1. Group information

The following section provides basic information on our team members and describes various tools and platforms used by our team for this project.

* 1. Member details

Our team has 3 members. Their information is listed below in the format [Full Name – Student Code]:

* Vo Van Tuong Quang – GCS190882
* Pham Chi Bao - GCS190381
* Mai Phuoc Thien - GCS190733
  1. Working platforms

Communication across team members was done informally with Messenger groupchats and in-person meetings in class. Formally tracked communication was largely through a Discord server.

The server had 3 main text channels and 2 voice channels for spoken communication online:

* #freetalk was for open discussions in text.
* #deadlines-and-deliverables was to track any major work products, as well as setting times to submit.
* #reports was for individual reports, sent when minor progress or issues arose.
* VC1 was the main spoken communication channel, with VC2 being a standby lobby for anyone that is available for contact but is not partaking in a discussion and wants to be undisturbed.

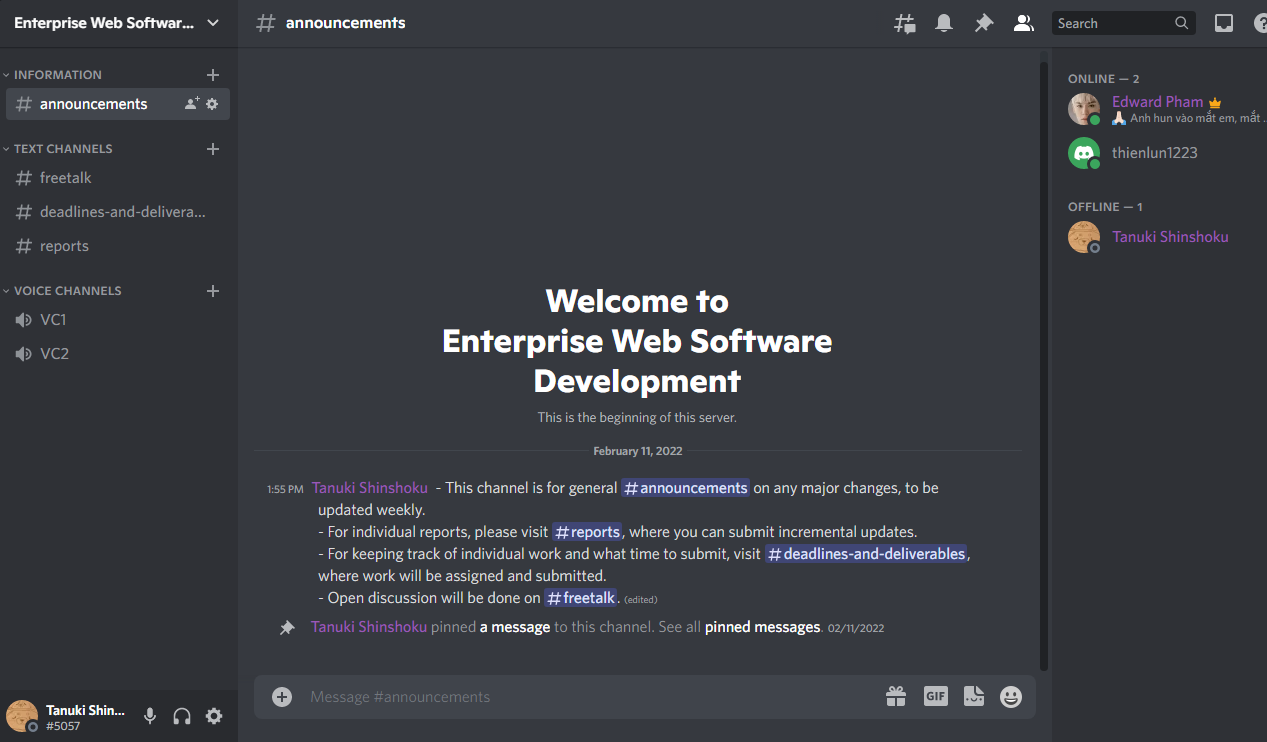


Figure - Discord server welcoming message

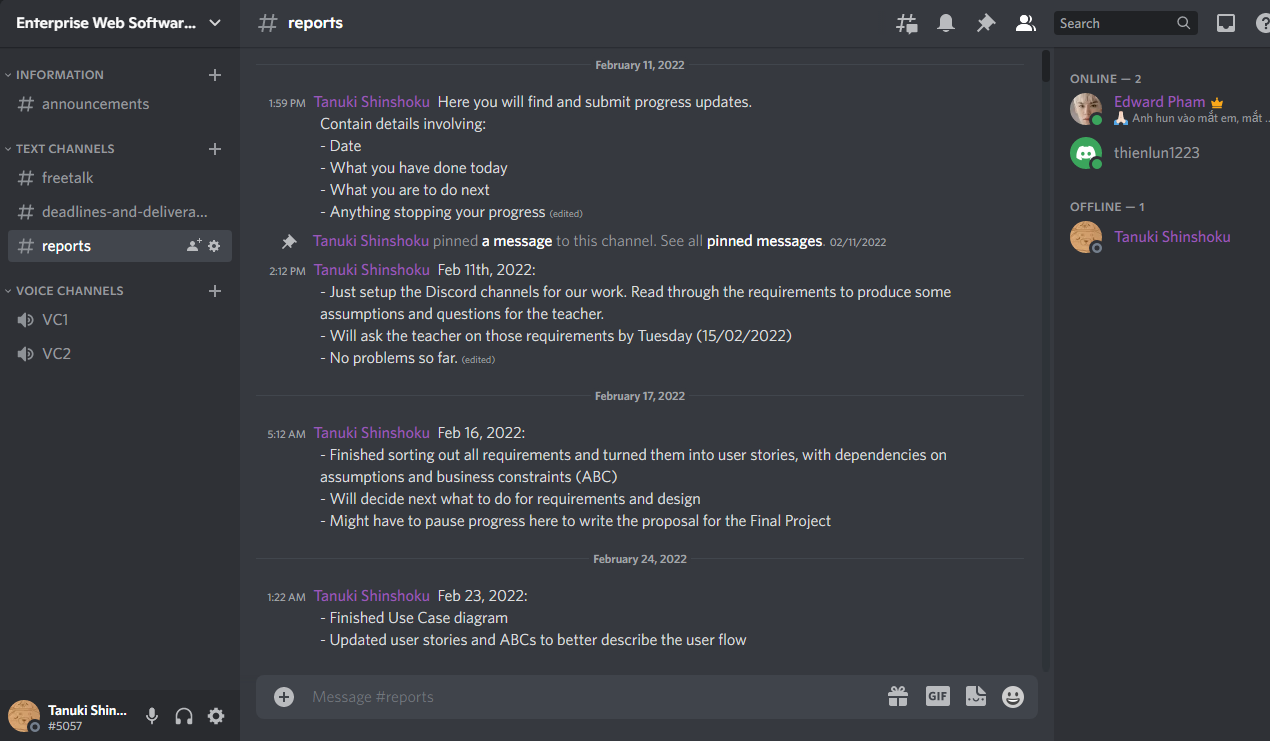


Figure - First sprint reports in the Discord server

We managed our work products through a group repository Google Drive (Link: [drive.google.com/drive/folders/1w8v\_bVAIHMD5fy7QK4PS5Og40pAB4lwC?usp=sharing](https://drive.google.com/drive/folders/1w8v_bVAIHMD5fy7QK4PS5Og40pAB4lwC?usp=sharing)) and managing different versions through our Discord’s #deadlines-and-deliverables channel.

1. Backlog

The project officially started in January 23rd and ended on April 18th, roughly a duration of 12 weeks. Each of our sprints lasted 2 weeks, making a total of 6 sprints throughout. Progress and requirements was tracked largely through user stories, and attached to each user story is an assumption/business constraint. Priorities scores and statuses for each user story was given to track their importance and progress, with the following syntax:

* Status:
  + New: For new user stories that have not been started.
  + Active: For user stories in progress and under development.
  + Closed: For user stories that are not to be addressed onwards, either due to time constraints or because implementation is complete.
* Priority:
  + 1: This user story is a must-have and has to be addressed first
  + 2: This user story is a must-have but does not have to be addressed immediately
  + 3: This user story is a should-have but is optional depending on resources and time.
  + 4: This user story is a nice-to-have and so is least prioritized.
  1. Sprint1: Gathering requirements

The first sprint was spent to properly understand and specify requirements for our system. We took to reading the coursework’s scenario and making a draft of the requirements and setting some initial assumptions to be confirmed with the lecturer that our interpretation is reasonable.

The following table describes constraints and assumptions derived in the first sprint.

|  |  |
| --- | --- |
| ID | Assumption/Business Constraints |
