Lappeenranta University of Technology

School of Business and Management

Degree Program in Computer Science

LTC-Otso Code Camp Winter 2015

Project Report

The Propeller Hat team

Student: Hieu Tran
0460261

Student: Kurosh Farsimadan
0460148

Student: Muninder Adavelli
0460287

Student: Lakshmi Prasanna Kuchimanchi
0433913

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

API(s)	Application programming interface(s)
JSON	JavaScript object notation
REST	Representational state transfer
SSH	Secure shell
UI(s)	User interfaces

1 INTRODUCTION

The code camps are organized by Lappeenranta University of Technology to provide students with an opportunity to collaborate and solve business problems in one week. From the code camps, students gain more hands-on coding experience, skills of project management and team works. Participating in the winter 2015 code camp, our team - the Propeller Hat, addressed the challenge provided by company Ltc-Otso. The challenge was to solve a business case to provide a business services for insuring clients with various business partners or sub-contractors. Such service helps insurance companies to manage the risks of insured partners based on the feedback and risk data.

2 CASE AND SOLUTION

The business case demanded an innovative service for insurance company to evaluate the risks of the insured companies according to their performance and risk data. In order to do that, the customer's feedbacks together with their task's performance are needed for the risk evaluation. Therefore, our team analyzed carefully the case and defined all stakeholders of the business, which will impact on the system. Such stakeholders are the manufacturer (producing the product), sub-contractors (having contracts with the manufacture to make certain components), and customers (who are end-users using the products). Subsequently, we started to build an application prototype to express our findings and provide an initial view of the system.

We decided to assume that the manufacturer is a mobile phone creator developing various kinds of mobile phones. The company supposes to deal with ten different sub-contractors who help create different phone components. These subcontractors are responsible for delivering high-quality spare parts according to the manufacture's requirements without spoiling the reputation. To solve the situation, we built an application with three different views for the manufacturer, sub-contractors and the customers. We aimed at create a system for multiple devices (phones, tablets, computers) and platforms (iOS, Android, Windows Phone).

- Manufacturer's view: the manufacturer can use the application to create new assignment and assign it to appropriate sub-contractor. They are also able to track the existing assignments for the progress and rating. Last but not least, due to the needs for customer feedbacks, the manufacturer can choose the group of customers, depending on configurable category setting, to send an email to ask for feedback.
- Sub-contractor's view: The sub-contractor can view their performance score board and track their delegated tasks as well as feedback from customers.
- Customer's view: The customer can send the feedback or complain about the product via feedback form, and track the feedback status.

3 WORKING PROTOTYPE

We successfully built an application prototype for the demo session. There are three concretely working business processes with full front-end and back-end implementation. These business flows are concerned about the manufacturer as our main target and achievements. The other views of sub-contractor and customer are implemented as static pages, which can be seen on the code camp's wiki page.

