Capstone Project Cover Sheet

Capstone Project Name: ABC Resource Allocation Module

Student Name: Ashwin Asani

Degree Program: BS-IT-Software

Student Mentor Name: [redacted]

Capstone Project Waiver/Release Statement Acknowledgement

It is the policy of Western Governors University ("WGU") that student Capstone projects should not be based upon, and should not include, any proprietary or classified information or material belonging to your employer or any other organization ("Restricted Information") without appropriate authorization.

Please confirm (by signing below) that you will complete (and upload into TaskStream) the IT Capstone Waiver Release form (verbiage is available in Appendix 2 of this document) indicating that your project does not include any restricted content. If you have included restricted content, please confirm that, in addition to the IT Capstone Waiver Release form, you will upload a suitable release letter giving you permission to use restricted information (A sample release letter is available in Appendix 3 of this document).

[redacted]
Student's Ink or Electronic Signature

4|21|2015
Date Signed

Table of Contents

Capstone Project Summary

Every company has several internal projects going on at any given time. It gets increasingly difficult to track which employees are assigned to which project since most assignments occur over email or paper service level agreements (SLAs), that are extremely difficult to track. ABC Company is one such company with 200+ employees and conducts at least 20 projects at any given time. As with any software project implementation, it is critical to understand the current business process, so that the software that is delivered fills the gaps in the process. We met with the stakeholders, starting with the CEO all the way down to the Project Managers and Project Coordinators from the Project Management Office (PMO). The organization is a matrix organization. The employees report to their functional managers and are assigned to projects during which they report to the Project Manager for that project. The sponsor for this project is the PMO. They have three main business requirements:

1. Only the functional manager is aware of the availability of his/her direct reports. This causes lack of transparency and makes it difficult to track the status of an employee in case of manager absence.

2. There are no organization wide reports that provide a listing of employees assigned to a project and the duration for which they are assigned.

3. Historical report of employees involvement in different projects.

In a nutshell, if a project coordinator from PMO needed to know when a particular employee would be available for assignment, they would have to setup a meeting with the functional manager to get a report so that they could plan towards the next project. This would be much easier if they could just use an online system that gives them a report on demand. Project

Coordinators can perform all the planning tasks without meeting a single person and then fix only exceptions if required.

This project also had time and budget constraints. It was imperative that the project got done within four weeks because the company was initiating several projects for which they were hiring contractors. The system was expected to enable efficient tracking of every contractor assigned to project tasks, such that they could be quickly reassigned to other projects as soon as they complete their project duties.

The functional and project managers also agreed that a web based online system with reporting capabilities would greatly benefit the organization. The office of the CEO was also onboard with the requirements as the application offered greater transparency and flexibility for all employees. The CEO was interested in technology that is robust and scalable due to the expansion plans of the company.

Based on inputs from all the stakeholders and approval from management, it was decided to use Open Source Development framework of Spring, Hibernate and MySql database in Model-View-Controller (MVC) architecture. These technologies are robust, scalable and offer modular development. Development was done using a bottom-up approach wherein we started with developing the database, then the middle tier and finally the presentation layer. There is typically much lesser coding involved in using frameworks. Once the prototype was developed, it was a matter of developing everything in a similar fashion to achieve the end result. This also meant that when the scope changed, new features could be added to the application without a major impact to the timeline OR the new requirements were slated for a later implementation as a separate project.

By and large, the purpose of this project was to provide PMO with easily accessible reports on employee involvement in current and past projects and to facilitate assignments to new projects. This report highlight all the stages of software development life cycle such as Initiating, Planning, Implementation, Monitoring and Control and Closure. The report described in detail the challenges that were experienced for completion of each phase. This final report describes the details of the final implementation until the delivery of the working application. Monitoring and Control phase were outside the scope of this assignment.

Review of Other Work

This project was essentially a subset of human resources planning framework of a company. According to a study by Rajesh K. Yadav and Nishant Dabhade in 2013, establishing a HR Planning framework and audit practices is a major achievement, that enables an organization to ensure how best to use its human resources to achieve outputs and outcomes. Employee plays crucial role in the success of business organization, it is important that organization should put consideration and careful planning into human resource practices. It is imperative that your Human resource practices should correspond with the business plans of a company. Human resource planning is an ongoing process. Organization must continually monitor and forecast personnel needs and concerns. Human resource planning is something that you can learn and improve on through experience and effort. The authors go on to state that formerly this process was mainly a clerical function but now human resource is a strategic partner in planning and attaining organizational policies and goals (Yadav & Dabhade, 2013).

Essentially the current system in ABC Company reflected a clerical function as stated by the authors in the study. The assignment of personnel to projects by email and the lack of

transparency was clearly an archaic method of project management. Having a web based application was in line with organizational process improvement, as indicated by the study.

In her book “Project Management”, author Adrienne Watt describes that resource planning is critical to the success of a project. The author postulates a case study of planning a wedding that involves several different tasks and the wedding planners being overwhelmed by the fact that they have no idea who is going to accomplish which task (Watt, 2014). It is clearly evident that resource assignments need to happen in the planning stage of any project. To be able to do that, the project coordinators need to have clear view of the availability of all personnel in the company workforce. This was achieved by having a web based resource allocation module that provided reports based on a employee to project relationship.

Considering that the project was essentially in line with improving business practices and that there was a definite need to manage resource allocation, there needed to be a make or buy analysis to determine whether there are any out of the box products that could be used to achieve the same functionality. According to an article in the Forbes Magazine, contributor Chuck Cohn explains that building custom software can unlock a host of benefits, but companies should only pursue that strategy if the software can provide a competitive advantage relative to the competitors, and the cost of a proprietary system can be spread over a large number of clients. The author goes on to state that even though it might be more expensive to build custom software, the return on investment makes it worth the effort. He also explains that if a decision to buy is made in a hurry, the repercussions of that decision are felt at a later stage when customizations are required to suit the business needs of the company process (Cohn, 2014).

