

**Instructor Input**

*Project*



**Notes to Faculty**

There will be unique case studies available for the students to select. In addition, you can ask the students to refer to the sample case study and its solution.

In phase one of the project, a Web application is created. The solution for the sample case study is provided in the **STMS** folder. Double-click the **STMS** solution file in the **STMS** folder to load the project in Microsoft Visual Studio. However, before executing the solution, you need to create the database named “**STMS**” in Microsoft SQL Server. After creating the database, open the **script.txt** file and execute in Microsoft SQL Server. It will create all the required tables along with the initial records. Further, update the database connection string in the **Web.config** file. Now, you can switch to the Visual Studio 2012 and press the **F5** key to execute the application.

In phase two, a hybrid mobile app is created. The solution for the sample hybrid mobile app is provided in the **Datafiles For Faculty\Solutions\Phase-II\App\STMS.zip** file. You can extract the **STMS.zip** file in your desired location so that you can execute it in Microsoft Visual Studio.

The students can form a group of maximum four participants to develop the project.

The topic selection will be opened to the students in a week as per the milestone and will be available for a limited period. If a student fails to select a topic within the specified time, you may allocate the same to the student and complete the development process.

After the topic selection date is over, groups are created by the system based on the main topics selected by the students. The details of the participants in each group and the final topic assigned are published for the entire batch. Once the groups are created, the students need to select the Team Lead (TL) by using a polling system. The TL should be responsible for assigning tasks to the team members. You should also encourage the students to keep track of the various activities to be completed in every session by maintaining a planner for the same.

The project document needs to be submitted as per the milestone.

Project walkthrough should be done in the classroom or machine room environment. During walkthrough, you, along with the expert faculty, need to evaluate the students based on the predefined parameters.

You need to encourage the students to coordinate with each other and ensure that the project is completed with mutual coordination and cooperation.

It is advisable that at regular intervals, the progress of the project work is tracked to check if all the students are actively participating in the project development.

### Phase One: Web Application Development

Each group will be assigned 60 hours to develop a Web application in the first phase. During this period, they need to perform various tasks, as shown in the following table.

|  |  |  |
| --- | --- | --- |
| ***S. No.*** | ***Tasks to be performed*** | ***Time limit in hours*** |
| ***Analysis Phase*** | | |
| *1* | *Create a requirement elicitation document for the project to gather more specific information on how the system should work.* | *2* |
| *2* | *Create a presentation of the project life cycle model for the client/stakeholders.* | *2* |
| *3* | *Create a vision document of the project.* | *2* |
| *4* | *Create the project plan.* | *2* |
| *5* | *Create a weekly status report template.* | *2* |
| ***Design Phase*** | | |
| *6* | *Create a functional requirement document of the system based on the user requirements gathered from the client.* | *2* |
| *7* | *Model the system using UML Use Case diagrams.* | *2* |
| *8* | *Create Use Case descriptions.* | *2* |
| *9* | *Create the activity diagrams of the system.* | *2* |
| *10* | *Create the class diagram of the system.* | *2* |
| *11* | *Create the entity–relationship model.* | *2* |
| *12* | *Create the software requirement specification.* | *2* |
| ***Implementation Phase*** | | |
| *13* | *Create the Models.* | *4* |
| *14* | *Create the ViewModel, Repository classes.* | *6* |
| *15* | *Create the Controllers.* | *6* |
| *16* | *Create the Layout.* | *4* |
| *17* | *Create the Views.* | *6* |

|  |  |  |
| --- | --- | --- |
| ***S. No.*** | ***Tasks to be performed*** | ***Time limit in hours*** |
| *18* | *Use CSS styles, JQuery, and JavaScript.* | *8* |
| *19* | *Test the functionality of the application manually and perform debugging of any existing errors.* | *2* |

*The Tasks to Be Completed in Phase One*

### Phase Two: Mobile Application Development

Each group will be assigned 30 hours to develop a mobile app in the second phase. During this period, they need to perform various tasks, as shown in the following table.

|  |  |  |
| --- | --- | --- |
| ***S. No.*** | ***Tasks to be performed*** | ***Time limit in hours*** |
| ***Analysis Phase*** | | |
| *1* | *Create a requirement elicitation document for the project to gather more specific information on how the system should work.* | *1* |
| *2* | *Create a presentation of the project life cycle model for the client/stakeholders.* | *1* |
| *3* | *Create a vision document of the project.* | *1* |
| *4* | *Create the project plan.* | *1* |
| *5* | *Create a weekly status report format to be updated weekly.* | *1* |
| ***Design Phase*** | | |
| *6* | *Create a functional requirement document of the system based on the user requirements gathered from the client.* | *1* |
| *7* | *Model the system using UML Use Case diagrams.* | *2* |
| *8* | *Create Use Case descriptions.* | *1* |
| *9* | *Create the activity diagrams of the system.* | *2* |
| *10* | *Create the entity–relationship model.* | *1* |
| *11* | *Create the software requirement specification.* | *1* |

|  |  |  |
| --- | --- | --- |
| ***S. No.*** | ***Tasks to be performed*** | ***Time limit in hours*** |
| ***Implementation Phase*** | | |
| *12* | *Create the HTML views.* | *7* |
| *13* | *Use CSS styles, JQuery, JQuery Mobile, JavaScript, and JSON to read–write data.* | *8* |
| *14* | *Test the functionality of the application manually and perform debugging of any existing error.* | *2* |

*The Tasks to Be Completed in Phase Two*



**Evaluation Guidelines**

Use the following points as guidelines while evaluating students:

* Correctness in terms of logic and requirements specified
* Usability and user-friendliness
* Implementation of validations
* Accuracy, reliability, and performance of the application
* Percentage completion measured on the basis of the number of functional requirements of the project versus the number of requirements completed on time
* Number and type of errors pointed out in the project
* Understanding of the technology used
* Completion of all the formats in documents and their readability

Communicate these evaluation parameters to the students in the first session. You must communicate your feedback and the rationale for awarding score. Any discrepancy arising should be immediately sorted out with the student. Do not postpone the evaluation of a student. For example, if a student does not submit the documents in time, the student will not be awarded any score for the documentation. This will ensure that all the students submit their documents in advance and are fully prepared for the presentation or walkthrough.

You need to validate and approve each activity in the Project Plan submitted by the students. The project will be evaluated on the following parameters:

* **Quality**: Conforms to the following requirements: (40 Marks)
  + The solution must map to the requirements specified in the case study.
  + The solution must be implemented by ensuring a good design.
  + The solution code must have meaningful variable names.
  + The solution must have explanatory comments in the code.
  + The solution code must be optimized for performance.
  + Quality must be maintained in the documentation.
* **Timeliness**: Timely project implementation (40 Marks)
* **Query handling**: Handling queries during project walkthrough, which may include: (20 Marks)
  + Queries regarding the flow of control in the application
  + Queries regarding the validations performed in the application
  + Queries regarding the logic implemented to accomplish the requirements specified in the case study



**Solution to Sample Case Study: Smart Training Management System**

Solution for the sample case study is developed in the following two phases:

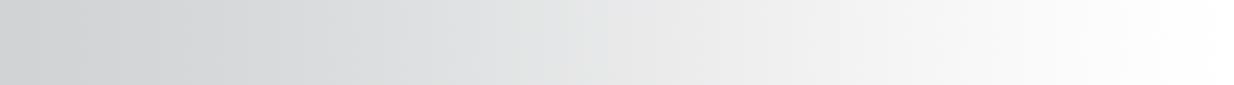
* Web application development
* Mobile application development

### Phase one: Web application development

In phase one, the following tasks need to be performed to meet the specifications of the project:

* Analyze the requirements to create the solution.
* Create the login view page to check users’ authenticity based on their roles.
* Create the layout page, partial layout page, and login layout pages, which display the role-based menu bar and footers. The login layout page should be displayed when a user clicks the “Sign In” button.
* Create the Home page that displays details of the role-based activities, such as search training and view nominations.
* Create a partial view for a search activity. Ensure that all role-based users should be allowed to access the existing approved training title in a drop-down list and to get the training details on the same page.
* Create the Nomination Index view page that displays the following role-based activities:
  + If a developer (with the EMP role) is logged in to the application, then his/her nomination details should be displayed.
  + If a department manager (with the DM role) is logged in to the application, then he/she can view the developer’s nomination details. In addition, he/she can view the dynamic “Manage” action link that will be visible only when the nomination status is “Pending.”
  + If a talent development heads (with the TDH role) is logged in to the application, then he/she can view the DM nomination details along with the dynamic “Manage” action link that will be visible only when the nomination status is either “Pending” or “Rejected.”
  + If a training hub manager (with the THM role) is logged in to the application, then he/she cannot access the Nomination Index view page.
* The navigation panel should comprise the following navigation options as per the users’ roles:
  + Developers’ (with the EMP role) navigation panel should have links to navigate to the Home, trainings, search training, and nomination view pages.
  + Department managers’ (with the DM role) navigation panel should have links to navigate to the Home, Manage TNI, Team Nomination, Search Training, My Nomination, and View Nomination view pages.
  + Talent development heads’ (with the TDH role) navigation panel should have links to navigate to the Home, Manage TNI, Team Nomination, Search Training, and View Nomination view pages.
  + Training hub managers’ (with the THM role) navigation panel should have the option to navigate to the Home, Manage TNI, Search Training, and View Nomination view pages.
* Create the My Nomination view page that displays the role-based training nomination details.
* Create the Edit Nomination view page that should be accessible only by DM and TDH. The DM users can approve/reject the respective EMP nominations and remarks. Similarly, the TDH users can approve/reject the respective DM nominations and remarks.
* Create the All Nomination view page that displays information of all nominations exist in the database. This page should be accessible by the DM, TDH, and THM.
* Create the Nominate view page that displays the confirmation message or the error message after clicking the Nominate action link either by the EMP or DM users.
* Create a notification page to display roles-specific notifications, such as these:
  + Developers with the “EMP” role should view the pop-up notification on the launched training in between the last visit and the current date.
  + The DM should view pop-up notifications on the nominations submitted by the EMP between the last visit date and the current date.
  + The TDH should view pop-up notifications on the nominations submitted by the DM between the last visit date and the current date.
* Create the All Search page that displays all approved training details and ensure that this page should be accessible by the developers only.
* Create the New Training page that should have all training details fields.
* Create the Delete Training view page that should delete the training from a database. Ensure that this page should be accessible by the DM and TDH users only.
* Create the Edit Training view page that should update the training details from a database. This page should be accessible by the DM, TDH, and THM. The DM can edit only those training details that are not approved by the TDH. The TDH can update the training details and can approve, reject, or block any training. Ensure that the THM can also access this page but is only allowed to edit the Status field.
* Create the Index Training view page that displays the training details to the users according to their roles. In addition, the training calendar should have the action links based on the following conditions:
  + If the DM is logged in to the application, then the Create New Training link should be displayed. If any training approval status is “Approved,” then the Nominate action link should be displayed. Similarly, if a training status is “Pending” or “Rejected,” then the Edit and Delete action link should be displayed.
  + If the TDH is logged in to the application, the Create New Training link should be displayed. All training details should be displayed with the Edit action link. If any training approval status is “Pending” or “Rejected,” then the Edit and Delete action link should be displayed.
  + If the THM is logged in to the application, then all training details along with the Edit action link should be displayed.

