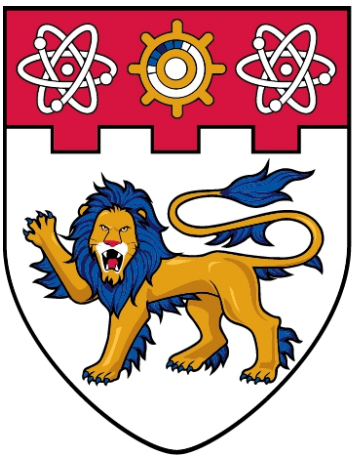


NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

CZ 3002 - ADVANCED SOFTWARE ENGINEERING ASSIGNMENT 2

GROUP 54

LEE ZHEN WEI (U1922813B)

RAJURAVI VISHAL RAJ (U1822268B)

WANG DAINI (U1721818F)

Overview of MVC structure

MVC is short for the *Model-View-Controller* software design pattern, it is a design pattern that categorizes software components into 3 groups the model, view and controller, to allow for better design especially popular for use in designing web applications where user interfaces are separated from program logic. MVC frameworks have been implemented in many different programming languages and some popular examples include Apache Struts, React, ASP.Net MVC and AngularJS, that help to implement MVC patterns and design good software architecture that is easy to debug, test and extend.

- The *Model* contains the main logic of the application and handles the data changes.
- The *View* is the visualization containing data and information, which is presented to the users.
- The *Controller* serves as a bridge between the model and the view by managing the data flow and reflecting it in the model or the view.

Architecture Diagram

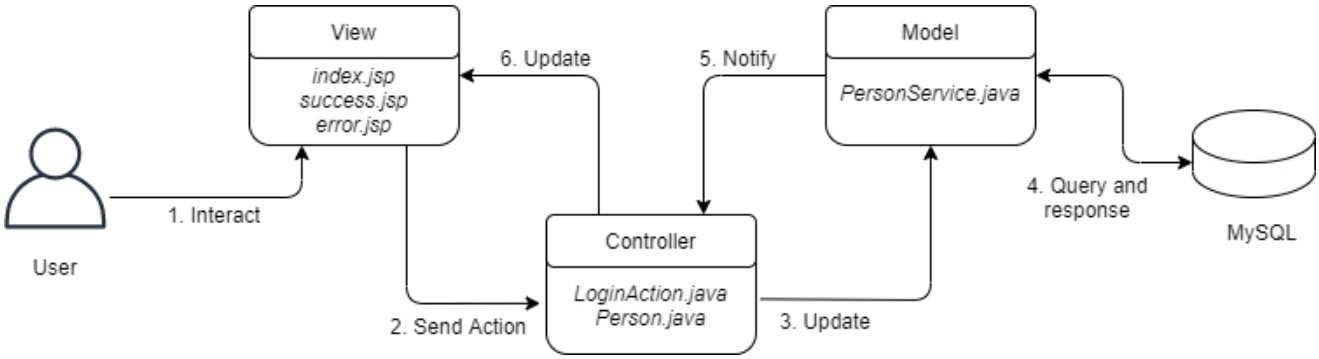


Figure 1. Architecture Diagram of Login Application

The *Model* component contains *PersonService.java* which contains the main logic of checking the input credentials with the database and returning the validation results. The *Controller* component contains *LoginAction.java* and *Person.java*. The *LoginAction.java* instantiates a *Person* class and passes the user input to the *Model* component for validation. It also receives results from the *Model* component. The *View* contains *index.jsp*, *success.jsp*, and *error.jsp*. The *index.jsp* shows the main page of the login page. The *success.jsp* shows the successful login page after the user keys in correct credentials. The *error.jsp* shows the error page when the inputs are incorrect.

Execution Flow

The user begins interaction by first interacting index.jsp, which is configured by the stuts.xml to control which view to serve and render to the user. The index.jsp holds the View component for the initial login page that consist of UI elements such as login form and submit buttons for the user to interact with.