As a result, it was decided to examine two products that have subtle difference in features. The first product that was examined is Microsoft Project. According to Atchison, S., & Kennemer, B. (2011), the software allows to accurately measure resource utilization and better manage resource allocation that aligns with your strategy. The software also helps to better manage the project pipeline and what employees are working on by capturing their ideas in SharePoint Online task lists and evaluating them in Project Online (Atchison & Kennemer, 2011). From this description it was evident that Microsoft Project was a tool for ongoing project management. So it gave a view of task based allocation of resources along with their end dates but it is restricted to just the current project. One of the business requirements of ABC Company was to view the past projects the employee has worked on and that would not be available using Microsoft Project. Essentially, the use of Microsoft Project would come in only after the project coordinators have already gone through the custom web application to pick the resources.

The second product that was evaluated was Hub Planner. As per the product website, this product is a highly complex resource management system. It involves components like booking grids, time sheets, dynamic reporting, and overall resource management. It has interesting features like resource tagging that can help dynamic searches for employees with a particular skill set. The product also scales easily with company growth. In terms of features, this product largely met several business requirements for the ABC Resource Allocation Projects in terms of tracking resources from the perspective of detailed monitoring. ABC company in that sense already has a Human Resources Management system that is linked to their payroll. So features like time sheets, etc were redundant provisions that are unnecessary. Moreover, the pricing for Hub Planner seems largely based on the number of resources. So the cost is expected to rise with the growth of the company.

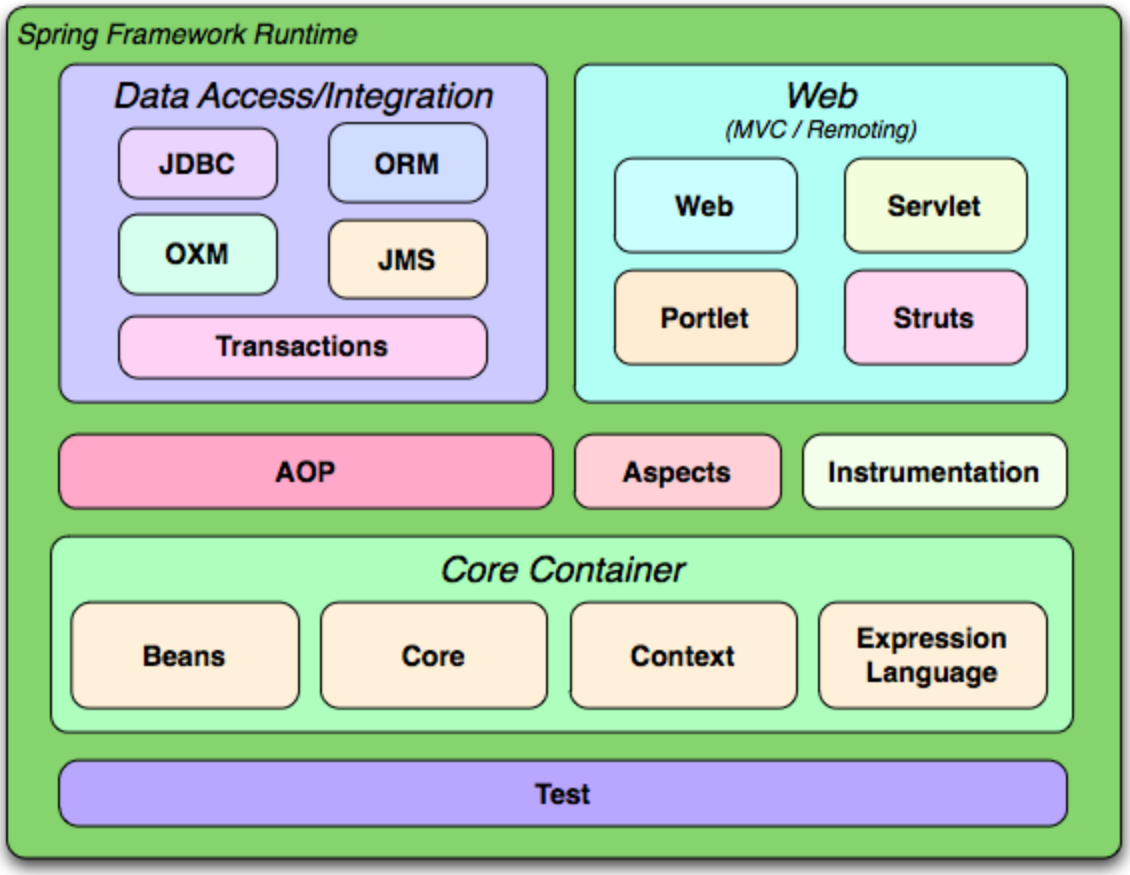
In conclusion, it made overall business sense to develop the custom solution that is easily customizable to add more features and is completely controlled and managed internally, rather than depend on external entities to manage and support in house project allocation of resources. Moreover, it did not make business sense to divulge and share internal company project details with online software and resource management.

Project Rationale

The solution was to develop a web based application to manage resource allocation to projects and generate reports based on the same. This was the main business requirement and primary scope of the project. The technology used for the project needed to allow for easy plug and play for custom components and addition of new features like email integration or WAP enabling the application and so on. The application was developed using Model View Controller (MVC) architecture. There were several advantages to using this architecture for web development. The concept involves the following:

- Clear separation of presentation, business logic and data model.
- Flexibility in developing components in isolation. For example, because the view doesn't care about the underlying model, supporting multiple file formats is easier: just add a model subclass for each.
- Parallel development. Developers can develop the different layers separately and integrate as and when components are completed.
- Ability to use experts for each area. Data model experts can focus on the data layer, programming experts can focus on business logic while user interface experts can focus on presentation.