| ABC1 | Any 1 QA coordinator is attached to only 1 Department. |
| ABC2 | Staff members only have to agree to the Terms and Conditions once before being allowed to submit. |
| ABC3 | Ideas can be uncategorized. (The category is “null”) |
| ABC4 | Ideas can have their category changed by the author before and after posting. |
| ABC5 | Categories can only be deleted when they are not in use. (No ideas are tagged with that category) |
| ABC6 | Staff members can opt to anonymously submit an idea or comment. |
| ABC7 | Staff can only submit ideas during an open duration for submission; after the idea submission closure date, staff cannot post new ideas. |
| ABC8 | Staff can only comment on ideas during an open duration for commenting per idea; after that idea’s commenting closure date, staff cannot post new comments on that idea. |
| ABC9 | The system displays ideas by departments. |
| ABC10 | A department’s list of ideas needs to be paginated (5 ideas per page). |
| ABC11 | Uploaded documents can be downloaded in a ZIP file. |
| ABC12 | The interface must be suitable for all devices (e.g. mobile phones, tablets, desktops). |
| ABC13 | 1. A number of reports need to be made available. For example:    1. Statistics       1. Number of ideas made by each Department.       2. Percentage of ideas by each Department.       3. Number of contributors within each Department.    2. Exception reports       1. Ideas without a comment.       2. Anonymous ideas and comments. |

The following describes user stories created in the first sprint.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7 | 1 | New |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 2 | New |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 2 | New |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC10 | 1 | New |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8 | 3 | New |
| US6 | All staff can give thumbs up or down once per idea. |  | 4 | New |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments |  | 4 | New |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 2 | New |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | New |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | New |
| US11 | The QA manager can add categories. |  | 2 | New |
| US12 | The QA manager can delete categories | ABC5 | 2 | New |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | New |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | New |

In short, the most basic functions of a content management system were specified and prioritized, but no implementation work could yet be done.

* 1. Sprint2: Picking technology, designing the system, updating requirements

The design work began in the second sprint, including:

* Selection of the technology stack
* Drawing up design diagrams (ERD and Use Case Diagrams)

The chosen framework to develop this project was ASP.NET MVC and Entity Framework, seeing that those were the technologies our team members were most familiar with for web development, and that we wanted to save on time from having to investigate other technologies. The technology stack implementation and design diagrams is elaborated further below.

