Meet Me For Language

Profiles for Socializing via Language Learning

Priyasmita Bagchi
Bahareh Shakibajahromi
Mike Bok
Ismail Kuru

Drexel University

2016

1 Description

This project aims providing a platform for users to learn and improve their non-native languages via socializing with other people within a university. It is a profile based platform which enables users to create their profiles and get matched to a language learning event. This platform provides not only a pure learning activity but also getting socialized via meet-ups of users in created events.

2 Functionality

2.1 Creating a Profile

A user creates a profile to get involved in the platform.

2.1.1 Indentification

Since this is a platform for university community, their employee, student identification number will be their user identification number.

2.1.2 Lecturing Languages In Native

Each user have languages that they are interested in teaching.

2.1.3 Learning Languages

Each user have languages that they are interested in learning.

2.2 Qualification Aspect

Each language stated in the profile, for teaching and learning, needs to be attached with a qualification, native, advanced, intermediate, basic etc.

2.3 Type of Lecturing Method

A user may want to meet with a group of people to teach. Maximum size of group needs to be identified. Besides, she may prefer to meet one-to-one for lecturing.

2.4 Type of Learning Method

A user may want to meet with a group of people to learn. Maximum size of group needs to be identified. Besides, she may prefer to meet one-to-one for learning.

2.5 Type of Events to Involve

A user states the type of events that are interested in attending. For example, a user may not want to attend any outdoor event etc.

2.6 Availability for Events

Each user states her available time intervals for the events.

2.7 Events Attended

A user has list of events that she attended to lecture or learn.

2.8 Creating an Event

Teaching and learning activities are performed via events. A user can create an event for lecturing or learning.

2.8.1 Event Description

Since it is an social activity, event creator needs to write the details of the event.

2.8.2 Event Creator and Lecturers

Each event has a creator and lecturers. A lecturer can both a creator and lecturer of an event.

2.8.3 Event Type

Event are classified against couple of parameters such as:

- **Indoor/Outdoor**. Where is it going to be held? Is it an indoor or outdoor activity?
- **Content of Event**. An indoor activity can be playing a game, having dinner etc.
- **Any Payment**. Does it include any sort of payment such as everyone pays its dinner?
- **Language Level of Audiance**. Level of audiance suggested to attend

2.9 Event Notification System

Once an event is created or potential meet-up exists other users are notfied in a guided fashion.

2.9.1 Guided Concrete Notification for User

Once an event is created filtered notifications are sent according to the preferences of an user. For example, a available time, type of event, level of the event are used for filtering the notifications for a user.

2.9.2 Guided Ghost Notification for User

If common interests for type of events in common available time exists for couple of people, a notfication is sent to create an event.

2.10 Confirmation System

Once the event created set of users/a single user gets notification for the event and

- After getting notified for an event a user may send a request for being an lecturer or learner for this event.
- A creator can get buch of request for being lecturer/learner for the event. Creator builds the group/single user among the requests.
- People who are confirmed by creator are notified for confirmation of thier attendance.
- Limits of the group (number of lecturers and learners) are specified.

2.11 Review System

Once the event is over, users can review via using two methods.

2.11.1 Commenting System

Commenting system can provide a way of reviewing the issues which are not quantified/qualified with teaching/learning activities. For example, an attendee may not be polite to other attendees and this may be mentioned in a comment.

2.11.2 Ranking System

- **Event**. Event creator can be review by other users.
- **Lecturer**. Lecturers can be reviwed by other users.

3 Development Infrastructure

3.1 Programming Languages

3.1.1 Front-End

Java-Script, PHP and HTML. [Mike,Bahareh] what do you think about front-end? Do you have suggestions ?

3.1.2 Backend

Java is going to be our main development language for backend functionalities.

3.2 Integrated Development Environment

We do not have our final decision yet but it is going to be one of IntelliJ/Eclipse/Netbeans.

3.3 Testing Infrastructure

There are couple of important points that we consider for testing infrastructure:

- We target using as much machinery as possible instead of human effort on generating specifications/test cases for functional correctness . Generating automated pre/post conditions, generating invatiants for modules in Class granularity.
- We have currently couple of options for machinery of testing infrastructure. Arquilian, The Grinder, Mockito or PowerMock can be one of our Java side testing frameworks. Arquilian enables you easily to create functional, integration and acceptance tests. However, we have experience on using Mock-Object-Based testing so we can also use PowerMock or Mockito.

4 Architectural Infrastructure

[Priyasmita] is going to write this part.

4.1 Data Base System

[Priyasmita] is going to write this part.

4.2 Web Server

[Priyasmita] is going to write this part.

5 Maintainability

- **Design**. We target a modular design which enables us to maintain the code base from the following aspects.
  - **Bug Isolation**. Our modular design provides localization of bugs so that refinement can be made fast.
  - **Extensibilty**. Our modular design provides fast integration for new features.
- **Code Reviews**. We review the source code of a written feature in function granularity. This review can be requested from implementer or any other group member can pick an update and review it.
- **Readable Code**. We use a structured naming convention. We follow rules on style of commenting, size of class methods etc.
- **Integration/Acceptance Tests**. We use integration tests to make building and functionality testing easier. We use regression tests for accepting a new feature to the system.
- **Version Control/Hosting**. We use version control and integrated issue tracking system. This enables us to keep hierarchy on status of stability of software. We keep release, in-test and in-development versions in repository. We use git as version control system and GitHub as remote repo client. Our repo is publicly available and resides at MeetMeForLang.
- **Issue Tracking**. We use issue tracking system integrated with GitHub client. We create issue with domain tags such as "Backend", "Testing", "FrontEnd" or "Documentation" so that related implementer can attack the issue fast.