**Problem definition**

Every year, millions of individuals attend seminars, conferences, and workshops. To expand their knowledge, promote their ideas, market their goods, and network. It is impossible to respond to all questions posed by attendees that are pertinent to the topic at the end of the seminar.

To overcome this problem, we have developed a web application in which seminar attendees can post their questions, if any, during the seminar, and the questions will be prioritized based on relevance and frequency, so that the most nearly identical query is given top priority, followed by the next most closely related query, and so on. so that the speaker can answer them in a shorter amount of time. The software is fast, easy to use, economically feasible, and user-friendly.

**Economic Feasibility**

This section will explain the current system, the proposed system, and the comparison between the two, as well as the merits of the proposed system over the current one.

**Existing System**

We reviewed one website before developing our question aggregator project, which may be found at the URL below which is our first reference

<https://totalinternalreflectionblog.com/2016/05/21/catechism-a-short-guide-to-asking-questions-in-seminars/>

and the other reference which we have is

<https://apps.apple.com/ca/app/apc-fall-seminar/id1484394088> which is available in the playstore

and anyone can download it from there.

**Reference 1**

If we consider the link I provided before as the first preference, it provides a quick guide on asking questions at seminars. After reading this article about seminars, I learned that it is a difficult task for a speaker to answer all the questions that are asked in a seminar and that there is a chance that someone may ask questions that are irrelevant to the subject, that is, questions about an unfamiliar topic and that clarification are difficult for the speaker to understand if the hall is full. Due to the attendant's random selection of questions to ask, some vital questions remain unanswered, while some less important and duplicate questions are answered. Another issue that arises in seminars is that the question-and-answer process takes time, and it can take even longer if there is a communication gap between the speaker and the person who is asking the question, or if the person who is asking the question has different comfort zones, such as some people are not good at communicating in front of a crowd, others do not have a loud voice, and so on. and finally it is impossible to respond to all questions posed by attendees that are pertinent to the topic at the end of the seminar.

**Reference 2**

If a user wanted to log in to that application, he or she must first create an account. This is mostly to schedule a meeting and give a seminar for the construction industry. This app includes a list of speakers and a list of topics. In a seminar, however, no questions can be asked. This is a little restrictive in that only specific people can schedule seminars, and these have limited features.

**Proposed system**

The proposed answer to the concerns outlined above is "Question Aggregator." After registering, the coordinator can log in with his or her user id and password. The organizer can edit seminar information such as the speaker's name, abstract, content, date, end date, venue, title, and time, as well as see a list of prioritized queries. Attendees can log in using the seminar name and password provided on the homepage. They can post queries on related topics after picking a seminar. The queries will be prioritized, and the coordinator will see them. The topic's associated keywords are extracted first, then questions are prioritized by comparing query keywords to the topic's actual list of keywords. Finally, a prioritized list of queries is formed, which can be accessed by the coordinator. Speakers can respond to questions from the list provided by the coordinator at the end of the lecture.

**Comparison between both the systems**

In terms of comparison, there are several shortcomings in both reference 1 and reference 2. When compared to the article provided in reference 1, our proposed model will undoubtedly eliminate problems such as communication gaps, question duplication, time consumption, and random question selection.

If we look at the second reference, it was all about a certain issue seminar, and it didn't have the option to add seminar questions. Moreover, the users have only restricted capabilities.

This is a fundamental flaw in the app, and it's from this that we've gotten the idea to improve it by adding features and providing priority based on the questions that were asked in the seminar based on the keywords

**Benefits of proposed application compared to existing systems**

* Low cost
* User flexible
* Secure registration of users must be done.
* Duplicate questions must be removed.
* Databases must update when needed.
* Questions must be ranked properly according to their relevancy.
* Better design of user interface to get user friendly.
* Better performance.
* Easy to use**.**
* User friendly
* No economic cost
* Open source

