COM741 Web Applications Development

Practical 10

Adding Features Described via User Stories

The process of defining "functional requirements" as user stories and then implementing these requirements, is something you will need to be able to do for your final project dissertation.

1. Ticket Resolution Feature:

   As a user (admin/manager role), **I would** like to be able to record Ticket resolution details when closing the ticket, **so that** this information can be used as a resource when resolving other tickets.

   Steps to implement feature include:

   - Amend the SMS.Data Ticket model to include `ResolvedOn` (DateTime) and `Resolution` (string) properties.
   - Update the service `CloseTicket` interface and implementation to accept the additional `string resolution` parameter and ensure the ticket Resolution and ResolvedOn values are updated.
   - Amend CloseTicket tests to prove the operation of the amended ticket service methods
   - Amend the Ticket `Index.cshtml` view to contain a link to each Tickets "Details" action (removing Close form)
   - Complete the Ticket `Details.cshml` view to display the Ticket. Add a button that displays the close ticket modal. See example in Student Details view that displays the Delete modal. Use a modal id of e.g. "CloseTicketModal"
   - Complete the Ticket `_CloseModal.cshtml` partial (see example in student delete modal). Ensure the you give the modal the same id as configured in the close modal button on the details page e.g. "CloseTicketModal".
   - Amend the `TicketController Close` action to call the service method to close the ticket, passing in the resolution contained in the model. Also use bind to ensure we only bind the required values Id and Resolution.

2. Ticket Search Feature:

   As a user (admin/manager role) **I can** search all, open or closed ticket issues or student names using a search query, **so that** I can easily find relevant tickets

   Steps to implement feature include:

   - Complete the outline service method `public IList<Ticket> SearchTickets(TicketRange range, string query) {...}` Note use of `TicketRange` enum found in the `SMS.Data.Models` namespace

   - Verify operation via additional test cases (see outline tests)

   - Create a `TicketSearchViewModel` in `SMS.Web.Models` namespace to contain properties `string Query, TicketRange Range` and `IList<Ticket> Tickets`.

   - Create a `_Search.cshtml` partial form in Ticket views folder to collect search data (Query, Range)

   - Complete the outline `Search` action in the Ticket controller. This should use the Query and Range atttibutes of the viewmodel parameter to call the new SearchTicket service method and assign the results to the view model Tickets property, before returning the View with the viewmodel as a parameter

   - Complete the outline `Search.cshtml` view by copying the table from the existing Index view and amending as necessary to loop over the tickets contained in the view model.

3. Use dependency injection system to

   - add a Scoped instance of IStudentService as StudentServiceDb in Program.cs
   - configure the Student/Ticket and User controllers to use the dependency injection system and verify the application still works