|  |
| --- |
| **Developing a Backend Administrative portal**  **for**  **Learner’s Academy**  **(Specifications and Scrum Details)** |

**The sections in this document are:-**

|  |
| --- |
| 1.Project details and Developer details |
| 2.Sprint Planning and Task Completion |
| 3.Algorithms and Flow Diagram of the Application |
| 4.Core concepts used in the Project |
| 5.Link to the Git Hub repository |
| 6.Demonstrating the Project Capabilities, appearance, and user Interactions |
| 7.Unique selling points of the Application |
| 8.Conclusions |

**The code for this project is hosted at**

|  |
| --- |
| **Git-Hub link:-**  **https://github.com/swathikanduri7/phase-2-finalproject-Learners-Academy.git** |

**The project is developed by Swathi Kanduri**

**1.Project Details:-**

|  |
| --- |
| Learner’s Academy is a school that has an online management system.  The system keeps track of its classes, subjects, students, and teachers.  It has a back-office application with a single administrator login.  **The administrator can do the following:-**   * Set up a master list of all the subjects for all the classes * Set up a master list of all the teachers * Set up a master list of all the classes * Assign classes for subjects from the master list * Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects) * Get a master list of students (Each student must be assigned to a single class) * Option to view a Class Report which will show all the information about the class, such as the list of students, subjects, and teachers.   **Developer Details:-**  **This project is developed by Swathi Kanduri** |

**2.Sprint Planning and Task Completion**

|  |
| --- |
| The project is planned to be completed in 2 sprints. |

**Tasks completed in this sprint are**:-

|  |
| --- |
| * Creating the flow of the application. * Initializing the git repositories to track changes as development progresses. * Configuring Web Application using Maven tool * Configuring Hibernate utility to manage data base connection and perform data base actions * Completed below stories * User management with the tasks to perform( add new user, edit user, delete user, display list of users using jsps, servlets) * Student management with the tasks to perform( add new student, edit student, delete student, display list of students using jsps, servlets) * Classes management with the tasks to perform( add new class, edit class, delete class, display list of classes using jsps, servlets) * Teacher management with the tasks to perform( add new teacher, edit teacher, delete teacher, display list of teachers using jsps, servlets) * Subject management with the tasks to perform( add new subject, edit subject, delete subject, display list of subjects using jsps, servlets) * Assign( add new user, edit user, delete user, display list of users using jsps, servlets) * Assign classes for subjects from the master list * Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects) * Get a master list of students (Each student must be assigned to a single class). * Develop a Option to view a Class Report(which will show all the information about the class, such as the list of students, subjects, and teachers) * Testing the program with different types of user input. * Pushing the code to Git Hub. * Creating this specification document highlighting application capabilities, appearance and user interaction. |

**3.Algorithms and Flow Diagram of the Application**

|  |
| --- |
| **Algorithm for display List of Users**  **Step1:-Login to the Application**  **Step2:-Select the option to display User details**  **Step3:-Prepare a Servlet end point with user/list with doget() method.**  **Step4:-Fetch the list of users from the database using Hibernate util**  **Step5:-Assign the user list to HTTP request**  **Step6:-Forward the request to userList.jsp**  **Step7:- UserList.jsp performs the following actions**  **a)Prepare a hyperlink to “adduser” when user clicks on adduser it**  **redirects to “user/new” Servlet endpoint.(to perform add user action).**  **b)Prepare a hyperlink to “listofusers” when user clicks on listofusers it**  **redirects to “user/list” Servlet endpoint.(to display latest records from**  **database).**  **c)Display list of users in a table format with options as Edit, Delete.**  **d) Prepare a hyperlink to “edituser” when user clicks on edit it**  **redirects to “user/edit” Servlet endpoint.(to perform update action).**  **e) Prepare a hyperlink to “deleteuser” when user clicks on delete it**  **redirects to “user/delete” Servlet endpoint.(to perform delete action**  **from the database).** |

|  |
| --- |
| **Algorithm for Adding a New User:-**  **Step1:-Login to the Application**  **Step2:-Select the option to display User details**  **Step3:-Select the option for Add new user by clicking “adduser” hyperlink**  **Step4:-Develop a servlet “user/new” to perform add user action**  **Step5:-When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest users list from the database and it**  **assigns the request to HTTP request object.**  **Step6:-Forward the request to userList.jsp** |

|  |
| --- |
| **Algorithm for Edit User:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Edit User details from the users list.**  **Step3:- Develop a servlet “user/edit” to perform display users details where**  **user can edit the details.**  **Step4:- When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest users list from the database and it**  **assigns the request to HTTP request object.**  **Step5:- Forward the request to userList.jsp** |