You can find the solution for this case study in the **02\_Datafiles For Faculty\Solutions\ Phase-I \ 02\_DotNet \STMS.zip** file provided in the TIRM.



**Executing the Solution**

To execute the project solution, you need to perform the following steps:

1. Copy the **STMS.zip** file to your machine from the location, **Project Solution**.
2. Unzip the **STMS.zip** file at the desired location. The **STMS** folder is created.
3. Double-click the **STMS** folder. The **STMS** folder is opened.
4. Double-click the **STMS.sln** file. The **STMS - Microsoft Visual Studio** window is displayed.

***Note***

*If you get the message box prompting STMS.psess does not exist or is invalid, then click the* ***OK*** *button.*

1. Open the **Web.config** file and update the connection string as per SQL Server path and the database location.
2. Open the **database script** file and execute it in SQL Server. It will create all required tables along with the initial data.

In the project solution, various files are created, as shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *1.* | *HomeController.cs* | *Controller* | *Contains the code to create the home page of the STMS Web application.* |
| *2.* | *LoginController.cs* | *Controller* | *Contains the code to enable role-based “Sign In” and “Sign Off” from the STMS Web application.* |
| *3* | *NominationController. cs* | *Controller* | *Contains the code for role-based Notifications, Manage Nominations, All Nominations, and Self-Nominations.* |
| *4.* | *SearchController.cs* | *Controller* | *Contains the code that provides search training details feature. This Controller is responsible for displaying the search title for the approved training and listing all the training from a database.* |
| *5.* | *TrainingController.cs* | *Controller* | *Contains the code to manage the training details, such as list existing trainings, add new training, update training, delete training, and approve/reject/ block trainings.* |
| *6.* | *EMP.cs* | *Model* | *Contains a class that contains fields to refer all the employees, such as developers, the DM, the TDH, and the THM.* |
| *7.* | *Login.cs* | *Model* | *Contains a class that represents login details for each role-based users.* |
| *8.* | *Nomination.cs* | *Model* | *Contains a class that represents nomination details for a training.* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *9.* | *SearchTraining.cs* | *Model* | *Contains a class that enable users to search training details.* |
| *10.* | *Training.cs* | *Model* | *Contains three classes: the Training class that represents training details, the TitleList class that represents the training titles only, and the SearchFilter class that represents the list of training titles.* |
| *11.* | *NominationViewModel. cs* | *ViewModel* | *Contains two classes: the NominationViewModel class that contains the details of all other model classes, such as Nomination, Training, EMP, and Login and. the Trainings class that contains only the list of training.* |
| *12.* | *GenericDbContext.cs* | *Repository* | *Contains the Database Context for the model classes named EMP, Login, Nominations, and Trainings.* |
| *13.* | *GenericRepository.cs* | *Repository* | *Contains the GenericRepository<T> class, which has standard generic functions from the Template class, such as GenericRepository, SelectAll, SelectByID, Insert, Update, Delete, and Save.* |
| *14.* | *IGenericRepository.cs* | *Repository Interface* | *Implements the Generic class.* |
| *15.* | *Site.css* | *CSS* | *Contains the html tags definitions to create the layout of Web pages.* |
| *16.* | *bootstrap.min.css* | *CSS* | *Contains the style sheet tags, classes, and ID definitions for managing the look of the layout and feel of Web pages.* |
| *17.* | *Home/Index.cshtml* | *View* | *Contains the Home page for each role-based user.* |
| *18.* | *Login/Login.cshtml* | *View* | *Contains the Login page form for each role-based user.* |
| *19.* | *Nomination/Index.csht ml* | *View* | *Contains nomination history details as per the users’ role. This page is accessible by all users except the THM.* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *20.* | *Nomination/AllNomina tion.cshtml* | *View* | *Contains all nominations details of the users, such as the developers and DMs. This page is accessible by all users.* |
| *21.* | *Nominatiion/Edit.csht ml* | *View* | *Contains the edit nomination details of the developers and DMs. This page is accessible by the DM and TDH users only.* |
| *22.* | *Nomination/MyNomina tion.cshtml* | *View* | *Contains the individual nomination details of the developers and DMs. Except the THM, all users can access this page.* |
| *23.* | *Nomination/Nominate. cshtml* | *View* | *Contains the feedback message after the nomination activity. This page should be accessible by the developers and DMs only.* |
| *24.* | *Nomination/Notificatio n.cshtml* | *View* | *Contains the code that displays new training information if the developers access the application; however, if the DM or TDH access the application, then new nomination will be displayed. The THM cannot access this page.* |
| *25.* | *Search/Index.cshtml* | *View* | *Contains the code to search the approved training titles and should be accessible by all users.* |
| *26.* | *Search/\_SearchResult. cshtml* | *View* | *This partial view page contains the training details based on the request received from the Search/Index.cshtml files.* |
| *27.* | *Search/All.cshtml* | *View* | *Contains a code that displays all training details from a database and is accessible by the developers only.* |
| *28.* | *Training/Index.cshtml* | *View* | *Contains the code that displays all training details with dynamic action menu for each user.* |
| *29.* | *Training/Edit.cshtml* | *View* | *Contains the selected training details form to update the role-based training details into a database. This page is accessible by all users, except developers.* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *30.* | *Training/Delete.cshtml* | *View* | *Contains the selected training details for deleting the training details from a database. This page is accessible by the DM or TDH only if the training status is “Pending,” “Blocked,” or “Rejected.”* |
| *31.* | *Training/Create.cshtml* | *View* | *Contains the training details form for adding new training details into a database. This page is accessible by the DM and TDH only.* |
| *32.* | *Shared/Error.cshtml* | *View* | *Contains an error information and displays it only when the Model Exception occurs.* |
| *33.* | *Shared/\_LoginPartial. cshtml* | *View* | *Contains the partial Login page layout along with Login user names from a database.* |
| *34.* | *Shared/\_LoginLayout.c shtml* | *View* | *Contains the Login page layout and the carousel effect for the application. This page is accessible by all users for Login into the application.* |
| *35.* | *Shared/\_Layout.cshtml* | *View* | *Contains the Layout format of the application for each user.* |

*The Project File Details*

### Phase Two: Mobile application development

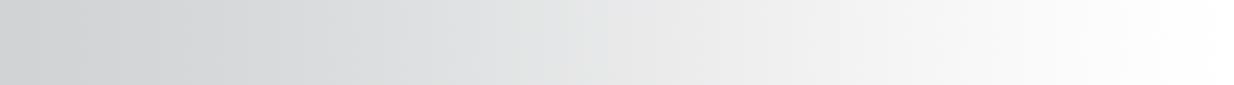
In phase two, the following tasks need to be performed to meet the specifications of the project:

* Analyze the requirements to create the solution.
* Create Apache Cordova Apps project and copy the **css**, **scripts**, and **images** folders from the **STMS**

folder into the **www** folder.

* Create a splash screen named **index.html** that displays the app logo on the load event. This screen stays for two seconds and then redirects to app login activity.
* Create the **login.html** file having two divisions named role layout and sign in. The role layout division consists of four action links that are mapped to the individual roles by defining ***data-name*** property. The sign in division consists of a login form and a **Sign In** action link. Make both username and password form fields mandatory and write JQuery for form field data validation.
* Open the **login.json** file from the **scripts** folder to get the predefined username and passwords.
* Create the role-specific home screens named **emp-home.html**, **dm-home.html**, **tdh-home.html**, and **thm-home.html** that display general guidelines for the actions users may perform under their login profile. Place **Continue** action link to navigate to the role-based activity views.
* Create the role-specific view files named **emp-activity.html**, **dm-activity.html**, **tdh-activity.html**, and **thm-activity.html** that display the navigation panel with the list of action links against respective roles. The navigation panel should have the following navigation options:
  + Developers’ (with the EMP role) navigation panel should have links to navigate to the home, view training calendar, and view nomination list pages.
  + Department managers’ (with the DM role) navigation panel should have link to navigate to home, view training calendar, add trainings, view nomination list, and manage nominations.
  + Talent development heads’ (with the TDH role) navigation panel should have links to navigate to home, view training calendar, manage trainings, and view nominations.
  + Training hub managers’ (with the THM role) navigation panel should have option to navigate to home, view training calendar and view nominations.
  + Display the “Back” and “Sign Out” action link at the top left and right corner of screen header respectively for all the users.
* Using JQuery mobile, create a division that contains collapsible list view component for populating training calendar. This calendar should have the records that are approved by TDH, otherwise division should display an alternate message, “no record not found”. Place this division in all four role specific activity files and define their IDs as **calendar**. Open the **myscript.js** file and refer to the function, under the EMP section, which is written to dynamically populate **Nominate** action link within the calendar.
* Using JQuery mobile, create a division that contains the list view component for populating nomination list. This list should have the records that are approved by DM; otherwise, division should display an alternate message, “no record not found”. Place this division in all four role specific activity files and define their IDs as **list-nomination**. Open the **myscript.js** file and refer to the function, under the DM section, which is written to dynamically populate **Approve** and **Reject** action links with each nomination record if the user belongs to the DM role.
* Using JQuery mobile, create a division that displays a form to add new raining. Place this division within **dm-activity.html** and define ID as **create-training**. Define form field validation rules in JavaScript.
* Using JQuery mobile, create a division that contains the list view component for populating training list. This list should have the records of new training courses which are added by DM, otherwise division should display an alternate message on record not found. Place this division within the **tdh-activity.html** file and define ID as **manage-training** for this division. Further, open the **myscript.js** file and refer to the function, under the TDH section, written to dynamically populate **Approve** and **Reject** action links with each training record if the user belongs to the TDH role.
* Create external style sheet to format all the view elements of the STMS mobile app.
* Include the following **css** and **js** files to each of the html files:
  + css/jquery.mobile-1.4.5.css
  + css/style.css
  + cordova.js
  + scripts/platformOverrides.js
  + scripts/jquery-1.11.3.js
  + scripts/jquery.mobile-1.4.5.js
  + scripts/myscript.js

You can find the solution for this case study in the **STMS.zip** file provided in the **02\_Datafiles For Faculty\Solutions\Phase-II\App** folder in the TIRM.



