Advanced Challenges in Web Technologies

Group Coursework
(40 Marks)

Deadline: Tuesday Week 11 at 16:00

The coursework should be submitted electronically as a zipped file of the final working version of the developed Web application as a NetBeans project (or as a Maven project), containing all the source code and other necessary files via the SurreyLearn system, before the deadline. The report should be submitted separately as a PDF (as is required by TurnitIn). Remember that depended on the size of your project, the upload time on SurreyLearn may vary, so plan for that (i.e. try to submit at least 30min before the deadline).

General guideline:

For all exercises it is required that you comment the source code appropriately. It is also important that there is no unused code within your program, and that the code is properly indented. Each HTML page should contain some basic instructions as to what the user is supposed to do on the page (e.g. in a popup help window, or similar), and how to do that. Finally, your implementation should handle inappropriate data entries (should not crash) and provide appropriate feedback messages to the user.

Brief Description of Topic:

The main aim of this coursework is to develop a Web application of your choice that demonstrates your understanding of the core module learning objectives. The application should include client and server side development using Java and Spring, reflecting the core objectives (demonstration of patterns, scalability, security, emerging web technologies) and at least four out of the six topics taught. Furthermore, the application should be supported by a technical document that describes the system specification and the project development details. Your group collaboration and how you handle problems is also taken into account. Finally, you will make a presentation of your achievements.

Detailed Coursework Requirements:

A. ARCHITECTURE

The architecture of the developed application should be separated in several layers (tiers) according to the needs of your application. Some of the layers that you could consider are:

i. Presentation layer (i.e. JSP views and anything that deals with the presentation of data)
ii. Reporting layer (e.g. security or traffic stats)
iii. Management layer (e.g. admin console)
iv. Security layer (i.e. authentication and authorisation is sufficient)
v. Business logic layer (i.e. your services)
vi. Database layer (i.e. models and data access objects). It is allowed but NOT required to access a database, so instead you can just hardcode some data in the data access layer.

NOTE: Be careful when adding additional layers! Do not add them if they do not provide a logical grouping of related components that manifestly increases the maintainability, scalability, or flexibility of your application.

B. IMPLEMENTATION

The implementation of your system MUST include the following characteristics:

- Utilising (at least one) relevant software design patterns such as MVC, Dependency Injection (Inversion of Control), Singleton, etc.
- Scalability capabilities, depended on the requirements of the system. Appropriate evaluation needs to be included in the report to support this.
- Basic security to include authentication and authorisation will be sufficient. This can either be achieved via an existing library or can be custom made.
- Although the user interface is not marked for styling, the user experience has to be at a level where one can use the system in a reasonably easy way.
- The system should be offering its services via a Web interface. Presentation logic should be well separated from the system's services.
- The system should be utilising the Object Relational Mapping principle to achieve persistence or to convert data into a different format.

Additionally, at least FOUR of the following SIX generic functionalities are required to be implemented in the system:

i. Rich client experience (e.g. usage of Ajax for dynamic data load, or jQuery manipulation of the user interface)
ii. Usage of the Spring MVC framework (i.e. using either XML configuration or annotations to setup the MVC of the application)
iii. Additional security (e.g. server side input validation and sanitisation, password encryption, encrypted communication, etc.)
iv. RESTful services (i.e. using RESTful services to request, update, etc records)
v. Intelligent Web behaviour (e.g. using clustering methods, classification of data, making recommendations for users, etc.)
vi. Distributing the processing needs (e.g. using Hadoop, Spark, etc.). This can be done as an offline process or as a live process, and appropriate processing updates should be reported in the Web application interface.

C. CODING

Coding quality and management requirements:

i. Each class or script should be commented appropriately (i.e. all classes and methods should contain comments and all important parts of code should be commented to explain usage)
ii. All unused code or classes must be removed from the final system!
iii. Classes should be appropriately separated into packages and layers
iv. HTML files should comply with W3C standards (please verify this at: http://validator.w3.org). Add a proof of that in your report (screenshot as an Appendix).
v. Separate programming logic from presentation logic (i.e. no Java in a JSP page and no JavaScript in an HTML page, etc.).
vi. Use a private GIT repository to store and manage your source code as a group. Several GIT repositories exist such as BitBucket or GitHub. Provide proof of using such a repository in your appendix.

