Table of Contents

Design and Architecture (describe each part you did and label them): 3

Overall architecture of the application 3

Flow Chart 4

Group 4

Movies.XSL (Ian) 5

Movies.XSD (Ian) 6

Movies.Java(Ian) 6

Search functionality (Ian) 6

startSearch.jsp 6

searchVar.java 6

searchResult.jsp 6

Movies.java 7

Movie.java 7

Reflections (individual): 7

Darsh 7

Claudia 7

Qiyuan 7

Ian 7

Issues And Challenges (group, list of challenges): 8

Group 8

Architecture and Code 8

Documentation 9

Version Control 9

Problem Solving 10

Research: 11

Darsh 11

Version Control 11

Bibliography (Harvard): 11

NOTE

Different highlighting means 2 different write ups eg here it means both me and Darsh had written something, so we need to decide which to use or how to mish mash them together

RED MEANS THAT THE AREA HAS CLEAR ISSUES AND MUST BE FIXED BEFORE SUBMITTING!

We have a bit under 2000 words, we are missing **Reflection** and **Architecture** please fill in your respective pieces!

# Design and Architecture (describe each part you did and label them):

## Overall architecture of the application

Most basic pages to initialise the project could be achieved with JSP, sessions and Javabeans. Basic starter pages included:

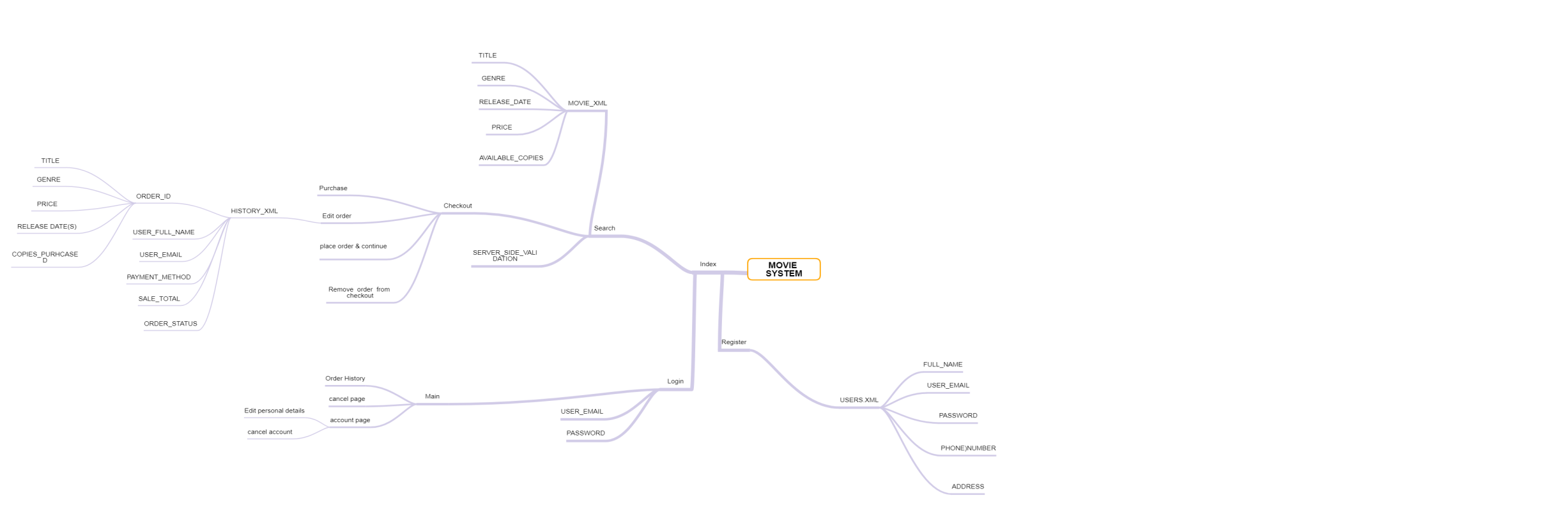
* login.jsp & loginAction.jsp
* register.jsp & registerAction.jsp
* welcome.jsp, main.jsp
* header.jsp