To facilitate the use of MVC architecture, we used Spring MVC framework that is integrated with Hibernate Persistence layer. Spring brought several advantages to the table. Firstly, it is an open source development framework and ABC Company requires that we stay away from proprietary technologies. Secondly, because of the Spring integration, applications can leverage all the features of spring framework. For example POJO style serve interfaces, IoC, Aspects, Remoting etc. Spring framework provided all these features out-of-box. And a good thing about Spring framework is that it's really easy to plug any custom component into the framework.



Spring Overview. Digital image.*Spring IO*. N.p., n.d. Web. 13 Apr. 2015.

With spring POJOs, IoC and Remoting it was very easy to develop services layer. Otherwise, we would have needed to implement EJB Session Bean (stateful or stateless) or depended on the frameworks like JSF, Struts or Servlets to implement the business logic which is not ideal. Another advantage of Spring is that it provides template for managing sessions, transaction across the application. Without the hibernate template, applications need to manage these on their own. All the custom business logic was implemented with simple POJOs and managed by Spring's IoC container. Additional services include support for sending email and validation that is independent of the web layer, which let us choose where to execute validation rules. So this framework allowed for implementation of additional features at a later date as well.

Hibernate over JDBC brought several advantages as well. Here are some of them that are especially important in easing development:

- Hibernate supports Inheritance, Associations, Collections.
- In hibernate if we save the derived class object, then its base class object will also be stored into the database, it means hibernate supporting inheritance
- Hibernate supports relationships like One-To-Many,One-To-One, Many-To-Many-to-Many, Many-To-One. This will also supports collections like List,Set,Map (Only new collections)
- In jdbc all exceptions are checked exceptions, so we must write code in try, catch and throws, but in hibernate we only have Un-checked exceptions, so no need to write try, catch, or no need to write throws. Actually in hibernate we have the translator which converts checked to Un-checked.

- Hibernate has capability to generate primary keys automatically while we are storing the records into database

- Hibernate has its own query language, i.e hibernate query language which is database independent. So if we change the database, then also our application will works as HQL is database independent. HQL contains database independent commands

- While we are inserting any record, if we don't have any particular table in the database, JDBC will rises an error like "View not exist", and throws exception, but in case of hibernate, if it not found any table in the database this will create the table for us ;)

- Hibernate supports caching mechanism by this, the number of round trips between an application and the database will be reduced, by using this caching technique an application performance will be increased automatically.

- Hibernate supports annotations, apart from XML

- Hibernate provided Dialect classes, so we no need to write sql queries in hibernate, instead we use the methods provided by that API.

- Getting pagination in hibernate is quite simple.

Finally in the development architecture we had MySql Database as the relational database management system. Although there are skeptics that would suggest that MySql has several limitations and that it does not scale very well, it is still one of the more popular open source database implementations. Even though it was purchased by Oracle, the company has not limited the use of MySql and the source is still available openly. Some of the advantages of MySql include its ease of use and is relatively inexpensive when compared with other

implementations. It also supports Novell clusters, so failover capabilities can be added by making the database instance available on a novell cluster. It comes with a development tool called MySql Workbench to facilitate GUI development of database entities and their relationships.

In conclusion, the use of Spring MVC, Hibernate and MySql was an apt set of technologies to develop this web based application. In future if the company intends to have its own developers add features to the product, then the learning curve would be remarkably lesser compared to other technologies. The entire implementation was on the Java development stack that is an industry standard for web applications.

System Analysis

The process of employee assignment to projects was quite archaic as mentioned previously. The system was essentially clerical in the sense that it involves merely sending an email to respective parties i.e. project manager and employee along with other stakeholder, informing them that the assignment has taken place. The project management office needed a more efficient way of allocating resources to projects and also a better reporting mechanism.

This report has already explained the rationale behind developing a web based system. Since the earlier system was purely manual mode of assignment, all that needed to be assessed, was the business needs. As stated earlier, the main requirement was to actually develop a web based application because of a lack of any automated system currently being available. The methodology as per the proposal document was to follow an industry wide standard of SDLC i.e. Initiating, Planning, Implementing, Monitoring and Control, Closure. Following this standard

ensured not only that the application is delivered on time but it will also ensure quality of the product.

During initiation of the project, meetings were held with users to understand their exact needs. During these meetings the scope document was written and approval was taken for the project from upper management. The meetings involved several brainstorming activities and negotiating the scope based on time and budget constraints. Initiation resulted in approved project charter and assignment of project manager.

This was followed by project planning during which the scope document was finalized and then broken down into individual tasks that form the various milestones in the project. It was also decided to plan phases to ensure there is tangible progress in terms of percentage completion. The three phases included Model completion, Controller Completion and View Completion. This was in line with MVC architecture. Resources were also allocated to each of the tasks in preparation for implementation. In terms of personnel, in this particular case, there was only one developer assigned to developing the entire application.

Once the resources were allocated, the development moved into the implementation phase. The project moved through this development phase with close monitoring by the project manager. This is where new requirements went through change control process to add additional budget and/or time to the project for additional implementation.

Finally, the project was delivered during the closure phase, during which a formal acceptance of the project is received. The upper management granted the final approval and release of resources.

As part of [redacted] there was also a possibility of maintenance and upgrade cycle to add new features but those activities were outside the scope of this proposal. In conclusion, following this methodology ensured there was tangible progress at every stage and made monitoring easier, so that any deviations could be addressed efficiently to avoid catastrophic failures.

Goals and Objectives

There was one main goal for this project. That goal was to deliver a web based application that will allow creation of employee to project relationship and provide reporting capabilities for current and historical data of employee project assignment. This goal was met using the following objectives:

1. Create a data model or entity-relationship diagram. See the data model in Appendix A. This objective was completed by creation of the ER Diagram.

