PhD Thesis Monitoring System

Dhruv Patel - 201501452@daiict.ac.in
Chirag Chandnani - 201501145@daiict.ac.in
DA-IICT
Supervisor: Prof. P M Jat

***Abstract*—This paper contains description of our project PhD Thesis Monitoring System, on which we worked on during our internship period. We have developed the web application for PhD Thesis Monitoring System project which supports all work flows required by student,supervisor, dean and reviewer in order to successfully monitor a thesis under review and it is developed according to the software specification requirement documentation. This web application project is developed using tools and technologies used in industries. We have mentioned all the tools and technologies and implementation details that we have done during our BTP period, in this report.**

***Index Terms*—Web development, Struts2, Hibernate, AJAX**

I. Introduction

The project PhD Thesis Monitoring System fully supports all work flows required by student, supervisor, dean and reviewer in order to successfully monitor a thesis under review. The current system requires to handle all the task manually like student have to submit thesis via mail to supervisor and then supervisor forwards it to the dean AP with the list of reviewers and then dean AP selects reviewer and mail them for thesis review and then reviewer accepts it and sends the review to the dean AP via mail. All this tasks are done through mail which is very inconvenient.

So the PhD Thesis Monitoring System makes the tasks digitally on one platform including sending mails. On this system student can submit their synopsis and thesis, supervisor can accept the thesis and select reviewers for that thesis, dean AP finalize the reviewers from selected reviewers for thesis review and send mail to reviewer from this system, reviewer will review the thesis and submit the review on this system and finally dean AP can see the review submitted by reviewer.

II. Methodology and Approach

A. WORK FLOW

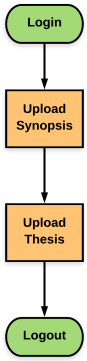

Fig. 1. Student flowchart.

1) Student: Student can upload synopsis and after that student can upload the thesis on this system.

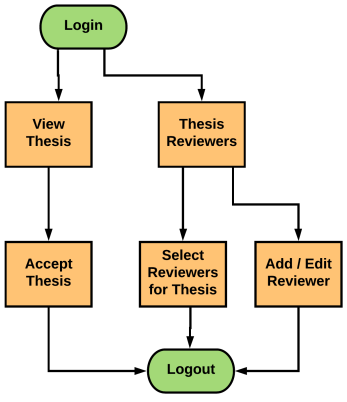

Fig. 2. Supervisor flowchart.

2) Supervisor: Supervisor can view thesis of students under his/her mentorship. After viewing thesis supervisor can accept that thesis. After accepting the thesis supervisor can select the reviewers for that thesis. Also

supervisor can add new reviewer or edit the reviewer details.

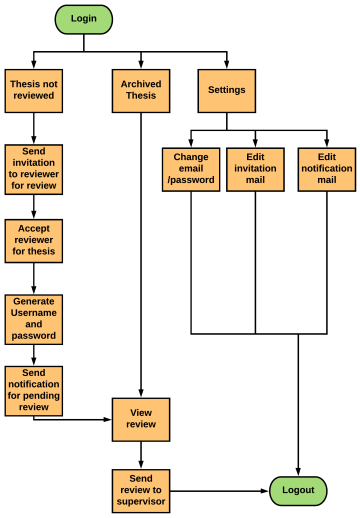

Fig. 3. Dean AP flowchart.

3) Dean AP: Dean can view all the thesis whose reviewers are selected and dean can send the invitation to reviewers from that list. Once reviewer accept the thesis for review then dean can also accept the reviewer. Then dean can generate username and password for that reviewer. If review is pending then dean can also send notification to reviewer for remainder. Once reviewer submits review dean can view the review and also he can send that review to supervisor. Once the thesis is defended dean can send the thesis to archives. Also dean can change email and password for sending mail, default invitation subject and body and default notification subject and body.

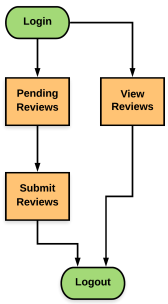

Fig. 4. Reviewer flowchart.

