**RECOMMENDER SYSTEM USING MACHINE LEARNING**

**A Project Report**

*Submitted in partial fulfilment for the award of the degree*

*of*

**Master of Technology**

***in***

**Information Technology**

*by*

**MANISH PRASAD**

**15MIN0017**

*under the guidance of*

**Prof. Nagesh Babu**

Internal Supervisor

VIT University



**School of Information Technology and Engineering**

Feb, 2019



**School of Information Technology and Engineering**

**DECLARATION BY THE CANDIDATE**

I hereby declare that the thesis entitled **“RECOMMENDER SYSTEM USING MACHINE LEARNING”** submitted by me to Vellore Institute of Technology University Vellore, in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Information Technology** is a record of bonafide project work carried out by me under the supervision of **Prof. Nagesh Babu., Internal Supervisor**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**Place**: Bangalore

**Date**: **Signature of the Candidate**



**School of Information Technology and Engineering**

**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled **“RECOMMENDER SYSTEM USING MACHINE LEARNING”** by MANISH PRASAD **(15MIN0017),** to Vellore Institute of Technology University, Vellore, in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Information Technology**, is a project bonafide work carried out by him/her under my supervision. The project fulfills the requirement as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

**Prof. Nagesh Babu**

Internal Supervisor

VIT University

**Internal Examiner(s) External Examiner(s)**

**ABSTRACT**

Recommender systems are found in almost all walks of life helping their users to decide, compare, explore and discover on the basis of information available through online resources or gathered from relevant community. The idea is to extract meaningful information from large quantity of raw data, which can result in affordable, personal and high-quality recommendations. Educational domain is the one field that can take immense advantage from this concept.

The project aims at developing a web based application that would provide academic module recommendations on the basis of students’ expertise level in different domains. The application aims to base its decisions on reliable, authentic and filtered source of information, thus saving its users from the problem of unnecessary information overload. It captures user specific expertise through set of dialogue boxes, requiring simple click actions and generates module recommendations by relating it to underlying knowledge models through suitable mathematical algorithm. While recommending suitable modules, the application takes into account domain constraints such as compulsory modules, credit hour limitations and modules offered in each semester etc. It also enables saving of user profile and module recommendations data for future analysis.

Throughout the project, an established software development methodology was adopted with scheduled timelines and milestones. The application was developed for Windows platform in two iterations. The second iteration was used to address user observations raised on working prototype developed during first iteration. The final product effectively provides a unified, online platform to potential students, requiring guidance in choosing right post-graduate courses.

Project report describes the process of developing the application from requirements gathering phase to evaluating the final application and analyses all relevant points that are worth of discussion.

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to my supervisor/mentor Prof. Nagesh Babu. Whose motivation and guidance helped me in doing this project and has given his part evaluating my efforts on this project and motivating me to improve on the aspects which was required. This project would not have been possible without the co-operation and help from the respected Professors, Wipro's staff and friends I have made over the last three and a half years. My sincere thanks to TT team members, Mr. Hari Prasad and Mr. Prashanth K S, who have given their valuable time in guiding throughout my project completion. Thank you everyone for your help and support through this thesis, I greatly appreciate it.

**Table of Contents**

**Abstract iii**

**Acknowledgements iv**

**Table of Contents v**

**List of Figures ix**

**List of Tables x**

**Chapter 1 Introduction 1**

* 1. Introduction 1
     1. Overview 1
     2. Problem Statement 2
  2. Project Scope 3
     1. Aim 3
     2. Objectives 3
     3. Deliverables 3
  3. Project Management and Planning 4
     1. Development Methodology 4
     2. Milestones and Timeline 5

**Chapter 2 Background Research 8**

* 1. What is a Recommender System 8
     1. Existing Applications in Relevant Domain 9
  2. Possible Implementation Approaches 10
     1. Collaborative Recommendation 10
     2. Content Based Recommendation 11
     3. Knowledge Based Recommendation 12
     4. Hybrid Approach 14
     5. Adopted Approach 14
  3. Relevant Mathematical Algorithms 15
     1. Pearson Correlation Coefficient 15
     2. Cosine Based Similarity 16
     3. K Nearest Neighbours 16
     4. Algorithms Based Upon Machine Learning 16
     5. Adopted Method - 16
  4. Appropriate Development Methodologies and Technologies 18
     1. Salient Features 19
  5. Conclusion 20

**Chapter 3 System Design 21**

* 1. Requirement Analysis 21
     1. Stakeholders Input 21
     2. Functional Requirements 22
     3. Non - Functional Requirements 24
     4. Use Case Diagram 25
     5. Application Workflow / Activity Diagram 26
     6. User Requirement Documents (URDs) 27
  2. Application Design Process 28
     1. System Architecture 28
     2. Conceptual Database Design 29
     3. Logical Database Design 30
     4. Sequence Diagrams 31
     5. Class Diagram (Part -1) 33
     6. Class Diagram (Part - 2) 34
     7. Class Diagram (Part - 3) 35
     8. Graphical User Interface (GUI) Design 36
  3. Conclusion 38

**Chapter 4 System Implementation 39**

* 1. Development Environment 39
  2. Business Logic 39
     1. Academic Module Model 39
     2. User Profile Model 40
     3. Mathematical Algorithm 40
  3. Business Logic Implementation 42
  4. Implementation Details 43
     1. Presentation Layer 43
     2. Business Layer 48
     3. Data Access Layer 50
  5. Source Code 51
  6. Conclusion 69

**Chapter 5 Testing and Evaluation 70**

* 1. Testing Machine Learning70
  2. The oracle problem70
  3. Metamorphic testing71
  4. Test Cases72
  5. Conclusion78
  6. Future Work79

**List of Figures**

[Figure 1.1 Higher education decisions: Information sources 2](#_bookmark0)

[Figure 1.2 Waterfall software development lifecycle 4](#_bookmark1)

[Figure 2.1 Collaborative recommendation approach 11](#_bookmark3)

[Figure 2.2 Content based recommendation approach 12](#_bookmark4)

[Figure 2.3 Knowledge based recommendation approach 13](#_bookmark5)

[Figure: 2.4 Hybrid approach 14](#_bookmark6)

[Figure 3.1 Application use case diagram 25](#_bookmark13)

Figure 3.2 Activity diagram / application workflow 26

Figure 3.3 Application logical architecture 28

Figure 3.4 Conceptual database design 29

[Figure 3.5 Logical database design 30](#_bookmark14)

[Figure 3.6 Sequence diagram: User / admin log-in 31](#_bookmark15)

[Figure 3.7 Sequence diagram: Adding programming expertise 32](#_bookmark16)

[Figure 3.8 Sequence Diagram: Calculate result 32](#_bookmark17)

Figure 3.9 Class diagram: Business and data access layer 33

[Figure 3.10 Class diagram: Presentation layer (log-in, modify module](#_bookmark18) [requirements, result calculation) 34](#_bookmark18)

[Figure 3.11 Class diagram: Presentation layer (configuring user profile) 35](#_bookmark19)

Figure 3.12 GUI: Adding programming expertise 36

Figure 3.13 GUI: User profile summary 37

Figure 3.14 GUI: System recommended modules 38

**List of Tables**

[Table 1.1 Project milestones and timelines 6](#_bookmark2)

[Table 2.1 Confusion Matrix 17](#_bookmark7)

[Table 2.2 Confusion Matrix metrics 17](#_bookmark8)

[Table 3.1 Functional requirements: User log-in/out, registration 22](#_bookmark9)

[Table 3.2 Functional requirements: Academic user profile 23](#_bookmark10)

[Table 3.3 Functional requirements: Module recommendations 23](#_bookmark11)

[Table 3.4 Functional requirements: Modify module requirements 24](#_bookmark12)

**Chapter 1 Introduction**

The first chapter of this report highlights the predicament being faced by the students, and defines the problem statement for its possible solution (Section 1.1). In the light of this, project scope, aims and objectives are defined in Section 1.2. Development methodology used, milestones achieved viz-a-viz scheduled timelines and risk assessment is deliberated upon in Section 1.3. Lastly, the chapter explains the layout followed in the remaining parts of the report (Section 1.4).