|  |
| --- |
| **Algorithm for Delete User:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Delete User details from the users list.**  **Step3:- Develop a servlet “user/delete” to perform delete user from the**  **Database using hibernate util.**  **Step4:- Fetch the latest users list from the database and it**  **Assigns the request to HTTP request object.**  **Step5:- Forward the request to userList.jsp** |

|  |
| --- |
| **Algorithm for display List of Students**  **Step1:-Login to the Application**  **Step2:-Select the option to display Student details**  **Step3:-Prepare a Servlet end point with student/list with doget() method.**  **Step4:-Fetch the list of users from the database using Hibernate util**  **Step5:-Assign the user list to HTTP request**  **Step6:-Forward the request to userList.jsp**  **Step7:- student-list.jsp performs the following actions**  **a)Prepare a hyperlink to “addstudent” when user clicks on addstudent**  **it should redirect to “student/new” Servlet endpoint.(to perform add**  **student action).**  **b)Prepare a hyperlink to “listofstudents” when user clicks on**  **listofstudents it should redirect to “student/list” Servlet endpoint.(to**  **display latest records from database).**  **c)Display list of students in a table format with options as Edit, Delete.**  **d) Prepare a hyperlink to “edit” when user clicks on edit it**  **redirects to “student/edit” Servlet endpoint.(to perform update action).**  **e) Prepare a hyperlink to “delete” when user clicks on delete it**  **redirects to “student/delete” Servlet endpoint.(to perform delete action from the database).** |

|  |
| --- |
| **Algorithm for Adding a New Student:-**  **Step1:-Login to the Application**  **Step2:-Select the option to display Student details**  **Step3:-Select the option for Add new student by clicking “addstudent” hyperlink**  **Step4:-Develop a servlet “student/new” to perform add student action**  **Step5:-When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step6:-Forward the request to student-list.jsp** |

|  |
| --- |
| **Algorithm for Edit Student:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Edit Student details from the students list.**  **Step3:- Develop a servlet “student/edit” to perform display student details**  **where admin can edit the details.**  **Step4:- When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step5:- Forward the request to student-list.jsp** |

|  |
| --- |
| **Algorithm for Delete Student:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Delete Student details from the students list.**  **Step3:- Develop a servlet “student/delete” to perform delete student from the**  **Database using hibernate util.**  **Step4:- Fetch the latest list from the database and**  **Assigns the request to HTTP request object.**  **Step5:- Forward the request to student-list.jsp** |

|  |
| --- |
| **Algorithm for display List of Classes**  **Step1:-Login to the Application**  **Step2:-Select the option to display Classes details**  **Step3:-Prepare a Servlet end point with classes/list with doget() method.**  **Step4:-Fetch the list of classes from the database using Hibernate util**  **Step5:-Assign the list to HTTP request**  **Step6:-Forward the request to classes-list.jsp**  **Step7:- classes-list.jsp performs the following actions**  **a)Prepare a hyperlink to “addClass” when user clicks on addClass**  **it should redirect to “classes/new” Servlet endpoint.(to perform add**  **Class action).**  **b)Prepare a hyperlink to “listofcalsses” when user clicks on**  **listofcalsses it should redirect to “classes/list” Servlet endpoint.(to**  **display latest records from database).**  **c)Display list of classes in a table format with options as Edit, Delete.**  **d) Prepare a hyperlink to “edit” when admin clicks on edit it should**  **redirect to “classes/edit” Servlet endpoint.(to perform update action).**  **e) Prepare a hyperlink to “delete” when Admin clicks on delete it**  **should redirect to “classes /delete” Servlet endpoint.(to perform delete**  **action from the database).** |

|  |
| --- |
| **Algorithm for Adding a New Class:-**  **Step1:-Login to the Application**  **Step2:-Select the option to display Classes details**  **Step3:-Select the option for Add new student by clicking “addclass” hyperlink**  **Step4:-Develop a servlet “classes /new” to perform add classes action**  **Step5:-When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step6:-Forward the request to classes-list.jsp** |

|  |
| --- |
| **Algorithm for Edit Class:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Edit Class details from the classes list.**  **Step3:- Develop a servlet “classes/edit” to perform display classes details**  **where admin can edit the details.**  **Step4:- When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step5:- Forward the request to classes-list.jsp** |

|  |
| --- |
| **Algorithm for Delete Class:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Delete Class details from the classes list.**  **Step3:- Develop a servlet “classes/delete” to perform delete class from the**  **Database using hibernate util.**  **Step4:- Fetch the latest list from the database and**  **Assigns the request to HTTP request object.**  **Step5:- Forward the request to classes-list.jsp** |