D. APPLICATION

The "wow" factor! You should try and identify the emerging web technologies needed and then implement some functionality in the system that is impressive and state-of-the-art. Some examples include:

i. Manipulating the biggest proportion of the GUI with Ajax (i.e. single page Web application)
ii. Intelligent searching for data records (e.g. the government has many resources available online: https://data.gov.uk)
iii. Enhanced system security (e.g. SSL for encryption, and defence against XSS and SQL injection attacks)
iv. Handling complex database transactions and rollback
v. File uploading and management
vi. Integration with third party services such as Google maps, Amazon, Facebook, Twitter, etc.
vii. Using APIs for intelligent processing of data, such as sentiment analysis from news agents or Twitter (many APIs are free!)
viii. Distributing the processing to a cluster of computers (or the cloud)
ix. Scalability and load balancing considerations
x. Using other libraries (or APIs) not taught in the lectures

E. REPORT

Report requirements:

i. Description of the stakeholders and the requirements & objectives (SMART) of the system (1 page max).
ii. Description of the system architecture and supporting UML diagrams. The main diagrams should include:
    1. Use case diagrams (functionality as seen by users) with a short description (1 paragraph per diagram).
    2. Structural diagrams concerned with the functionality of the system, defining the object-oriented analysis and design elements, system decomposition into layers and subsystems, interfaces of the system and its components. Such diagrams include any of class diagrams, package diagrams and/or composite structure diagrams; so choose the most

   appropriate. Diagrams should include a short description (1 paragraph for each).
   3. Behavioural view diagrams of only three (most important) functions of the system. Such diagrams include sequence diagrams, activity diagrams, and/or state diagrams; so choose the most appropriate. Each diagram should include a short description (1 paragraph for each diagram).

iii. Description on the choice of technologies and patterns used in the implementation

iv. Short deployment and user manual of the system describing the main setup requirements of the system and the available functionality (1 page max for the deployment manual and 2 pages max for the user manual).

v. State how you worked as a group (brief roles and group structure) and how the GIT repository was utilised (2 paragraphs and a group diagram).

vi. For each member of the group: explain your role and contribution (1 or 2 paragraphs max per group member). Describe any problems or issues encountered (1 paragraph). This part should be done separately and included in the group section of the report.

vii. As a group, state clearly if the work effort was equal effort from all members or it was of a different proportion. If the members did not contribute equally, then suggest what should the proportions be, so that it can be taken into consideration during marking.

viii. Use the Appendix to include any evidence to support your work.

F. PRESENTATION

Presentation requirements:

i. You are required to give a 15 minute presentation about your system (date and place of the presentation will be announced at a later stage).

ii. The presentation should be structured as follows:
   1. Quick overview of the system (5 min)
   2. Demo (5 min)
   3. Main challenges & issues encountered (5 min)

iii. At the end of the presentation there will be 5 minutes of questions, at least one per group member. During the questions each member is expected to be familiar with ALL parts of the system. Familiar means that you should know the technologies used and the reason for the choice, so that you can justify it. So it is NOT expected that you will know the details of the work done by another member, but you have to be familiar with it.

NOTES:

- ALL presentations are open for anyone to attend. So you are welcome to come and see what others have to present.
- Make sure you have your system ready to go, so that you don't waste time booting up and setting up the demo during the presentation!
- You are not obliged to have a power point presentation, but you can do that if that will help you form your thoughts and the flow of the presentation.

