# Project #2 Due: 8th December @ 23:59

**You have all the material you need within Canvas - examples, Powerpoints, boilerplate code. Please do not Google because there is too much information out there, not all of which is good.**

Reverse auctions are becoming increasingly popular online.

Registered users publish jobs they need done, e.g. fit an Ikea kitchen or install a new bathroom suite, and businesses will bid for those jobs. The lowest bid wins the job.

You are to develop a website with the following features:

- Users can register to for the service using an email, a password and personal details i.e. name and phone number. An address is not needed - it would be communicated to the interested bidders directly by the customer.
- Authenticated users can
    - add a job, giving it a name, and description.
      The system should store the date on which the job was first published.
      Suggestion: use `LocalDate` which has many useful time/date related methods including `LocalDate.now()` for today's date.
    - bid for a job, if the user does not own the job
        - Bids cannot be made to Inactive (closed) jobs.
        - A bid must be lower than the current lowest bid.
    - A job is closed after 20 days.
        - This can be achieved using a scheduler. Every day the system change the status of jobs more than 20 days old to closed.
    - Any user can view jobs to see all the details of each job including the list of bids
    - If a job is closed, the user can view the job's details including who won that job and the winning bid for the job.

You must also provide

- Two REST API endpoints to return the following data in json format to authorised users
    - List all currently active jobs
    - List the bids made by a user, given an id
- An example of consuming these two APIs (requires another Spring project) – nothing fancy is needed here, just another project which has two controllers that send authentication data to the REST APIs and present retrieved data as, for example, a list – no CSS is needed. Do not waste time on making it look good – the focus is on functionality.

# Technical Notes

Develop a **WebMVC** Spring Boot application.

Use an **in-memory h2** database to create an "out of the box" application. This database must be populated with sample data.

The application must be implemented using be implemented using **JPA**, making use of all that offers. This project essentially has 3 entities: registered user, job and bid; a user can have many jobs, and a job has many bids.

Use the **Security** module for authentication and authorisation. This will require another entity i.e. Role.

I am not interested in visual styling – just make the web site well structured, readable, navigable, etc. Credit any templates (CSS/HTML) if you use them.

Forms must be validated using form binding and suitable error messages should be displayed by the view if the user makes a mistake.

It must be possible to **change the language** of the website. User input must be validated with suitable (international) error messages. You do not have to translate content. I will accept subtle changes e.g. "Balance" becoming "Balance_FR" to indicate the language has changed to French.

You **must** use the following:

- Spring Boot
- Spring MVC
- Thymeleaf (not JSP)
- Spring Data JPA
- Spring Security
- H2 for an embedded database
- Maven


Unit tests are not required.

You may use **Project Lombok** if you wish but it is not a requirement.

Provide a **brief document** outlining the high-level design of your system (1 or 2 A4 pages) including but not limited to your database design and class diagrams and the beans which you used.

# Marking

Consult the rubric for a breakdown of the marks and guidance on what is expected in each category.

# Submission

You may work in pairs or alone. Either way please sign up for a group even if it is a group consisting only of you. If you work in pairs, you will both get the same mark.

Submit as a 7z or zip file including

- A **brief document** outlining the high-level design of your system (2/3 A4 pages) including but not limited to your database design along with a list of issues (if you have any) e.g. functionality not implemented, limitations in code etc.
- Your **Spring project** as a .7z or .zip file.

# Penalties

Penalties for late submission are applied as per section 4.4.2 of the Exam Regulations. (Links to an external site.)

Do not plagiarize – don’t copy code, don’t provide code, don’t work with anyone outside your team

on the project. Trying to be clever by taking someone else's code and renaming classes and variables is not a good strategy and will likely result in a zero grade or worse.

# Need help?

If you don't know where to begin, some help is available.

Information about scheduling is available here.

