Projects For Lebanon Documentation

How to install:

Eclipse EE and tomcat are needed. This link can be used to install and connect the two together: http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat-7-with-eclipse.html#undefined

After installing both, the files can be copied from the source folder (with this documentations). It is advised to manually create the folders using the Eclipse Project Explorer, and then copying the source files to these folders using drag and drop.

The source files also contain a MySQL JAR file; this file can be copied to any place inside the workspace but it needs to be added to the build path in the eclipse project.

Download and install MySQL server. The java files are configured to connect to:

Database name: projectleb

Username: root

Password: 1234

Be aware that MySQL server has a bug on windows 10, when installing it will ask for a name for the server (initially called MySQL followed by a number), when this name has capital letters, the windows won't be able to start the service. The solution is to call it a name with no capital letters (just change the M and SQL to small letters).

After installing the server, execute the commands found in db.sql (found in source folder).

Description of project:

The project is using JSP. We mostly rely on java files in which we print the html pages since the html will change based on the type of user. The only page that is written in pure html is the index, since it is the same for everyone.

One of the java files is called DumbDatabase.java. This file started as a dumb database (in order to start implementing the user interface before creating the database, we created this file to assume that we have a database however it only showed static results), now it is developed and it contains the real implementation of the database. In other words, the DumbDabase.java is a library used to interact with the database now containing a real working implementation. So inside the java files, we call static methods from the DumbDabase.java in order get information or update the database.

We have 4 types of users: Organization, Mentor, Donor, and Group of volunteers. They all share most of the information however each type has some specific fields; the mentor has a skill which they can help others with, the donor has a 'Donation' field where he puts what he can donate (money, clothes…), the organization and group are only different by definition.

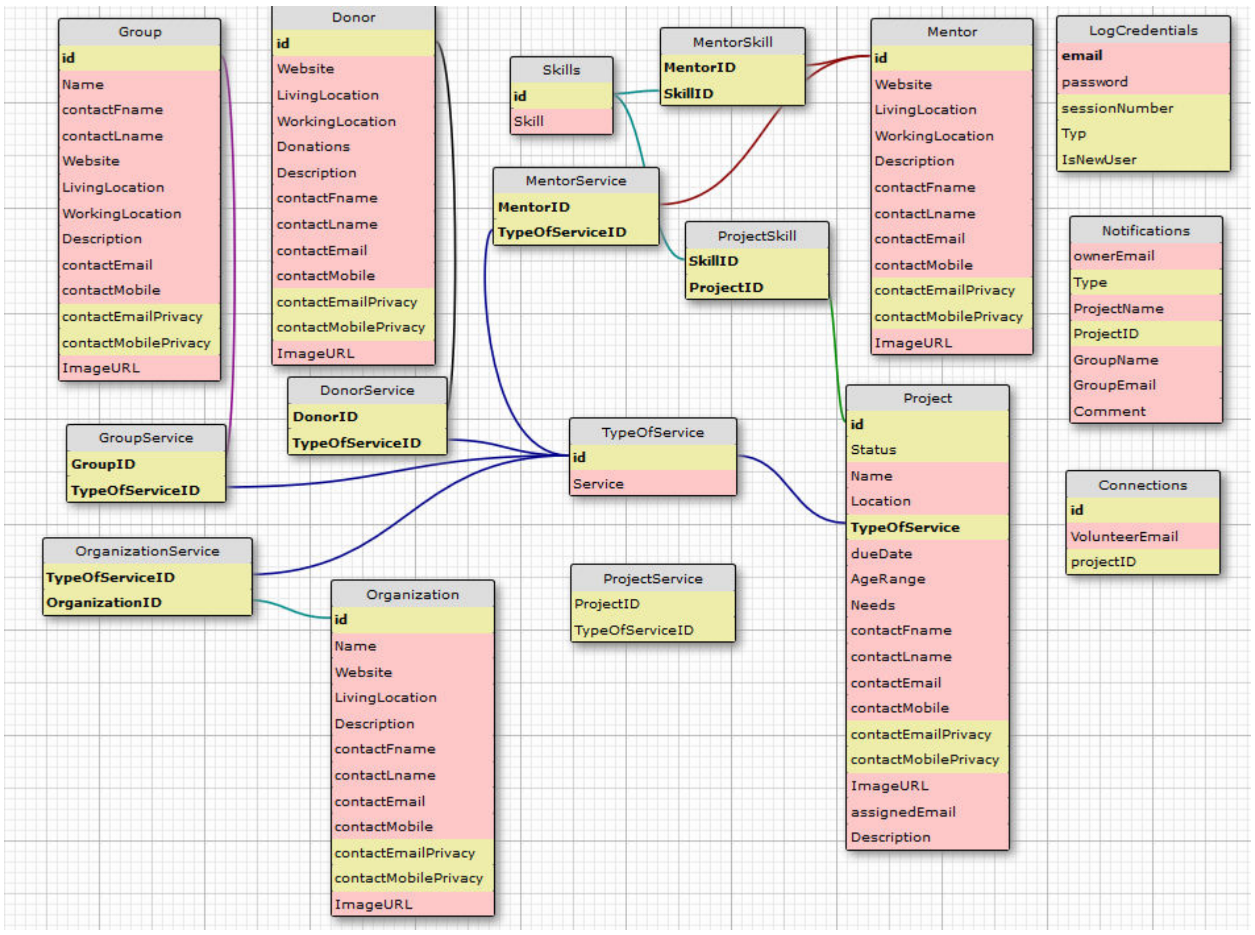
There is also an admin who should be able to accept and reject projects before displaying them publicly, however we did not implement this feature.