**Executing the Solution**

To execute the project solution, you need to perform the following steps:

* 1. Copy the **STMS.zip** file to your machine from the **Project Solution\Phase-II** folder given in the TIRM.
  2. Unzip the **STMS.zip** file at desired location. The **STMS** folder is created. Double-click the folder and ensure **STMS** folder is created along with its file structure.
  3. Start Microsoft Visual Studio IDE. Select the **Open** option from the **File** menu. Select **Project/Solution** Browse and open the project solution folder you unzipped. Select **STMS.jsproj** and click **Open**. The project with the template and dependent files will be loaded in the IDE.
  4. Select **STMS** from Solution Explorer and press **F5** key to run the project.
  5. This will execute the project and open the app in the selected emulator.

In the project solution, various files are created, as shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *1* | *Index.html* | *View* | *Contains the layout code of the app’s splash screen, which will be displayed for two seconds and then the calls Login.html on timeout event.* |
| *2* | *Login.html* | *View* | *Contains the layout code of the login page along with a sign-in form.* |
| *3* | *Emp-home.html* | *View* | *Contains the layout code of the home page for the EMP role.* |
| *4* | *Dm-home.html* | *View* | *Contains the layout code of the home page for the DM role.* |
| *5* | *Tdh-home.html* | *View* | *Contains the layout code of the home page for the TDH role.* |
| *6* | *Thm-home.html* | *View* | *Contains the layout code of the home page for the THM role.* |
| *7* | *Emp-activity.html* | *View* | *Contains the layout code of the action view for the EMP role.* |
| *8* | *Dm-activity.html* | *View* | *Contains the layout code of the action view for the DM role.* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No.*** | ***File Name*** | ***Component Type*** | ***Description*** |
| *9* | *Tdh-activity.html* | *View* | *Contains the layout code of the action view for the TDH role.* |
| *10* | *Thm-activity.html* | *View* | *Contains the layout code of the action view for the THM role.* |
| *11* | *Login.json* | *Data* | *Contains users’ login credentials details.* |
| *12* | *Index.js* | *Script* | *Contains JQuery code for the app’s splash screen activity.* |
| *13* | *Login.js* | *Script* | *Contains JQuery code for the login activity.* |
| *14* | *Myscript.js* | *Script* | *Contains JQuery code for the app activities.* |
| *15* | *Login.css* | *Stylesheet* | *Contains the CSS code for styling login view elements of the STMS app.* |
| *16* | *Index.css* | *Stylesheet* | *Contains the CSS code for styling the app splash screen.* |
| *17* | *Style.css* | *Stylesheet* | *Contains the CSS code for defining custom styles to view elements.* |
| *18* | *Jquery.mobile- 1.4.5.css* | *Stylesheet* | *Contains the CSS code for styling the app JQuery mobile elements.* |

*The Project File Details*

**PROJECT ON SMART TRAINING MANAGEMENT SYSTEM (STMS) - WEB APPLICATION**

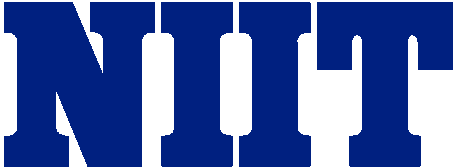
Developed by



**Sample Project Documentation**

**Name**: Patrick Smith

**Reg. No.**: RNG3366



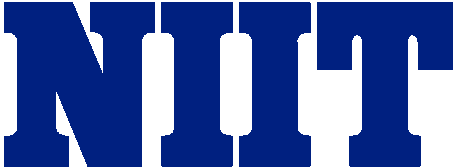
**SMART TRAINING MANAGEMENT SYSTEM (STMS)**

**Batch Code**: 001 **Start Date**: 1/08/2016 **End Date**: 26/08/2016

**Name of the Coordinator**: John Williams

**Name of the Developer**: Patrick Smith

**Date of Submission**: 29/08/2016



# CERTIFICATE

This is to certify that this report, titled Smart Training Management System (STMS) Capstone Project on Web Application Development, embodies the original work done by Patrick Smith in partial fulfillment of the course requirement at NIIT.

Coordinator: John Williams

**ACKNOWLEDGEMENT**

We have benefited a lot from the feedback and suggestions given to us by Mr. John Williams and other faculty members.

**REQUIREMENT ELICITATION DOCUMENT**

Q1. What are the objectives of your company?

Q2. Tell us briefly about the stakeholders of the company. Q3. How many departments does your company have?

Q4. How many employees work in your company? Q5. How many roles types does your company have?

Q6. How many roles do you have for each training requirement? Q7. Who are your primary competitors?

Q8. How do the operations of your competitors differ from your operations? Q9. How do you maintain the training information?

Q10. How do you maintain the training nomination information?

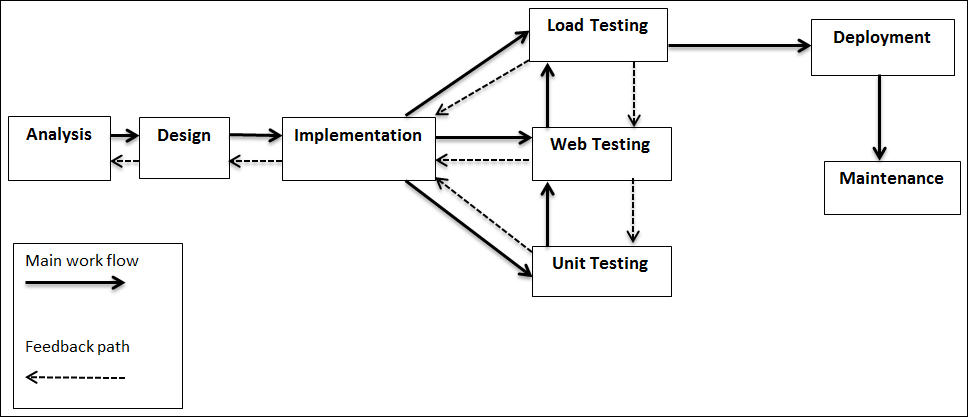
Q11. How do you keep track of nominations that are submitted for a day? Q12. How many trainings can an employee submit at a time?

Q13. How do you inform a participant if a nomination is already submitted for a training? Q14. How is a training detail modified or canceled?

Q15. Do you provide any status change for a training execution?

Q16. How do you restrict a training from modification or deletion if training is released? Q17. How many reports are required by the company?

**PROJECT LIFE CYCLE MODEL**



*The Project Life Cycle Model*

**VISION DOCUMENT**

### Vision of the Project

To create a responsive Web-based solution for the Training Hut Inc. company for its training management services.

### About the Company

Training Hut Inc., founded in 2005, currently operates within Boston and Minnesota. The company provides professional training programs to their employees in different corporate offices or client locations. Currently, Training Hut Inc. is in the process of establishing itself across the USA, UK, Australia, and Asia.

### Requirements Summary

The management of Training Hut Inc. wants to automate its training management processes. In the first phase, the management wants the respective stakeholders to be able to access the Web application over the Internet to create, publish, search, and nominate for trainings. The management also wants that an employee of the company should be able to log on to the application and manage the training requests.

### Project Goals

The project goals are:

* + - Analyze the requirements to create the solution.
    - Design the application based on the results of the analysis phase.
    - Create a responsive Web application based on the MVC model of the application design.
    - Test the implemented application.

### Project Stakeholders

The primary stakeholders of the projects are:

* + - Department Managers (DM) of Training Hut Inc.
    - Developers (with EMP role) of Training Hut Inc.
    - Talent Development Head (TDH) of Training Hut Inc.
    - Training Hub Managers (THM) of Training Hut Inc.
    - Management of EarnestPro Inc.

**PROJECT PLAN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***S. No*** | ***Task Name*** | ***Planned Start Date*** | ***Planned Finish Date*** | ***Actual Start Date*** | ***Actual End Date*** | ***Person Responsible*** |
| *1* | *Analyze the project requirements* | *1/08/2016* | *2/08/2016* | *1/08/2016* | *2/08/2016* | *Patrick* |
| *2* | *Design the project* | *3/08/2016* | *5/08/2016* | *3/08/2016* | *5/08/2016* | *Patrick* |
| *3* | *Implement the project* | *8/08/2016* | *19/08/2016* | *8/08/2016* | *19/08/2016* | *Patrick* |
| *4* | *Test the project* | *22/08/2016* | *26/08/2016* | *22/08/2016* | *26/08/2016* | *Patrick* |
| * *Patrick will be working four hours per working day to complete the project.* * *Saturdays and Sundays being holidays are not considered as working days.* | | | | | | |

**WEEKLY STATUS REPORT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Weekly Status Report** | | | | |
| **Developer**: Patrick **Phase**: 1 and 2  **Period**: **From** 1/08/2016 **to** 5/08/2016 | | | | |
| **Activity/Artifact** | **Responsibility** | **Planned Completion Date** | **Completion Status (%)** | **Reason for Incompletion** |
| * *Create a requirement elicitation document for the project to gather more specific information on how the system should work.* * *Create a presentation of the project life cycle model for the client/stakeholders.* * *Create a vision document of the project.* * *Create the project plan.* * *Create a weekly status report format.* * *Create a functional requirement document of the system based on the user requirements gathered from the client.* * *Model the system using UML use case diagrams.* * *Create use case descriptions.* * *Create the activity diagrams of the system.* | Patrick | 5/08/2016 | 100% | NA |

**FUNCTIONAL REQUIREMENT DOCUMENT**

The functional requirements of the system have been derived from the user requirements and interviews with the project stakeholders. The functional requirements have been designed to highlight the services that the system should provide. These requirements specify how the developers should design and develop the system that is responsive to devices such as PCs, tabs, and smart phones in order to meet the expectations of the project stakeholders from the developed system.