**Functional requirements**

**Requirement Elicitation:**

Two Types of users: Attendant and Seminar Coordinator

**The functionality of Attendant:**

**View Seminar:** Attendants can view the upcoming seminars.

**Seminar Registration:** Attendants can enroll themselves for seminars available from the portal.

**Sign in**: Attendants can sign into the website by provided id password during registration.

**Post Query**: Attendants can post their questions during the seminar.

**View Query**: Attendants can view their posted questions.

**The functionality of the Coordinator:**

**Registration**: Coordinators can register themselves and make the profile to add their seminars.

**Sign in**: The coordinator can sign in to the portal by using their id password.

**Post Seminar:** The coordinator can post their upcoming seminar so that the interested crowd can register.

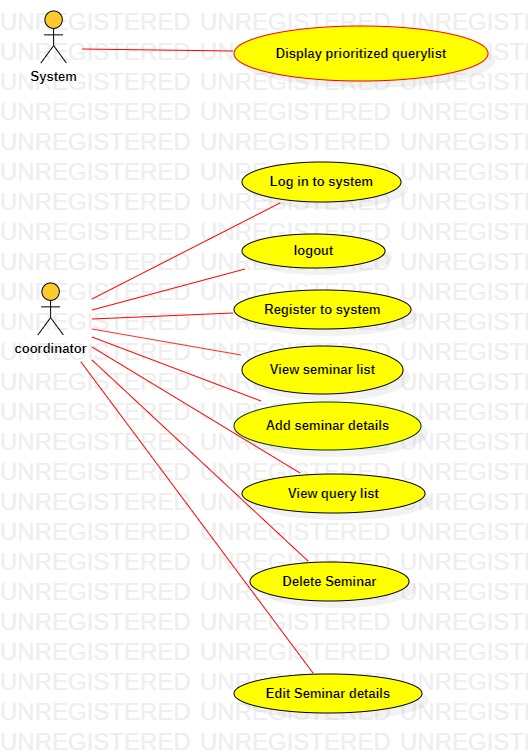
**View Seminar:** The coordinator can view the details of their posted seminars.

**Edit Seminar:** The coordinator can edit the details of the posted seminars.

Delete Seminar

**View Query:** The coordinator can view the prioritized query list posted by attendees.

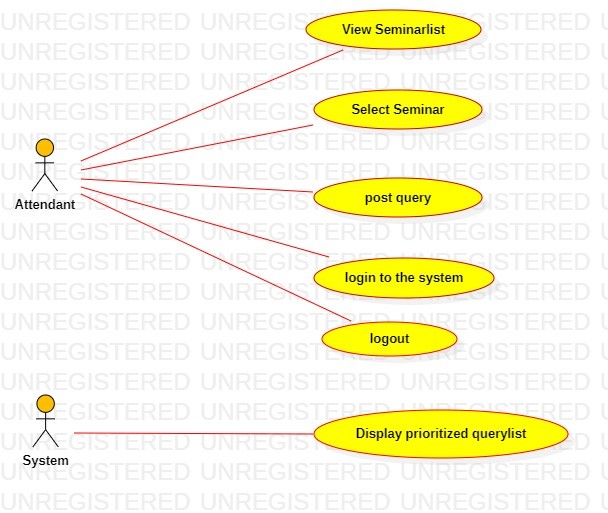
**3 b) Use Case Diagram for coordinator**

****

There are two actors in the coordinator use case diagram. The first is the system, while the second is the coordinator. The coordinator can log in, log out, register, view the seminar list, add seminar details, view the query list, delete the seminar, and edit seminar details.

In addition, the system can show a prioritized query list.

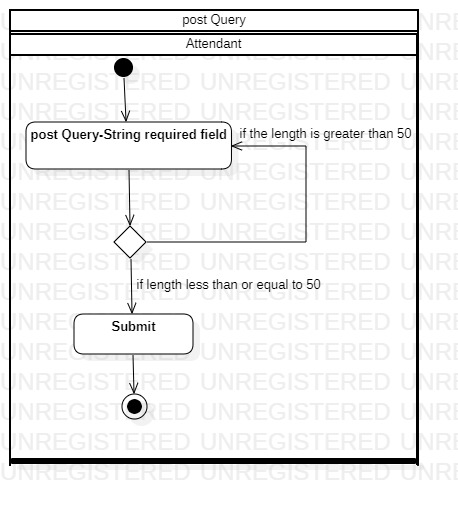
**Use case diagram for attendant:-**

****

There are two actors in the attendant use case diagram. The first is the attendant, while the second is the system. The coordinator can view the Seminar list, select seminar, post query, login to the system, and logout.