3.1 Register new assignment

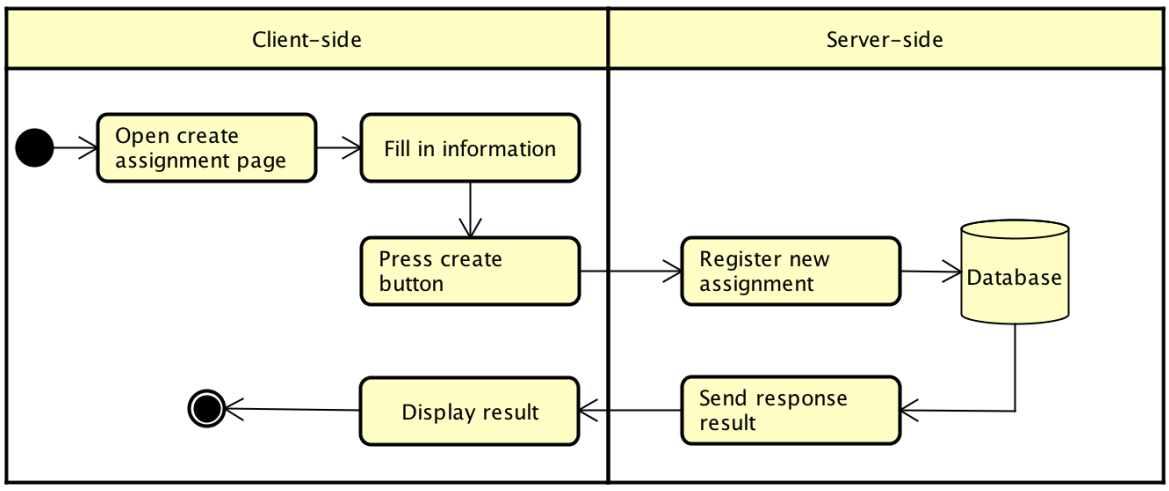

Figure 1. Register new assignment

The manufacturer opens the page to create new assignment by filling in all information. They are deadline, task name and description, sub-contractors picking, and the components choosing. Once button is pressed, all the data is sent to server and saved to database and the result is displayed on the screen.

3.2 View existing assignment

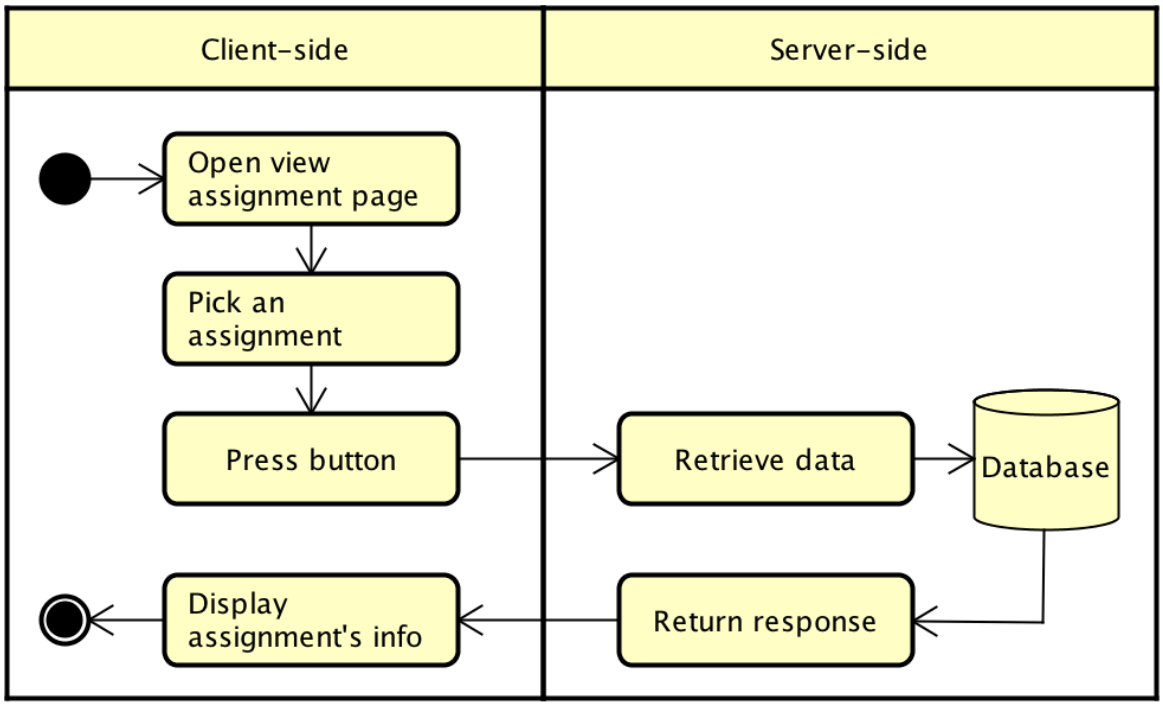

Figure 2. Track existing assignment

By opening the page, the manufacturer has a list of created assignments which are pulled from the database. The manufacturer chooses a task based on its name and press button to check the detail.

3.3 Send feedback request to customers

The manufacturer can choose the customers according to their group to send the feedback request. Such group can be classified based on their resident location, ages, or gender. In this prototype, we implemented the grouping of customer's location so that the manufacturer can pick customers from different cities in Finland to send the feedback request. One the request

is sent, the customers will receive a notification email, in which a hyperlink to the feedback form is provided.