JSP pages allow dynamic pages to be created that make use of Java features and code that is processed on a Java server. This is incredibly important to implement and understand as organisations require websites to operate within the specified business logic and seperate that from the presentation of the web page itself (Hock-Chuan 2012). The labs and demos were used to create a template in which we could start implementing the specified requirements. Various features of JSP were utilised such as the jsp:include function to include header.jsp on every page without have to duplicate code. Sessions were implemented through the JSP scope session.setAttribute. Additionally, a JSP logout script was implemented via <% session.invalidate(); %>

A key component and feature of java web applications includes the use of Javabeans. This component allows encompassing data from various sources and also allows creating ‘reusable components' (Oracle 2018) to be used a later time or place within applications. This includes features like Serializable interfaces and the use of getters and setters to pass and edit properties. An example of a javabean used within the project is MovieStoreUserApp.java which is used as bean through <jsp:useBean id=... class=... scope=...> . This allows getting and settings Users for example along with the crucial updateXML aspect that will be discussed later on.

We have been tasked with creating a movie purchasing site using the knowledge we have gained from our weekly lectures and tutorials. The web application should provide the user the ability to **search the catalog of movies** and by **logging in** or **registering** an account they can then complete the process and **check out** their chosen movies. The site also would provide additional features connected to this process including the ability to **view order history**, **cancel orders** and **edit personal details** or **cancel their account** completely. In addition there were certain technical requirements such as the inclusion of **REST**, **SOAP** and **server side validation**.

## Flow Chart



Note:

* lowercase = pages and functionality
* uppercase = XML structure
* xsl missing but assumed
* movie system == index

This flow chart was one of the first things we made. We used it to help with visualizing how each part of the system would connect with each other and also to assist with sectioning off each part of the project and allocating parts to each member of the team.

## Group

<class diagram>

The basic architecture of the OMS web application includes:

* **JavaServer Pages** jsp - these pages house both the html and additional code that allows the page to be dynamic. Various features of jsp were used to further deliver and enhance the minimum requirements. For example, <jsp:include page="header.jsp" /> was included in all relevant jsp pages to avoid having to manually include the header html code. Another benefit if using jsp is that regular HTML
* **XML, XSD, XLST** - Various components that act as the data store, schema language, and offer transformative capabilities. In summary, the components above use XML and the various frameworks available to allow storing data, defining what data is stored, and communicating with the java web server to transform data into valid and dynamic pages.
* **Java Source Files and Packages** - This is where most of the logic of the web application resides. The parent package is uts.wsd which includes the majority of java source code. For example, Movie.java stored the relevant fields, getters and setters, and crucially maps fields to their XML element counterparts. Additionally, Movies.java is a relatively simple program that allows the web application to access an ArrayList of movies that are stored in pre-mapped XML elements.
* **REST and SOAP** - WIP
* **Glassfish Server -** Glassfish was used as the Java Web Server as all labs and lectures relied upon this.

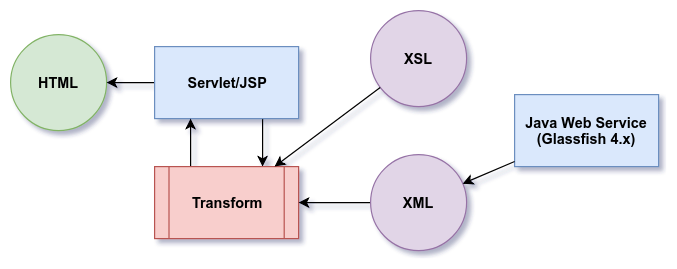


Figure 1a, as demonstrated in Lecture 6

## Movies.XSL (Ian)

As the XSL would be used for displaying the search information I wanted to create something that was as simple as possible to display, I did this by placing all the required information in cells with each movie making up its own row, The intended outcome would be something similar to this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Genre | Release Date | Price | Available Copies |
| Citizenfour | Horror | 2014 | 25.00 | 4 |
| Who Am I – Kein Sysyem ist sicher | Action | 2014 | 17.00 | 5 |

We had to make sure that we were having the correct names under the XML in order to have the schema work correctly when applying templates to each aspect EG “<xsl:template match="**movie**">” vs “<xsl:template match="**movies**">”.