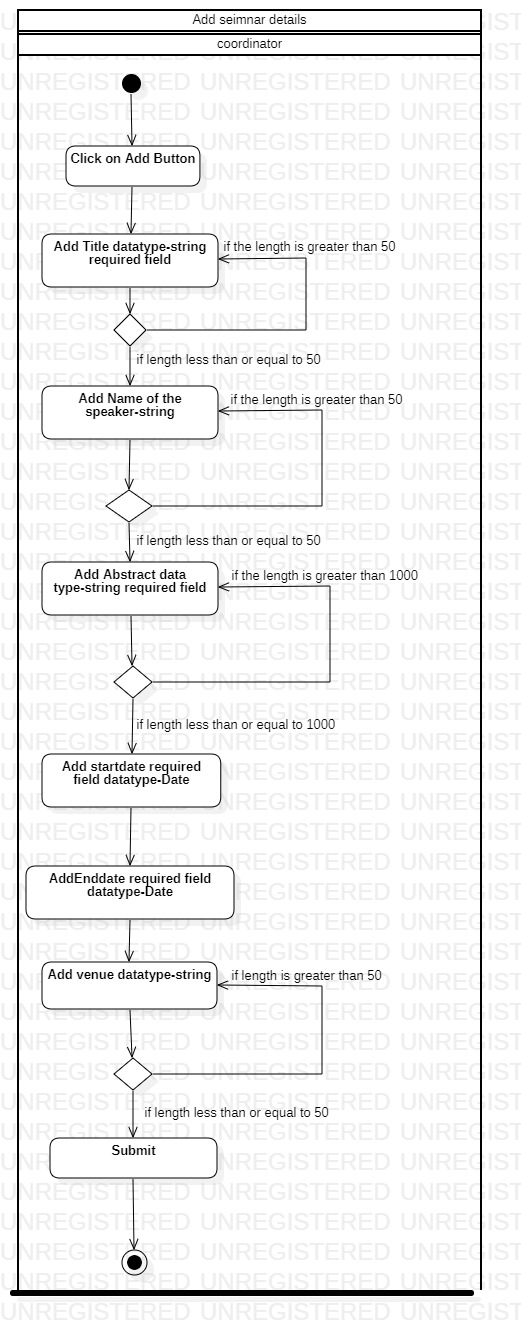
In addition, the system can display a prioritized query list.

**activity diagram for Post query:-**

****

This is the activity diagram for the post query in the attendant difficult use scenario. The data type we use for a post query is a string, and it will be validated there. If the total length exceeds 50 It will return to submit the query, or if the length is less than or equal to 50 characters, it can be submitted.

**Activity Diagram for adding seminar details**

****

This is the activity diagram for the complex use scenario of adding seminar details in the coordinator. After that, click the "Add" button. Then we can add a title, which is a data type string, a speaker's name, which would be a data type string, an abstract, a start date, an end date, and a venue, which is of data type string, and finally, we can submit after entering all of the details and making sure they are valid.

**Software qualities**

**Correctness:**

* The correct credentials should be able to make Attendants and Coordinators sign in.
* Coordinator should be able to see the correct prioritized sequence of queries.

**Example: -** Attendees can login to the system with correct credentials. Access can be granted only to AUTHORIZED USERS (who are matched with the database). The user will not be able to access the application if he/she enters invalid details.

The frequency and relevancy of queries are used to prioritize and display them. If the seminar's topic is artificial intelligence, questions about abstract and AI will be prioritized over queries that are unfamiliar with the topic, such as "what is your name."

**Robustness:**

* If a user (Attendants and Coordinator) inserts wrong credentials, the system should print an error message and redirect to the login page for retry.
* If Attendants want to login before the seminar starts, the system should give notification of seminar timing and they should not be able to log in.