In addition, some requirements were updated:

* Assumptions/Business Constraints: Changed ABC9 and added ABC14-21

|  |  |
| --- | --- |
| ID | Assumption/Business Constraints |
| ABC1 | Any 1 QA coordinator is attached to only 1 Department. |
| ABC2 | Staff members only have to agree to the Terms and Conditions once before being allowed to submit. |
| ABC3 | Ideas can be uncategorized. (The category is “null”) |
| ABC4 | Ideas can have their category changed by the author before and after posting. |
| ABC5 | Categories can only be deleted when they are not in use. (No ideas are tagged with that category) |
| ABC6 | Staff members can opt to anonymously submit an idea or comment. |
| ABC7 | Staff can only submit ideas during an open duration for submission; after the idea submission closure date, staff cannot post new ideas. |
| ABC8 | Staff can only comment on ideas during an open duration for commenting per idea; after that idea’s commenting closure date, staff cannot post new comments on that idea. |
| ABC9 | The system can display preview of ideas in a list per submission. |
| ABC10 | A department’s list of ideas needs to be paginated (5 ideas per page). |
| ABC11 | Uploaded documents can be downloaded in a ZIP file. |
| ABC12 | The interface must be suitable for all devices (e.g. mobile phones, tablets, desktops). |
| ABC13 | 1. A number of reports need to be made available. For example:    1. Statistics       1. Number of ideas made by each Department.       2. Percentage of ideas by each Department.       3. Number of contributors within each Department.    2. Exception reports       1. Ideas without a comment.       2. Anonymous ideas and comments. |
| ABC14 | The system can sort idea previews by views, popularity, and recency. |
| ABC15 | The system can filter idea previews by categories. |
| ABC16 | The system can filter idea previews by departments. |
| ABC17 | The system defaultly filters idea previews (sorted by recency) by the user’s departments if the user has a department, or by all departments if the user is in department “Support Staff". |
| ABC18 | The user can view an idea in detail (more infothan preview) |
| ABC19 | The user can only comment and view comments from the details view |
| ABC20 | The user can thumb up/down and view thumbs/up down from both the details view and the list view |
| ABC21 | The user can submit ideas to any department submission. |

* User Stories: Created US15, updated dependencies for US1, US4, US5, US6, US7, and US15.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7, ABC21 | 1 | New |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 2 | New |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 2 | New |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC18 | 1 | New |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8, ABC19 | 3 | New |
| US6 | All staff can give thumbs up or down once per idea. | ABC20 | 4 | New |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments | ABC10, ABC14 | 4 | New |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 2 | New |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | New |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | New |
| US11 | The QA manager can add categories. |  | 2 | New |
| US12 | The QA manager can delete categories | ABC5 | 2 | New |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | New |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | New |
| US15 | All staff can sort and filter when viewing ideas and comments. | ABC14, ABC15, ABC16, ABC17 | 4 | New |

* 1. Sprint3: Implementation of basic functions

In the 3rd sprint, requirements are well described enough so as not to have to be updated, and so implementation began for some basic functions, including US1, US4, and US14.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7, ABC21 | 1 | Active |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 2 | New |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 2 | New |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC18 | 1 | Active |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8, ABC19 | 3 | New |
| US6 | All staff can give thumbs up or down once per idea. | ABC20 | 4 | New |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments | ABC10, ABC14 | 4 | New |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 2 | New |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | New |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | New |
| US11 | The QA manager can add categories. |  | 2 | New |
| US12 | The QA manager can delete categories | ABC5 | 2 | New |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | New |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | Active |
| US15 | All staff can sort and filter when viewing ideas and comments. | ABC14, ABC15, ABC16, ABC17 | 4 | New |

* 1. Sprint4: Further implementation and testing

In the 4th sprint, some basic functions were completed, and implementation for more advanced functions began. The priority and statuses were updated for user stories: US1, US, US3, US8, US10, US11, US12, US13, US14.

Testing was also done for previously active user stories, with US1 and US14 being closed for successful completion.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7, ABC21 | 1 | Closed |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 1 | Active |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 1 | Active |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC18 | 1 | Active |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8, ABC19 | 3 | New |
| US6 | All staff can give thumbs up or down once per idea. | ABC20 | 4 | New |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments | ABC10, ABC14 | 4 | New |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 1 | Active |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | New |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | Active |
| US11 | The QA manager can add categories. |  | 1 | Active |
| US12 | The QA manager can delete categories | ABC5 | 1 | Active |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | Active |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | Closed |
| US15 | All staff can sort and filter when viewing ideas and comments. | ABC14, ABC15, ABC16, ABC17 | 4 | New |

* 1. Sprint5: Further testing and implementation

Many user stories were closed after completion of implementation and successful testing:

* US2, US3, (dealing with with posting ideas, terms and conditions, and attaching documents)
* US8, US11, and US12 (creating and deleting categories, tagging ideas with categories)

In addition, US6 and US9 were put into implementation, now that the must-haves and could-haves have all been addressed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7, ABC21 | 1 | Closed |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 1 | Closed |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 1 | Closed |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC18 | 1 | Active |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8, ABC19 | 3 | Active |
| US6 | All staff can give thumbs up or down once per idea. | ABC20 | 4 | Active |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments | ABC10, ABC14 | 4 | New |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 1 | Closed |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | Active |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | Active |
| US11 | The QA manager can add categories. |  | 1 | Closed |
| US12 | The QA manager can delete categories | ABC5 | 1 | Closed |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | Active |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | Closed |
| US15 | All staff can sort and filter when viewing ideas and comments. | ABC14, ABC15, ABC16, ABC17 | 4 | New |