## Movies.XSD (Ian)

For the XSD I did some research regarding appropriate limits for each field and tried to implement them this included limiting the year of release for movies to only years where movies have been around (min = 1870) and I also set the data type for each field appropriately such as the price being a double and the year being an integer.

## Movies.Java(Ian)

# Search functionality (Ian)

The implementation of search functionaility in our project primarily relies of 5 key components/files, these are **startSearch.jsp** which collects the users search input and categorizes it, **SearchVar.java** which stores all the information about the users search for use in later functions, **searchResult.jsp** which is embedded in the index page and gets all the movies that match the users search and displays it as well as code in **Movies.java** that creats an array list of all the movies that meets the search criteria and **Movie.java** that returns if a particular movie meets the search criteria.

## Reflections (individual):

## Darsh

This was my first foray into helping develop web applications and services at such a relatively large scale. Ideally, with a subject that covers so many different concepts and methodologies, all topics were equally important in order to understand this project. I had a tendency to reach for low hanging fruit, especially when starting out, which resulted in dedicating time and energy into elements that should have been prioritised further down. This is where the leadership skills of both Claudia and Ian helped me stay on track and correctly prioritise deliverables. For example, I may have dedicated too much time into trying to fix some styling elements (Bootstrap 4). By looking at the deliverable schedule I could see that I was behind and was able to shift my focus onto something more substantial like debugging the checkout functionality.

## Claudia

During this assignment, I faced two specific challenges; one is that I have limited coding experience, and the other is that I have never interacted with code which had dependencies with each other. This assignment is the first time I am required to use the code that I have written previously and built upon over time. Additionally, the marking criteria of this project requires an in-depth knowledge of java. This is not familiar to me and required me to adapt to the challenges of this project.

Challenges we faced as a group

Coding as a group is challenging for us as all of us have different coding styles. Due to other university and work commitments, time we spent face to face to code was limited. We relied on a combination of github and sending each other whatever progress we made through our group chat. Additionally, we encountered difficulties with integrating our code to the project, often needing to re-read our work to understand how to apply it. Although there was communication between me, Ian, and Darsh; the means of communication added a layer of complexity towards successful code integration as we had varying ideas on how to build the project. As a result, whenever we realized the product was not working, time was wasted trying to get to know how each other’s code works.

As someone with no coding background, this assignment has been incredibly difficult and stressful. However, this has been exacerbated due to one of our team members doing no work on the project. During our very first audio enabled meeting on Discord, he was playing video games the entire time and not contributing.

## Qiyuan

## Ian

Similar to some other members of the group I didn’t have any experience with developing web based applications so I ended up having a lot to learn and it took me some time before I was able to fully understand how each aspect of the project would be connected. I started of by doing tasks that had the most reference available to it such as the XML’s however by focusing on that I ended up not working on other tasks that were more important and were dependencies for other tasks. In hind sight I should have instead taken on more important tasks and at least started on having a frame work for them and taking advantage of the workshop times and the regular meetings in order to seek feedback and bring each aspect to completion.

# Issues And Challenges (group, list of challenges):

## Group

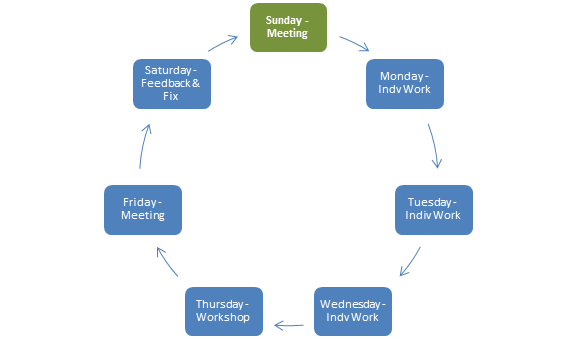
Various issues were encountered in the development of web applications and services for this project.  These issues were subsequently reflected upon in this report and has allows the group to appreciate the complexities of developing an application within a team.