|  |
| --- |
| **Algorithm for display List of Subjects**  **Step1:-Login to the Application**  **Step2:-Select the option to display Subjects details**  **Step3:-Prepare a Servlet end point with subjects/list with doget() method.**  **Step4:-Fetch the list of subjects from the database using Hibernate util**  **Step5:-Assign the list to HTTP request**  **Step6:-Forward the request to subjects-list.jsp**  **Step7:- subjects -list.jsp performs the following actions**  **a)Prepare a hyperlink to “addSubject” when user clicks on addSubject**  **it should redirect to “subject/new” Servlet endpoint.(to perform add**  **Subject action).**  **b)Prepare a hyperlink to “listofsubjects” when user clicks on**  **listofsubjects it should redirect to “subject/list” Servlet endpoint.(to**  **display latest records from database).**  **c)Display list of subjects in a table format with options as Edit, Delete.**  **d) Prepare a hyperlink to “edit” when admin clicks on edit it should**  **redirect to “subject/edit” Servlet endpoint.(to perform update action).**  **e) Prepare a hyperlink to “delete” when Admin clicks on delete it**  **should redirect to “subject /delete” Servlet endpoint.(to perform delete**  **action from the database).** |

|  |
| --- |
| **Algorithm for Adding a New Subject:-**  **Step1:-Login to the Application**  **Step2:-Select the option to display Subjects details**  **Step3:-Select the option for Add new Subject by clicking “addsubject” hyperlink**  **Step4:-Develop a servlet “subject /new” to perform add classes action**  **Step5:-When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step6:-Forward the request to subjects-list.jsp** |

|  |
| --- |
| **Algorithm for Edit Subject:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Edit Subject details from the subjects list.**  **Step3:- Develop a servlet “subjects/edit” to perform display subject details**  **where admin can edit the details.**  **Step4:- When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step5:- Forward the request to subjects-list.jsp** |

|  |
| --- |
| **Algorithm for Delete Subject:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Delete Subject details from the subjects list.**  **Step3:- Develop a servlet “subjects/delete” to perform delete class from the**  **Database using hibernate util.**  **Step4:- Fetch the latest list from the database and**  **Assigns the request to HTTP request object.**  **Step5:- Forward the request to subjects-list.jsp** |

|  |
| --- |
| **Algorithm for display List of Teachers**  **Step1:-Login to the Application**  **Step2:-Select the option to display Teachers details**  **Step3:-Prepare a Servlet end point with subjects/list with doget() method.**  **Step4:-Fetch the list of subjects from the database using Hibernate util**  **Step5:-Assign the list to HTTP request**  **Step6:-Forward the request to subjects-list.jsp**  **Step7:- teachers-list.jsp performs the following actions**  **a)Prepare a hyperlink to “addTeacher” when user clicks on**  **addTeacher it should redirect to “teacher/new” Servlet endpoint.(to**  **perform add teacher action).**  **b)Prepare a hyperlink to “listof teachers” when user clicks on**  **listof teachers it should redirect to “teacher/list” Servlet endpoint.(to**  **display latest records from database).**  **c)Display list of teachers in a table format with options as Edit, Delete.**  **d) Prepare a hyperlink to “edit” when admin clicks on edit it should**  **redirect to “teacher/edit” Servlet endpoint.(to perform update action).**  **e) Prepare a hyperlink to “delete” when Admin clicks on delete it**  **should redirect to “teacher/delete” Servlet endpoint.(to perform delete**  **action from the database).** |

|  |
| --- |
| **Algorithm for Adding a New Teacher:-**  **Step1:-Login to the Application**  **Step2:-Select the option to display Teachers details**  **Step3:-Select the option for Add new Teacher by clicking “addteacher” link**  **Step4:-Develop a servlet “teacher/new” to perform add teacher action**  **Step5:-When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step6:-Forward the request to teachers-list.jsp** |

|  |
| --- |
| **Algorithm for Edit Teacher:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Edit Teacher details from the teachers list.**  **Step3:- Develop a servlet “teacher/edit” to perform display teacher details**  **where admin can edit the details.**  **Step4:- When admin clicks on save option and it saves the details to database**  **using hibernate util. Fetch the latest list from the database and it**  **assigns the request to HTTP request object.**  **Step5:- Forward the request to teachers-list.jsp** |

|  |
| --- |
| **Algorithm for Delete Teacher:-**  **Step1:-Login to the Application**  **Step2:-Select the option to Delete Teacher details from the teachers list.**  **Step3:- Develop a servlet “teacher/delete” to perform delete class from the**  **Database using hibernate util.**  **Step4:- Fetch the latest list from the database and**  **Assigns the request to HTTP request object.**  **Step5:- Forward the request to teachers-list.jsp** |