4) Reviewer: Reviewer can see the thesis whose review is pending and he can submit the review for thesis. Also reviewer can see the submitted reviews.

B. CLASS DIAGRAM

Below is the class diagram for the system. We have made use of 7 classes in our project. Class diagram contains all classes, their attributes and the way these classes are related.

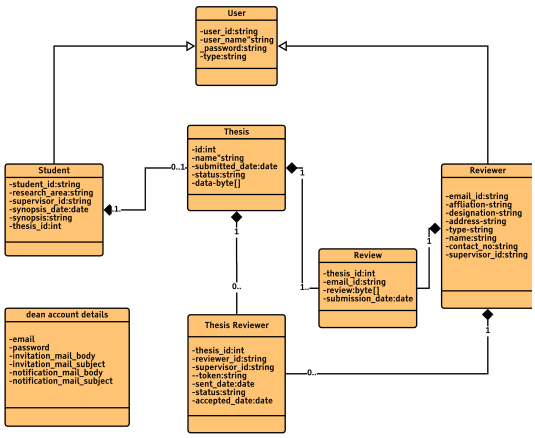

Fig. 5. Class Diagram.

C. IMPLEMENTATION AND TECHNOLOGIES USED

For the implementation of our project, we have used a MVC framework in Java named struts 2. We have used various tools and technologies for our project each serving a different purpose during the course of our project. Below i have described the implementation and where a particular technology was used.

*1) **MVC (Model–View–Controller)**:* We used a MVC framework(Struts2) for developing our web application. MVC is widely used as the programmer can isolate the application logic from user interface. Controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response. [7] It can be graphically represented as follows.

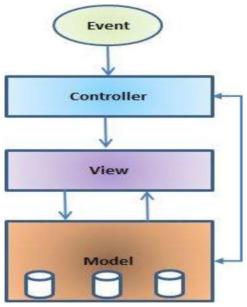

Fig. 6. MVC framework.

*2) **Struts 2**:* Struts2 is a web application framework which implements the MVC design pattern. It supports AJAX and JSON through plugins.
For loading a web page, it goes through the following steps:

- User sends a HTTP request for requesting some resource.
- Filter Dispatcher servlet in struts looks at the request and determines appropriate response.
- Configured interceptor functionalities applies such as validation, file upload etc.
- Selected action(a java class) is performed based on the configuration specified in struts.xml and the action returns a string(result) which is mapped to a view(jsp page in our case)in struts.xml.
- Configured interceptors do post processing.
- The result which is configured by the programmer is prepared by the view(a jsp page in our case) and the user sees the webpage.

The diagram below shows the process for loading a webpage in struts:

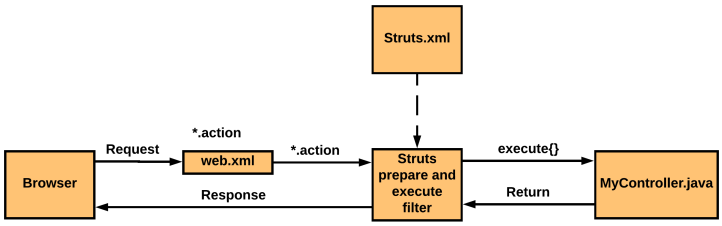

Fig. 7. Struts2 Flow.

This struts convention plugin enables zero-configuration for Struts2 application by enforcing the standard naming conventions for actions classes and jsp pages and enables the programmer to get rid of writing anything in struts.xml. In our program, we have used struts convention plugin which does automatic mapping between the actions(java code) and the views(jsp page) based on the convention naming patterns. The Convention plugin searches for actions in the java packages having the following words in their names: struts, struts2, action or actions Under the matching java packages (and sub-packages), the plugin looks for the classes that are either implementing the com.opensymphony.xwork2.Action interface or having their names end with Action. Now based on the result returned,the action class is automatically mapped to the jsp page which struts looks for WEB-INF/content directory by default. [1]

*3) **Hibernate**:* Hibernate is an open source Java persistence framework project which implements the Java persistence API. In Java persistence API, we define objects for each and every entity in the actual database and then there is a mapping between these real entities and classes in java so java can understand the database entities. This is called Object-Relational mapping(ORM). Hibernate performs powerful object-relational mapping and query databases using HQL and SQL. [6] It helps the developers by cutting down a lot of complexity by performing conversions between Java objects and SQL tuples behind the scene. We have used PostgreSQL which is widely used as our Database.