## Architecture and Code

**Team management**

Right from the beginning of the project we agreed to have regular twice a week group meetings with one longer meeting on Sunday over a voice call on discord where we could allocate work, check up on what we are doing and also ensure that our code would match (referencing correct file names in imports, packages, etc). And we also had planned a shorter meeting every Friday over text where we could ask for help and feedback. This would help avoid our Sunday meetings being just to fix issues and instead more of an opportunity to plan on what we do next and move forward.

With the inclusion of the weekly group meetings our work cycle included individual work, learning and getting feedback from the workshop, getting help on allocated parts and a meeting to allocate the next step. Unfortunately this didn't end up working as well as we would have liked,  we only had one meeting a week where we would allocate new work, because of this we ended up having insufficient allocation of work.



The planned work cycle

## Documentation

The report for this project was a great chance to document our findings along with providing individual reflections. Google Docs was the initial solution the team had in mind. The main issue with Google Docs was the lack of task delegation and of course syntax highlight for code that group members shared. Initially, this resulted in some disarray as it was somewhat difficult to keep a track on what everyone was working at any given time.

Claudia showed the group the collaborative features of Dropbox Paper. This was similar to Google Docs but included the option to attach files, jot down ideas, create to-do lists, include code with syntax highlighting, and most importantly assign tasks to team members. This allows us to tick off tasks that were assigned to team members, record minutes of each meeting with detail, and provided a scratchpad for everyone implement draft solutions or report deliverables.

The group now appreciates the issues that must have been encountered by teams working on web applications - especially if team members are remotely located. As a result, tools and services like Atlassian’s Jira, Confluence, Trello, Asana have been created and extensively used within industry.

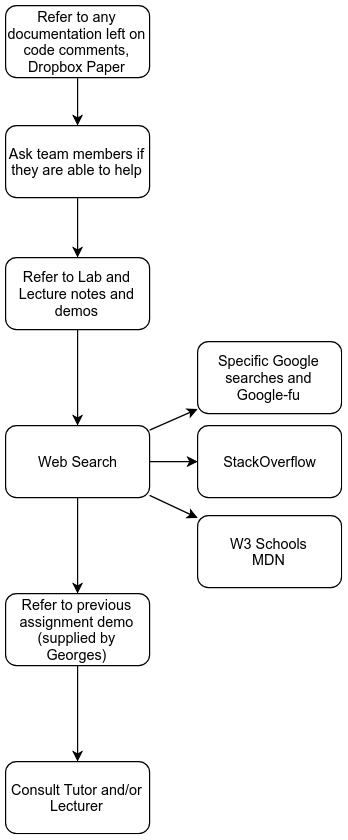
## Version Control

It was quickly apparent that any team working on a technical project requires some form of version control. Our group had a mixed experience level with version control systems like git. Initially, the project started out with various files being pasted on a Dropbox Paper (as described above) which was adequate to start with as they were mostly xml schemas.

Since no one in the group had actively worked in a project using git, there was considerable research conducted in the early stages of the project. This research is later in the document. One key problem with a team project involving web services is viewing the changes rendered by the server in a continuous fashion. This is where continuous integration and delivery (CI/CD) would play a large part in ensuring projects are steadily tested and updated with code changes. While we were unable to solve this as a group, a great deal was understood on how it allows developers and businesses to delivery high quality applications (Heroku 2018). If our group was able to integrate CI/CD using Heroku, then the Movie Application would have been visible on a URI such as xxxxx.herokuapp.com using a Glassfish server hosted in the Heroku cloud.