* 1. **Introduction**

The significance of choosing the right subject to study cannot be overemphasized, as it is directly linked to students’ performance and personal contentment. The following section explains the traditional factors that affect students’ course selection decision and consequences of making a wrong decision. Additionally, a problem statement is elaborated, proposing a possible solution to resolve the problem at hand.

* + 1. **Overview**

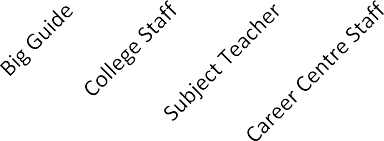
Post-graduate / Higher education plays a critical role in future pathway for students. That is why educationists emphasize upon students to take the most important decision of their life, while keeping all the relevant factors in mind. In general, these factors include, but are not limited to employment prospects, programs offered, educational background, academic interests, future plans and personal capabilities etc. [2].

According to Connor H, et al [3], students adopt different formal and informal ways to decide the most appropriate course / program, they want to pursue for their future studies. Figure 1.1 illustrates different information sources used by students in for higher studies. (Information sources with percentages below 3 % are not referred).

Percentage Cases

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

**Figure 1.1** Higher education decisions: Information sources



45 40.4

40

35

30

25

26.1

20

15

10

5

0

15.9

9.7

9.6

7.8

7

6.7

4.5

Source of Information

Surprisingly, program / module selection by an individual is also affected by some other seemingly unrelated factors like age, country of origin, economic background, ethnicity and sex etc. [3]. Students may also resort to hearsay, internet searches and peer discussions for modules / program selection, which sometimes further aggravates the problem.

Being a multi-dimensional problem with no straight answer, and with all the anxiety the students experience at the start of a new era in their life, they often tend to make mistakes, which stays with them for rest of their life. Even changing modules / programs in the middle of the course, if allowed at all, affects the students’ concentration levels, educational progress and morale adversely. As a result, students fail to continue with the same enthusiasm, with which they started the program resulting in a qualification an individual is not proficient or interested enough to pursue as future career path, or failing it altogether.

As described, there is a dire need of a system that can suggest and recommend potential students regarding their future educational goals, while taking into consideration some of the relevant factors, briefly discussed above.

* + 1. Problem Statement

The aim is to enable potential Masters Students applying for School of Computing (VIT University) to become ‘informed consumers’ and make best decisions for themselves on the basis of reliable, relevant and integrated information instead of relying on irrelevant and anecdotal sources. The proposed software based recommender system would consider individuals’ expertise level in selected domains and would suggest a suitable list of academic modules, on the basis of some mathematical algorithm.

Moreover, the system will apprise the potential student regarding academic implications of accepting system suggested recommendations in terms of entry requirements, course pre- requisites, credit hours info, tentative timetable, teaching methods, summary syllabus, learning outcomes, tentative effort in terms of man-hours, mode of evaluation etc.

* 1. **Project Scope**

Project scope defines the amount of work that needs to be accomplished to deliver a product that meet its stakeholders’ requirements. It involves determining and documenting project goals, objectives and deliverables etc. in the context of project.

* + 1. **Aim**

The aim of this project is to develop a web based software product that can assist Master’s program students in making a personalized decision regarding modules selection according to their own academic and professional circumstances at the start of program.

* + 1. **Objectives**

The following objectives were specified in order to meet the project aim:

* + - * Development of *Academic Module Model* representing academic module requirements for each module, covering different aspects.
      * Development of appropriate *User Profile Model* (representing information that can be fed in by user to get required guidance from the software).
      * Mathematical algorithm interlinking modules information available in the system (*Academic Module Model*) with information provided by user (*User Profile Model*).
      * Integration of above three modules through an interactive, user friendly web based application.
    1. **Deliverables**

The main deliverables within this project will include:

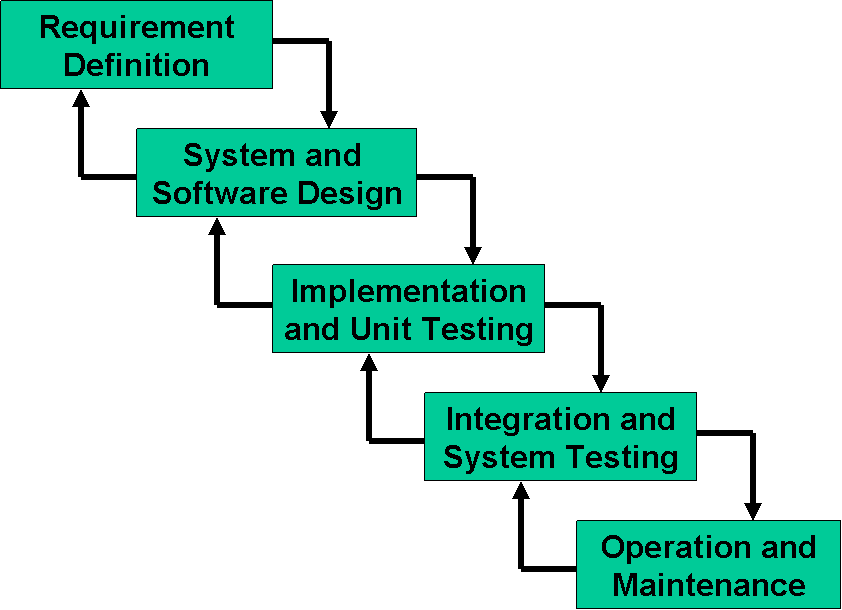
* + - * Academic modules / user profile database.
      * Web based software application.
      * A project report to document rationale of an application, providing background research, design documents, and evaluations.
      * User guide.
  1. **Project Management and Planning**

The process involves choosing appropriate methods required to meet project goals viz-a-viz allocated time and resources. Project management requires organizing activities according to pre-set schedule and controlling them accordingly. Challenges likely to face during the project, and their mitigation strategies are also need to be considered. Management and planning details in the context of this project are discussed below.

* + 1. **Development Methodology**

Software development methodology selected for this project was Waterfall model also referred to as linear sequential model. The model has been slightly modified to add an element of iteration and feedback. The modified methodology allows sequential approach to software development, while also leaving allowance to refer back to previous development stages in case of any mistake.

In principle, the result of each phase is one or more documents which must be approved. No phase is complete until the documentation for that phase has been completed and products of that phase have been approved. The following phase should not start until the previous phase has finished [5]. Different stages of this methodology and relevant project activities are as follows:



**Figure 1.2** Waterfall software development lifecycle

* + - 1. **Requirement Analysis and Definition**

This phase answered the question; “*What is to be made*”. The phase was used to formulate functional and non-functional software requirements and use case scenarios for the project. Apart from consulting relevant literature from Internet sources; informal interviews / use case studies and online surveys were conducted with current Masters Students, faculty staff and project supervisor, in order to have comprehensive user requirements. User requirements document was vetted informally, by project supervisor at the end of this phase.

* + - 1. **System and Software Design**

This phase answered the question; “*How it is to be made*”. The phase was used to design *Academic Module Model*, *User Profile Model* and a mathematical algorithm to link these two models. Moreover, system architecture and conceptual and logical database designs were also finalized during this phase. Relevant design documents were vetted informally, by project supervisor at the end of this phase.

* + - 1. **Implementation and Integration**

System was implemented on the basis of design documents, finalized in previous phase, while using C# and ASP.Net Web Forms technology. Moreover, GitHub will be used as version control system for better code management.

* + - 1. **System Testing**

System was tested for its anticipated functionality according to aim and objectives, initially finalized at the start of project. Moreover, system results were also validated by comparing them with independent expert opinion, while provided with same set of information. Moreover, current Masters Students were also engaged to test its user friendliness, functionality and result output, so as to ensure that all requirements finalized during requirement analysis phase have been met.

* + - 1. **Operation and Maintenance**

After due acceptance from project supervisor and assessor, the product would go into operation and maintenance phase. User guide, explaining basic working of the system has been prepared, to support its smooth operations by the potential students.