### View training calendar

1. The system must allow all the users to view the list of released training details.
2. The system should display only those training records that are approved by the Talent Development Head.

### Create training

1. The system should allow the Department Manager and Talent Development Head to create new training and submit the details.
2. The system should designate approval status as “Pending,” by default, for newly created training.
3. The system should ensure that all the form fields must be duly filled in by the user.
4. The system should generate the training ID automatically.

### Modify/delete training details

1. The system should allow Talent Development Head to modify the details.
2. The system should ensure that the Department Manager can modify the details only if the training is not approved.
3. The system should ensure that the approved training sessions cannot be deleted.
4. The system should ensure that only the Talent Development Head can update the status of a training session (approved/rejected).
5. The system should update the date of release or update of a training session.

### Approve/reject training

1. The system should allow the Talent Development Head to update approval status of a training session as “Approved” or “Rejected.”
2. The system should allow to update the training calendar to add on a training session that is approved and remove a training session that is rejected.
3. The system should allow the Talent Development Head for approval or rejection of all the training sessions created and submitted by the Department Manager.

### Search training

1. The system should allow users to search training by the training title.

### Submit nomination for training

1. The system should allow all the developers (with EMP role) and department managers to select a training session and submit nomination.
2. The system should ensure that a user who has already applied for a training session cannot nominate himself/herself again for the same.
3. The system should ensure that the Department Manager can view only those training sessions that are created and released by the Talent Development Head.
4. The system should allow developers (with EMP role) to view all the training sessions created by the Department Manager and Talent Development Head.
5. The system should designate approval status as “Pending,” by default, for newly created training.
6. The system should ensure to keep track of the nomination submission date.

### Approve/reject a nomination

1. The system should allow the Department Manager to view, select, and perform action for approval or rejection to a nomination submitted by his/her team member.
2. The system should allow the Talent Development Head to view, select, and perform action for approval or rejection to a nomination submitted by the Department Manager.

### User login

1. The system should allow users to log on and perform role-based actions by specifying a user name and password. The users’ details are predefined and already available in the Human Resource database.
2. The system should ensure that the user name and password fields are not empty.
3. The system should authenticate the login credentials before providing access to the application.
4. The system should provide role-based action menu to the user when the authentication is successful.

### Modify training status

1. The system should allow the Training Hub Manager or Talent Development Head to modify training status as “Not Started,” “Being Executed,” “Completed,” or “Cancelled.”
2. The system should allow the Talent Development Head to update the training status as “Blocked.”

### View nomination details

1. The system should allow all developers and department managers to view self-nomination summary report.
2. The system must allow all the users of “Department Manager” roles to view team-nomination summary report.
3. The system must allow Training Development Manager and Training Hub Manager to view nomination summary report.

### View popup notification

1. The system should display a popup notification for developers (with EMP role) that depicts all the training sessions released between the last visited date and the current date.
2. The system should display a popup notification for the Department Manager and the Talent Development Head that depicts all the nominations submitted between the last visited date and the current date and whose nomination status is “Pending.”
3. The system should ensure that no popup notification should be displayed if there is no training submitted and approved between the last visited date and the current date.
4. The system should ensure that no popup notification should be displayed if there is no nomination submitted between the last visited date and the current date and which is pending for approval.

**UML USE CASE DIAGRAM**

Nominate for Training

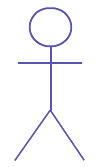
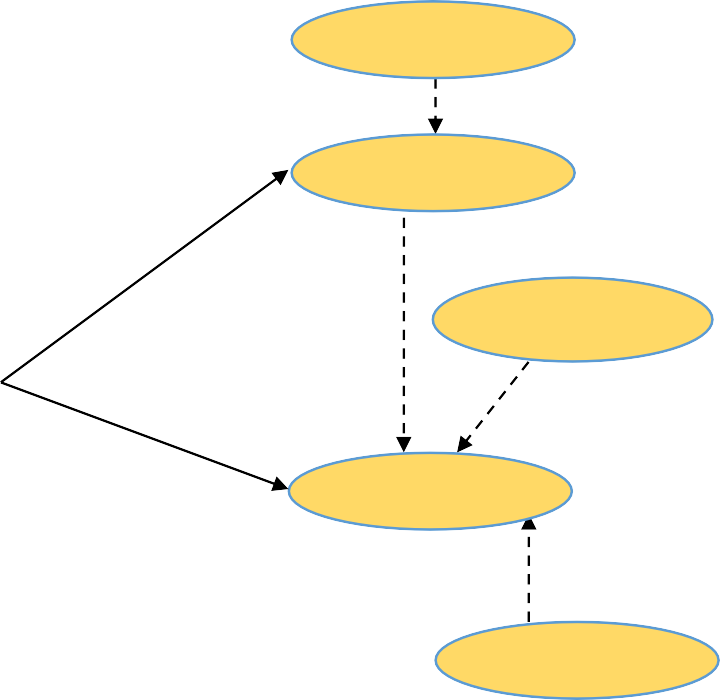
Search Training

Popup Notification

Smart Training Management System

**Developer**

Login



Logout

View Nominations

*The Use Case Diagram for a Developer*

Nominate for Training

Search Training

Popup Notification

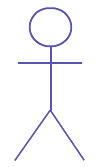
Login

Smart Training Management System

**Department Manager**

View Nomination

Logout



Create Training Update Training

Delete Training Approve Nomination

*The Use Case Diagram for the Department Manager*

Block Training

Search Training

Popup Notification

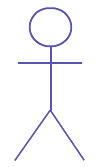
Login

Smart Training Management System

**Talent Development Head**

View Nomination

Logout



Create Training Update Training

Delete Training Approve Nomination

*The Use Case Diagram for the Talent Development Head*

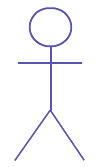
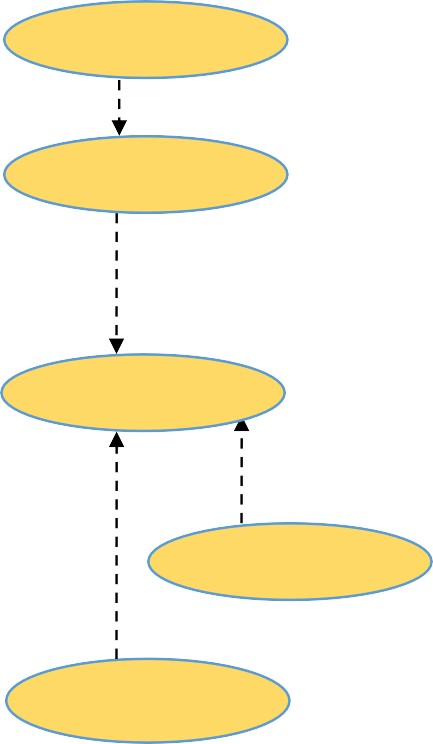
Update Training