* 1. Sprint6: Final Testing and Documentation

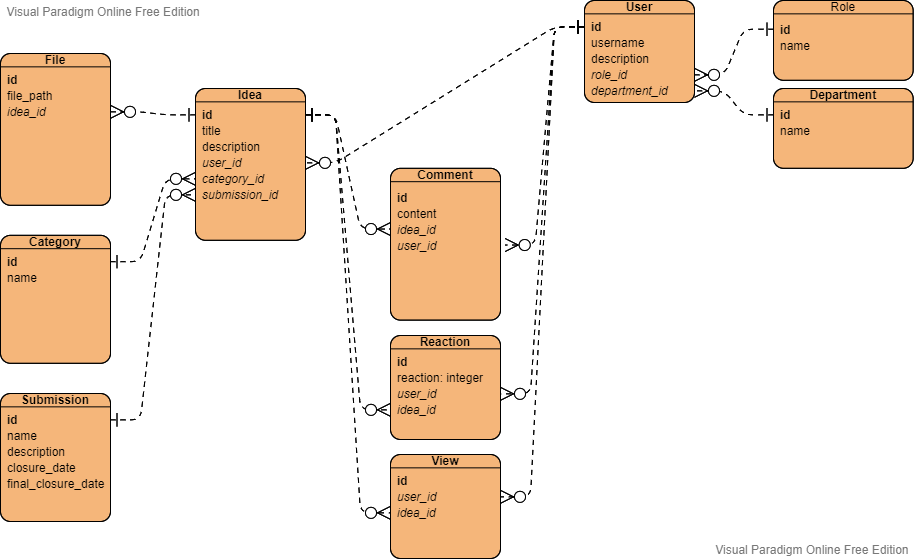
Due to time constraints, the final build had US6 and US9 – which was previously trying to be implemented - closed due to time constraints, as was US7 and US15. However, US4 and US5 were closed for completion, ending the 6th sprint with all must-haves and could-haves implemented, and only nice-to-haves were not.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Dependency (Assumptions/Business Constraints) | Priority | Status |
| US1 | All staff can submit ideas to a submission. | ABC6, ABC7, ABC21 | 1 | Closed |
| US2 | A staff member must agree to a Terms and Conditions before submitting. | ABC2 | 1 | Closed |
| US3 | All staff can attach documents when submitting ideas to a submission. |  | 1 | Closed |
| US4 | All staff can view submitted ideas and their comments. | ABC9, ABC18 | 1 | Closed |
| US5 | All staff can make comments to a submitted idea. | ABC6, ABC8, ABC19 | 3 | Closed |
| US6 | All staff can give thumbs up or down once per idea. | ABC20 | 4 | Closed |
| US7 | All staff can view list of Most Popular Ideas, Most Viewed Ideas, Latest Ideas, Latest Comments | ABC10, ABC14 | 4 | Closed |
| US8 | All ideas can be categorised (tagged) before posting. | ABC3 | 1 | Closed |
| US9 | A QA coordinator can view reports from their department. | ABC13, ABC1 | 4 | Closed |
| US10 | A QA coordinator receives an email when an idea is submitted in their department | ABC1 | 2 | Closed |
| US11 | The QA manager can add categories. |  | 1 | Closed |
| US12 | The QA manager can delete categories | ABC5 | 1 | Closed |
| US13 | The QA Manager can download CSV of all posts or select multiple files to be put in zip. | ABC11 | 2 | Closed |
| US14 | The Administrator has top level access to all system data, e.g. open-closure dates for each academic year’s idea submission time, staff details. |  | 1 | Closed |
| US15 | All staff can sort and filter when viewing ideas and comments. | ABC14, ABC15, ABC16, ABC17 | 4 | Closed |

1. Design diagrams

For design diagrams we took to a Use Case diagram to quickly model a generalization of functionality through various roles, and an ERD to model our database and how we initially intended to have various entities interacting with each other in the system.

* 1. Entity Relationship Diagram (ERD)



The first and foremost entity was the **Idea entity**, which stores information of the suggested idea to be submitted in a submission in the Idea Collection system. It requires knowledge of the author of the idea, the category it is tagged in, and the submission it belongs to, and so takes *user\_id, category\_id, submission\_id* as **corresponding foreign keys** with **each user, category, and submission being able to author, tagged, and submitted with many** Ideas.

**Submissions** in particular have further information, most notably *closure\_date* and *final\_closure\_date*, to be used for user stories involving inability to comment on posts or post in a submission after a particular date.

**Users** have basic information, but more importantly they have a *department ID* and *role ID*, enabling Many-To-One relationships with departments and roles, with the latter being used for authorization purposes. Of course, each user can only have 1 role/department.

The **File** entity was designed for the purpose of having files be associated with the idea they are attached to. It stores a *file\_path* string to later query for and retrieve the file, and references the foreign key *idea\_id* to know which idea it was uploaded to.

**Comments** are stored with its content and *user\_id* of the author of the comment, as well as *idea\_id* of the idea it is posted on. Naturally, each idea can have many comments, but each comment can only be made by 1 user and is posted on only 1 idea.