|  |
| --- |
| **Algorithm for adding subject to a class**  **Step1:-Select Menu option and add subject to a class**  **Step2:-Fetch subject list and classes from the database and assign the request to HTTP.**  **Step3:-Forward the request to “assign-subject-class.jsp”**  **Step4:-Display the subject and classes drop down menu, select the subject and class and click on save, it redirects to assign subject class servlet endpoint.**  **Step5:-In servlet save the details to database using Hibernate.**  **Step6:-Fetch the classes and subjects form database, assign HTTP request.**  **Step7:-Forward the request to “assign-subject-class.jsp”** |

|  |
| --- |
| **Algorithm for assign-teacher-to a class for the subject**  **Step1:-Select the option to assign a class and subject**  **Step2:-Select the subject list, teachers list and classes list from the database using hibernate util.**  **Step3:-assign list to HTTP request.**  **Step4:-forward the request to “teacher-class-subject.jsp”**  **Step5:-display drop down list for teacher, subject and classes.**  **Step6:-Select the data from the drop down menu, click save button and it redirects to “assign-teacher-to a class”**  **Step7:-Servlet will save the data and fetch the latest list from the database and assign to HTTP request.**  **Step8:-Forward the request to “teacher-class-subject.jsp”** |

|  |
| --- |
| **Algorithm for class report-information about the class such as**  **List of subjects, teachers and students**  **Step1:-Select the option to assign class report from the menu**  **Step2:-Select the class details from the database and assign to HTTP request**  **Step3:- forward the request to “class-report.jsp”**  **Step4:-“class-report.jsp” will display the list of teachers, students and subjects in a table format.** |

**Flow Diagram of the Application**

|  |
| --- |
|  |

**4.Core concepts used in the project are:-**

|  |
| --- |
| * **IDE: Eclipse is used to code for the application** * **Core Java:-A programming language to develop web pages, databases and others.** * **Databases:-Mysql** * **ORM tools:-Hibernate Framework for mapping an object-oriented domain model to a relational database.** * **WebTechnologies:-HTML,CSS,JSP** * **Maven:-Maven is used to build the application** * **J2EE:-Servlet/Jsp are used to collect data from the users to create the web pages dynamically and present the results** * **SVN Tools:-Git-Hub** * **Scrum:-An efficient agile framework to deliver the product.** * **Specification document:-Ms-word is used for specifications for open-source documentation.** |

**5.Link to the Git Hub Repository:-**

|  |
| --- |
| **Git-Hub Link:-**  **https://github.com/swathikanduri7/phase-2-finalproject-Learners-Academy.git** |

**6.Demonstrating the product capablilites, appearance and user interactions:-**

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1)Display List of Students:-

2)Add Student Details,

3)Edit Student Details,

4) Delete Student Details,.

5)Display List of Users:-

6)Add Users Details,

7)Edit Users Details,

8) Delete Users Details,

9)Display List of Teachers:-

10)Add Teachers Details,

11)Edit Teacher Details,

12) Delete Teacher Details,.

13)Display List of Classes:-

14)Add Class Details,

15)Edit Class Details,

16) Delete Class Details,

17)Display List of Subjects:-

18)Add Subject Details,

19)Edit Subject Details,

20) Delete Subject Details,

21) Assign Subject to a Class

22) Assign a teacher to a Subject and Class

23) Display Class Report : students,subjects,teachers

**7. Unique Selling Points of the Application:-**

1. The application is designed to keep on running and taking user inputs even after  exceptions occur. To terminate the application, appropriate option needs to be  selected.

2. The application can take any file/folder name as input. Even if the user wants to  create nested folder structure, user can specify the relative path, and the application  takes care of creating the required folder structure.

3. User is also provided the option to write content if they want into the newly created  file.

4. The application doesn’t restrict user to specify the exact filename to search/delete  file/folder.

5. They can specify the starting input, and the program searches all  files / folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

6. The application also allows user to delete folders which are not empty.

7. The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

8. The application is designed with modularity in mind. Even if one wants to update the  path, they can change it through the source code. Application has been developed  keeping in mind that there should be very less “hard coding” of data.

**8.Conclusions:-**

|  |
| --- |
| Further enhancements to the application can be made which may include:   * Conditions to check if user is allowed to delete the file or add the file at the  specific locations. * Asking user to verify if they really want to delete the selected directory if it’s not  empty. * Retrieving files/folders by different criteria like Last Modified, Type, etc. • Allowing user to append data to the file. * The Learner’s Academy system is bringing a great difference in the lives of students, teachers, parents, and the admin. * Good management offers better productivity and hence more progress towards development. * It helps the school to achieve the target, reduce work, increase efficiency, eliminating error, and monitoring progress. |

**Pushing the code to Git Hub Repository**

* Open your command prompt and navigate to the folder where you have

created your files.

**cd <folder path>**

* Initialize repository using the following command

**git init**

* Add all the files to your git repository using the following command

**git add .**

* Commit the changes using the following command

**git commit . –m<commit message>**

* Push the files to the folder you initially created using the following command

**git push -u origin master**