Search Training

Smart Training Management System

**Training Hub Manager**

Login



Logout

View Nomination

*The Use Case Diagram for the Training Hub Manager*

### USE CASE DESCRIPTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case**: **Login**  *The Login Use Case*  **Use Case**: **Logout** | | | |
|  | Use Case Number | *UC-02* |  |
|  | Use Case Name | *Logout* |  |
|  | Priority | High |  |
|  | Actor | Developer, Department Manager, Talent Development Head, and Training Hub Manager |  |
|  | Description | This use case describes how a user logs off from the system. |  |
|  | Precondition | User has signed in the system. |  |
|  | Post-condition | If the use case is successful, the user’s session will be invalidated and the user will be returned to the login page. |  |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-01* | |
| Use Case Name | *Login* | |
| Priority | High | |
| Actor | Developer, Department Manager, Talent Development Head, and Training Hub Manager | |
|  |  | |
| Description | This use case describes how the users sign in to the system. | |
| Precondition | None | |
| Post-condition | If the use case is successful, a user gets access to his/her respective home page as per the role; otherwise, an error message is displayed. | |
| Basic course of Action | **User Action** | **System Response** |
| 1. The user is on the system login page of the application. 2. The user specifies the login user name and password and clicks the “Sign In” button. | 1. The system verifies that the specified user name and password are not empty and valid. 2. The system authenticates the login credentials. 3. The system displays the home page of action area when the authentication is successful. 4. Use case ends. |
| Alternate Course of Action | 3.1 If the user name and password fields are empty, the system displays an error message.  4.1 If the login credentials do not match, the system displays an error message. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user has logged on to the system and is working on any of the activity page. 2. The user clicks the Sign Out link.   4. The user clicks the “Click here…” link. | 3. The system invalidates user’s session and displays the Sign Out notification page given with a link to return to the login page.   1. The system displays login page and invalidates browser history for the application. 2. Use case ends. |  |
|  | Alternate Course of Action | 5.1 If the cab category, model, and license number fields are empty or not valid, the system displays an error message. | |  |
| *The Logout Use Case*  **Use Case**: **Search Training**  *The Search Training Use Case* | | | | |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-03* | |
| Use Case Name | *Search Training* | |
| Priority | High | |
| Actors | Developer, Department Manager, Talent Development Head, and Training Hub Manager | |
| Description | This use case describes how users can search/view a training session. | |
| Precondition | User has signed in to the system, and system identifies the role of the user and creates a role-based session. | |
| Post-condition | If the use case is successful, the user can search and view released training details in the system. | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user is on the home page of the application. 2. The user clicks the “Search Training” tab from the top navigation menu.   5. The user selects number of records to display.  7. The user selects search value in the search box. | 1. The system verifies the role stored in the session while signing in by the user. 2. The system queries database for released training sessions to be published based on the user role identified.   6. The system displays the selected training details.   1. The system displays filtered records based on training database details. 2. Use case ends. |
| Alternate Course of Action | 3.1 If database connection is not available, system displays an error message. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case**: **Nominate for Training**  *The Nominate for Training Use Case*  **Use Case**: **View Nominations** | | | |
|  | Use Case Number | *UC-05* |  |
|  | Use Case Name | *View Nominations* |  |
|  | Priority | High |  |
|  | Actor | Developer, Department Manager, Talent Development Head, and Training Hub Manager |  |
|  | Description | This use case describes how users view role-based nomination report. |  |
|  | Precondition | User signs in the system. |  |
|  | Post-condition | If the use case is successful, system displays role-based report on summary of nominations submitted. |  |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-04* | |
| Use Case Name | *Nominate for Training* | |
| Priority | High | |
| Actor | Developer and Department Manager | |
| Description | This use case describes how the users submit nominations for training in the system. | |
| Precondition | User has signed in the system and currently on “Trainings/Search Training” page displaying training calendar. | |
| Post-condition | If the use case is successful, the users submit nomination for a training session and view summary of nominations submitted till date; otherwise, an error message will be thrown by the system. | |
| Basic course of Action | **User Action** | **System Response** |
| 1. The user is on the “Trainings/Search Training” page of the application displaying training details. 2. The user clicks “Nominate” link for the training opted for submitting nomination. | 1. The system identifies training ID and runs database query to submit nomination for the selected training. 2. The system runs database query to fetch nomination records for the user. 3. The system redirects to display the feedback message after validating the Nomination Id from database, either nomination action is successful or already nominated for the same training. 4. Use case ends. |
| Alternate Course of Action | 3.1 System displays error message on failure of database update for connection/system error. System displays alternate message if record already exists, or selected training status is “Blocked,” and use case ends. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user clicks the “View Nominations” tab from the application navigation menu. | 1. The system queries for nomination records from database. 2. The system displays table with nomination records if exist, else displays alternate message for record not found. 3. Use case ends. |  |
|  | Alternate Course of Action | * 1. System displays error page on any system or application error if occurs.   2. Each DM/TDH can view only his/her team nominations by comparing from the Emp table. | |  |
| *The View Nominations Use Case*  **Use Case**: **Popup Notification**  *The Popup Notification Use Case* | | | | |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-06* | |
| Use Case Name | *Popup Notification* | |
| Priority | High | |
| Actor | Developer, Department Manager, and Training Development Head | |
| Description | This use case describes how users view a role-based popup notification post logging on to the system. | |
| Precondition | User signs in to the system. | |
| Post-condition | If the use case is successful, the system displays role-based popup notification post logging on to the system and landing on the home page. | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user signs in to the system. | 1. The system verifies role, if role is Employee (EMP), then system queries from database for new training sessions where release date is between user’s last login date and current system date. If role is Department Manager (DM) or Talent Development Head (TDH), then system queries from database for nominations submitted where submission date is between user’s last login date and current system date. 2. The system displays popup notification in modal window if any record exists in result set, else system does not display any popup notification. 3. Use case ends. |
| Alternate Course of Action | 2.1 System displays error page if any application error occurs. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

**Use Case**: **Create Training**

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-07* | |
| Use Case Name | *Create Training* | |
| Priority | High | |
| Actor | Department Manager and Talent Development Head | |
| Description | This use case describes how user submits request with training details in the system for a new training requirement. | |
| Precondition | User signs in to the system. | |
| Post-condition | If the use case is successful, the system creates and saves training details record in the database and defines the training status as “Pending.” | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user is on the Trainings page of the application. 2. The user clicks the Create Training link.   4. The user enters details in respective form fields, such as training title, training duration, start date, end date, training objectives, and training location, and clicks the Submit button. | 3. The system displays the Create Training page.   1. The system checks that the form field values are complete and valid. 2. The system saves the details in database. 3. The system displays Manage Trainings page with the updated list of trainings. 4. Use case ends. |
| Alternate Course of Action | 5.1 If the training title, training duration, start date, end date, training objectives, and training location are empty or not valid, the system displays an error message.  6.1 System displays error page if there is any application or system error. | |

**Use Case**: **Update Training**

*The Create Training Use Case*

|  |  |
| --- | --- |
| Use Case Number | *UC-08* |
| Use Case Name | *Update Training* |
| Priority | High |
| Actor | Department Manager, Talent Development Head, and Training Hub Manager |
| Description | This use case describes how users update training details and/or status based on privileges assigned to the respective roles in the system. |
| Precondition | User signs in to the system. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Post-condition | If the use case is successful, the system updates training details and/or status successfully to a selected training. | |  |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user is on the Trainings or Manage Trainings page of the application. 2. The user clicks the Edit link of Action column provided to the respective training record.   6. The user edits the required form field details and clicks the Save button. | 1. The system displays Modify Training page along with the details of the training opted, already filled in the respective form fields. 2. The system verifies the role of the user and does not display training status and verifies the approval status form fields. 3. If role is Department Manager, the system disables all form fields except training status. If role is Training Hub Manager, the system displays status only. All form fields are enabled if role is Talent Development Manager. 4. The system verifies the form field values, if valid, the system updates the database record for the specific training. 5. The system displays the Manage Trainings page with the updated record set for the training. 6. Use case ends. |  |
|  | Alternate Course of Action | 5.1 If the respective form field(s) are empty or not valid, the system displays an error message.  3.1, 4.1, 6.1, 7.1 System displays error page if there is any application or system error. | |  |
| *The Update Training Use Case*  **Use Case**: **Delete Training** | | | | |
|  | Use Case Number | *UC-09* | |  |
|  | Use Case Name | *Delete Training* | |  |
|  | Priority | High | |  |
|  | Actor | Department Manager and Talent Development Head | |  |
|  | Description | This use case describes how users delete training record from the system, if training status is not “Approved,” and how the record is released in the training calendar. | |  |
|  | Precondition | User signs in to the system. | |  |
|  | Post-condition | If the use case is successful, the system deletes the selected training record from database successfully. | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user signs in to the system. 2. The user clicks the Trainings or Manage TNI tab from the application navigation menu.   4. The user clicks the Delete link of Action column against a training record opted.  6. The user clicks the Delete button. | 3. The system verifies the training status if not “Approved” and displays training calendar along with the “Delete” link in the “Action” column.  5. The system displays the Delete Training page along with the details of the training opted, already filled in the respective form fields.   1. The system deletes the training record from the database for the specific training. 2. The system displays the Manage Trainings page with the updated training list. 3. Use case ends. |  |
|  | Alternate Course of Action | 3.1, 5.1, 7.1, 8.1 System displays error page if there is any application or system error. | |  |
| *The Delete Training Use Case*  **Use Case**: **Approve Nomination** | | | | |
|  | Use Case Number | *UC-10* | |  |
|  | Use Case Name | *Approve Nomination* | |  |
|  | Priority | High | |  |
|  | Actor | Department Manager and Talent Development Head | |  |
|  | Description | This use case describes how users approve or reject a nomination submitted. | |  |
|  | Precondition | User signs in to the system. | |  |
|  | Post-condition | If the use case is successful, the system updates the approval status of a nomination as “Approved” or “Rejected.” | |  |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user is on the Nomination page if role is Talent Development Head, or Team Nominations of role is Department Manager. 2. The user clicks the Manage link provided in Action column.   4. The user selects option from Approval Status dropdown list, enters remarks in the Remarks for Approval/Rejection form field, and clicks the Submit button. | 3. The system displays Approve/Reject page filled with the nomination details to the respective form fields.   1. The system verifies form field data and updates nomination approval status. 2. The system displays the Manage Nomination page if role is Talent Development Head, or Team Nominations of role is Department Manager along with the updated list of nominations. 3. Use case ends. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Alternate Course of Action | 5.1 If the respective form field(s) are empty or not valid, the system displays an error message.  3.1, 5.1, 6.1 System displays error page if there is any application or system error. |  |
| *The Approve Nomination Use Case*  **Use Case**: **Approve Training**  *The Approve Training Use Case* | | | |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-11* | |
| Use Case Name | *Approve Training* | |
| Priority | High | |
| Actor | Talent Development Head | |
| Description | This use case describes how users approve or reject a training session submitted. | |
| Precondition | User signs in to the system. | |
| Post-condition | If the use case is successful, the system updates the approval status of a training session as “Approved” or “Rejected.” | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user is on the Manage Trainings page. 2. The user clicks the Edit link provided in the Action column.   4. The user selects option from Approval Status dropdown list, enters remarks in the Remarks for Approval/Rejection form field, and clicks the Submit button. | 3. The system displays the Modify Training page filled with the training details in the respective form fields.   1. The system verifies form field data and updates training approval status. 2. The system displays the Manage Trainings page. 3. Use case ends. |
| Alternate Course of Action | 5.1 If the respective form field(s) are empty or not valid, the system displays an error message.  3.1, 5.1, 6.1 System displays error page if there is any application or system error. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

**ACTIVITY DIAGRAMS**

**Activity diagram**: **Login activity**

Start

Display Sign-In page

Submit login credentials

Invalid login credentials

Valid login credentials

Display home page

*The Login Activity Diagram*

## Activity diagram: Logout activity

Stop



Start

Display sign-out page

Invalidates user session

Display login page

Stop

*The Logout Activity Diagram*

## Activity diagram: Search Training activity

Start

Display trainings page

Submit search text

Display search result

Stop

*The Search Training Activity Diagram*

## Activity diagram: Create Training activity

Start

Display add training page

Enter training details

If invalid training details

If valid training details

Save record into database



Stop

*The Create Training Activity Diagram*

## Activity diagram: Modify Training activity

Start

Display modify training page

Update training details

If invalid training details

If valid training details

Save record into database



Stop

*The Modify Training Activity Diagram*

## Activity diagram: Delete Training activity

Start

Display delete training page

Delete training details

Remove from database

Stop

*The Delete Training Activity Diagram*

## Activity diagram: Approve Existing Training activity

Start

Display search training details page

If status is rejected/pending/blocked

Check

Approve

status Not allowed for edit

Allowed for edit

*The Approve Existing Training Activity Diagram*

## Activity diagram: Approve Nomination activity

Stop

Start

Display training details page

Select and nominate training

Check nomination status

New nomination Approved

Update records into database

If already nominated Display error information

Reject nomination

Stop

*The Approve Nomination Activity Diagram*



## Activity diagram: Nominate for Training Approval activity

Start

Display nomination details page

Select nomination request for approval

Check for existing nomination

Approve

Update records into database Reject

Not allow for edit nomination

Stop



*The Nominate for Training Approval Activity Diagram*

## Activity diagram: Popup Notification activity

Start

Display home page

Record does not exist where training release date is between last sign-in date and current date

role= ‘EMP’

role= ‘DM’ or ‘TDH’