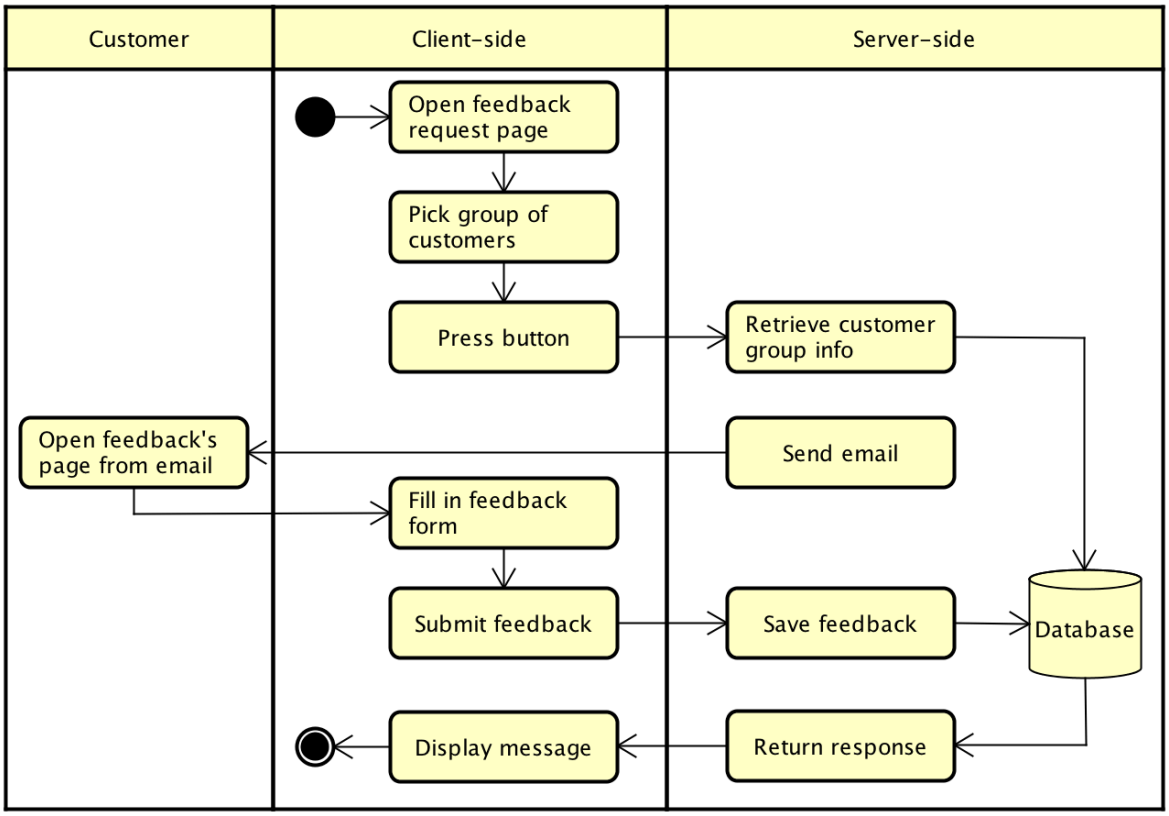

Figure 3. Feedback request

4 DEVELOPMENT PROCESS

The project started with a kick-off event on Monday to fully equip teams with the business case, schedule, tasks and expectation. After the kick-off, our team organized the very first meeting in a short duration to brainstorm and check on the requirements, tasks and agree on the development process. In the meeting, we also defined the roles for team members and setting up working environments. In consequent meeting on Tuesday, we came up with the final agreement on features to be built and started the coding implementation. In the following days, we organized one meeting at the starting of the day to review and discuss about the progress as well as plan the to-do list. On last day - Friday, we had a final meeting

to revise the whole work, test the application and prepare for the presentation and demo session.

The general roles of members are identified as in the following – Prasanna as business analyst, Kurosh and Muninder were responsible for front-end development, and Hieu as system admin as well as back-end developer. Prasanna analyzed the requirements and produced mock-up design of front-end on papers. She also supported to the front-end development using her skills on HTLM, CSS and JavaScript. Kurosh and Muninder were in charge to move all mock-up papers to the code, run and test on the server. Meanwhile, Hieu contributed to the setup of server and working repositories, implement the server coding and also support front-end coding. Moreover, to support team communication, we use Slack as the chatting tools when not be presenting in the computer lab.