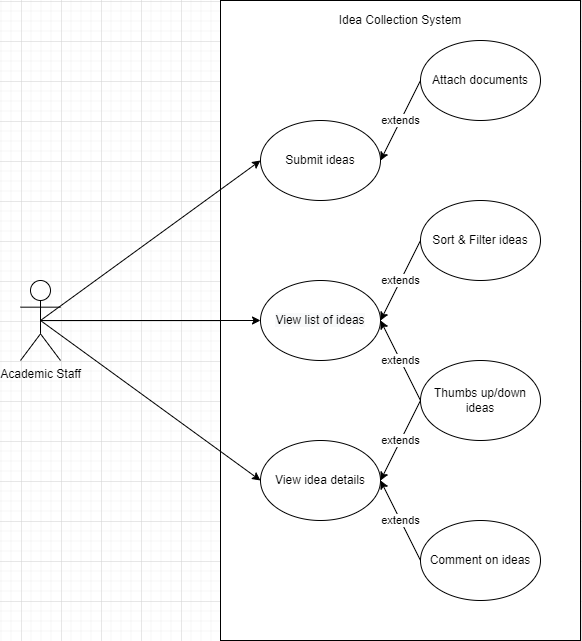
Lastly, there are the **Reaction and View** entities, used to store and track respectively the “Thumbs Up/Thumbs Down” and views of ideas posted. Similar to comments, they have to know which user made them and which idea they are attached to with *user\_id* and *idea\_id*. For Reactions in particular, each record stores a *reaction* value to be used as a mapping value for further processing. The datatype **“integer” was used for the sake of extensibility**, allowing for more than just Like and Dislike (e.g. 1 for Like, 2 for Dislike, 3 for Confused, and so on) as opposed to just a boolean value.

* 1. Use-Case Diagram

Based on the requirements elicited around Sprint2, a high-level summary of the user stories can be described based on the Use-Case Diagrams below.

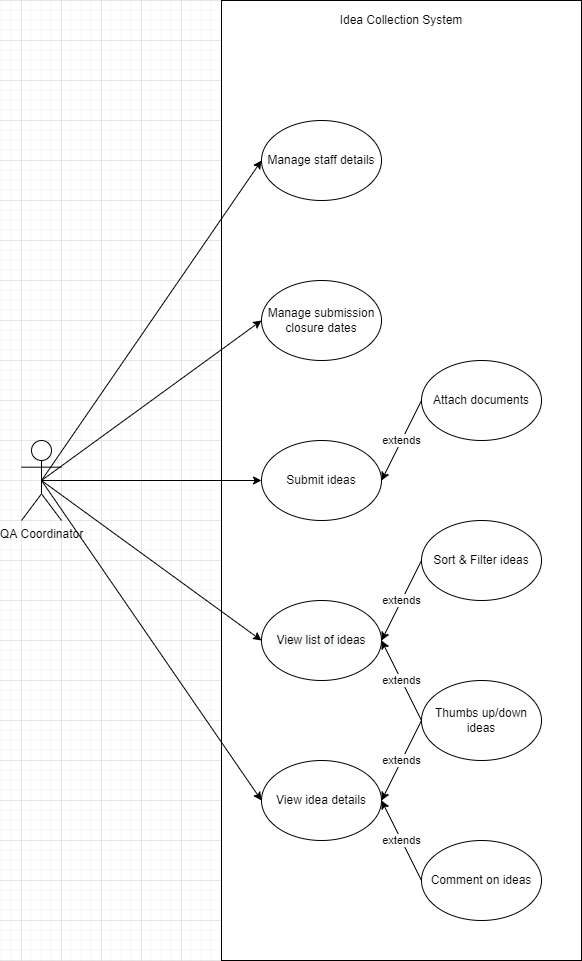
Every actor can submit ideas, attach documents, make comments, thumbs up or down once per idea, view list of ideas, view idea details, and sort&filter ideas.

* + 1. Academic Staff



The academic staff has the least amount of power, only being able to submit ideas and viewing idea submissions and details and the subsequent extensions.

* + 1. Administrator



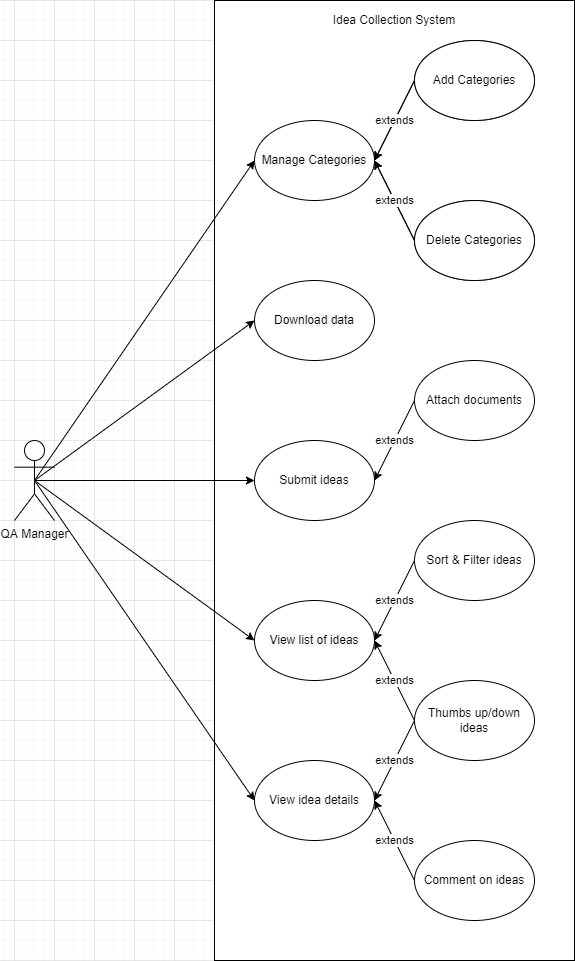
Extra power of the Administrator role: maintains closure dates and staff details

* + 1. QA Coordinator



Extra authority of the Quality Assurance Coordinator of each Department: receive email notifications when ideas are submitted, view statistical reports.