Record exists where training release date is between last sign-in date and current date

Fetch records where training release date is between last sign-in date and current date

Record does not exist where nomination submission date between last sign-in date and current date

Record exists where nomination submission date between last sign-in date and current date

Fetch records where nomination submission date between last sign-in date and current date

Display records in popup modal screen

Stop



*The Popup Notification Activity Diagram*

**CLASS DIAGRAMS**

**Class diagrams for the Models**, **ViewModel**, **and Controller**

|  |
| --- |
| **modelEMP** |
| +Employee\_Id : String  +Name : String  +Address : String  +Phone : String  +Email : String  +DeptId : String  +ManagerId : String |

|  |
| --- |
| **modelLogin** |
| +Uid : String  +Password : String  +Role : String  +Lastsignedin : DateTime |

*The modelLogin Class Diagram*

*The modelEMP Class Diagram*

|  |
| --- |
| **modelNomination** |
| +Nomination\_Id: Integer  +empid : String  +Tid : Integer  +IsApproved : String  +Remarks: String  +ManagerId: String  +Role: String  +Ndate: DateTime |

*The modelNomination Class Diagram*

*The modelSearchTraining Class Diagram*

|  |
| --- |
| **modelSearchTraining** |
| +Tid: String  +Title : String  +Duration : Integer  +StartDate: DateTime  +EndDate: DateTime  +Objective: String  +Hub: String  +Role: String  +IsApproved:String |

|  |
| --- |
| **modelTraining** |
| +Tid: Integer  +Title : String  +Duration : Integer  +StartDate: DateTime  +EndDate: DateTime  +Objective: String  +Hub: String  +Role: String  +IsApproved:String  +Remarks : String  +Status : String  +Rdate : DateTime |
| +TitleList : Class  +SearchFilter : Class |

|  |
| --- |
| **LoginController** |
| -db : GenericRepoContext  -repository : IGenericRepository<Login>  -emprepository : IGenericRepository<EMP>  -rolelist : List<SelectListItem>  -role : var  -log : var  -name : var  +LastLogin : Session  +Uid : Session  +Role : Session  +Name : Session |
| #Login() : ActionResult  #Login(Login : Object) : ActionResult  #LogOff() : ActionResult |

*The modelTraining Class Diagram*

|  |
| --- |
| **NominationViewModel** |
| +Nomination : IEnumerable Class  +Training : IEnumerable Class  +Emp : IEnumerable Class  +Login : IEnumerable Class  +Trainings : List Class |

*The NominationViewModel Class Diagram*

|  |
| --- |
| **interface IGenericRepository<T> where T : class** |
| +SelectAll() : IEnumerable<T>  +SelectByID(object id): T  +Insert( obj T) : void  +Update( obj T) : void  +Delete( obj T) : void  +Save( ) : void |

*The IGenericRepository Interface Diagram*

*The LoginController Class Diagram*

|  |
| --- |
| **GenericRepoContext** |
| +GenericRepoContext () : base("STMSContext")  + Login : DbSet<Login>  +EMP : DbSet<EMP>  +Trainings : DbSet<Training>  +Nominations : DbSet<Nomination> |

*The GenericRepoContext Class Diagram*

|  |
| --- |
| **HomeController** |
| #Index() : ActionResult |

*The HomeController Class Diagram*

|  |
| --- |
| **TrainingController** |
| -db : GenericRepoContext  -training : Training  -mytraining : Training  -hublist : List<SelectListItem>  -approvedlist: List<SelectListItem>  -approvedlist : ViewBag  -hubslist : ViewBag  -statuslist : List<SelectListItem>  -hublist: List<SelectListItem>  -role : var  -status: ViewBag |
| #Index() : ActionResult  #Details(id : String) : ActionResult  #Create() : ActionResult  #Create(training : Training) : ActionResult  #Edit( id : int) : ActionResult  #Edit(training : Training) : ActionResult  #Delete( id : int) : ActionResult  #DeleteConfirmed(id : int) : ActionResult  #Dispose(disposing : bool) : Protected |

*The TrainingController Class Diagram*

|  |
| --- |
| **NominationController** |
| -db : GenericRepoContext  -repository : IGenericRepository<EMP>  -nrepository: IGenericRepository<Nomination>  -ntraining: IGenericRepository<Training>  -details : var  -empid : var  -role : var  -lastlogin : DateTime  -logindata : var  -traindata : var  -NewMessage : Session  -nomdata : var  -empdata : var  -ntitle : var  -data : var  -num : Nomination  -managerid : var  -nomination : Nomination  -approvedlist : List<SelectListItem> |
| #Notification () : ActionResult  #Index() : ActionResult  #AllNomination () : ActionResult  #MyNomination () : ActionResult  #Nominate(Tid : int ) : ActionResult  #Edit (id : int) ActionResult  #Edit ( nomination : Nomination) :ActionResult  #Dispose(disposing : bool) : protected |

*The NominationController Class Diagram*

|  |
| --- |
| **GenericRepository<T>** |
| -db : GenericRepoContext  - table : DbSet<T> |
| #SelectAll () : IEnumerable<T>  #SelectByID (id : Object) : T  #Insert(T : Object) : void  #Update(T : Object) : void  #Delete(id : Object) :void  #Save() : void |

*The GenericRepository Class Diagram*

|  |
| --- |
| **SearchController** |
| -repository : IGenericRepository<Training>  -objTraining : List<Training>  -\_objTraining: Trainings  -data: SearchFilter  -TypeList: List<TitleList>  -list : var  +t1 : TitleList  +service: var |
| #Index () : ActionResult  #Index (title :String) : PartialViewResult  #All() : ActionResult |

*The SearchController Class Diagram*

The preceding class diagram contains the following model classes:

* + - modelEMP: This class is used for retrieving the field values for employee details, such as

Employee\_Id, Name, Address, Phone, Email, DeptId, and ManagerId.

* + - modelLogin: This class is used for retrieving the field values for login details for all user types, such as Uid, Password, Role, and Lastsignedin.
    - modelNomination: This class is used for retrieving the field values for nomination details, such as

Nomination\_Id, empid, Tid, IsApproved, Remaraks, ManagerId, Role, and Ndate.

* + - NominationViewModel: This class is used for retrieving the class values, such as Nomination, Training, Emp, Login, and Trainings.
    - modelSearchTraining: This class is used for retrieving the search training details, such as Tid, Title, Duration, StartDate, EndDate, Objective, Hub, Role, IsApproved, and Status.
    - modelTraining: This class is used for retrieving the training details, such as Tid,Title, Duration, StartDate, EndDate, Objective, Hub, Role, IsApproved, Remarks, Status, and Rdate. Another two classes TitleList and SearchFilter are used for retrieving the training details.

The preceding class diagram contains the following controller classes:

* + - HomeController: This class is used for initializing the following function.
      * Index(): This function is used for loading the index page for each user.
    - LoginController: This class is used for initializing the following functions.
      * Login(Login): It validates the entered user id and password. If the validation is successful, it redirects to the Notification view page, otherwise page will redirect to the Login page. The Notification view page will be only displayed if any training or nominations between the last login date and the current date is arrived; otherwise the rolse-based index view page will be loaded.
      * Login(): It will validate the login session details. If any user is logged in, the users role will be stored in ViewBag variable; otherwise, page will be redirected to the Training Index page.
      * LogOff(): This function will remove all session details and log off from the system. After successfully logging off from the application, the users will be redirected to the role-based Home Index page.
    - NominationController: This class is used for initializing the following functions.
      * Notification(): This function will authenticate role-based user’s login. If an employee is logged on to the application and any new training is initiated, the new training notification will be displayed. If DM logs on to the application and any new nomination is submitted by an employee, the system displays the nomination notification. Similarly, if TDH logs on to the application and new nomination is submitted by DM, then the system displays the nomination notification.
      * AllNomination(): This function is used for displaying the role-based nomination details. If DM is logged on to the application, then this function will return all employees’ nominations, such as Pending, Approved, and Rejected. Whereas, if TDH or THM are logged on to the application, then this function will return all DM nomination details, such as Pending, Approved, and Rejected.
      * Index(): This function will display the training nomination notification details for each

role-based users. If a developer is logged on to the application, it will display all nomination done by itself for each training. If DM is logged on to the application, it will display all nomination details of Employee from database. If TDH is logged on to the application, it will display all nomination details of DM from database.

* + - * MyNomination(): This function will allow each DM to view the individual nomination details.
      * Nominate(Tid): This function will allow each user to nominate for any approved training. In addition, it ensures that each user can nominate him/her self once for each approved training. If users are able to successfully nominate, the notification message “Your Nomination for the training has been forwarded to the concerned department!” should be displayed; otherwise, an error message “You have already submitted your nomination for the training” along with existing nomination id should be displayed.
      * Edit(id): This function is used to approve or reject any nomination, which are either nominated by a developer or DM.
      * Edit(Nomination): This function is used to get the nomination status and Nomination Date

details from Edit(id)function and update into database.

* + - SearchController: This class is used for initializing the following functions.
      * Index(): This function will display all approved training details to all role-based users and will also check if the logged on user’s session is valid. If logged on user’s session is invalid, then redirect the current page to the Login page.
      * Index(Title): This function will retrieve all distinct training title from a database and display them to all role-based users. Using this, all users can search those training sessions that already exist into the database. This partial function will be invoked by the Index() function.
      * All(): This function will display all approved training details to all role-based users.
    - TrainingController: This class is used for initializing the following functions.
      * Index(): This function will return all the training details from database.
      * Create(): This function will display the training details from database.
      * Create(Training): This function will create the new training details into database.
      * Edit(Id): This function will return the role-based options to each user into application. If TDH is logged on to the application, then this function will display the "Approved,” “Reject,” or “Blocked” options to all TDH users. Similarly, if THM is logged on to the application, then training status should be “Not Started,” “OnGoing,” “Completed,” or “Cancelled.”
      * Edit(Training): This function will update all training details into database and redirect the edit page into the Training Index page.
      * Delete(): This function will find the selected training details from database.
      * DeleteConfirmed(id): This function will delete the selected training details from database.