## Problem Solving

Our group spent a lot of time problem solving and some time was spent on not only working through challenges but anticipating future challenges. As noted by Chan (2017), it is a perpetual problem that all development teams will face which is why a good framework and methodology is required. An example of this problem included trying to get in-line server validation working via  Validator Java class. Our group did not know how to validate effectively using Java initially. The process we went to get answers for this issue lead us to develop a brief diagram on how to approach problems as seen below in *diagram 3a.*



# Research:

## Darsh

### Version Control

A variety of websites were used to solve the issue of selecting and implementing version control systems. Initialising the project and git was a priority when the project was started. Gregg (2018) provided fantastic instructions on how to initialise a project with git using Netbeans. After following the instructions I was able to see the repo on both github and on my local workstation. Github was chosen because of the amount of documentation available along with private repos availability for students.

For teams working on software projects, git branches are a useful feature to utilise (Atlassian 2018). Separate branches were created for each member of the group. A brief meeting was held where I described the process we should be aiming for by committing regularly and pushing upstream to individual branches. Ideally, in an agile and fast-paced environment, branches should be used for developing individual features or sprints which are then merged to the master (Vallandingham 2015).

## Team management

We were unable to fix the largest issues with regards to this during the project however we realize there are many things that if applied earlier could have helped mitigate some issues. Firstly, having an agile or scrum development process with a scrum board and potentially 15 minute daily scrums this would help with keeping track of peoples work and letting those who finish a task to on their own to chose the next requirement in the process and begin working on it (Wikipedia.org/Scrum 2018). Although there are issues with scrum such as for dependencies that fall out of scope of the scrum system most of this would not be applicable to our much smaller scale project.

## Documentation

Overall most of the issues we had with regards to this was the lack of commitment for example we created a to do list however we weren’t keeping this up to date. This is another reason why a scrum approach would have been beneficial as this could have been one of the things ticked of in the daily scrum (as was previously discussed)

# Bibliography (Harvard):

**USED: [http://www.citethisforme.com/](http://www.citethisforme.com/" \t "_blank)**

**TODO**: Harvard!

En.wikipedia.org. (2018). *Scrum (software development)*. [online] Available at: https://en.wikipedia.org/wiki/Scrum\_%28development%29 [Accessed 4 Oct. 2018].

Www2.lawrence.edu. (n.d.). *How to Use github with NetBeans*. [online] Available at: [http://www2.lawrence.edu/fast/GREGGJ/CMSC250/git/github.html](http://www2.lawrence.edu/fast/GREGGJ/CMSC250/git/github.html" \t "_blank) [Accessed 30 Sep. 2018].

Atlassian. (2018). *Learn Git- Git tutorials, workflows and commands | Atlassian Git Tutorial*. [online] Available at: [https://www.atlassian.com/git](https://www.atlassian.com/git" \t "_blank) [Accessed 29 Sep. 2018].

Chan, V. (2017). *10 Steps to Solving a Programming Problem – codeburst*. [online] codeburst. Available at: [https://codeburst.io/10-steps-to-solving-a-programming-problem-8a32d1e96d74](https://codeburst.io/10-steps-to-solving-a-programming-problem-8a32d1e96d74" \t "_blank) [Accessed 30 Sep. 2018].

Vallandingham, J. (2015). *A Git Workflow Walkthrough - Feature Branches - deployment, git, software development process, tools and workflow - Bocoup*. [online] Bocoup.com. Available at: [https://bocoup.com/blog/git-workflow-walkthrough-feature-branches](https://bocoup.com/blog/git-workflow-walkthrough-feature-branches" \t "_blank) [Accessed 30 Sep. 2018].

Chacon, S. and Long, J. (2018). *About - Git*. [online] Git-scm.com. Available at: [https://git-scm.com/about](https://git-scm.com/about" \t "_blank) [Accessed 30 Sep. 2018].

Heroku.com. (2018). *Heroku & Continuous Delivery on Heroku | Heroku*. [online] Available at: [https://www.heroku.com/continuous-delivery](https://www.heroku.com/continuous-delivery" \t "_blank) [Accessed 30 Sep. 2018].