: define a payment DTO where only necessary data are included and transfer it back and forth between MVC and REST

- Batch Job Issues

1. Batch and Spring Integration is FAILED because the JobExecutionListener always sending message to router even JOB has not executed yet and getting loop in cycle. Currently, we disable this integration by comment out configuration in file **daily-job.xml.**

```
<job id="dailyJob" >
            <step id="step1">
            <tasklet>
                        <chunk reader="itemReader" processor="reportProcessor"
                        writer="customWriter"  commit-interval="1" />
            </tasklet>
</step>
            <!--
                        <listeners>
                                    <listener ref="notificationExecutionsListener" />
                        </listeners>
            -->
</job>
```

2. Both ItemReader and ItemWriter are only process one item at one time although the input of writer is list but the list contains only one item**.** This can cost connection to database. The idea is how possible to fetch a batch records from database instead of one by one. **Solution** is both ItemReader and ItemWriter need to implement ItemStreamReader/Writer.
3. Another issue related report is missing data. For example, there are two record of payment but the writer is writing only the first row. **Solution** is still looking
4. The reader works correct with current date only. if enter past/future , batch will not triggered. Solution is preventing user enter invalid date from UI

## 7. Future Considerations

Move all services to Jersey REST so that it can be integrated with different clients such as browser, mobile device, …

Appy EHCAche to catch request/response if invoke many times with the same parameter

## 8. References

http://docs.spring.io/spring-batch/reference/html/whatsNew.html