**Chapter 2 Background Research**

The chapter describes the background research, carried out in order to attain requisite knowledge required for the application development. *Module Chooser* can be categorized as a recommender system from educational domain. Keeping the fact in mind, literature related to recommender systems and relevant applications was reviewed, with special emphasis on educational domain. (Section 2.1). Possible implementation approaches (Section 2.2) and appropriate mathematical algorithms (Section 2.3) were also studied to understand complete problem, at hand. At the end, appropriate development methodology and technologies were considered for effective implementation (Section 2.3). Brief discussion at the end of each section give reasons behind particular choices made in the context of this project.

* 1. **What is a Recommender System**

Recommender system is a software agent that improves decision making process of its users, by making appropriate recommendations on the basis of their interests, preferences or peculiar requirements, elicited explicitly or implicitly [12]. Additionally, recommender systems have also become valuable mean for dealing with the information overload problem. Explosive growth of World Wide Web and countless sources of information for single query has not only overwhelmed users, but has also led to poor decision making. Thus over-provision of choices and unprecedented load of data has killed the very purpose of it, for which it was initially designed. Recommender systems relieve user from the problem of “too many choices” and present information to the user according to his context and needs [11].

These systems try to mimic the basic human psychology of relying on others’ suggestions and advice, while making decisions in one’s day to day life. The tendency becomes even more prevalent, when that source of recommendation is considered knowledgeable, resourceful, transparent and unambiguous in its approach. For example, it is a common human norm to select a movie on someone else’s suggestion or newspaper review, one trust, or even hiring people on the basis of recommendation letters from renowned institutions, without much hesitation [11].

Similarly, recommender systems are directed towards individuals, not having enough knowledge or competence to decide among various alternatives. In doing so, they generate recommendations on the basis of information available about users, items and previous relationship history between the two, if any. These recommendations may be in the form of ranking list on the basis of user’s preferences, capabilities and constraints etc. The user may select or decline the options with relevant feedback that is stored in the database; and can be used for new recommendations in any of the next sessions [11].

* + 1. **Existing Applications in Relevant Domain**

Today’s technological world is replete with recommendation systems in one form or other; knowingly and even sometimes without any explicit knowledge of its naive user. All big names of digital world like Google, Amazon, Microsoft, Dell, Ali Baba, e-Bay, Netflix etc. uses different recommendation strategies to educate or for that matter lure audience. Apart from these big names, other digital platforms associated with domains like hotels, rentals, tourism, beauty salons, bars - almost anyone and everyone - employs some form of recommendation approach, while dealing with its users [13].

However, exception to this general trend is the education domain, where recommender systems are not so common. Although, being actively researched in academic circles, its usage is still limited in academia. At the moment, potential objectives being pursued in this domain are finding relevant online educational resources, identifying users with similar / advanced / lower academic interests and filtering alternative learning pathways (modules / degrees / courses) to achieve specific goal [9].

Apparent reasons for its limited usage can be attributed to three following facts:

* + - * Recommender systems application in educational domain is fundamentally difficult because the system is not only guided by learner’s preferences, but also by academic criterion [14]. Thus it requires implementation of pedagogical principles, domain ontology and academic pre-requisites etc [9].
      * System evaluation is tricky. In case of product or blog recommendations; sales and number of clicks can be observed respectively to validate system performance, but in case of educational domain, learning output as a result of some recommendation is not only a long-time affair but also a qualitative attribute that can’t be measured directly [9].
      * The systems may be used for in-house recommendations at college and university level and not for general public use.

Nevertheless, some examples from educational domain are discussed briefly for illustration purposes.

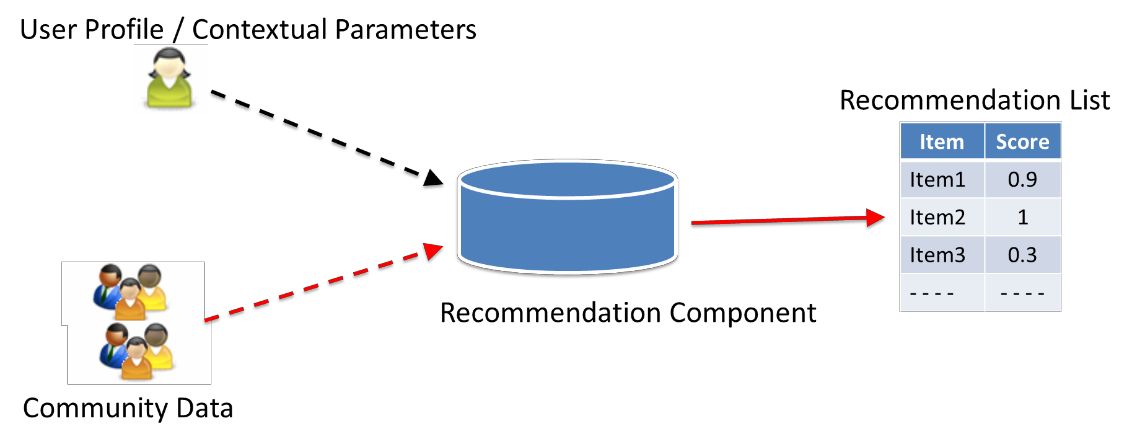
* + - * *CourseRank* system used at University of Stanford, USA assists students, in making informed choices against offered modules. Apart from detailed official information, it provides course ranking system on the basis of students’ recommendations and reviews, while also cross-commenting on other’s point of view. Same system is used as feedback mechanism for faculty as well as administrators. The system had been developed at *Infolab*, Stanford and is open sourced with Berkeley Software Distribution (BSD) license at <http://infolab.stanford.edu/db_pages/courserank/source.html>[16].
      * Research conducted at University of Illinois, USA designed a course recommendation algorithm, suggesting modules not only on the basis of program constraints and course requirements but are also desirable (popular or selected by students of similar interests). System had been implemented by using heuristic algorithms to avoid computational complexity [15].
      * Research conducted at Desh Bhagat University, India proposes a recommender system suggesting graduate courses to potential students on the basis of their previous marks / grades and future job interests. System had been implemented by using clustering techniques to identify structures and relationships within relevant data [17].
      * Joe Manley et al. [18] proposed online recommendation system, assisting students in choosing 10 suitable USA universities in the order of priority, on the basis of user profile (CGPA, GRE score, research experience, Journal publications etc.). The system was implemented by using three different classification algorithms; Support Vector Machine, K-Nearest Neighbours and Random Forest, with best results produced by Support Vector Machine method.
      * Koper et al. [19] proposed an online recommender system on the basis of learning networks, which connects distributed learners and educational resources in relevant domains. The idea was highly flexible and learner centric, extending beyond the formal course and program centric education. Over the period of time, learning network shapes itself on the basis of learning resources provided by stakeholders and their respective evaluation by the users. These higher level user-activity patterns are used as basis for recommendations to new users.
      * Marije De Heus [20] proposed recommender system for suggesting high school courses in the Netherland. The system recommends courses and potential grades in them, on the basis of choices made by similar students in the past. The system had been developed using java implementation of different recommender algorithms provided in *Apache Mahout,* and works on top of *Hadoop* with *map reduce* paradigm, thus also enabling application scalability.
  1. **Possible Implementation Approaches**

Recommendation system may use one of the following approaches / paradigms for its implementation. No one approach is suitable in all domains, and choosing one over another, purely depends upon the problem at hand.

* + 1. **Collaborative Recommendation**

The approach is based upon the universally believed notion that people generally like to take advice from the people with interests similar to their own. The approach takes into consideration the behavior, opinions, preferences and personality traits of large group of users and make appropriate recommendations to them, by matching people with similar interests [7].