* + 1. QA Manager



Extra authority of the Quality Assurance Manager: can add and delete categories, downloading data.

1. Implementation
   1. Authentication and Authorization

With there are different roles in this system, implementation must be such that it can recognize legitimate users, as well as grant or restrict access to them depending on which role they have. Through the ASP.NET Identity API should be able to login by way of inputting a username and password for an existing account, and what they are allowed to do (and similarly what they will be seeing from the page on their end) will depend on what role the account they’re logged into has.

* 1. Posting ideas into submissions
     1. CRUD for submissions

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Enteripse\_web.Models

{

public class Submission

{

public int Id { get; set; }

public string name { get; set; }

public string description { get; set; }

public DateTime closureDate { get; set; }

public DateTime FinalDate { get; set; }

}

}

**Submission model**

[AuthLog(Roles = "Administrator")]

// GET: Submissions/Create

public ActionResult Create()

{

return View();

}

// POST: Submissions/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,name,description,closureDate,FinalDate")] Submission submission)

{

if (ModelState.IsValid)

{

db.Submissions.Add(submission);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(submission);

}

**Action Create Submission**

[AuthLog(Roles = "Administrator")]

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Submission submission = db.Submissions.Find(id);

if (submission == null)

{

return HttpNotFound();

}

return View(submission);

}

// POST: Submissions/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,name,description,closureDate,FinalDate")] Submission submission)

{

if (ModelState.IsValid)

{

db.Entry(submission).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(submission);

}

**Edit Action for Submission**

[AuthLog(Roles = "Administrator")]

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Submission submission = db.Submissions.Find(id);

if (submission == null)

{

return HttpNotFound();

}

return View(submission);

}

// POST: Submissions/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Submission submission = db.Submissions.Find(id);

db.Submissions.Remove(submission);

db.SaveChanges();

return RedirectToAction("Index");

}

Delete Action for submission

* + 1. CRUD for ideas

[model/controller/views for ideas and submissions]

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

using System.Data.Entity;

namespace Enteripse\_web.Models

{

public class Post

{

public Post()

{

this.IsAnonymus = true;

this.Time = DateTime.Now;

this.Comments = new HashSet<Comment>();

}

public int PostId { get; set; }

[Required]

[MaxLength(200)]

public string Title { get; set; }

[Required]

public DateTime Time { get; set; }

public string AuthorId { get; set; }

public virtual ApplicationUser Author { get; set; }

[MaxLength(400)]

public string AuthorName { get; set; }

public string Description { get; set; }

public bool IsAnonymus { get; set; }

public string DocumentName { get; set; }

public int submissionId { get; set; }

public int categoryId { get; set; }

public virtual Category Category { get; set; }

public virtual ICollection<Comment> Comments { get; set; }

}

}

**Model for Idea**

public ActionResult Create()

{

ViewBag.CategoryId = new SelectList(context.Categories.ToList(), "Id", "Name");

return View();

}

// POST: Posts1/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Post post, int id)

{

if (ModelState.IsValid)

{

post.AuthorId = User.Identity.GetUserId();

if (Request.Files != null)

{

var file = Request.Files[0];

if (file != null && file.ContentLength > 0)

{

string path = Path.Combine(Server.MapPath("~/Files"),

Path.GetFileName(file.FileName));

//Save file using Path+fileName take from above string

file.SaveAs(path);

post.DocumentName = Path.GetFileName(file.FileName);

}

}

MailMessage mail = new MailMessage();

{

ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls

| SecurityProtocolType.Tls11

| SecurityProtocolType.Tls12;

}

SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

smtpServer.Credentials = new System.Net.NetworkCredential("thienlun15082001@gmail.com", "thien15082001");

smtpServer.EnableSsl = true;

smtpServer.Port = 587; // Gmail works on this port

mail.From = new MailAddress("thienlun15082001@gmail.com");

var userId = User.Identity.GetUserId();

var user = db.Users.Where(x => x.Id == userId).FirstOrDefault(); //tim entity user

if(user.Department.Name == "Human Resources")

{

mail.To.Add("qahumanr@gmail.com");

}

if (user.Department.Name == "IT")

{

mail.To.Add("qainformationt@gmail.com");

}

if (user.Department.Name == "Bussiness")

{

mail.To.Add("qabussiness06@gmail.com");

}

if (user.Department.Name == "QA")

{

mail.To.Add("thienlun26062001@gmail.com");

}

mail.Subject = "New post";

mail.Body = "Your department has a new post. Please check it!";

smtpServer.Send(mail);

post.submissionId = id;

var submit = db.Submissions.Find(id);

if (DateTime.Now > submit.closureDate)

{

return RedirectToAction("Fail", "Home");

}

else

{

db.Posts.Add(post);

db.SaveChanges();

return RedirectToAction("Index","Home");

}

}

ViewBag.AuthorId = new SelectList(db.Users, "Id", "FullName", post.AuthorId);

ViewBag.CategoryId = new SelectList(context.Categories, "Id", "Name", post.categoryId);

return View(post);

}

Create Action for Idea

* + 1. Closure dates
    2. Terms and Conditions before submitting

[controller/view of create idea]

<script language="javascript" src="http://code.jquery.com/jquery-1.11.1.js"></script>