The JSCSS folder used to contain all the CSS and JS files, however after migrating to bootstrap, the most of the CSS files in this folders are not used anymore, the only important files here are the JavaScript files, they are all named the same name as the page that uses them. For example, CreateProfile.java uses createProfile.js and so on.

Notifications: We have 5 types of notifications (types of notifications may increase with the development of the project):

Accepted Project: A project proposed by a user is accepted by the admin.
Rejected Project: A project proposed by a user is rejected by the admin.
Connected: Some volunteer (donor, mentor, or group of volunteer) chose to implement a proposed project (which is accepted by the admin).
Accepted User: A volunteer is accepted to work in a specific project.
Rejected User: A volunteer is rejected to work in a specific project.

Database:



A file called 'database.xml' is included in the source files. This file can be used to load and edit above database preview in this website: http://ondras.zarovi.cz/sql/demo/

It is important to note that the Type of Services and skills will be preset in the database, that's why we created tables to link each of the 4 types to these preset Types of Services and skills using their IDs.

The way we are implementing multiple users is by using a session number for each user. This number is initially -13, and the server will have a static counter. Each time a user logs in, we will add a value in his session called 'sessionNumber.' The value of this number will be assigned equal to the static counter for the server and stored in the database, then the server will increment its counter by one. On logout, the session number should be back to -13 in the database.

The Connections table is used to link the volunteers (volunteer group, mentor, donor, or organization) with their respective project. So each time a volunteer decides to get involved in a project, a record is added in the Connections table. Although this feature is added in the database, we did not yet implement it on the website (one possible implementation is to add a button on the project profile page, when the user clicks it an AJAX request will be sent to the java page where it will update the database accordingly). It is important to note that we did not decide if the organization is allowed to volunteer in a project (or just propose projects) since if they do volunteer, there will be no difference between an organization and a group of volunteers. Another issue we did not solve is how to know if this user is the owner of the project or just a volunteer in it.

Description of the pages:

Index.html: This is the home page where each user who accesses the website will reach first.

> Sign in feature: It sends an AJAX request with the email and password to Login.java where it checks if the username and password are found in the LogCredentials table in the database.
> Sign up feature: After making sure that all the fields are filled, it sends an AJAX request to SignUp.java with all the information to sign up.
> Contact us feature: not implemented

Upload Picture: If this is the first time the user enters the website, after logging in he will reach upload picture where he needs to upload a picture, then he is sent to Create Profile page with the picture being uploaded to the server.

Create Profile: The page will receive the uploaded picture and save it on the server, however the code needs to be edited in order to update the database with the link for the picture (each of the 4 types has in its table a field called 'imageURL' where the link to the user image is found). Then, upon clicking submit, the information filled in the fields will be sent to EditProfile.java.

Dashboard: If the user is an old user (that is, signed into the website more than once), he will be directly landing on this page after signing in. If the user is new, he will land here after the passing through Upload Picture and Create Profile.

Projects: The page will send an AJAX request to get the projects from GetProjects.java. The data will be displayed on the java page (/GetProjects) as JSON objects, processed by javascript and displayed on the dashboard.

Notifications: Similarly, this page will get the notifications from GetNotifications.java.

The user can also navigate to update/view his profile or create a project.

User Profile: A user can edit any of the fields using the edit button. This is implemented by having a Boolean field called isOwner, the database will check using the sessionNumber if the person is viewing the profile is the owner or not and accordingly sets this Boolean value. In case this person is the owner, the edit button appears. All the user information are inside <input> html elements, when the edit button is clicked this <input> is made editable (it is initially readonly). After changing this input element, the user will click the edit button again, the data are sent through AJAX to EditProfile.java. The projects this person is involved in are also displayed on this page.

Create Project: Similar to Create Profile.

Project Profile: Similar to User Profile.

The status field:

Potential:

If the admin didn't accept it yet, the status field will show potential and only the owner can see it.
If the admin accepted it, the status will be potential and the page will be public.

In Process: Next to the status field, there should be a button (not implemented) that is only displayed to the admin, when he clicks this button it means he wants to start the project, all the volunteers should be notified.

Finished: The project is done. Patricia was suggesting to make a feedback page or form that can be filled by the owner.

Search feature: This is not a page; it is a feature that should be implemented on every page. It was implemented before, however we did not try it after adding bootstrap, hence it is only a matter of fixing the bugs (changing the names of the <div>s that should be hidden and so on). The implementation is found in search.js, particularly it is one function (searchOnLoad()) which should be called at the end of each JavaScript file (in the window.onload function).

List of known bugs and unimplemented features:

- When signing up, the database doesn't check if the user already exists, so anyone can sign up and overwrite an older user.
- Uploading pictures doesn't work, the method that uploads pictures on Create Profile is giving an error.

- The project status is not implemented.
- The Home and Profile icons in the header of each page are not linked, they just need to be put inside an anchor with references to the Dashboard and User Profile pages.
- The way creating a new project is implemented is by creating a new project and calling it temp at first, then filling the rest of the information. This is a dangerous practice since if many users are creating projects at the same time, the server will have many projects named temp. The implementation needs to be fixed.