To illustrate the concept with the help of an example; if user X and Y shares same purchase history, that largely overlaps in the past, then it can be safely assumed that an item purchased by X would probably has quite high chances of being bought by Y as well, and may be recommended to him. Thus, having similar preferences in the past can be an indication of same interests in the future [8]. Figure 2.1 illustrates collaborative paradigm i-e “*Tell me what’s popular among my peers, having similar interests and attributes etc.*”



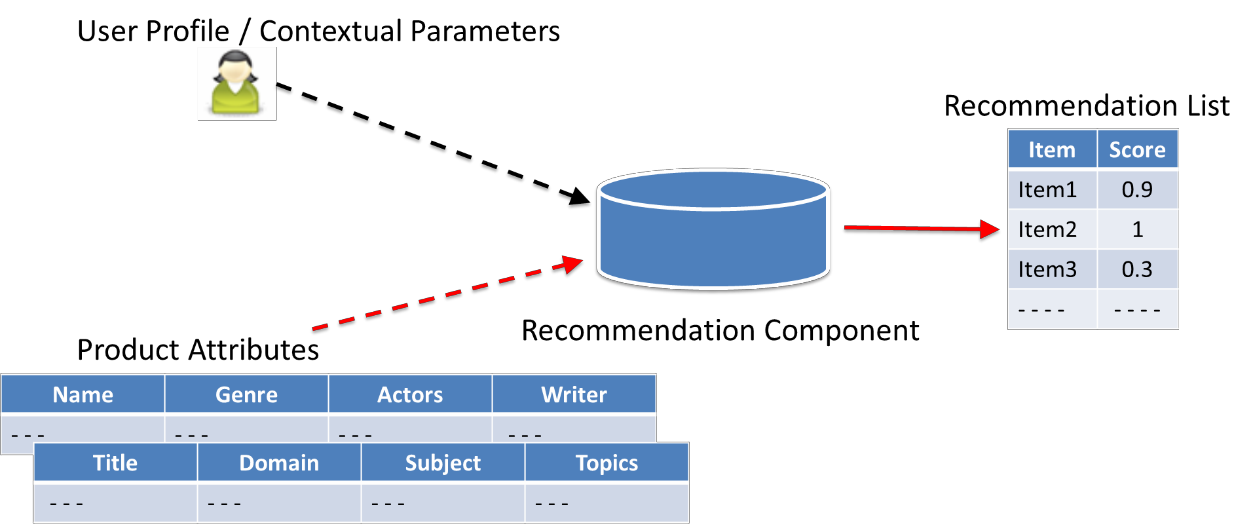
**Figure 2.1** Collaborative recommendation approach

Obvious benefit of using this strategy is that it does not require any domain knowledge about the items themselves. For an online entertainment recommender system using this approach; it does not need to have information about what the movie is about, it’s genre, cast etc. Thus this saves the costly effort of acquiring, maintaining and subsequently exploiting such data for a large scale system. However, the approach suffers from *cold start* problem; i-e it is neither suitable for new items that nobody has bought or used yet, nor applicable to users who do not have relevant history in the system to be used as basis of recommendation [8].

Recommender systems proposed by Koper et al. [19] and Marije De Heus [20] (discussed in Section 2.1.1) employs collaborative approach to recommend ‘educational resources’ and ‘high school courses’ respectively to their users. Similarly *CourseRank* system also employs the same approach to identify popular courses for potential students [16].

* + 1. **Content Based Recommendation**

The approach is based upon the fact; that recommender systems should suggest items to the user, that are similar to those what the user liked in the past. The approach takes into consideration technical and qualitative features of all items and prioritizes new items on the basis of feature similarities between them and the items user has used earlier [8]. Figure 2.2 illustrates content-based paradigm i-e “*Show me more of the same what I've liked.*”



**Figure 2.2** Content based recommendation approach

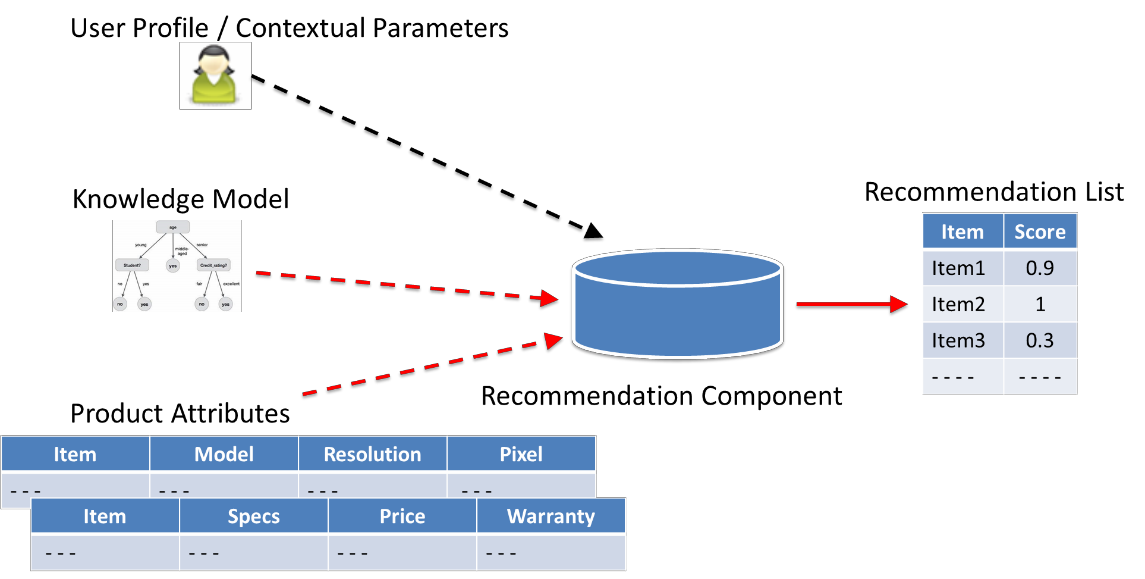
To illustrate the concept with the help of an example; an online book recommender system using this approach would recommend books of specific genre or certain writer to user X, if the system knows through his previous purchase history, that X likes them [8].

The approach is beneficial, as it does not rely on other users for recommendations. Moreover, newly introduced items can easily be recommended, once their features are known. The paradigm is comparatively more transparent as compared to collaborative recommendation approach. However, it does require past purchase history / user activity to suggest him / her more of the same type. Moreover, it also requires efforts in terms of collecting extensive domain knowledge and features which may or may not be automatically extractable. Collecting subjective or qualitative data like “ease of use” or “design elegance” about certain item may be tricky to collect and prone to human indulgence and errors [9].

Another potential pitfall of this approach is filter bubble; an intellectual isolation caused due to recommender system inbuilt inclination of suggesting things that abide by user’s past activity. Perfect industry examples of this approach can be seen in the form of personalized search results from Google and Facebook news stream [10].

* + 1. **Knowledge Based Recommendation**

The approach aims to deal with information overload by discriminating between relevant and irrelevant items on the basis of what the user actually requires or needs. Its simplest implementation would be to use conversational approach and ask user directly about his or her peculiar requirements through set of appropriate options. These options would be correlated with already available domain related features of each and every item, and prioritizing them subsequently, on the basis of some mathematical algorithm / knowledge model [8]. Figure 2.3 illustrates knowledge - based paradigm i-e “*Tell me, what fits based on my needs.*”



**Figure 2.3** Knowledge based recommendation approach

The approach may give an option of prioritizing relative importance of features or take into account the contextual background (user location, gender or profession etc.) to give personalized touch to resultant outcome. Another variant of this approach may also adopt incremental conversational style by shaping-up personalized future queries and dialogs on the basis of earlier responses and chosen options [8].

To illustrate the concept with the help of an example; a property recommender system may use detailed house features like type of property, mortgage options, number of rooms, room size, and price etc. and compare it against peculiar requirements of user X. System may also prioritize these features as per user’s discretion, by giving them different numerical weights or contextualize the output on the basis of user’s location. It is evident, that X in this case may be more interested in viewing recommendations based upon his / her current property requirements, instead of viewing recommendations on the basis of what other people of similar purchase habits have bought over the period of last ten years (collaborative recommendation) or what he / she has bought previously in the relevant domain (content-based recommendation). Additionally, it is highly likely, that system may not have any data at all, for user X in the later case [11].

Thus, it can be concluded that the approach is suitable for domains; where no previous usage / purchase history exists for the item or for the user himself who is opting to choose or buy it. Similarly the approach is very much applicable in the situations where the relevant items, because of their nature or associated cost, are traditionally chosen not too frequently like car, house, electronic products etc. [9]. However, for its successful implementation, the approach must efficiently utilize the explicit user requirements acquired during the process and must also have the deep knowledge of underlying business domain [11].