<script language="javascript" type="text/javascript">

function AcceptTermAndcondition() {

if ($("#chkcondition").is(':checked')) {

return true;

}

else {

alert("Please accept term and condition");

return false;

}

}

</script>

Javascript code for term and condition

<div>

@Html.TextArea("Termandcondition", "Term 1, Term 2, Term 3, Term 4,...", new { @height = "135px", @width = "251px", @id = "txtcondition" })

</div>@Html.CheckBox("Condition", false, new { @id = "chkcondition" })<a href="#">Term and condition</a>

Check box in View for Term and Conditions

* + 1. Uploading files to an idea

[model/view/controller of idea and file]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Enteripse\_web.Models

{

public class FileModel

{

public string FileName { get; set; }

public string FilePath { get; set; }

public bool IsSelected { get; set; }

}

**}**

**File model**

if (Request.Files != null)

{

var file = Request.Files[0];

if (file != null && file.ContentLength > 0)

{

string path = Path.Combine(Server.MapPath("~/Files"),

Path.GetFileName(file.FileName));

//Save file using Path+fileName take from above string

file.SaveAs(path);

post.DocumentName = Path.GetFileName(file.FileName);

}

}

**Controller action for upload file**

* 1. Categories for ideas
     1. CRUD for categories

[model/view/controller of category]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Enteripse\_web.Models

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

}

}

**Model for category**

[AuthLog(Roles = "Administrator")]

// GET: Categories/Create

public ActionResult Create()

{

return View();

}

// POST: Categories/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,Name,Description")] Category category)

{

if (ModelState.IsValid)

{

db.Categories.Add(category);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(category);

}

[AuthLog(Roles = "Administrator")]

// GET: Categories/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

Create action for category

[AuthLog(Roles = "Administrator")]

// GET: Categories/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// POST: Categories/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,Name,Description")] Category category)

{

if (ModelState.IsValid)

{

db.Entry(category).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(category);

}

Edit action for category

[AuthLog(Roles = "Administrator")]

// GET: Categories/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// POST: Categories/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Category category = db.Categories.Find(id);

var check\_category = db.Posts.Where(x => x.categoryId == id).ToList();

if (check\_category.Count() != 0)

{

return RedirectToAction("FailDelete", "Home");

}

else

{

db.Categories.Remove(category);

db.SaveChanges();

return RedirectToAction("Index");

}

}

Delete action for category

* + 1. Choosing a category when posting an idea

[controller of idea, create ideaview]

public int categoryId { get; set; }

public virtual Category Category { get; set; }

Model Idea co khai bao category Id va ke thua toan bo Category model

ViewBag.CategoryId = new SelectList(context.Categories.ToList(), "Id", "Name");

ViewBag.CategoryId = new SelectList(context.Categories, "Id", "Name", post.categoryId);

* 1. Comments on ideas

[model of comment and controller and details view of Idea]

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace Enteripse\_web.Models

{

public class Comment

{

public Comment()

{

this.Date = DateTime.Now;

}

public int CommentId { get; set; }

[Required]

public string Content { get; set; }

[Required]

public DateTime Date { get; set; }

public string AuthorId { get; set; }

public virtual ApplicationUser Author { get; set; }

public bool IsAnonymus { get; set; }

public int PostId { get; set; }

public int submissionId { get; set; }

public virtual Post Post { get; set; }

}

}

Model of Comment

this.Comments = new HashSet<Comment>();

public virtual ICollection<Comment> Comments { get; set; }

comment in Idea model

[HttpGet]

public ActionResult Comment(int id)

{

return PartialView("\_PartialComment", new Enteripse\_web.Models.Comment { PostId = id });

}

[HttpPost]

public ActionResult Comment(Comment data, int id)

{

try

{

data.AuthorId = User.Identity.GetUserId();

data.submissionId = id;

var submit = db.Submissions.Find(id);

if (DateTime.Now > submit.FinalDate)

{

return RedirectToAction("FailComment", "Home");

}

else

{

db.Comments.Add(data);

db.SaveChanges();

}

if (data.IsAnonymus == true)

{

data.AuthorId = "Anonymous";

}

else

{

data.AuthorId = data.AuthorId;

}

Create action for comment

* 1. Email notification

[create action method from idea controller]

using System.Net.Mail;

MailMessage mail = new MailMessage();

{

ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls

| SecurityProtocolType.Tls11

| SecurityProtocolType.Tls12;

}

SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

smtpServer.Credentials = new System.Net.NetworkCredential("thienlun15082001@gmail.com", "thien15082001");

smtpServer.EnableSsl = true;

smtpServer.Port = 587; // Gmail works on this port

mail.From = new MailAddress("thienlun15082001@gmail.com");

var userId = User.Identity.GetUserId();

var post = db.Posts.Where(x => x.PostId == data.PostId).FirstOrDefault();

var author = db.Users.Where(x => x.Id == post.AuthorId).FirstOrDefault();

var email = author.Email;

mail.To.Add(new MailAddress(email));

mail.Subject = "New comment";

mail.Body = "Your post has new comment. Please check it!";

smtpServer.Send(mail);

action for email to the author when have new comment

SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

smtpServer.Credentials = new System.Net.NetworkCredential("thienlun15082001@gmail.com", "thien15082001");

smtpServer.EnableSsl = true;

smtpServer.Port = 587; // Gmail works on this port

mail.From = new MailAddress("thienlun15082001@gmail.com");

var userId = User.Identity.GetUserId();

var user = db.Users.Where(x => x.Id == userId).FirstOrDefault(); //tim entity user

if(user.Department.Name == "Human Resources")

{

mail.To.Add("qahumanr@gmail.com");

}

if (user.Department.Name == "IT")

{

mail.To.Add("qainformationt@gmail.com");

}

if (user.Department.Name == "Bussiness")

{

mail.To.Add("qabussiness06@gmail.com");

}

if (user.Department.Name == "QA")

{

mail.To.Add("thienlun26062001@gmail.com");

}

mail.Subject = "New post";

mail.Body = "Your department has a new post. Please check it!";

smtpServer.Send(mail);

Action for email notification to QA Cordinator when have new post in their department.