2. Create the actual relational database in MySql Database. This objective was met by creating the SQL scripts for table creation. See Appendix B in this document for the scripts.

3. Create POJOs corresponding to each database entity. This objective was also met during the implementation processs. The compiled java classes are part of the deployable WAR file.

4. Create Hibernate XMLs for each POJO and Hibernate Configuration file. This objective was met during implementation and created a type mapping between the database columns and java class fields.

5. Create Data Access Objects interfaces with implementation for each entity. This objective was met by creating the DAOs with business logic for CRUD operations.

6. Create the Spring Controller and write Business Logic. This objective was met in the Controller phase during which the business flow of the application was developed using RequestMapping annotations.

7. Add the a DAO bean reference for each entity in the servlet-context XML file. This objective was met to facilitate autowiring of DAO implementations in the Spring Controller which in turn facilitated dependency injection.

8. Create a CSS File for UI consistency. This was a newly added objective to create UI consistency. This was completed during implementation phase.

9. Create JSP files that represent the UI. This objective was met during the creation of View phase of the project. This is where the user interface was implemented.

10. Build and Deploy the application and test core functionality. Finally just before closure this objective was met by deploying and testing the application functionality and receiving the approval from stakeholders.

11. Meet the deadlines and project milestones to stay within the budget. This objective was met partially. The reason we say that is because the project went an extra 8 days on implementation, either due to some implementation challenges or due to new requirements that were added. So partially met would be correct for this objective.

Each object represented a critical part of project development life cycle. The success of the project was dependent on successful completion of each of the objectives.

Creating an entity relationship diagram was the key to a successful implementation. The ER diagram was used to visually represent database entity relationships. Developing an ER diagram had several advantages:

1. Straightforward relation representation: Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straightforward.

2. Easy conversion for E-R to other data model: Conversion from E-R diagram to a network or hierarchical data model can· easily be accomplished.

3. Graphical representation for better understanding: An E-R model gives graphical and diagrammatic representation of various entities, its attributes and relationships between entities. This is turn helps in the clear understanding of the data structure and in minimizing redundancy and other problems.

The next step after creating the ER Diagram, was to create the physical database tables using the Data Model. There are several things that needed to be kept in mind here. The tables need to be normalized to avoid redundant columns. The data model more or less takes care of these aspects but it is good to verify that the appropriate keys have been created. For example, one department can have many employees but one employee can belong to only one department. So this would form a one to many relationship between the department and employees. In another case, one employee can belong to one active project but they could have taken part in earlier projects. So in the database terms, one employee can belong to many projects and at the same time, one project can have several employees. This would mean that the data model should show an intersection table that has two foreign keys, one for employee and other for project.

Verifying the actual tables have the correct relationships is paramount step before moving on to the next step of creating POJOs.

POJOs, stand for plain old java objects. These objects are essentially as the name suggests a direct map of the database table. The columns in the database table form the fields in the java object. It is important to ensure that the datatype of the fields matches the datatype of the column. Otherwise there will be errors while storing the data. The POJOs also have getters and setters for each of the fields. These are the objects that are used in the middle tier as entity representatives and hence their implementation is necessary before moving on to the next step.

Each POJO has a corresponding Hibernate XML file. This xml file essentially creates a mapping between the java class name and the database table name along with the mapping of the class field name and the database field name. We can think of this as glue that binds the java objects to the entities. Finally to complete the database layer this xml file is entered as a mapping entry in the hibernate configuration as a resource file for the spring framework.

Having completed the Database Layer, it was time to create the Data Access Objects (DAO). So far the project implementation had completed the POJOs and Hibernate mappings. That defines the “what” of the implementation. To define the “how” of the implementation, we needed to create the DAOs. These are java interfaces that are autowired in the Spring Controller by way of dependency injection. Creating interfaces is what makes the plug and play feature of spring. The controller layer does not have any issues about “how” the database is accessed. If the implementation were to change sometime in the future, it would not impact the rest of the project. The implementation classes of DAOs have the actual CRUD (Create, Update, Delete) or transactional code. This is where Hibernate Query Language (HQL) was used to fetch the

records. Again, as mentioned earlier, HQL is database independent, so it is possible to use a different database in the future, if so required, without impacting the rest of the project implementation.

After creating the DAOs, it was time to create the actual business flow calls using the Spring controller. To facilitate dependency injection of DAOs in the Spring Controller the servlet-context xml file had to have an entry of the DAO implementation as a bean tag entry. Having that entry enabled the autowiring of the DAO in the spring controller. So even though the interface reference is used in the controller, it receives an object of the interface implementation class at runtime, so that it can make the database calls. The spring controller also maps each method call using request mapping. Based on the web context, a particular method is called to process the request. Spring controller business logic methods return the "ModelAndView" object that contains all the session variables that can be accessed in the user interface layer to display the data to the end user.

Creation of JSPs was done in the end to finally display the session data to the end user. This is where the user interface elements come into picture. Data presentation in appropriate format that conformed to the ABC company standards was important. The reason this was important is because the end users only see the final user interface. Any glitches here would signify a failure of the project.

Finally it was important to deploy and test the application in the web server. Here we had to ensure that the scope of the project had been met in terms of the main functionality. The following was the success criteria:

1. Can the user create new employees?

2. Can the user get a listing of projects?
3. Can the user select a project and get a listing of employees?
4. Can the user get a listing of all employees?
5. Can the user select an employee and get a listing of all the projects?
6. Can the user assign an employee to a project?

It was not just the completion of all this functionality that determined successful completion of the project. The time taken to completely deliver a deployable application was extremely relevant. Any overshooting of time, escalated the budget constraints for this project and required exceptions from upper management. To ensure confidence of upper management, the project had to be completed within the time frame and deliver all the functionality mentioned in the success criteria. In view of this, change control had to be established to avoid scope creep. Any deviation from these requirements required approvals from upper management.

