EE6052/ED5022/CE4208 Group Project

1.Overview

This is a group project with groups of 3-5 students (groups with less than 3 students will not be accepted). Please add the names and ID numbers of your group members to the wiki page on the module's SULIS page. You can also leave messages there if you are looking for additional group members of for a group to join.

Your task is to develop a web application (outlined below) and deploy it on a virtual machine. Feel free to use any Java EE container and database you like. However, your sources must be submitted as a NetBeans project. All features should be implemented using Java EE (EJB, entity classes, persistence API) and JSF/HTML **only**. **Do not use any other frameworks** such as Hibernate or Spring. If you really "need" to use any other framework/library, you must first confirm with me that it is ok to use these.

For any queries, please refer to the question and answer section on the module's SULIS page.

2. Description

Your task is to write an online job marketplace (think freelancer, upwork, gigster or something similar). Job requester post their job descriptions to the marketplace and registered freelancer can offer to undertake the job. For this project, requester include a fixed price for the job to be undertaken – there is no bidding/negotiating. Further, there is no need to implement a real payment system – freelancer are paid with virtual tokens only.

2.1 Functional Requirements

Access to the job marketplace is limited – you must provide an authentication scheme. Access rights are role based, where your system provides three roles: provider, freelancer and administrator.

- Provider can perform the following tasks:
    - Create a job description (job is marked "open"). A job description includes:
        - Title
        - Unique Job ID (should be generated by the system)
        - Keywords
        - Job description
        - Payment offer
    - List all job descriptions posted by the provider.
    - Remove an open job description.
    - View profiles of any freelancer that offered to undertake a job description.
    - Accept a freelancer for a posted job description (assign the job to a freelancer that offered to undertake the job). Once a freelancer has been accepted, the job will be marked as "closed".
    - Mark a job description as completed (when a freelancer has completed the job). Once a job is closed, the "payment" will be assigned to the freelancer.
- Freelancer can perform the following:
    - Browse through all open job offers (offers that have not yet been assigned to a freelancer and have not yet been completed).
    - Search job offers by keyword (list all offers that include the specified keyword) and browse through the search result.
    - Search job offer by unique Job ID.
    - Offer to undertake an open job description.
    - Revoke an offer to undertake a job (only before the requester has accepted the freelancer).

    - Edit their profile - must contain at least name, Freelancer ID (unique, assigned by system), list of skills and a message to job requestors (allow at least 500 characters).
    - View the amount in their "payment" account.
- Administrators can perform:
    - Register freelancer to the database.
    - Remove freelancer from the database.
    - Register job provider to the database.
    - Remove job provider from the database.
    - Remove any job description from the system (in any state, i.e. "open", "closed" or "completed".
- A logging facility:
    - Every time a requestor accepts a freelancer or marks a job as closed a corresponding entry is added to the log (either a log-file or database table).
    - Every time a freelancer offers to undertake a job description a corresponding entry is added to the log (either a log-file or database table).

2.2 Technical Requirements

Your solution must use the following:

- All features should be implemented using Java EE (EJB, entity classes, persistence API) and JSF/HTML **only.**
- All posted job descriptions must be kept in a database.
- All users of the system (provider, freelancer, administrators) and their details must be kept in a database.
- You **must** use at least one Session Enterprise JavaBean (either stateless or stateful) that is remotely accessible.
- You **must** use Message Driven Beans for the logging facility.
- Your web interface must utilize:
    - At least one RequestScoped managed bean.
    - At least one SessionScoped managed bean.
    - At least one composite component.
    - At least one custom converter tag **or** one custom validator tag.
    - JSF Templates

3. Deadline and Deliverables

Deadline for submission of your solution is Friday of week 12 (Friday, 12.04.2019 – submissions will be accepted without penalty until 13:00h on Monday 15.04.2019). Please submit your solution as a **single** zip, 7-zip or rar archive (please do not use any other format and do not remove the extension from the archive) via the module's SULIS page (only submit your NetBeans project and your Report via SULIS). Only one group member should submit the solution (if multiple students submit, I will mark the first submission that I find)

A complete solution includes the following items:

- Demonstration of the working application in week 12 (lab slots).
- Well documented and formatted source code as a NetBeans project (submit the entire project folder, not only the individual source code files!)

- Report (MS Word or PDF) that contains a security evaluation of your application against OWASP Top 10 2017 (it is ok if your application is vulnerable!). This report features two parts:

1. A practical security test, where you outline how you tested the security of your application.
2. A discussion that establishes the following for each Top 10 item:
    - Is the Top 10 item applicable?
    - If yes, does your application suffer from the vulnerability?
    - If it does, outline:
        - How an attacker could take advantage of the vulnerability.
        - How you could secure your application (no need to implement your proposal).
    - If it does not, outline what you did to avoid the vulnerability.

5. Marking

The project is worth 40% of the module. In general, all students of a group will receive the same mark. However, if any group members are not contributing sufficiently, please let me know and marks will be adjusted correspondingly (no contribution means 0 marks).

Application meets all functional requirements (0.5 per bullet point)	10
Application meets all technical requirements (1 per bullet point)	10
Practical Security Test	10
Security Evaluation against OWASP Top Ten	10
Total	**40**