Upon receiving input from the user, (when the submit button is pressed) control will be passed over to the LoginAction(Controller component) which will then query the PersonService (Model) for data such as list of login id and passwords, which will be retrieved by the model from the MySQL database before being returned to the LoginAction component, to perform a matching in the LoginAction (Controller component). The controller will then decide based on whether the login has matched, to serve the success.jsp to show a successful login attempt or error.jsp to show that the login has been unsuccessful.

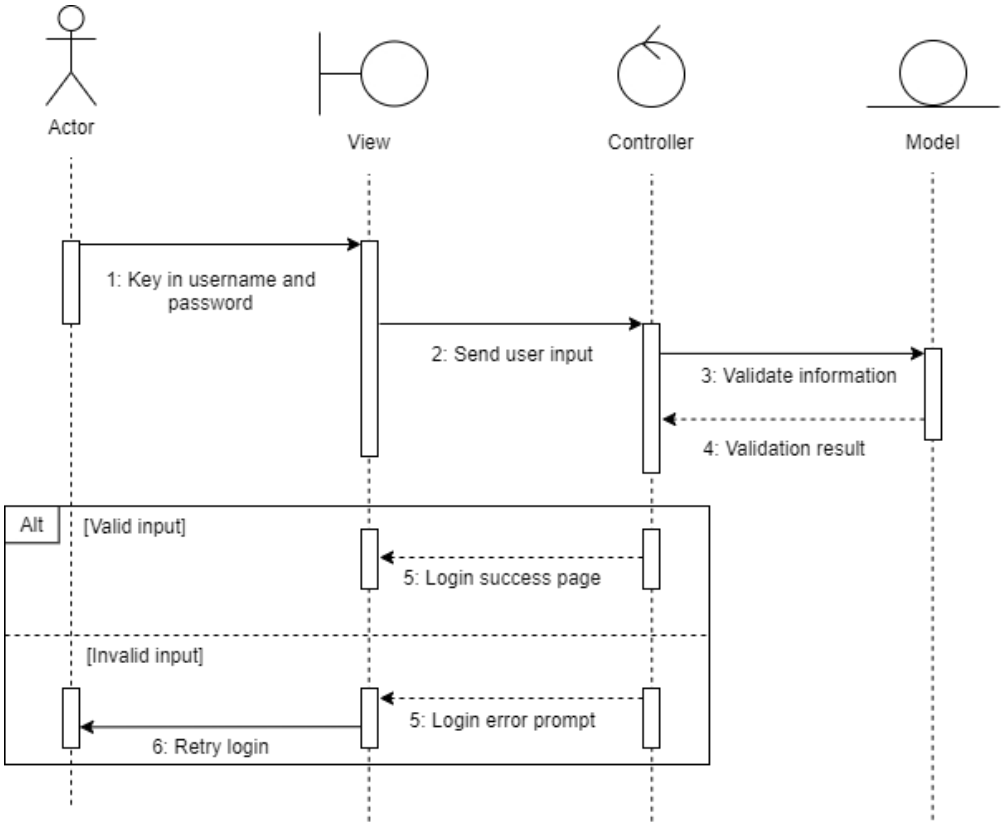


Figure 2. Sequence Diagram of Login Application

Dynamic Binding

Dynamic Binding is a process in which the object or function being called in the program is being overlooked during the runtime. In the current system, the action component named *'loginaction'* in *struts.xml* is linked to the *LoginAction.java* class. When a user clicks on the submission button, the method *execute()* is called upon for the first time where it creates a new object *Person()* assigned to the variable *personBean*. Now consider the scenarios where the username or password submitted during login is wrong and the user attempts to login with the right credentials. Thereafter, the method *execute()* is called upon the second time where it overwrites the existing object *Person()* with the new object containing the right credentials. Thus, we notice how dynamic binding is implemented in this system.