**Example: -** If the user enters invalid credit information, the user should be directed to the login page.

 If an attendee registers before the seminar, they will not be able to log in, Proper error pop up messages will be displayed

**Time-efficiency**

* System will not take a long time when a large number of users are using it.

Example- Typically, a seminar will begin at a specific time, and many users will use the same application to post their queries. Even in this instance, using the app will not take any longer than usual. It will act quickly and will not become stuck as a result of the large load.

**Friendliness:**

* Improved user interface to make it more user-friendly.

Example- The website is user-friendly and accessible to everybody, even if they are unfamiliar with how to use the seminar app.

**Web Scrapping:**

* Web scraping is a method of extracting information from web pages. We can get the contents of the page and then extract the information we need for processing, saving, or simply displaying on our app.
* It's useful when the app/website we're trying to scrape doesn't have an external API that's open to the public.

**Example: -** We can scrap related words from the website ([**https://relatedwords.org/**](https://relatedwords.org/)) and use those words to prioritize the queries if the coordinator only placed a tiny abstract as we don't have any more keywords from that abstract.

**4. Top-Level and Low-Level design**

**MVC Architecture:-** MVC is useful for our project development. This section explains why we chose the MVC pattern for our application. The MVC pattern is simple to maintain and test. This is merely a structure. The model, view, and controller will all be decoupled.

MVC is a programming language that may be used to create mobile, online applications, etc., A data controller is nothing more than a controller in this case. If we don't use MVC, it will be tough to alter something in the data. It is three times faster than a standard application when utilizing MVC. The term "model" refers to the data that will be retrieved from the server. It can be accessed from any server that has a database. The data fetched from the model will be shown in the view. The controller resides between the Model and the View. It will change the data.

MVC is generally classified as

* Model
* View
* Controller

**Model:-**

The model will contain most of all the logic. This model layer will be split into two parts: the Business layer and the DAO layer which stands for Data Access Object layer. If we are to do any task. The business layer will be used to write the code. For example, suppose we type our email address into a webpage. The model will determine whether or not there is another account with the same email address. and makes a call to the controller if the model informs you that the email address is already in use. It will display an error message. Another example is if we need to conduct a task that requires us to add two numbers. This is where the method of doing addition will be applied. These are perhaps the most ardent supporters of data validation. The model logic that we used in my model coordinator is a decision, questions, seminar details, and seminar-login-info.

**View:**

It's simply a user interface, as the title implies. If we want to access the webpage, we can either use a valid user name and password to login to the system. The view is everything, and entering all the details is nothing but the view. If we want to create a webpage, we'll utilize HTML, JavaScript, Angular, and so on. CSS is used to style the website. I used the views coordinator, Display-seminar, Home, Infopopup, and student in my application.

**Controller:-**

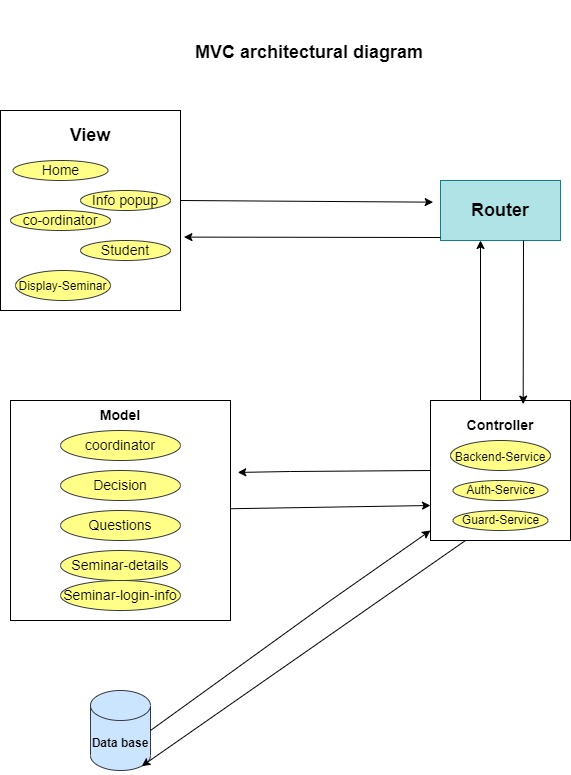
The controller is in charge of coordinating the user's interaction with the interface. That is, if we enter a value in the view, the model will be redirected. If we take the registration page as an example and enter the correct values, these credentials will be checked using the controller, and the data will be written back if the values are correct, or an error notice will be displayed if the values are incorrect. Back-end service, Auth-service, and Guard-service were all utilized in my application.