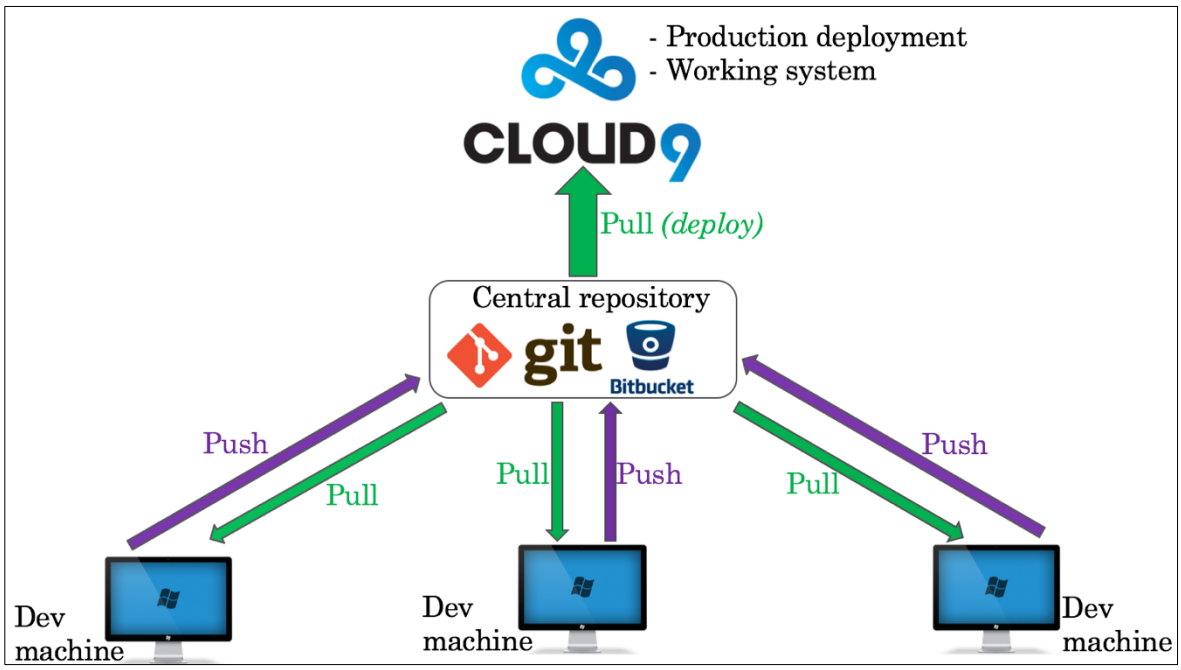

Figure 4. Organization of team works

We used GIT for controlling the version of source code. The central repository was setup on Bitbucket, allowing the private access through the usage of SSH keys. Each member wrote the code on his own machines, and then pushed their code to this central repository. On the other hand, the code could be pulled from the repository to local machines to be updated. We decided to have one Cloud9 workstation to serve as the production environment to run our application. We deployed all of works after testing to this environment by pulling code

from the central repository. In other words, the aim of this Cloud9 environment was solely for running application and we did not write any code there.

5 IMPLEMENTATION

The Propeller Hat team decided to organize the implementation to two concretely different systems - the client-side and the server-side. We defined the works for client-side and server-side and then delegate to proper team members to code. The client-side works referred to the user interface (UI) design, places for users to input data, and places for showing data to users. On the other hand, the server-side tasks involved in implementing database, retrieving data from database, and inserting or manipulating records to database. In a nutshell, there was no database installed on the client's side, meanwhile, all data were retrieved from the server by sending suitable requests using REST APIs. The server was responsible to listen such requests and prepare responses to send back to the client in the JSON data format. Afterwards, the client handled the responses and displayed information for users. The system design of this application is illustrated more clearly on the following figure.

