Patrice Boulet (6583832), Nicholas Gagnon (7453294)

Professor Timothy Lethbridge

SEG2105 Introduction to Software Engineering

4 November 2014

Assignment 5

PROBLEM STATEMENT

Students need part-time work in order to fund their studies and gain experience, while startups need smart and creative employees who are willing to work for lower hourly wages.

REQUIREMENTS

StudentsMeetStartups.com is a website that connects students with local startups for part-time work. It will allow debuting entrepreneurs to meet with students through informal meet-ups that, in contract with traditional interviews, will provide them with a much deeper and realistic impression of those students.

Students have a name, an age, a short self-description, an address, an optional personal website, an email address, a telephone number, an academic record, a resume, and a set of credentials to allow them to sign into the website. In order to attend meetups, a student has to be a member of the website. Both the academic record and resume will be uploaded as PDF documens.

A startup has a company name, a description, a postal address, a telephone number, an optional fax number, a website, an email address and a set of credentials to allow them to sign into the website. In order to post meetups, an employer must be a registered member.

An address has a street number, an optional apartment number, a street name, a postal code, a province and a municipality.

The meet-ups have start and end times, an address, an employer and participating students. Participants do not necessarily meet physically, hence the address could be omitted. After the meet-up has taken please, the startup will write a review of each student, whil each student will write a review of the startup. Such reviews include a 5-star based rating as well as a short textual comment.

ARCHITECTURE & TECHNOLOGIES

Since today's personal computers and mobile devices have the capabilities to perform very advanced computations, our system will be designed around a fat-client architecture in which the server component will merely serve as an adapter for the underlying SQL server.

The JavaScript language will be used for both the client and the server components. Our system will favour re-use of existing components, hence the KnockoutJS and jQuery frameworks will be used on the client-side, while Expressjs will be used by the server-side.

The MySQL database server will be used for persistent storage of information. At first, it will be running on the same server as our server component.

Our codebase will be hosted on GitHub, and as a consequence, we will use git as a version control system in order to keep track of changes.

The client and the server components will communicate using the HTTP protocol following the RESTful principles. The body of the HTTP requests and responses will contain JSON-formatted objects that are perfectly compatible with our choice of programming language.

Bootstrap 3 will be used for creating responsive and cross-platform layouts. This easy-to-use CSS framework will thus allow our website to run both on traditional personal computers and mobile devices such as tables and smartphones.

Finally, we will use the Zombie.js framework to provide us with an easy-to-use API to write functional tests for our whole system. This framework emulates a Web browser an navigates through the pages as a normal user would.

USER STORIES

All user stories are carried out in the same manner. A form is displayed to the user, and he or she has to fill it in, and then click the submit button. The system will ensure that all required values have been provided by the user. Figure 1 illustrates such process in more details using a use case map. The actual user stories follow.

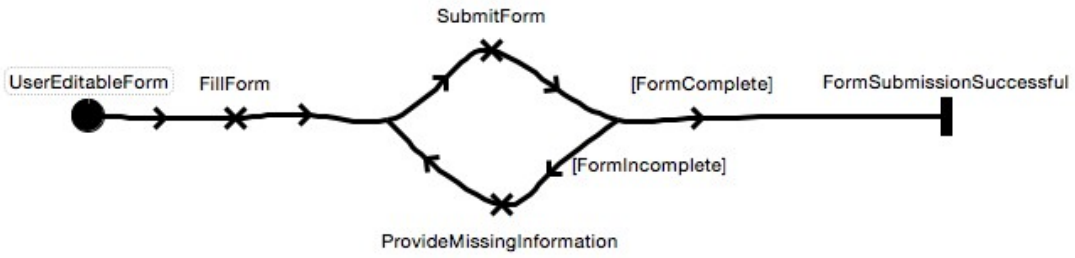


Figure 1 – General form submission use case map

1. As a student, I would like to sign in to the website in order to participate in meet-ups.

2. As a startup representative, I would like to sign in to the website in order to schedule meet-ups with interested students.