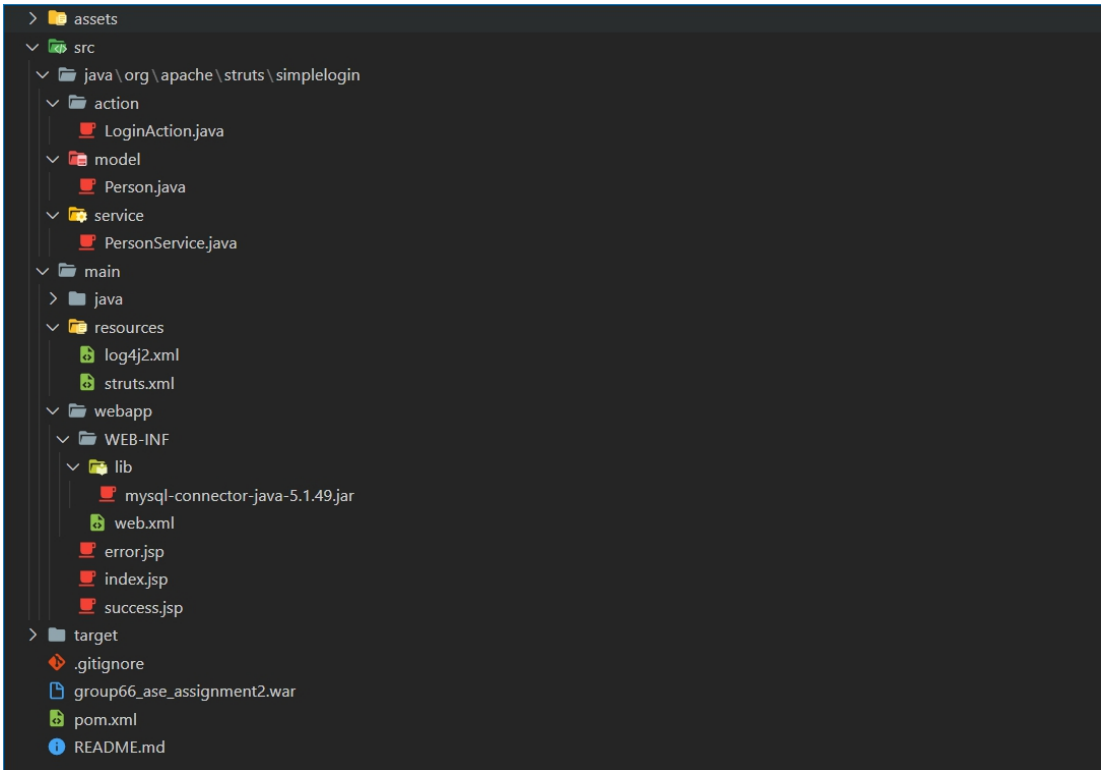
Assuming that there is a change to the login() function in the LoginAction() class , which is called upon the user pressing the submit button, the change this binding from the button to the action, we simply need to change the specifications in the strut.xml file, which performs this

binding (determining which method to call upon button submission interaction). For example in struts.xml we can change the method mapped to the ‘login’ name and by mapping ‘login’ name to a different method, for example FingerprintLogin() we would have been able to change the dynamic binding performed by struts based on the struts.xml file.

name	login
class	controllers.action.LoginAction
method	login

Organization of Code and Manual

The code is structured as the following:



The installation and user manual is as the following:

1. Install MySQL from the website
2. Open command prompt in the downloaded *MySQL/MySQL Server/bin* folder and set up MySQL username and password using the tutorial here
3. Download latest Maven
4. Set the downloaded *maven/bin* folder path to the environment variable, e.g. *“set PATH="c:\program files\apache-maven-3.8.1\bin";%PATH%* “ in command prompt
5. Change MySQL credentials in *PersonService.java*(line 16, 17) using the credentials set up from step 2
6. In the login application project folder, run *“mvn jetty:run”*
7. The web will be up at this link
8. Key in “admin” as user and “password” as password and login success page will show