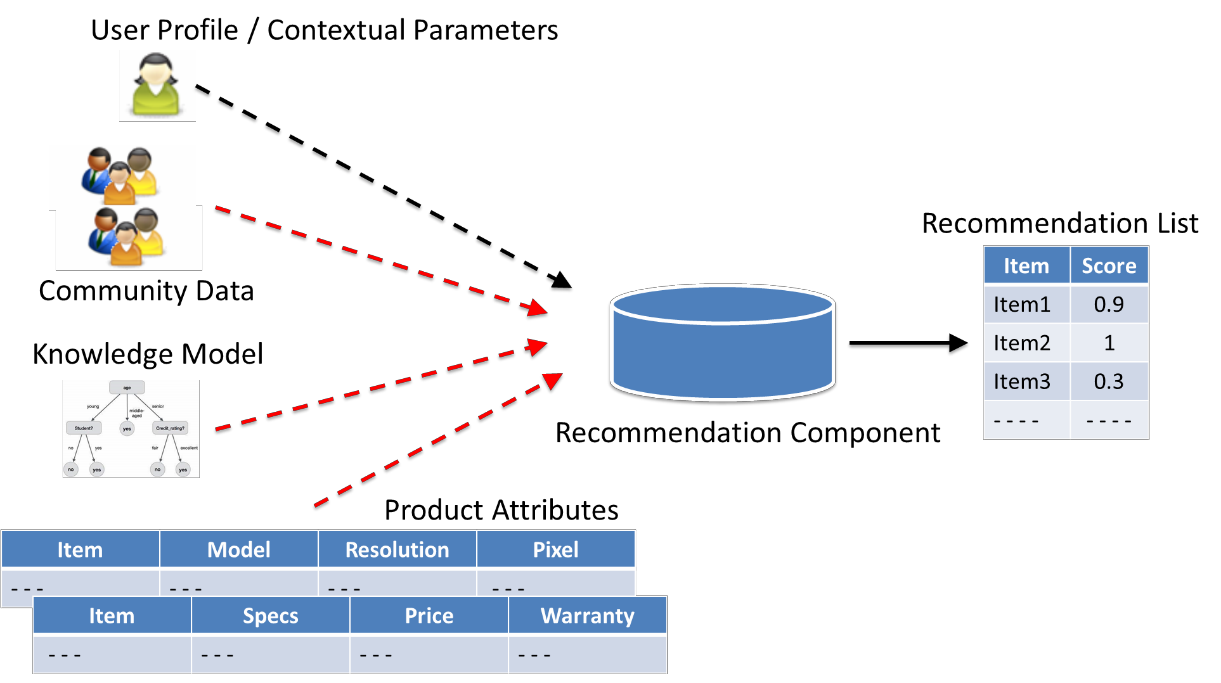
Joe Manley et al. [18] (discussed in Section 2.1.1) uses same approach to recommend universities to the students, in which they have better chances of being admitted.

* + 1. **Hybrid Approach**

The approach is based upon the fact that, no single approach is perfect and has its own set of advantages and disadvantages depending the domain under consideration. So the most pragmatic strategy would be to use a set of approaches in combination, thus overcoming innate shortcomings of one approach through advantages of the other.

In hybrid approach, collaborative filtering can be implemented effectively along with knowledge based recommendation approach. The idea is to take into account all the benefits of collaboration (past input from users of similar interests) and one’s own current peculiar requirements, while also avoiding *cold start* problem at the same time [9]. Moreover, from the design point of view, such a recommender system can either use combination of approaches in parallel with varying associated weights, or alternatively can connect two or more approaches in pipelining architecture, in a way that output of first recommender acts as an input to the next one. Hybrid approach implementation is an important candidate for future work and is also discussed in Section 5.5 in the context of this project [8]. Figure 2.4 illustrates hybrid paradigm.

The approach has been employed by research work [15] at University of Illinois. (Discussed in Section 2.1.1). It successfully implements collaborative as well as knowledge based approaches to design a system that recommends post-graduate courses.



**Figure: 2.4** Hybrid approach

* + 1. **Adopted Approach - Masters Project**

While deciding an appropriate recommendation approach, it is important to consider salient features of problem, we are trying to solve through current project.

* + - * Assisting **Masters** Students (users) in choosing academic modules (items) at the start of their academic session through some well-defined recommendation process.
      * The process would generally be required to recommend modules for new students, who have no past history of using this application.
      * The process may have to recommend new modules to students, which have never been chosen by other students.
      * Because of application domain, it is safe to assume that:
        + The application may only attract large number of one-time users (at the start of programme); unlike an online store; which may see frequent visits by a single user.
        + It is possible to ask users about their academic expertise in specified academic domains that can be used as basis of a user profile.
        + It is reasonably possible to identify and define underlying domain knowledge in terms of module requirements, constraints etc.

Consequently, it can be concluded that the ideal case would be to use hybrid approach, however considering the time constrained environment, it was deemed feasible to adopt knowledge based approach. The approach would be useful to avoid cold start problem, as user requirements (user profile) would be elicited explicitly within the same recommendation session. Moreover, being a **Masters** project under controlled environment, domain knowledge acquisition is possible, which is a pre-requisite for knowledge based approach. Lastly, the approach would not be hindered by new user / new module problem and would be able to handle one-time visits of users.

* 1. **Relevant Mathematical Algorithms**

Finding an appropriate relationship between user profile and items’ attributes is a fundamental step of any recommendation system. This relationship defines a similarity measure between user - user or item - items or between user requirements and item’s attribute depending upon the type of recommendation approach, being implemented. Some of the mathematical algorithms reviewed are briefly discussed below.

* + 1. **Pearson Correlation Coefficient**

The concept defines similarity between two variables X and Y by identifying linear correlation between the two. Coefficient value varies between +1 and -1, where +1 represents complete positive linear correlation, -1 indicates total negative correlation and 0 being no correlation at all. It can be derived as:

𝑃𝑒𝑎𝑟𝑠𝑜𝑛 (𝑋, 𝑌) = 𝐶𝑂𝑉 (𝑋,𝑌)

𝜎𝑋 × 𝜎𝑌

Where 𝐶𝑂𝑉 is the covariance and 𝜎 indicates standard deviation [11].

* + 1. **Cosine Based Similarity**

Similarity between the two variables is measured by representing them as vectors, 𝑗 in m-directional user space, and calculating the cosine angle between the two. The value varies between 0 (not similar at all) and 1 (100% similar) [22]. It can be computed as:

𝑠𝑖𝑚 (, 𝑗) = cos (𝑖 , 𝑗) = ( 𝑖 . 𝑗 )

||𝑖|| ||𝑗|

Where the nominator indicates dot product of two vectors, and ||𝑖|| is the norm of vector.

* + 1. **K Nearest Neighbours**

It is a non-parametric method, used for pattern recognition and classification. An instance / item is classified by majority vote of its neighbours i.e. its classification depends upon the most common class among its K nearest neighbours. In this case, choosing K value is crucial, as too high value would include too many neighbours without much similarity, and a lesser value may adversely affect the output by listing very few results [8].

* + 1. **Algorithms Based Upon Machine Learning**

Apart from methods discussed above, there are number of other algorithms from machine learning domain, that have been used successfully to classify the two main components of recommender system; users and items, on the basis of their associated features. Prominent among them are linear / logistic regression, neural networks, support vector machines, decision trees, clustering and probabilistic methods like Naïve Bayes etc.

* + 1. **Adopted Method - Masters Project**

The mathematical algorithms discussed in section 2.3.1, 2.3.2 and 2.3.3 are more suitable for collaborative and content based recommendation paradigms, where similarities among users or within items are important to be identified and recommendation process works either on the basis of “*show me what my peers of similar interests like*” or “*show me more of what I liked in the past*” paradigm. Moreover, machine leaning algorithms referred in Section 2.3.4 are suitable for all type of recommendation approaches, but none of them was adopted in this project because of their relative complexity and lack of historical data sets in relevant domain. Instead an innovative but simple approach was adopted, by using confusion matrix and fuzzy logic; details of which are discussed in Section 2.3.5.1 and 2.3.5.2.