Project Deliverables

The main project deliverable was the WAR file that could be deployed on a web server. This war file contains all the dependencies in the lib directory and so was self-sufficient in its entirety. As per the stages described in the goals and objectives section, each stage was comprised of a deliverable that eventually lead to the production of the WAR file. Wherever possible, documentation was made available so as to assist future developers with implementing additional functionality.

Following the project management development lifecycle, the first deliverable of the project was a document that defines the scope. This document mainly consisted of the success criteria defined earlier. The scope document functioned as a baseline for change control.

Moving on to the implementation phase, a printable ER Diagram was made available for records. This showed the relationships between the database entities as described earlier. This can be in the form of a pdf file. The ER Diagram consists of four Entities called Dept, Emp, Project and Project_Emps. In addition to the ER Diagram, a set of table creation sql scripts were delivered as well, to allow for database schema and table creation.

Next is a java package containing four POJOs and corresponding Hibernate XML and Hibernate Configuration Files. This is followed by a package containing the four DAO interfaces and their implementation classes.

Spring Controller is delivered next with all the business or process flow logic. This is where the maximum amount of development time was spent. An efficient process flow is the key to a high performing web application. Special care was taken to significantly test this flow before delivery. The spring controller consists of one class file and one xml file that is the servlet-context xml file.

Next set of deliverables are the user interface elements. In this case these are the JSP files. Most of the JSPs that list the database rows as HTML rows had links against each row to edit or delete. For example a list of departments showed all departments in tabular format and against each row there were links to either edit the department information or delete the department all together and also get a list of all employees. There is also a link at the top of the page for creating a new department. Clicking the link to new or edit department takes the user to

another JSP that provides the user with form input. This means that, at the very least, we delivered two JSPs for each entity. Since we have 4 entities, we delivered nine JSPs including one home page jsp.

Finally, using all the project artifacts, other than the documentation as such, a WAR file was developed and delivered that can be deployed on tomcat web server. In addition, a deployment document with a step by step guide is to be delivered.

Project Timelines

Project Deliverable or Milestone	Planned Duration	Actual Duration	Start Date	End Date
	Initiating			
Onsite Meetings	2 days	2 days	03/02/15	03/03/15
Discussions with stakeholders	3 days	3 days	03/04/15	03/06/15
	Planning			
Prepare Scope Document	1 day	1 day	03/09/15	03/09/15

	Implement			
Develop ER Diagram	1 day	1 day	03/10/15	03/10/15
Develop Database Tables	1 day	1 day	03/11/15	03/11/15
Develop POJOs and Hibernate XML	1 day	1 day	03/12/15	03/12/15
Develop DAO and Implementation	5 days	6 days	03/13/15	03/19/15
Develop Spring Controlled	10 days	12 days	03/20/15	04/06/15
Develop CSS	0 days	3 days	04/07/15	04/09/15
Develop JSP	4 days	5 days	04/10/15	04/16/15

Presentation				
Create WAR and Deploy/Test	1 day	2 days	04/17/15	04/20/15
	Monitoring and Control			
Change Control	23 days	31 days	03/10/15	04/20/15
	Closure			
Prepare Deployment Document	1 day	1 day	04/21/15	04/21/15
Total	30 days	38 days	03/02/15	04/21/15

This timeline was essentially representative of a single resource working on the project. The times given are the reasonable amount of time required to deliver those milestones. Since the project was on tight timeline, there was not much contingency added to each activity. Although it was possible to accelerate the work and finish the same in $3/4^{th}$ the amount of time given by hiring an additional resource, it was not done in light of the tight budget. Overall the estimated amount of time required from start to finish was approximately 38 business days.

Project Development

This project was considered successful largely due to the fact that it did not deviate tremendously from the initially planned scope due to the tight change control process. The success is attributed mainly to the fact that the work breakdown structure matched the technology being used. This helped to plan the deliverable schedule based on the layers of technology. Secondly, a strong change control process was implemented to prevent scope creep. No change was implemented unless the budget and timeline changes were approved. Change control is often a neglected activity in projects but as seen in this project, it is paramount to the success of the project. While we deemed this project as a success, it does not mean that there were no challenges along the way. One of the main challenges were the slight learning curve involved with new technology. While using frameworks eases subsequent development, it does have a slightly larger initial learning curve. This is because we had to learn the intricacies of the framework and how the components worked together. Here is a list of problems that were encountered:

Problems Encountered	
Problem	Solution
Incorporating a many to one hibernate relationship caused a Column defined twice error. SEVERE: Servlet.service() for servlet [appServlet] in context with path [/SpringMVCHibernateXML] threw exception	The reason for this error was that Hibernate tries to insert the foreign key twice, one for the actual column and one for the relation itself. This is not allowed with mySQL and so it threw an error while processing. The

[Request processing failed; nested exception is org.hibernate.exception.SQLGrammarException: Column 'DEPTNO' specified twice] with root cause
 	
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Column 'DEPTNO' specified twice
	solution was to define the relation with insert=false and update=false attributes as follows:
	
	`<many-to-one name="project" class="com.wgu.capstone.model.Project" fetch="select" lazy="false" insert="false" update="false">`
	`<column name="project_id" not-null="true" />`
	`</many-to-one>`

Date Formatting for the UI was another problem faced during development. The form input would be a string but the field type was java.util.Date So there was a type conversion error in spring as follows:
 	
Field error in object 'projectEmps' on field 'startdate': rejected value [2012-10-01]; codes [typeMismatch.projectEmps.startdate,typeMismatch.startdate,typeMismatch.java.util.Date,typeMismatch]; arguments [org.springframework.context.support.Default
	This was resolved by defining the DateTimeFormat annotation for the date fields and using the fmt tag library for date formatting.
	