The preceding class diagram contains the following Generic Repository classes:

* + - GenericRepoContext: This class is used for initializing the entity framework database classes, such as Login, Emp, Training, and Nominations.
    - GenericRepository<T>: This class is used for implementing Generic Repository and database related functionality, such as SelectAll(), SelectByID(Object), Insert(T), Update(T), Delete(T), and Save(). This class has generic method description; therefore, any controller class can implement it.
    - IGenericRepository<T>: This interface is used for initializing all the functions defined by

GenericRepository<T> classes.

**ENTITY RELATIONSHIP DIAGRAM**

IsApproved

Duration StartDate

EndDate

Objective Remarks

Status

Role

Training

Title

Tid

Rdate Hub

Accepts

Role

Tid

IsApproved

|  |  |
| --- | --- |
|  |  |
| Nomination | |

ManagerId Nomination\_Id

Remarks

empid

Ndate

Creates Submits

Address Employee\_Id

ManagerId

Name

Emp

Has

Uid

Role

Login

Lastsignedin

Password

DeptId

Phone

Email



*The Entity Relationship Diagram of Smart Training Management System*

**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware**: 3.0 GHz or faster processor

: 2 GB of RAM

: 80 GB 7200 RPM hard drive

: DVD Drive

: 10/100 NIC

: Optical wheel mouse

: Keyboard

: 17” color monitor

**Operating system**: Windows 8.1 (X64)

**Software**: SQL Server 2012

:.NET 4.5

: Visual Studio 2012

: Entity Framework 6.0

**PROJECT ON SMART TRAINING MANAGEMENT SYSTEM (STMS) – MOBILE APP**

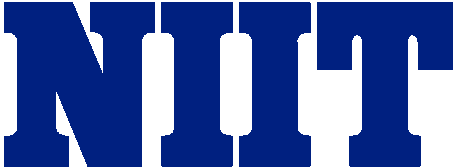
Developed by



**Sample Project Documentation**

**Name**: Patrick Smith

**Reg. No.**: RNG3366



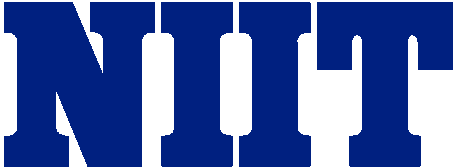
**SMART TRAINING MANAGEMENT SYSTEM (STMS)**

**Batch Code**: 001 **Start Date**: 1/09/2016 **End Date**: 15/09/2016

**Name of the Coordinator**: John Williams

**Name of the Developer**: Patrick Smith

**Date of Submission**:16/09/2016



# CERTIFICATE

This is to certify that this report, titled Smart Training Management System (STMS) Capstone Project on Mobile App Development, embodies the original work done by Patrick Smith in partial fulfillment of the course requirement at NIIT.

Coordinator: John Williams

**ACKNOWLEDGEMENT**

We have benefited a lot from the feedback and suggestions given to us by Mr. John Williams and other faculty members.

**REQUIREMENT ELICITATION DOCUMENT**

Q1. What are the objectives of your company?

Q2. Tell us briefly about the stakeholders of the company. Q3. How many departments does your company have?

Q4. How many employees work in your company? Q5. How many roles types does your company have?

Q6. How many roles do you have for each training requirement? Q7. Who are your primary competitors?

Q8. How do the operations of your competitors differ from your operations? Q9. How do you maintain the training information?

Q10. How do you maintain the training nomination information?

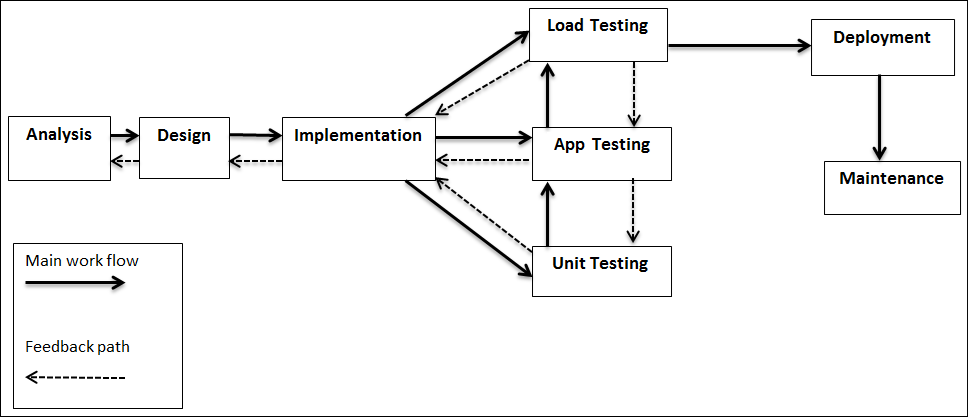
Q11. How do you keep track of nominations that are submitted for a day? Q12. How many trainings can an employee submit at a time?

Q13. How do you inform a participant if a nomination is already submitted for a training? Q14. How is a training detail modified or canceled?

Q15. Do you provide any status change for a training execution?

Q16. How do you restrict a training from being modified or deleted if the training is released? Q17. How many reports are required by the company?

**PROJECT LIFE CYCLE MODEL**



*The Project Life Cycle Model*

**VISION DOCUMENT**

### Vision of the Project

To create an efficient mobile app solution for Training Hut Inc. company for its training management services.

### About the Company

Training Hut Inc., founded in 2005, currently operates within Boston and Minnesota. The company provides professional training programs to their employees in different corporate offices or client locations. Currently, Training Hut Inc. is in the process of establishing itself across the USA, UK, Australia, and Asia.

### Requirements Summary

The management of Training Hut Inc. wants to automate its training management processes. In the second phase of the solution development, the management wants the respective stakeholders to be able to access the envisioned solution through a mobile app. In addition, the management wants the envisioned mobile app to work on most of the mobile platforms, such as Android, Windows, and IOS. Users can create, publish, search, and nominate for trainings through the app. The management also wants the app to be able to authenticate a user to authorize them for role-based actions.

### Project Goals

The project goals are:

* + - Analyze the requirements to create the solution.
    - Design the application based on the results of the analysis phase.
    - Create a mobile app compatible on different mobile platforms, such as Android, Windows, and IOS.
    - Test the implemented application.

### Project Stakeholders

The primary stakeholders of the projects are:

* + - Department Managers (DM) of Training Hut Inc.
    - Developers (with EMP role) of Training Hut Inc.
    - Talent Development Head (TDH) of Training Hut Inc.
    - Training Hub Managers (THM) of Training Hut Inc.
    - Management of EarnestPro Inc.

**PROJECT PLAN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***S. No*** | ***Task Name*** | ***Planned Start Date*** | ***Planned Finish Date*** | ***Actual Start Date*** | ***Actual End Date*** | ***Person Responsible*** |
| *1* | *Analyze the project requirements* | *1/09/2016* | *1/09/2016* | *1/09/2016* | *1/09/2016* | *Patrick* |
| *2* | *Design the project* | *2/09/2016* | *5/09/2016* | *2/09/2016* | *5/09/2016* | *Patrick* |
| *3* | *Implement the project* | *8/09/2016* | *12/09/2016* | *8/09/2016* | *12/09/2016* | *Patrick* |
| *4* | *Test the project* | *13/09/2016* | *13/09/2016* | *13/09/2016* | *13/09/2016* | *Patrick* |
| * *Patrick will be working four hours per working day to complete the project.* * *Saturdays and Sundays being holidays are not considered as working days.* | | | | | | |

**WEEKLY STATUS REPORT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Weekly Status Report** | | | | |
| **Developer**: Patrick **Phase**: 1 and 2  **Period**: **From** 1/09/2016 **to** 5/09/2016 | | | | |
| **Activity/Artifact** | **Responsibility** | **Planned Completion Date** | **Completion Status (%)** | **Reason for Incompletion** |
| * *Create a requirement elicitation document for the project to gather more specific information on how the system should work.* * *Create a presentation of the project life cycle model for the client/stakeholders.* * *Create a vision document of the project.* * *Create the project plan.* * *Create a weekly status report format.* * *Create a functional requirement document of the system based on the user requirements gathered from the client.* * *Model the system using UML use case diagrams.* * *Create use case descriptions.* * *Create the activity diagrams of the system.* | Patrick | 5/09/2016 | 100% | NA |

**FUNCTIONAL REQUIREMENT DOCUMENT**

The functional requirements of the system have been derived from the user requirements and interviews with the project stakeholders. The functional requirements have been designed to highlight the services that the system should provide. These requirements specify how the developers should design and develop the system for devices such as PCs, tabs, and smart phones in order to meet the expectations of the project stakeholders.

### View training calendar

1. The app should allow all users to view the trainings list with details approved and released by the Talent Development Head.
2. The app should display all the training records to the Talent Development Head.

### Create a training

1. The app should allow the Department Manager and Talent Development Head to create a new training and submit the details.
2. The app should allocate approval status as “Pending,” by default, for newly created training.
3. The app should generate the training ID automatically.

### Approve/reject a training

1. The app should allow the Talent Development Head to update the approval status of a training as “Approved” or “Rejected.”
2. The app should allow a user to update the training calendar and add-on a training that is approved or remove a training if rejected.

### Search a training

1. The app should allow users to search for a training by the training title.

### Submit nomination for a training

1. The app should allow developers (with EMP role) to submit their nomination for a training.
2. The app should allocate a nomination approval status as “Pending,” by default, when newly submitted by a user.

### Approve/reject a nomination

1. The app should allow department managers to view, select, and perform actions for approval or rejection of a nomination.

### User login

1. The app should allow users to login and perform role-based actions by specifying a user name and password. Employee details are predefined and already given in the app.
2. The app should authenticate the login credentials before providing access to enter the application.
3. The app should provide a role-based action menu to the user when the authentication is successful.