* + - 1. **Confusion Matrix**

A confusion matrix is a 2 × 2 table that is often used to describe the performance of a binary classification model on a set of test data for which the true values are already known [21]. It is represented as shown in Table 2.1:

**Table 2.1** Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **True Values (Module Requirements)** | |
| **Yes** | **No** |
| **Predicted Values**  **(User Profile)** | **Yes** | True Positive (T.P) | False Positive (F.P) |
| **No** | False Negative (F.N) | True Negative (T.N) |

Different metrics like *Precision*, *Recall*, *Accuracy*, *Specificity*, *Prevalence*, *True Positive Rate*, *False Positive Rate, Misclassification Rate* and *F1-Measure* etc. can be derived from confusion matrix and highlight different aspects of the model [21]. Details for some of them are shown in Table 2.2:

**Table 2.2** Confusion Matrix metrics

|  |  |
| --- | --- |
| **Metric** | **Formula** |
| Precision | 𝑇. 𝑃  𝑇. 𝑃 + 𝐹. 𝑃 |
| **Recall** | 𝑻. 𝑷  𝑻. 𝑷 + 𝑭. 𝑵 |
| Accuracy | 𝑇. 𝑃 + 𝑇. 𝑁  𝑇𝑜𝑡𝑎𝑙 𝐶𝑎𝑠𝑒𝑠 |
| Specificity | 𝑇. 𝑁  𝑇𝑜𝑡𝑎𝑙 𝐶𝑎𝑠𝑒𝑠 |
| F1 - Measure | 2 ∗ 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 ∗ 𝑅𝑒𝑐𝑎𝑙𝑙  𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 ∗ 𝑅𝑒𝑐𝑎𝑙𝑙 |

For this project; the concept has been adopted in a way that, *True Values* in Table 2.1 are assumed to be feature values relevant to the business domain i-e academic module requirements for each module, and *Predicted Values* are assumed to be user profile values based upon same features. We would be interested in *True Positives* (module requirements, required by specific module and also available in the user profile) and *False Negatives* (module requirements, required by specific module but not available in the user profile). Conversely, we would not consider *False Positives* (module requirements, not required by specific module and available in the user profile); as they do not affect user eligibility against any module and *True Negatives* (module requirements, not required by specific module and not available in the user profile); as these features are out of scope from problem domain.

Considering Table 2.2, the *Recall* metric was selected to relate the academic module requirements against respective user profile attributes and for subsequent priority rankings. Implementation details regarding mathematical logic are discussed in Section 4.2 and 4.3.

* + - 1. **Fuzzy Logic**

Fuzzy logic represents human reasoning in computing world and is used to represent natural language that cannot always be translated into absolute terms of 0 or 1. In context of this project, it is being implemented in certain cases, where it is required to implement ‘degree of truth’ or ‘partial truth’, rather than representing ‘completely true’ or ‘completely false’ facts [37].

* 1. **Appropriate Development Methodologies and Technologies**

In terms of development methodologies, various concepts like Waterfall, Rapid prototyping, Object Oriented lifecycle and eXtreme programming were reviewed. Waterfall methodology was selected because of its document driven approach, simplicity and small size of the project. It requires to have specific deliverables at the end of each phase to gauge the incremental progress which is an added advantage.

The web based application was implemented using ASP.Net (Active Server pages), which is built on top of *Common Language RunTime* (CLR), allowing use of any .Net language for its implementation. It is an open source server-side web application framework to develop dynamic websites and web applications, using HTML, CSS and JavaScript. It offers three implementation technologies for web development: *MVC, Web Forms*, and *Web Pages*. The three technologies differ in their developmental approach and level of expertise required to handle them [23].

*ASP.Net MVC* is a presentation technology, based upon *Model-View-Controller* (*MVC*) architectural pattern. It provides “separation of concerns” by dividing application components into *Model* (responsible for maintaining data), *View* (user interface) and *Controller* (logic that communicates between Model and View) [23]. *ASP.Net Web Forms* is an even driven model that gives an option of rapidly developing powerful, user interface driven web sites with database access. The technology comes with rich library of server controls, which have their own set of properties, methods and events, defining page design and behaviour [24]. *ASP.Net Web Pages* is a classic, light-weight Microsoft technology to create dynamic content on any web site, and to handle simple tasks like calculate values, reading / writing database or calling other programs. It supports *Single Page Application* (SPA) model, and has HTML and code in the same file [25].

All implementation technologies discussed above are based upon .Net framework, are supported by main stream languages like Visual basic and C#, and can be used to develop

web applications. Moreover, the technologies are not entirely independent and can be used simultaneously in one project [23].

Choosing one over another, depends upon the preferred development style, application scale, level of control required over an application behaviour and time available. Language C# with *ASP.Net Web Form* was used for this project because of its rapid development approach (through set of rich control libraries), even driven programming model, architectural support to isolate user, business and data layers, intuitive state management, simple database handling and comparatively small learning curve. The approach is more suited for developing small to medium scale applications as compared to *MVC*. Although, *MVC* has almost all the features discussed in context of *Web Forms*, but it is suitable for large scale applications requiring greater degree of control / flexibility and involving large development teams. Moreover, it is time consuming to implement and has a relatively large learning curve. Lastly, *ASP.Net Web Page* was not considered for this project, because it does not support any “separation of concerns” between business logic and user interface [26] [27].

* + 1. **Salient Features**

Relevant features / concepts of *ASP.Net Web Forms* that have been reviewed for implementation are discussed below.

* + - 1. **Code Behind Model**

Code-behind model isolates user interface (HTML tags, ASP.Net server controls etc.) from dynamic program code (page load / initialization code, event handlers etc.) by keeping them in separate files (*SamplePage.aspx and SamplePage.axpx.cs* respectively). In practice, this allows the developer to focus on one dimension, without disturbing the other. The approach is similar to isolating *View* from *Controller* in *MVC* framework [28].

* + - 1. **State Management**

ASP.Net uses stateless HTTP protocol to communicate between client and server. However, current value of all the controls / variables for current user in current session is required to be maintained, in order to keep the “state full” interaction. ASP.Net defines *View*, *Control*, *Session* and *Application* States, to do the job. *View* and *Control* states are configured by default. However, *Session* state needs to be managed effectively against globally unique session ID, which goes from client to server and back as a cookie, and stores requisite information throughout the session [29]. *Application* state is out of the scope of this project and is not being used.

* + - 1. **Database Handling**

ASP.Net uses ADO.Net (ActiveX Data Objects) to connect front end controls with back end database. ADO.Net encapsulates all data access operations and interact with controls to

provide data access services, while hiding all implementation details. It uses concepts such as *DataTable* (in-memory table representing table from database), *DataSet* (in-memory container having subset of data from database), *DataAdapter* (mediator between the DataSet and database) and DbConnection (connection to data source) to provide these services. Under ADO.Net framework, connections are only maintained long enough to fetch or update data, thus ensuring efficient data handling [29] [30].

* + - 1. **Security**

ASP.Net provide *Windows*, *Forms*, *Passport* and *Custom* based authentications to ensure application security. This project uses *Forms* based authentication that involves creation of database for saving user’s credentials and provision to register, log-in / out and change password etc. Moreover, ASP.Net may also uses *Secure Socket Layer* (SSL) protocol to encrypt / decrypt server-client communication by using digital certificates [29].

* + - 1. **Application Portability**

ASP.Net applications are developed using Microsoft Visual Studio on Windows platform. Moreover, the application can only be deployed on Windows machine, using Internet Information Server (IIS). However, once deployed, it is compatible to be accessed through all major browsers.

An alternative approach was studied to use *ASP.Net Core* (next generation of ASP.Net) as web framework and *Kestrel* hosting server, which are platform independent and would enable application development and deployment on any operating system - Windows, Mac or Linux. The approach was not further pursued, because of its being a relatively new concept (introduced in 2016) and lack of standard documentation / tools [31].

Other ASP.Net features such as application lifecycle, directory structure, event handling, server controls and error handling were also deliberated to ensure smooth application development.

**2.5 Conclusion**

The chapter explains the background research conducted in the field of recommender systems with special reference to educational domain, possible implementation approaches, appropriate mathematical algorithms and relevant development technologies. Particular choices made in the context of this project were also discussed. In the light of these choices, the next chapter would elaborate on different requirement and design documents developed, to realize the proposed system.