	`@DateTimeFormat(pattern="yyyy-MM-dd")`
	`public void setStartdate(Date startdate) {`
	`this.startdate = startdate;`
	`}`
	`<fmt:formatDate value="${projectemps.startdate}"`

MessageSourceResolvable: codes [projectEmps.startdate,startdate]; arguments []; default message [startdate]]; default message [Failed to convert property value of type 'java.lang.String' to required type 'java.util.Date' for property 'startdate'; nested exception is org.springframework.core.convert.ConversionFailedException: Failed to convert from type java.lang.String to type java.util.Date for value '2012-10-01'; nested exception is java.lang.IllegalArgumentException]	type="both" pattern="dd-MMM-yyyy" />
There was a problem with HQL – Hibernate Query language where it was complaining that the specified table was not mapped: org.hibernate.hql.internal.ast.QuerySyntaxException: Project_Emps is not mapped [from Project_Emps where id=2]	The reason for this error was that in case of HQL, it is the class data type that is used in the query and not the actual database table name. Changing the query to class data type resolved this issue.

These were the main errors that were encountered that took the maximum amount of research and time to resolve. This nearly added 3 days to the implementation as we researched to figure out the solutions to these issues. In addition to the above, there were some additional

requirements that were not exactly unanticipated but still were not part of the initial scope and in fact they were even mentioned as possible features that could be added later:

Additional Requirements
1. Deliver a CSS file that will provide user interface consistency. In the initial requirements it was mentioned that web application will conform to the UI standards of ABC Company but there was no mention that it was required to deliver or even create a CSS file. The project manager was under the assumption that this was understood. But we had to negotiate and add additional time for creation and delivery of this CSS file. Since the requirement was legitimate, upper management approved the increase of 3 days to the original budget.
2. Adding a Date Picker to the date fields in the UI. Initially there was no mention in the requirements that a date picker is required. But when the users got a taste of the prototype, they mentioned that it was not practical to manually enter a date. So to incorporate this functionality we added one more day to the implementation cycle. Only one additional day was required because there were not many date fields and implementation of one fields can easily be duplicated to the other.
3. Email functionality was discussed initially as possible future enhancement but after some more discussions during the change control process, the stakeholders realized this would be part of the core requirements in this first implementation. The reason for this requirement is to make sure that the employee receives an email notification when they are assigned or removed from the project. Even the employee manager should be notified. This functionality added another 3 days to the implementation.

As a result of these additional requirement and problems faced during implementation, nearly a whole week was added to the project causing a slight delay in delivery but management was fine with it because there was not a significant cost increase. Moreover, they had a delay in the launch of new projects which in turn gave additional buffer time for this implementation.

During the project implementation, as mentioned previously, we had a deviation of 8 days from originally planned duration. There were several reasons for these changes as highlighted in the table below:

Reasons for Change
1. One of the main reasons for deviation from original plan was the incorporation of CSS implementation. Due to the less number of UI elements the initial estimate did not incorporate creation of reusable UI element to form a consistent look and feel. It was originally perceived to implement the look and feel aspects directly in the nine JSPs that were part of the project. But the stakeholders realized that this project has a potential of being a prototype for future projects in the company. So it was decided to incorporate CSS into the project, so that it can be used in other projects in the future. The CSS file is included for reference as a deliverable in Appendix C.
2. Date Picker was another functionality that was not foreseen during the scoping of requirements. This became a crucial discussion point mid-way through the implementation during the review meetings as part of the change control process. There is only one UI form in the application that has 2 date fields and the stakeholders wanted consistently in the way the users would enter date values. Having a date picker was the perfect solution. This should have

been part of the original requirements but unfortunately was overlooked and as a result found its way in the change requests.
3. Finally the addition of notification capability to the application was another change that was added to the application. This requirement was actually not overlooked and it was part of the original discussion but due to budgeting reasons was not incorporated initially. Since the company projects were delayed, there was a buffer time frame that was available for implementation of additional features, due to which, management was able to allocate an additional budget to implement this functionality.