### View nomination details

1. The app should allow users to view the list of submitted nominations.

**UML USE CASE DIAGRAM**

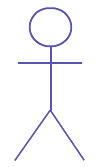
Nominate for Training

Search Training

Smart Training Management System

**Developer**

Login



Logout

View Nominations

*The Use Case Diagram for a Developer*

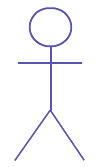
Search Training

Login

Smart Training Management System

**Department Manager**

View Nomination Logout



Create Training

Approve Nomination

*The Use Case Diagram for the Department Manager*

Approve Training

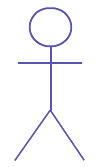
Search Training

Login

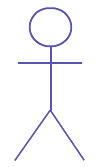
Smart Training Management System

**Talent Development Head**

View Nomination Logout



*The Use Case Diagram for the Talent Development Head*



Search Training

Login

Smart Training Management System

**Training Hub Manager**

Logout

View Nomination

*The Use Case Diagram for the Training Hub Manager*

### USE CASE DESCRIPTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case**: **Login**  *The Login Use Case*  **Use Case**: **Logout** | | | |
|  | Use Case Number | *UC-02* |  |
|  | Use Case Name | *Logout* |  |
|  | Priority | High |  |
|  | Actor | Developers, Department Manager, Talent Development Head, and Training Hub Manager |  |
|  | Description | This use case describes how a user logs out of the system. |  |
|  | Precondition | User has signed into the app. |  |
|  | Post-condition | If the use case is successful, the user’s session will be invalidated and the user will be returned back to the login page. |  |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-01* | |
| Use Case Name | *Login* | |
| Priority | High | |
| Actor | Developer, Department Manager, Talent Development Head, and Training Hub Manager | |
|  |  | |
| Description | This use case describes how the users sign into the app. | |
| Precondition | App is started and currently displaying the login page. | |
| Post-condition | If the use case is successful, the user gets access to their respective home pages as per their roles, otherwise an error message is displayed. | |
| Basic course of Action | **User Action** | **System Response** |
| 1. The user is on the login page. 2. The user enters username and password into the respective fields and clicks the “SIGN IN” button. | 1. The app authenticates the user’s login credentials. 2. The app displays the “Action” page if the authentication is successful. 3. Use case ends. |
| Alternate Course of Action | 3.1 If the login credentials do not match, the system displays an error message. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user has logged in. 2. The user clicks the sign out button situated at the top right corner on the app header. | 1. The app invalidates the user’s session and returns to the login page. 2. Use case ends. |  |
|  | Alternate Course of Action |  | |  |
| *The Logout Use Case*  **Use Case**: **Search Training**  *The Search Training Use Case*  **Use Case**: **Nominate for Training** | | | | |
|  | Use Case Number | *UC-04* | |  |
|  | Use Case Name | *Nominate for Training* | |  |
|  | Priority | High | |  |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-03* | |
| Use Case Name | *Search Training* | |
| Priority | High | |
| Actors | Developers, Department Manager, Talent Development Head, and Training Hub Manager | |
| Description | This use case describes how users can view and search for a training. | |
| Precondition | User has signed into the app. | |
| Post-condition | If the use case is successful, the user can view and search for a released training in the app. | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user navigates to the “Action” screen after successful login. 2. The user clicks the “View Training Calendar” button.   4. The user enters a search value in the search box. | 3. The app displays the training calendar with the training details approved by Talent Development Head.   1. The app displays filtered records based on the search value. 2. Use case ends. |
| Alternate Course of Action | 3.1 If there is no record(s) fetched, app displays an alternate message on the empty record list. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Actor | Developer | |  |
|  | Description | This use case describes how the user submits a nomination for training. | |  |
|  | Precondition | User has signed into the app and is currently on the “Training Calendar” screen displaying a list of published trainings. | |  |
|  | Post-condition | If the use case is successful, the user submits a nomination for training and views the summary of nominations submitted till date. | |  |
|  | Basic course of Action | **User Action** | **System Response** |  |
| 1. The user is on the “Training Calendar” page. 2. The user clicks the “Nominate” button for the training opted for submitting nomination. | 1. The app identifies the training ID and saves the nomination for the selected training. 2. The app fetches the updated nomination list saved in local storage. 3. The app displays the “Nomination Info” page displaying the list of nominations saved. 4. Use case ends. |  |
|  | Alternate Course of Action |  | |  |
| *The Nominate for Training Use Case*  **Use Case**: **View Nominations**  *The View Nominations Use Case* | | | | |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-05* | |
| Use Case Name | *View Nominations* | |
| Priority | High | |
| Actor | Developers, Department Manager, Talent Development Head, and Training Hub Manager | |
| Description | This use case describes how users view role-based nomination report. | |
| Precondition | User signs into the system and is currently viewing the “Action” screen. | |
| Post-condition | If the use case is successful, the app displays the list of nominations submitted. | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user clicks the “View Nominations Info” button from the “Action” screen after successful login. | 1. The app fetches the list of nominations with details. 2. The app displays the list of nominations. 3. Use case ends. |
| Alternate Course of Action | 3.1 The app displays an alternate message if record(s) not found. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case**: **Create Training**  *The Create Training Use Case*  **Use Case**: **Approve Nomination** | | | |
|  | Use Case Number | *UC-7* |  |
|  | Use Case Name | *Approve Nomination* |  |
|  | Priority | High |  |
|  | Actor | Department Manager |  |
|  | Description | This use case describes how the user approves or rejects a submitted nomination. |  |
|  | Precondition | User signs into the system and is currently viewing the “Action” page. |  |
|  | Post-condition | If the use case is successful, the app updates the approval status of a nomination as “Approved” or “Rejected.” |  |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-06* | |
| Use Case Name | *Create Training* | |
| Priority | High | |
| Actor | Department Manager | |
| Description | This use case describes how the user submits a request with training details for a new training requirement. | |
| Precondition | User signs into the system and is currently viewing the “Action” page. | |
| Post-condition | If the use case is successful, the app creates a new training record, saves it in local storage, and defines the training status as “Pending.” | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user clicks the “Add Training” button from the “Action” page.  3. The user enters details in the respective fields, such as training title, training duration, start date, end date, training objectives, and training location then clicks the Submit button. | 2. The app displays the “Add Training” page with required fields, such as training title, training duration, start date, end date, training location, and training objectives.   1. The app generates the training Id automatically. 2. The app saves the training details in local storage. 3. The app displays the “Action” page. 4. Use case ends. |
| Alternate Course of Action |  | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Basic Course of Action | **User Action** | **System Response** |  |
| 1. The user clicks the “Manage Nomination” button.  3. The user clicks the “Approve” or “Reject” button. | 2. The app displays the “Manage Nomination” page listing the nomination details along with the “Approve” and “Reject” buttons for each of the listed items.   1. The app updates the nomination status as “Approved” or “Rejected” for the selected nomination and saves the record in local storage. 2. The app redirects to “Action” page. 3. Use case ends. |  |
|  | Alternate Course of Action |  | |  |
| *The Approve Nomination Use Case*  **Use Case**: **Approve Training**  *The Approve Training Use Case* | | | | |

|  |  |  |
| --- | --- | --- |
| Use Case Number | *UC-8* | |
| Use Case Name | *Approve Training* | |
| Priority | High | |
| Actor | Talent Development Head | |
| Description | This use case describes how the user approves or rejects a training submitted. | |
| Precondition | User signs into the app and is currently viewing the “Action” page. | |
| Post-condition | If the use case is successful, the app updates the approval status of a training as “Approved” or “Rejected.” | |
| Basic Course of Action | **User Action** | **System Response** |
| 1. The user clicks the “Manage Training” button.  3. The user clicks the “Approve” or “Reject” button. | 2. The app displays the “Manage Training” screen listing the training details along with the “Approve” and “Reject” buttons.   1. The app updates the training approval status for the selected training and saves in local storage. 2. The app displays the “Action” page. 3. Use case ends. |
| Alternate Course of Action | 2.1 App displays alternate message if no record exists. | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |  |
|  |  |
|  |  | |

**ACTIVITY DIAGRAMS**

**Activity diagram**: **Login activity**

Start

Display login screen

Submit login credentials

Invalid login credentials

Valid login credentials

Display home screen

*The Login Activity Diagram*

## Activity diagram: Logout activity

Stop



Start

Nullify session objects

Invalidates user session

Display login screen

Stop

*The Logout Activity Diagram*

## Activity diagram: Search Training activity

Start

Display training list

Submit search text

Display search result

Stop

*The Search Training Activity Diagram*

## Activity diagram: Create Training activity

Start

Display add training screen

Accept training details

If invalid training details

If valid training details

Save record into database



Stop

*The Create Training Activity Diagram*

## Activity diagram: Nominate for Training activity

Start

Display trainings list

Display nomination info screen

Stop

User submits for nomination

Save new nomination record in local storage

Generate nomination Id

Accept training ID

*The Nominate for Training Activity Diagram*

## Activity diagram: Approve Nomination activity

Start

Display nominations list

Accept user action to approve or reject

On reject

On approve

Update nomination status



Display action screen

Stop

*The Approve Nomination Activity Diagram*

## Activity diagram: Approve Training activity

Start

Display trainings list

Accept user action to approve or reject

On reject

On approve

Update trainings approval status



Display action screen

Stop

*The Approve Training Activity Diagram*

## Activity diagram: View Nominations activity

Start

User requests for nominations info screen

Fetch nomination records from local storage

False

Check if record exists

True

Display records on nomination info screen



Display message on nomination page

Stop

*The View Nominations Activity Diagram*

**ENTITY RELATIONSHIP DIAGRAM**

IsApproved

Duration StartDate

EndDate Objective

Training

Title

Tid

Accepts

Role

Tid

IsApproved

|  |  |
| --- | --- |
|  |  |
| Nomination | |

ManagerId Nomination\_Id

Remarks

Status

Role

Releasedate Venue

Tstamp

Empi

Remarks

Creates Submits

Address Employee\_Id

ManagerId

Name

Emp

Has

Uid

Role

Login

Lastsignedin

Password

Dept\_Id

Phone

E-mail

*The Entity Relationship Diagram of Smart Training Management System*



**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware**: 3.0 GHz or faster processor

: 4 GB of RAM

: 500 GB 7200 RPM hard drive

: DVD Drive

: 10/100 NIC

: Optical wheel mouse

: Keyboard

: 17” color monitor

**Operating system**: Windows 8.1 (X64)

**Software**: Microsoft Visual Studio with Apache Cordova Extension