*4) **Maven**:* We used maven to manage use of various libraries and their dependencies. Maven is a project management tool that is based on POM (project object model). Project Object Model(POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project. Maven reads pom.xml file to accomplish its configuration and operations and to download the dependencies. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository. [8]

*5) **Interceptors**:* Interceptors allow for common functionality for actions to be implemented separately from the action as well as the framework. Many of the features provided in the Struts2 framework are implemented using interceptors. Examples include passing request params to action classes,making Servlet API request, response, session available to Action classes, validation etc. Most of them are defined in struts-default package and used in defaultStack interceptor stack. [5]
We have implemented a custom interceptor to deny access to URLs of the website to unauthorised users and also user of one type cannot access URLs of another type (Eg :Student cannot access any URLs without logging in and Student cannot access URLs of Supervisor). For applying this functionality using a interceptor is very useful as it provides cross cutting functionality and we don't have to write code in action class.
To achieve this, we have checked whether the userId attribute in session is empty or not. If its not empty then, whether the URL the user is trying to access is the URL

of that user only.

*6) **Javax Mail API**:* The use case of our project requires dean AP to send mails to reviewers within the website itself. To implement that, we have used javax mail library which uses SMTP protocol to send emails. We have also added the functionality that dean AP can set predefined mails for invitation and notification so that he doesnt have to type the mail subject and body again. For authentication, we have used imaps protocol.

*7) **AJAX**:* AJAX allows web pages to be updated asynchronously by communicating with a web server behind the scenes. It allows the web browser to reload part of a web page without the updating the whole page. AJAX uses XMLHttpRequest object to request data from web server.
In our project, we have used struts2 jQuery plugin which provides UI widgets and ajax functionality based on the javascript jQuery framework using a tag library. [2]
We have used this to create dialog boxes which load content from server like in deans account for send invitation dialog and send notification mail dialog and forget password dialog in login page. Also used while displaying thesis details in supervisors account and reviewers details in the same page itself.

*8) **Validation**:* We have implemented most of the validations in forms like login, uploading pdf and others using in-build struts2 annotations for validation. But for content which is loaded using ajax, we had to do validation using struts2-json plugin. In AJAX validation, the errors are returned as a json response to the client. The AJAX validation is performed by json validation interceptor. It is not found in the default stack so we need to define it explicitly. It is found in the json validation work flowStack.

III. CONCLUSION

In the course of our BTP, we got to learn lot of new things. Over the course of 3-4 months, we learned struts and various other technologies and implemented the project from scratch from the SRS we were given. We worked with database, backend and as well a bit of frontend. We have implemented the work flows of all the 4 users of the website namely: dean AP, student, supervisor and reviewer. Finally the website facilitates efficient monitoring of Phd thesis.

IV. ACKNOWLEDGMENT

The project would never have been completed without the continuous support and guidance of our mentor Prof. PM JAT. We would also to thank Dean AP for approving this project . We would also like to thank the people who made the SRS for this project because of which we were able to implement the project.

REFERENCES

[1] Struts2 convention plugin, https://struts.apache.org/plugins/convention/
[2] Struts 2 jquery plugin, https://github.com/struts-community-plugins/struts2-jquery/wiki
[3] Sending emails in struts2, https://www.tutorialspoint.com/struts_2/struts_sending_email.htm
[4] Struts 2 validation, https://www.concretepage.com/struts-2/struts-2-validation-annotation-example
[5] Building custom interceptor for login, http://sandeepbhardwaj.github.io/2010/12/01/struts2-with-login-interceptor.html
[6] Hibernate query language, https://www.tutorialspoint.com/hibernate/hibernate_query_language.htm
[7] MVC architecture, https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm
[8] Maven, https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/