* 1. Downloading CSVs and zips by QA Manager

downloadcontroller filemodel

csv controller

details view of post

public FileContentResult DownloadCSV()

{

string csv = "\"PostId\",\"Title\",\"Time\",\"AuthorId\",\"AuthorName\",\"Description\",\"DocumentName\" \n";

var List = db.Posts.ToList(); //get this list from database

foreach (Post item in List)

{

csv = csv + String.Format("\"{0}\",\"{1}\",\"{2}\",\"{3}\",\"{4}\",\"{5}\",\"{6}\" \n",

item.PostId,

item.Title,

item.Time,

item.AuthorId,

item.AuthorName,

item.Description,

item.DocumentName);

}

//StringWriter sw = new StringWriter();

//sw.WriteLine

return File(new System.Text.UTF8Encoding().GetBytes(csv), "text/csv", "Report123.csv");

}

Download CSV action

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Enteripse\_web.Models

{

public class FileModel

{

public string FileName { get; set; }

public string FilePath { get; set; }

public bool IsSelected { get; set; }

}

}

File model for download

[AuthLog(Roles = "Administrator")]

public ActionResult Index()

{

string[] filePaths = Directory.GetFiles(Server.MapPath("~/Files/"));

List<FileModel> files = new List<FileModel>();

foreach (string filePath in filePaths)

{

files.Add(new FileModel()

{

FileName = Path.GetFileName(filePath),

FilePath = filePath

});

}

return View(files);

}

[HttpPost]

public ActionResult Index(List<FileModel> files)

{

using (ZipFile zip = new ZipFile())

{

zip.AlternateEncodingUsage = ZipOption.AsNecessary;

zip.AddDirectoryByName("Files");

foreach (FileModel file in files)

{

if (file.IsSelected)

{

zip.AddFile(file.FilePath, "Files");

}

}

string zipName = String.Format("FilesZip\_{0}.zip", DateTime.Now.ToString("yyyy-MMM-dd-HHmmss"));

using (MemoryStream memoryStream = new MemoryStream())

{

zip.Save(memoryStream);

return File(memoryStream.ToArray(), "application/zip", zipName);

}

}

}

Download in zip file

* 1. Listed page(5 ideas per page)

public ActionResult Index(int page = 1, int pageSize = 5)

{

var posts = data.Posts.ToList();

var page\_post = posts.OrderByDescending(x=> x.Time).ToPagedList(page, pageSize);

return View(page\_post);

}

Action for listed page

@using PagedList.Mvc;

@using PagedList;

@Html.PagedListPager(Model, page => Url.Action("Index", new { page }))

Use listed page in view

**PHẦN THÊM VÀO THEO YÊU CẦU CỦA ÔNG**

@model Enteripse\_web.Models.Submission

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Create</title>

</head>

<body>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Submission</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.description, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.description, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.description, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.closureDate, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.closureDate, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.closureDate, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.FinalDate, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FinalDate, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FinalDate, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

</body>

</html>

**Trang view create của submission**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using Enteripse\_web.CustomFilters;

using Enteripse\_web.Models;

namespace Enteripse\_web.Controllers

{

public class SubmissionsController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Submissions

public ActionResult Index()

{

return View(db.Submissions.ToList());

}

// GET: Submissions/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Submission submission = db.Submissions.Find(id);

if (submission == null)

{

return HttpNotFound();

}

return View(submission);

}

[AuthLog(Roles = "Administrator")]

// GET: Submissions/Create

public ActionResult Create()

{

return View();

}

// POST: Submissions/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,name,description,closureDate,FinalDate")] Submission submission)

{

if (ModelState.IsValid)

{

db.Submissions.Add(submission);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(submission);

}

// GET: Submissions/Edit/5

[AuthLog(Roles = "Administrator")]

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Submission submission = db.Submissions.Find(id);

if (submission == null)

{

return HttpNotFound();

}

return View(submission);

}

// POST: Submissions/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,name,description,closureDate,FinalDate")] Submission submission)

{

if (ModelState.IsValid)

{

db.Entry(submission).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(submission);

}

// GET: Submissions/Delete/5

[AuthLog(Roles = "Administrator")]

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Submission submission = db.Submissions.Find(id);

if (submission == null)

{

return HttpNotFound();

}

return View(submission);

}

// POST: Submissions/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Submission submission = db.Submissions.Find(id);

db.Submissions.Remove(submission);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**SubmissionsController**

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

@if (User.IsInRole("Administrator"))

{

<li>@Html.ActionLink("Register", "Register", "Account")</li>

<li>@Html.ActionLink("Role", "Index", "Role")</li>

<li>@Html.ActionLink("Departments", "Index", "Departments")</li>

<li>@Html.ActionLink("Create Submission", "Create", "Submissions")</li>

<li>@Html.ActionLink("Create Category", "Create","Categories")</li>

}

View ở Layout

1. Test Cases