3. As a registered startup representative, I would like to schedule a meet-up in order to actually meet with students.

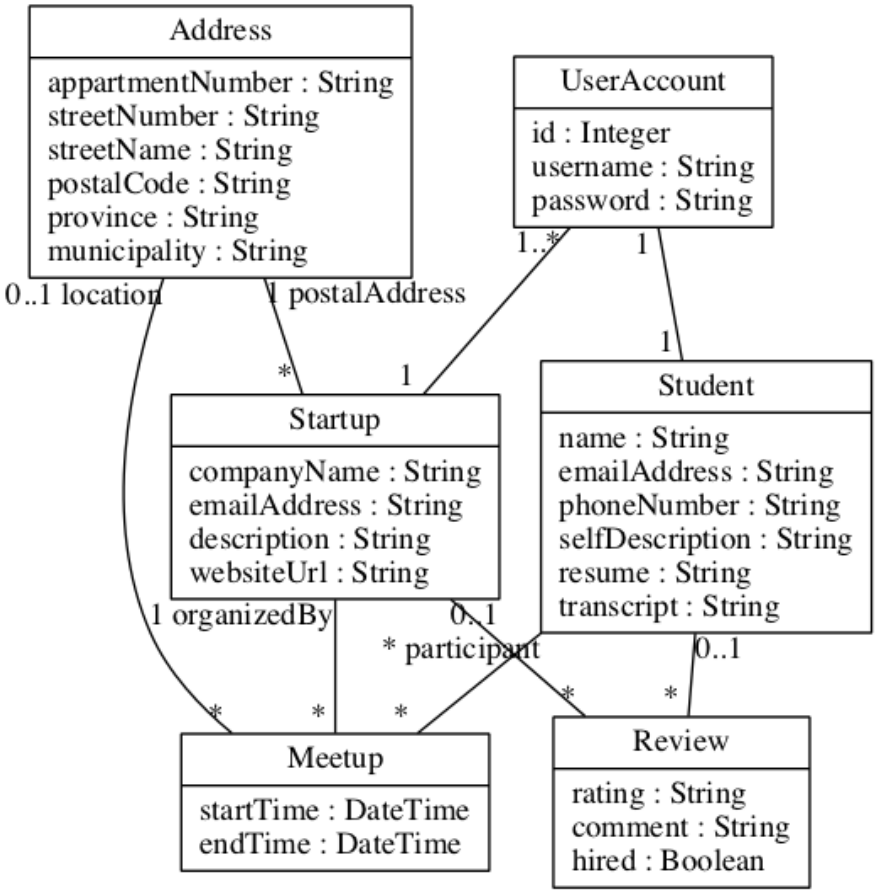
4. As a registered student, I would like to register to a meet-up in order to meet with a startup representative.

5. As a visitor, I would like to browse through the startup profiles registered with the website in order to discover potential employers.

6. As a registered student, I would like to modify my profile in order to display more up-to-date information about me.

7. As a startup employee, I would like to modify my profile in order to display more up-to-date information about my business.

CLASS DIAGRAM



MESSAGE EXAMPLES

All communication between the client and the server will follow the request-response pattern as dictated by our use of the HTTP protocol. The server will always respond with a JSON object that includes a confirmation as well as error messages, if any. Each user story needs its own request to the server, as shown by table 1.

The request body is typically a JSON-formatted version of the corresponding class in the class diagram. Therefore, the JSON key-values simply match the attributes of the corresponding class.

Table 1: HTTP requests associated with each user story

User Story	HTTP Request	Request Body
1	POST /student	{“student”: { “name”: “...”, “emailAddress”: “...”, ... }}
2	POST /startup	{“startup”: { “companyName”: “...”, “emailAddress”: “...”, ... }}
3	POST /meetup	{“meetup”: { “startTime”: “YYYY-MM-DD HH:MM:SS”, “endTime”: “YYYY-MM-DD HH:MM:SS” }}
4	PUT /meetup/:id	{“meetup”: { “addParticipants”: [“Student unique ID ...”] }}
5	GET /startup/:id	N/A
6	PUT /student/:id	{“student”: { ... }}
7	PUT /startup/:id	{“startup”: { ... }}