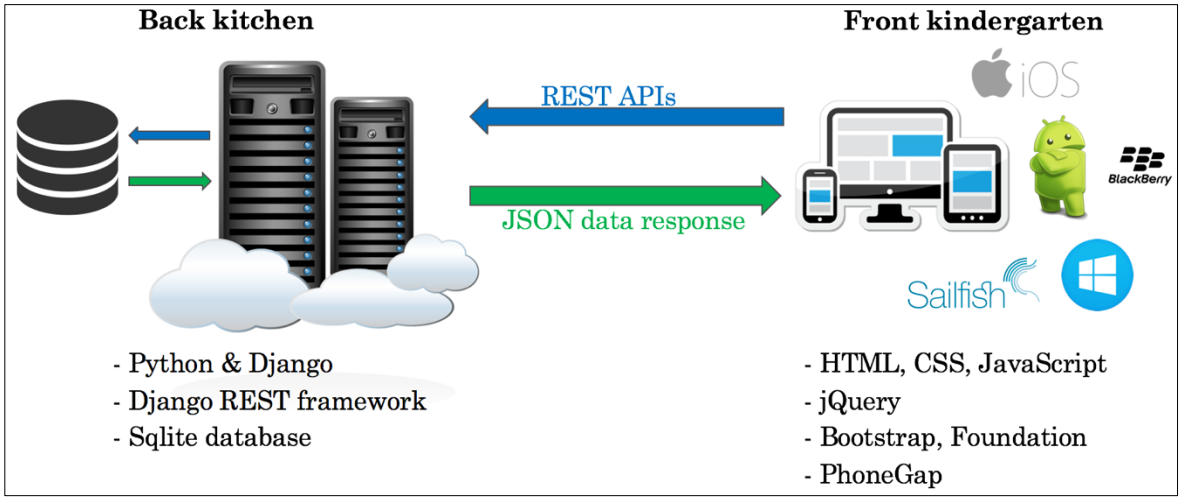

Figure 5. System design and architecture

On production, the system was running on the same domain for both client and server sides to unify the workspace. We setup the server to run at port 8081 and the client to run at port 8080 on Cloud9 workspace. On server, the programming language was Python with the usage of Django framework and Sqlite database. With the help of Django REST framework,

we were able to write REST APIs at a really fast pace to keep our schedule on time for the client-side development. The main reason our team chose Python and Django framework to build REST APIs was the rapid implementation of web APIs. Additionally, we would like to challenge our knowledge and skills on Python programming and REST APIs building. Besides, we picked Sqlite database due to the fast build within time limit, its light-weight and easy manipulation of data records. This solution was suitable for this short development iteration to build a demo prototype.

Meanwhile, we implemented PhoneGap technology to build the client side. Thanks to PhoneGap, we were able to easily conduct a server to test our design in different mobile devices, and certainly on the computer. We employed Bootstrap and Foundation frameworks to create the responsive UIs. In addition, we used JavaScript with jQuery library to send post and get request to the server through REST APIs, and manage the return response from the server.

6 CONCLUSION AND FURTHER DEVELOPMENT

Through the first iteration of this project, we successfully delivered a working prototype of the system and organized a demo session to the customer. Also, we have defined several tasks for the further development. Such tasks can be classified into two categories, namely technical aspect and business aspect. In the following, we will define several points for the further works.

In the perspective of technical implementation, we would like to implement security features to protect the APIs. Currently, the APIs can be easily accessed through the URL sources and all data can be retrieved in no time. Therefore, the implementation of security is necessary to prevent unexpected access, and hence we propose use token key authentication. Basically, the token key is required to send in any request and the server will check its validity to decide whether data can be returned. Moreover, the data type validation is required and a must for next step. The server has to perform data sanitization and examine whether their types are valid prior to proceed to the next steps. The same rule must be also applied to client side, in

which all input from users must be checked in order to prevent dummy data or system attacks.

On the other hand, we were thinking about the configurable ability of this application to fit all the business industry. In this prototype, we aimed at developing a system for mobile phone manufacturer and its sub-contractors as well as customers. We did not want to limit the expandable of the application, instead, we consider to build a platform which can be configured using a back control administration page to tailor the application for other fields, such as fashion, construction or catering.

7 APPENDIX

Example of system REST APIs

Get list of cities

API resource `/api/bigboss/feedback`

Method `GET`

Response

```
{
  “data”: [
    {
      “id”: 1,
      “name”: “Lappeenranta”
    },
    {
      “id”: 2,
      “name”: “Helsinki”
    },
    {
      “id”: 3,
      “name”: “Espoo”
    }
  ],
  “success”: true
}
```

Register a new assignment

API resource `/api/bigboss/create`

Method `POST`

Request

```
{
 "name": "Assignment 5",
 "component": "2",
 "contractor": "Phillips",
 "deadline": "2015-12-21 10:58:04.700125",
 "requirement": "Do do do!!!"
}
```

Response

```
{
 "data": {
   "rating": 0,
   "requirement": "Do do do!!!",
   "name": "Assignment 5",
   "contractor": "Phillips",
   "deadline": "2015-12-21 10:58:04.700125",
   "id": 24
 },
 "success": true
}
```