EXEMPLAR TOPICS:

- "Visualising data on Google/OpenStreet maps"
- "Movie recommendation system"
- "Chatting/messaging system"
- "Credit score classification system" [must find some data online]
- "RESTful service for a mobile device" [i.e. no mobile implementation, just the service]
- "JavaScript (i.e. client side) based game"
- "Searching for trending news from online news agents" [i.e. find free APIs]
- "Statistical data visualisation with charts" [e.g. from https://data.gov.uk/]
- "Social media aggregator for extracting personal data of people!" [e.g. email, job description, company, etc.]

HINTS:

1. If you cannot use a database for the system; you could generate a dummy JavaBean that contains and returns some hardcoded records.
2. Make sure that you do not use curly quotes (“) or (”) within your source code and use straight quotes (") instead. This sometimes is difficult to spot and can cause lots of problems!
3. Name all HTML elements you are going to use, giving them a unique name and id.
4. Keep the code simple and well separated, and use standard code conventions / style (e.g. https://google.github.io/styleguide/javaguide.html).
5. Give some friendly instructions (guidance) for the user accessing the page.
6. Test your application on several Web browsers!!! If you cannot make it use in a specific browser then just mention it and don't let the user find out!
7. Use a search engine to help your development! There are also sites like StackOverflow (https://stackoverflow.com) that can be helpful, but always check the answers and always reference the source if you are to just copy the solution found!

WARNING:

You are free to use any books or electronic resources for your assignment. However you should write your own programming code, based on the knowledge acquired from these sources. Copying programming code that belongs to other people including your classmates is called **plagiarism**. Every coursework deemed to contain copied material will **incur severe penalties in marking. It can be given a total mark of 0.** To avoid this you must reference all your sources (i.e. as comments within your source code). If in doubt as to what constitutes plagiarism please come and see me immediately.
It is also **unacceptable** to get others to complete the coursework by posting it on any website and requesting solutions, perhaps paying for them. To avoid doubt also check "B2: Regulations for academic integrity":
https://www.surrey.ac.uk/quality_enhancement/documents/B2%20Regulations%20for%20academic%20integrity%202017-18%20final.pdf

Marking scheme: This assignment forms 40% of your overall mark. Feedback will be provided within 3 semester weeks of submission. Each part of the coursework is marked according to the following schema:

Part	**Marks**	
A - Architecture	**8%**	6-8: The application is clearly separated into separate layers and each layer contains appropriate classes and functionality. 3-5: The application does show some separate layers but some classes or functionality is misplaced. 0-2: The application either does not have separate layers or the classes/functionality of each layer is confusing.
B - Functionality	**34%**	Up to 18% for the must have functionality. Up to 4% for each of the four selected functionalities (max 16%). 4: All working well without any errors. 2-3: Parts working and some basic implementation done. 0-1: Nothing done or some parts working but not doing much.
C – Coding	**18%**	Up to 3% for each listed code quality criteria and 3% for using GIT. Up to 3% for each of the five coding quality criteria (total 15%): 3: Good effort 2: Fair attempt 0-1: Nothing substantial done GIT: 3: Good setup and good use 2: Fair setup but little use of GIT 0-1: Not really used
D – Wow factor	**10%**	10-8: An excellent implementation of an impressive feature. 5-7: A good attempt at implementing something exciting. 0-4: Nothing special was implemented.
E - Report	**15%**	12-15: A well-structured report covering all main sections, clear and easy-to follow deployment instructions and a well-presented, easy-to-understand user manual. 9-11: A reasonable report which covers most required aspects but could be clearer, the deployment instructions are clear but might have omitted steps or contains errors, the user manual could be clearer. 6-8: A poor report which hardly covers the main aspects, the deployment instructions are incorrect, the user manual is difficult to understand. 0-5: Hardly any documentation was provided, it did not reflect the actual system, the deployment instructions did not work, the user manual was unclear.
F - Presentation	**15%**	13-15: Excellent presentation and all members answered all questions asked well. 9-12: A good presentation with a working demonstration of the system, and good reply to questions. 5-8: The presentation could have been clearer, the system was not fully functional. Members of the group could not answer well to questions. 0-4: A poor presentation, the system did not work or most of its intended functionality could not be demonstrated. Not good answers to questions asked.