# Rubric

Project #2 Rubric

Project #2 Rubric

| Criteria | Ratings | Pts | |
|---|---|---|---|
| This criterion is linked to a learning outcome Application Architecture & Quality of the code<br>The overall design e.g. use of architectural layering (packages, service layers etc), clean and readable code, commented where necessary. It reflects an understanding of the need for layers.<br>Spring Boot is used to create the default beans and application.properties is used to overwrite the defaults, if required.<br>Annotations are used to create beans.<br>MVC is used.<br>Code's documentation is concise and meaningful.<br>12.0 to >9.6 Pts Excellent<br>9.6 to >6.0 Pts Good<br>6.0 to >0.0 Pts Needs Work<br>0.0 Pts No marks | | 12.0 pts | |
| This criterion is linked to a learning outcome Functionality<br>Does the application do what it is supposed to do?<br>Go through the bullet points relating to functionality to ensure it does all that it is supposed to do.<br><br>NOTE: functionality does not include security - that is not included here. Security is a separate concern.<br>25.0 to >20.0 Pts Full marks<br>Excellent implementation of functionality e.g. a user can view all jobs; login; bid on jobs that are open and not her own; create jobs; closes jobs on schedule. Each view has a menu through which you can access other views/functionality.<br>20.0 to >12.86 Pts Good<br>Maybe you can create a new user, login, bid on jobs but perhaps the jobs cannot be closed and there are issues around bidding for your own job? Nearly there but not perfect.<br>12.86 to >0.0 Pts Needs Work<br>Basic functionality e.g. can create a user, can create a job but that might be it - maybe no bids.<br>0.0 Pts No marks | | 25.0 pts | |
| This criterion is linked to a learning outcome Internationalisation<br>Can the user change language? On all pages or perhaps just some of them? Do | | 10.0 pts | |

| Criteria | | | | Ratings | Pts | |
|---|---|---|---|---|---|---|
| error messages also change?<br>10.0 to >8.0 Pts<br>Full marks<br>The language can be changed in all views and even error messages change language. | 8.0 to >5.33 Pts<br>Good<br>The language can be changed in all views. | 5.33 to >0.0 Pts<br>Needs Work<br>The language may be changed in a few views. | 0.0 Pts<br>No marks | | | |
| This criterion is linked to a learning outcome Form Validaton<br>A user creates an account by providing data in a form.<br>A user creates a job by providing data in a form.<br>A user bids for a job through a form.<br><br>These forms should be validated so, for example, the user cannot bid a negative amount of money.<br>Some of this validation can happenclient-side (at a minimum by using the browser's checks) but should also happen through server-side validation using form binding.<br>12.0 to >9.6 Pts<br>Full marks | 9.6 to >6.4 Pts<br>Good | 6.4 to >0.0 Pts<br>Needs Work | 0.0 Pts<br>No marks | | 12.0 pts | |
| This criterion is linked to a learning outcome API REST endpoints and a Consumer for those endpoints<br>Create two REST API endpoints to return the list of jobs in Json format.<br>Most of the marks here go to creating a consumer website, another Spring project, that requests the data through the authenticated API and presents it in a webpage.<br>15.0 to >12.0 Pts<br>Full marks | 12.0 to >7.5 Pts<br>Good | 7.5 to >0.0 Pts<br>Needs Work | 0.0 Pts<br>No marks | | 15.0 pts | |
| This criterion is linked to a learning outcome JPA<br>Does the application take full advantage of JPA? There should be no SQL code, although perhaps a little JPQL might be needed.<br>14.0 to >11.2 Pts<br>Full marks | 11.2 to >7.47 Pts<br>Good | 7.47 to >0.0 Pts<br>Needs Work | 0.0 Pts<br>No marks | | 14.0 pts | |
| This criterion is linked to a learning outcome Security<br>Does the application take full advantage of Spring security? The user should be allowed to login. Once logged in, extra functionality becomes available to that user.<br>12.0 to >9.6 Pts<br>Full marks | 9.6 to >6.4 Pts<br>Good | 6.4 to >0.0 Pts<br>Needs Work | 0.0 Pts<br>No marks | | 12.0 pts | |
| Total points: 100.0 | | | | | | |

# Submission

Not submitted!

Submission details

You might not see all comments right now because the assignment is currently being graded.