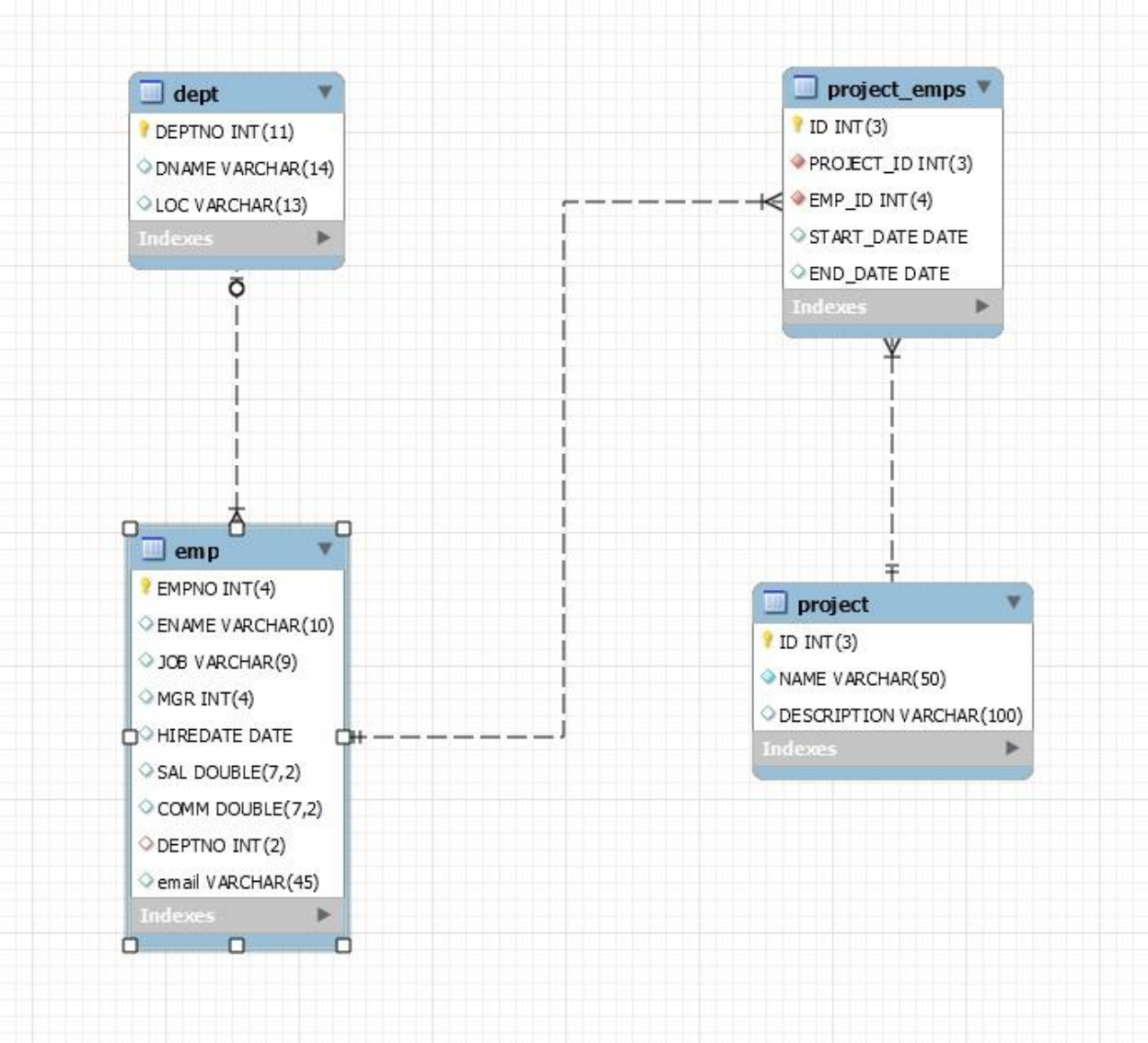
The potential effects of this project were quite significant from companies growth perspective. This project implementation using the MVC architecture with Spring and Hibernate has opened doors for ABC Company to perform similar implementation for other requirements like allocation of assets to projects, conference room management system across timezones and even reorganizing the company intranet portal using this technology. Due to the plug and play feature of the architecture, the company is even thinking of creating a WAP interface of the application for access using mobile devices. It is possible to implement a WAP controller and use XSLT to transform the output to suit the mobile device screens. All this can be done without changing the model layer at all.

Conclusion

In the end, the project was deemed successful as it achieved not only the requirements but also implemented the extended scope within the approved time period. Although there was a deviation of 8 days, it does not constitute a failure since the additional requirements were

erroneously assumed to be part of the original scope. The change control process ultimately recognized that fallibility and required an official sign off for additional time and budged. Eventually the stakeholders were satisfied with the web application and how it made their process of resource allocation much more efficient that the archaic process they were following. Moreover the project opened doors for the company for future enhancement of other systems and even enabling mobile implementations of this current project. Whenever a project leaves doors open for future enhancements and easy plug and play of newer components, it is deemed as a success as it incorporates the companies growth plans in its fold and constitutes several new potential effects for the company to look forward to.

Appendix A – ER Diagram



Appendix B – Database Creation Scripts

1. CREATE TABLE IF NOT EXISTS `abc`.`dept` (

`DEPTNO` INT(11) NOT NULL AUTO_INCREMENT,

`DNAME` VARCHAR(14) NULL DEFAULT NULL,

```
  `LOC` VARCHAR(13) NULL DEFAULT NULL,
  PRIMARY KEY (`DEPTNO`),
  UNIQUE INDEX `PK_DEPT` (`DEPTNO` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 41
DEFAULT CHARACTER SET = latin1
```

2.

```
CREATE TABLE IF NOT EXISTS `abc`.`emp` (
  `EMPNO` INT(4) NOT NULL AUTO_INCREMENT,
  `ENAME` VARCHAR(10) NULL DEFAULT NULL,
  `JOB` VARCHAR(9) NULL DEFAULT NULL,
  `MGR` INT(4) NULL DEFAULT NULL,
  `HIREDATE` DATE NULL DEFAULT NULL,
  `SAL` DOUBLE(7,2) NULL DEFAULT NULL,
  `COMM` DOUBLE(7,2) NULL DEFAULT NULL,
  `DEPTNO` INT(2) NULL DEFAULT NULL,
  `email` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`EMPNO`),
```

```
  UNIQUE INDEX `PK_EMP` (`EMPNO` ASC),
  INDEX `FK_DEPTNO` (`DEPTNO` ASC),
  CONSTRAINT `FK_DEPTNO`
    FOREIGN KEY (`DEPTNO`)
    REFERENCES `test`.`dept` (`DEPTNO`))
ENGINE = InnoDB
AUTO_INCREMENT = 7936
DEFAULT CHARACTER SET = latin1
```

3.
```
CREATE TABLE IF NOT EXISTS `abc`.`project` (
  `ID` INT(3) NOT NULL AUTO_INCREMENT,
  `NAME` VARCHAR(50) NOT NULL,
  `DESCRIPTION` VARCHAR(100) NULL DEFAULT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB
AUTO_INCREMENT = 2
DEFAULT CHARACTER SET = latin1
```