**Chapter 3 System Design**

The chapter lays out design outlines by discussing in detail first two stages of the development methodology, adopted for the development of *Module Chooser* application: requirement definition / analysis phase and software design phase. While discussing requirement analysis phase, Section 3.1 briefly touches upon different stakeholders’ input, functional and non- functional requirements, use case diagram and application workflow. The section concludes by discussing user requirements’ documents (URDs) related to application business domain, duly prepared during this phase. Section 3.2 of the chapter elaborates salient design documents like system architecture, conceptual and logical database designs and sequence diagrams representing dynamic behaviour of the system. The section also illustrates class diagrams, highlighting relationship between important classes and functions by categorizing them into presentation, business and data access layer. At the end, graphical user interface (GUI) design is shown, indicating salient application features.

* 1. **Requirement Analysis**

Requirement analysis phase is the process of formalizing user expectations or “statement of needs”, that triggers the development of a new software product [6]. The requirements must be complete, unambiguous, quantifiable, consistent and achievable within allocated time. Moreover, the primary focus at this stage is to finalize what to build and not how to build it.

* + 1. **Stakeholders Input**

Stakeholders’ input plays critical role in the success of any project, which cannot be overemphasized. The first step is this direction is to identify appropriate stakeholders, keeping in mind the project scope, aim and objectives. To formulate comprehensive user requirements within allocated time, the following stakeholders and sources of input were identified and consulted, in case of *Module Chooser* application.

* + - * Current / potential **Masters** Students
      * Project supervisor
      * Resource material available on Internet, university etc.

Out of all of the above, Masters students are primary stakeholder, as they are the actual users and can most realistically comment upon features the product must have, and be considered useful for them. In this regard, an online survey was conducted to take input from the maximum number of users within the shortest possible time. The survey was aimed to take respondents’ point of view regarding their academic background, preferred source of guidance for academic module selection, probable factors for selecting any module in the order of priority and future plans.

Of the total 24 responses received, an option of “*talking to module leader*” ranked at the top as a preferred source of guidance in module selection process. Other options selected by students in the order of priority were: Academic module catalogue, Internet, past students’ selections, educational recommender system and hearsay. Interestingly, an educational recommender system ranks quite low in the list, which can be attributed to its’ being a relatively new concept in education field. However, all the preferred sources of guidance as indicated by students were either taken as an input in proposed mathematical model or recommended for future work in Section 5.5 (e.g. past student’s selections, hearsay).

Similarly, while prioritising role of different factors in module selection process, students’ choices in the order of priority were: professional tools handling, related working experience, mathematics aptitude, future career plans, academic interest, research expertise, and programming expertise. Except related working experience, future career plans and academic interest, all other factors were implemented in current project. Out of left out factors, “academic interest” is further deliberated in Section 3.1.6.4 and recommended for future work in Section 5.5.

Moreover, regular weekly sessions with project supervisor during requirement analysis phase also refined ideas regarding application features. While doing so, initial draft of user requirements document (URD) and online survey feedback was also validated by project supervisor in terms of its scope, technical feasibility, consistency and time constraints; and amended accordingly.

* + 1. **Functional Requirements**

Functional requirements present the system’s perspective and indicate the minimum product functionality, required to satisfy requirements indicated in URD [6]. Functional requirements as derived from user requirements forwarded by different stakeholders of this project are described in following tables.

**Table 3.1** Functional requirements: User log-in/out, registration

|  |  |
| --- | --- |
| FR01-01 | System shall allow new user to register by providing required personal details. |
| FR01-02 | System shall allow existing user to log-in with user name and password to access application interface. |
| FR01-03 | System shall allow existing user to change one’s own password. |
| FR01-04 | System shall allow Admin user to log-in with required credentials to view admin Interface. |
| FR01-05 | System shall ensure that user is not able to register with already used user name. |
| FR01-06 | System shall allow existing user to view one’s own already saved academic user profile and finalized module selections. |
| FR01-07 | System shall allow existing user to modify one’s own already saved academic user profile and finalized module recommendations. |
| FR01-08 | System shall prompt user to submit application feedback / comments, at the time of logging out. |

**Table 3.2** Functional requirements: Academic user profile

|  |  |
| --- | --- |
| FR02-01 | System shall allow user to select options regarding programming language, programming platform and domain with relevant experience level, to indicate one’s programming expertise. |
| FR02-02 | System shall allow user to select options regarding theoretical and applied mathematics with relevant experience level, to indicate one’s math’s expertise. |
| FR02-03 | System shall allow user to select option regarding research domain with relevant experience level, to indicate one’s research aptitude. |
| FR02-04 | System shall allow user to select options regarding different tools with relevant experience level, to indicate one’s expertise in each tool. |
| FR02-05 | System shall allow user to view user profile summary / selected options, at the end of process to have broader picture. |
| FR02-06 | System shall allow user to modify any of the user profile options by moving back and forth on relevant GUI pages. |
| FR02-07 | System shall allow user to reset user profile options of each domain to a default selection i-e *No Experience* |
| FR02-08 | System shall allow user to save user profile in application database. |

**Table 3.3** Functional requirements: Module recommendations

|  |  |
| --- | --- |
| FR03-01 | System shall allow user to view priority rankings of each academic module, as calculated by underlying implementation logic. |
| FR03-02 | System shall allow user to view semester wise recommended modules. |
| FR03-03 | System shall allow user to view detailed information associated with each module like No of credit hours, evaluation mode, theoretical and practical portion, coursework’s weightage and university module catalogue link etc. |
| FR03-04 | System shall allow user to view the missing module requirements (False Negatives) in one’s user profile against each academic module. |
| FR03-05 | System shall allow user to manually change software recommended module options, on the basis of detailed module information available. |
| FR03-06 | System shall allow user to reset module selections to software recommended settings. |
| FR03-07 | System shall allow user to save finalized module selections for each semester to application database. |

**Table 3.4** Functional requirements: Modify module requirements

|  |  |
| --- | --- |
| FR04-01 | System shall allow admin user to view module requirements in programming, mathematics, research and tools domain with relevant experience level required for each academic module. |
| FR04-02 | System shall allow user to add, delete and amend module requirements for each academic module, from given options. |
| FR04-03 | System shall allow user to save modified module requirements to application database. |

* + 1. **Non - Functional Requirements**

Non-functional requirements specify the design restrictions, implementation constraints, quality and performance standards, to which the application must conform [6]. In case of *Module Chooser*, non-functional requirements are:

* + - * User-friendly interface with consistent use of layouts and colours.
      * Application may be extensible in terms of incorporating other master and undergraduate programs.
      * Application must be stable and reliable in performance.
      * User personal data and user academic profile must be securely stored in database.
      * Application should be browser independent.
    1. **Use Case Diagram**

Use case modelling is being used to structure the user requirements in a way, that is understandable for users and developers, both. The model indicates how the system is going to be used and normally has two components; use cases and actors. Figure 3.1 indicates use case diagram for *Module Chooser*, having two actors (user and administrator) and 11 use cases, inside system boundary (blue rectangular box with black boundary). Each use case indicates certain functionality within an application initiated by an actor.

For example, “*Add Programming Expertise*” use case (oval with dashed boundary) indicates the functionality of configuring programming expertise against a user profile as initiated by the potential student (actor). Moreover, the same use case diagram also acts as a reference for application testing and evaluation process [6].





User

User LogIn

Add Programming Expertise

Add Mathematics Expertise

Admin LogIn

Add Research Expertise

View Module Requirements

Add Tools Expertise

Modify Module Requirements

View / Amend User Profile Summary

View / Amend Recommended Modules

User feedback

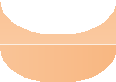
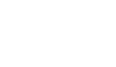
Admin

**Figure 3.1** Application use case diagram

* + 1. **Application Workflow / Activity Diagram**

Unified Modelling Language (UML) is being used to illustrate dynamic aspects of the system and represents application workflow from one activity to another activity [40].

Data Submission



Save Module Requirements

Start

Log-In / Registration

No

Registration

Yes

Data Submission

Existing User ?

Requirements Finalized ?

No

Modify Module Requirements

Yes

Log-In

Application

Database

Admin Credentials

User Credentials ?