**MVC working process:-**

* A request will be sent by the user.
* Using routing techniques, the request will be sent to the controller.
* The "controller" will now coordinate the user's interaction with the model.
* The model checks whether the data is there in its database and from the database, it will redirect to the controller after verifying the data with the model (Coordinator, decision, Question, Seminar-details, Seminar-login info).
* The controller (backend-service, Auth-service, Guard-service) will communicate the response to the view using routing strategies.
* It will be displayed to the user

**Advantages of using MVC-architecture:-**

* It can be easily maintained and tested
* If we are working with a team it is beneficial to use MVC
* It can be used to develop larger applications
* Any modification will not modify the entire system
* Faster the development
* It is very easy to identify the error
* It will offer multiple views

****

**7. Acceptance testing**

**(a) Functional testing with four test cases (screenshots of both inputs and outputs)**

**1)Attendant login to the system**

|  |  |
| --- | --- |
| **S.R No** | **Login of Attendant** |
| Description of the test case | Choose appropriate seminar and enter the corresponding password |
| Steps to be followed | 1. Visit the webpage  2. Select Attendant  3. Login with appropriate seminar and password |
| Results to be expected | Login was successful |

**2)Coordinator signup**

|  |  |
| --- | --- |
| **S.R No** | **Coordinator signup** |
| Description of the test case | Coordinators can register themselves by entering a Valid name, User id, and password |
| Steps to be followed | 1. visit the webpage  2.select coordinator  3. Type valid name, User id, and password |
| Results to be expected | Registration successful |

**3)Add seminar details by the coordinator**

|  |  |
| --- | --- |
| **S.R No** | **Add seminar details** |
| Description of the test case | The coordinator can add seminar details by clicking the add button |
| Steps to be followed | 1. coordinator login to the webpage  2. click on add button  3. Type valid Title, Name of the speaker, abstract, content, start date, end date, venue |
| Results to be expected | successfully added |

**4)Attendees post their queries**

|  |  |
| --- | --- |
| **S.R No** | **Post query** |
| Description of the test case | Attendee can post their query after successful login |
| Steps to be followed | 1. Attendee login to the webpage  2. Type a valid query |
| Results to be expected | Posted query successfully |

**(b) Robustness testing with four test cases (screenshots of both inputs and out-**

**puts).**

**1) Creating seminar details with blank fields**

|  |  |
| --- | --- |
| **S.R No** | **Creating seminar details with blank fields** |
| Description of the test case | Creating seminar with blank fields |
| Steps to be followed | 1. coordinator login to the webpage  2. click on add button  3. don’t enter any fields |
| Results to be expected | The message "Error" will be displayed. |

**2)register attendee with an invalid email address**

|  |  |
| --- | --- |
| **S.R No** | **Register attendee with invalid email** |
| Description of the test case | Attendee registration using an invalid email address |
| Steps to be followed | 1. click on the website  2. enter an unformatted email |
| Results to be expected | Enter proper mail format will be displayed |

**3.** **Choose an end date that is earlier than the start date.**

|  |  |
| --- | --- |
| **S.R No** | **Entering end date earlier than the start date** |
| Description of the test case | Entering end date before that of the start date |
| Steps to be followed | 1. click on Add Button  2. choose an end date that is earlier than that of the start date |
| Results to be expected |  |

**4. register coordinator using the same user-id**

|  |  |
| --- | --- |
| **S.R No** | **Registering coordinator using the same user-id** |
| Description of the test case | Registering with the same user-id |
| Steps to be followed | 1. open website  2. choose user id which we have already created |
| Results to be expected |  |

**Time-efficiency testing of two functions**

Three attendees posted queries at the same time-worst case

One attendee posting query-Best case