4.
```
CREATE TABLE IF NOT EXISTS `abc`.`project_emps` (
```

```
 `ID` INT(3) NOT NULL AUTO_INCREMENT,

 `PROJECT_ID` INT(3) NOT NULL,

 `EMP_ID` INT(4) NOT NULL,

 `START_DATE` DATE NULL DEFAULT NULL,

 `END_DATE` DATE NULL DEFAULT NULL,

 PRIMARY KEY (`ID`),

 UNIQUE INDEX `PROJECT_EMPS_UK1` (`PROJECT_ID` ASC, `EMP_ID` ASC),

 INDEX `PROJECT_EMPS_PROJECT_FK2` (`EMP_ID` ASC),

 CONSTRAINT `PROJECT_EMPS_PROJECT_FK1`

   FOREIGN KEY (`PROJECT_ID`)

   REFERENCES `test`.`project` (`ID`)

   ON DELETE CASCADE,

 CONSTRAINT `PROJECT_EMPS_PROJECT_FK2`

   FOREIGN KEY (`EMP_ID`)

   REFERENCES `test`.`emp` (`EMPNO`)

   ON DELETE CASCADE)

ENGINE = InnoDB
```

DEFAULT CHARACTER SET = latin1

Appendix C – CSS Implementation

```
body

{

  background-color: White;

  color: black;

  font-family: Arial, Helvetica, sans-serif;

  text-align:center;

}

h1

{

  color: navy;

  font-family: Arial, Helvetica, sans-serif;

  font-size: x-large;

  font-weight: bolder;

  text-transform: capitalize;

}

h2

{

  color: navy;
```

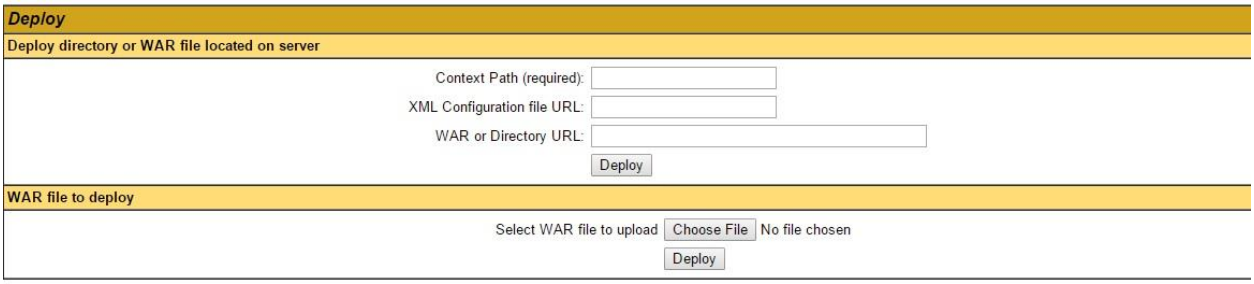
```
  font-family: Arial, Helvetica, sans-serif;
  font-size: large;
  font-weight: bolder;
  text-transform: capitalize;
}
h3
{
  color: navy;
  font-family: Arial, Helvetica, sans-serif;
  font-size: large;
  font-weight: lighter;
  text-transform: capitalize;
}
.caption{
    background: red;
    color: white;
    font-weight: bold;
    font-size: xx-large;
}
table{
```

```
    text-align: center;
    border: none;
}
table.center {
    margin-left:auto;
    margin-right:auto;
}
th{
    height: 50px;
}
tr:nth-child(even) {
    background-color: #D3D4D5;
}
tr:nth-of-type(even) {
    background-color: #D3D4D5;
}
pre.left{
    text-align: left;
}
```

Appendix D – Deployment Steps

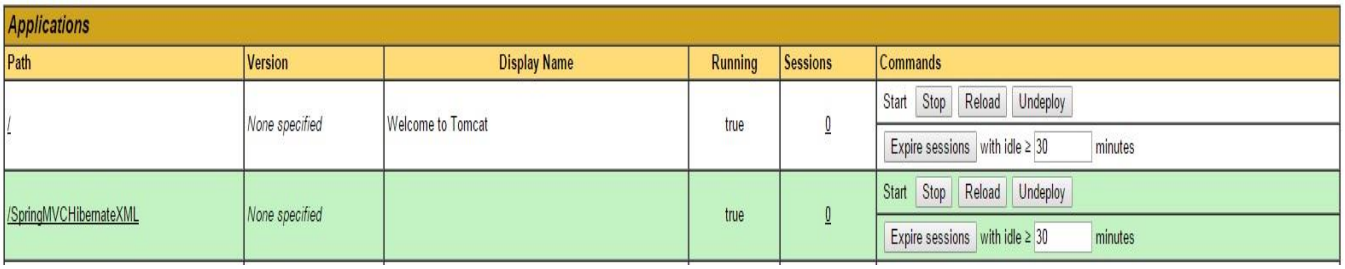
1. Open the Tomcat Manager

2. Go to the Deploy section and click on “choose file” button to pick the war file to deploy.



3. After picking the war file, click on the Deploy button.

4. The screen should refresh. Make sure the deployment status for ‘Running’ shows ‘true’ as follows:



5. Test the application by accessing http://<host>:<port>/SpringMVCHibernateXML/

References

Atchison, S., & Kennemer, B. (2011). Using Microsoft Project 2010. Retrieved from http://www.cob.unt.edu/itds/faculty/becker/BCIS3610/Syllabus/Atchison-MS_Project_2010_Textbook.pdf.

Cohn, C. (2014, Sept). Build vs. Buy: How to Know When You Should Build Custom Software Over Canned Solutions. Retrieved from http://www.forbes.com/sites/chuckcohn/2014/09/15/build-vs-buy-how-to-know-when-you-should-build-custom-software-over-canned-solutions/

Hub Planner FAQ (2015). Hub Planner. HubPlanner, Stockholm, Sweden. Retrieved from http://hubplanner.com/features/

Spring Overview. Digital image.*Spring IO*. N.p., n.d. Web. 13 Apr. 2015. Retrieved from http://docs.spring.io/spring-framework/docs/3.0.x/reference/overview.html

Yadav, R., & Dhabade, N. (2013). Human Resource Planning and Audit – a Case Study of HEG Limited. Retrieved from http://www.scipress.com/ILSHS.16.44.pdf

Watt, A. (2014, August). Project Management. Retrieved from http://opentextbc.ca/projectmanagement/