User Credentials

Invalid Credentials

Error Message

End

Error Message

Save Module Recommendations

No Yes

Data Submission

Valid Change ?

Make Manual No Changes

Data Submission

Accept ?

Yes

View Module Recommendations

Config User Profile (Programming)

Config User Profile (Mathematics)

Config User Profile (Research)

No

Config User Profile (Tools)

User Profile Summary

Submit ?

Yes

**Figure 3.2** Activity diagram / application workflow

Figure 3.2 illustrates three application paths at the time of log-in. First path marked with black arrows represents log-in with right user credentials and indicates complete user activities from start till end. Second path also marked with black arrows is for wrong user credentials, and does not do anything except showing an error message. The last one marked with red arrows, indicates admin log-in, used to modify academic module requirements, whenever required. Dotted arrow lines describe data services activities within an application.

* + 1. **User Requirement Documents (URDs)**

Apart from documents already discussed in the previous sections, the requirement analysis phase also produced critical documents related to application business domain. Details of these documents are as follows:

* + - 1. **Academic Module Model**

The document shows the proposed model used to represent module requirements for each Masters module in terms of programming experience, mathematics aptitude, research background and tools expertise through 30 different attributes with four possible values i.e. *No Knowledge Required*, *Novice*, *Intermediate*, *Expert*.’.

* + - 1. **Academic Module Requirements**

The document indicates actual module requirements for each Masters module on the basis of the *Academic Module Model*. It is the most critical part of requirement analysis phase, as the underlying business logic used during implementation phase solely relies on its correctness. The major source of this information was current Masters Students’ own experience and valuable discussion during one-to-one sessions with project supervisor.

* + - 1. **Academic Module - Associated Information**

The document describes modules associated information such as number of credit hours, evaluation mode, theoretical and practical portion, coursework’s weightage and university module catalogue link etc. The information is not part of *Academic Module Model*, but is considered useful, if user opts to make manual changes in module selection. Major source of this information was online academic module catalogues and School of Computing Handbook (Level 5) [38] [39].

* + - 1. **Academic Module - Keywords**

The document defines groups of keywords for major concepts and topics related to each Masters module. The keywords have been shortlisted on the premise that certain student with specific academic interests may be offered an academic module on the basis of their similarity with keywords associated with any academic module. The feature is not implemented in current release, but is strong candidate for future work.

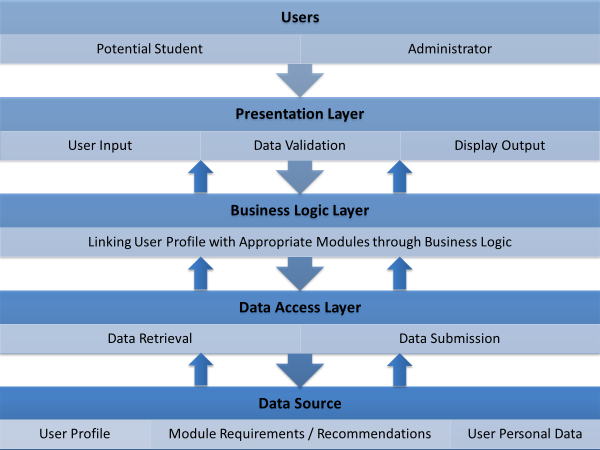
* 1. **Application Design Process**

Application design is the process of transforming user requirements into a blueprint that will help the programmer in software coding and implementation. It describes a roadmap to progressively transform requirements into a number of design models, while looking at the future system from different angles and at different abstraction levels. Primary focus of this phase is how to make a system, on the basis of what is already finalized in requirements phase. Salient aspects of the design process are discussed in following sections.

* + 1. **System Architecture**

The system was implemented as web based application. From a physical architecture point of view, it is a single tiered application running on a standalone PC / laptop. Whereas, from logical architecture point of view, the application has three-tiered architecture.

A tiered architecture is modular approach to design a system, within which, each tier (organizational unit) is a complete subsystem, encapsulating a single responsibility independently. As a result, implementation of any tier can be amended or changed altogether, without having to re-develop the entire application. Each tier provides some service to layers above, while consuming services from the below layers [4]. The three logical layers (tiers) and their respective role in the system is explained with the help of Figure 3.3.



**Figure 3.3** Application logical architecture

* + 1. **Conceptual Database Design**

As a result of system analysis, conceptual database design has been drawn with the help of Entity Relationship Diagram (ERD), using Crow's Foot notation [32]. (Figure 3.4). Entities are represented by rectangular boxes, whereas diamonds indicate the relationship between them. Inner line symbols represent minimum cardinality, whereas the outer symbols indicate maximum cardinality between relationships. *UserProfile* is a ternary relationship between *User*, *AcademicInfo* and *ExpertiseLevel* entities, whereas *ModuleRequirement* is another ternary relationship between *AcademicModule*, *AcademicInfo* and *ExpertiseLevel* entities. Rest all the relationships indicated in Figure 3.4 are binary in nature.

Furthermore, entities and relationships marked in red are incorporated, keeping future work in mind, but are not implemented in final release.



WebAddress

Academic Module

Has

Has

ModuleRequirement

Has

Has

UserProfile

Has

Chooses

ModuleManager

Category

AcademicInfo

ExpertiseLevel

AcademicInterests/ Keywords

User

**Figure 3.4** Conceptual database design

* + 1. **Logical Database Design**

Logical database design has been obtained by translating the conceptual database design (Figure 3.4) into relational data model and is shown as Figure 3.5. It is descriptive in nature and relatively less expressive as compared to conceptual database design. It is this design, which is actually going to be implemented. During the translation process, entities from conceptual database design would be converted in to tables and their attributes (not shown in Figure 3.4) would be replaced by tables’ columns, having appropriate data types. Moreover, ‘*one to many’* cardinality in contextual database design is represented by adding primary key of entity type against ‘*one’* side to entity type against ‘*many’* side as a foreign key. In case of many to many relationships, a third table is created for the relationship, having primary keys of the participating entity types as its composite primary key.

Figure 3.5 only shows the primary and foreign key columns against each table, for the sake of brevity. Complete design can be viewed on SQL Server Management Studio by accessing database files from GitHub.



**AcademicInterest**

**User**

User\_id



**Domain**

Domain\_ID



|  |  |  |
| --- | --- | --- |
| **AcademicInfo** | |  |
|  | AcademicInfo\_ID |
|  | Domain\_ID |
|  | |

|  |  |
| --- | --- |
| **ModuleManager** | |
|  | ModuleManager\_ID |
|  | |

|  |  |  |
| --- | --- | --- |
| **UserProfile** | |  |
|  | UserProfile\_ID |
|  | AcademicInfo\_ID |
|  | User\_ID |
|  | Expertise\_Level |
|  | |

|  |  |
| --- | --- |
|  | AcademicInterest\_ID |
|  | AcademicModule\_ID |

|  |  |
| --- | --- |
| **ModuleSelction** | |
|  | ModuleSelction\_ID |
|  | AcademicModule\_ID |
|  | User\_ID |
|  | |

|  |  |  |
| --- | --- | --- |
| **ModuleRequirement** | |  |
|  | ModuleRequirement\_ID |
|  | AcademicInfo\_ID |
|  | AcademicModule\_ID |
|  | ExpertiseLevel\_ID |
|  | |

|  |  |
| --- | --- |
| **AcademicModule** | |
|  | AcademicModule\_ID |
|  | ModuleManager\_ID |
|  | |

**Figure 3.5** Logical database design



|  |  |  |
| --- | --- | --- |
| **WebAddress** | |  |
|  | WebAddress\_ID |
|  | AcademicModule\_ID |
|  | |

|  |  |
| --- | --- |
| **ExpertiseLevel** | |
|  | ExpertiseLevel\_ID |
|  | |

* + 1. **Sequence Diagrams**

Sequence diagrams describe dynamic behaviour of a system. They are designed to model the way, a use case is realized through a sequence of messages between objects. Classes (marked in blue) are arranged on X-axis, whereas time is shown on Y-axis. Green rectangles on life-lines (dotted vertical lines) are activation boxes, that indicate for how long a particular message will remain active. Moreover, the arrows along with function names depict the actual messages being sent from one object to the other in direction of an arrow. Dotted arrows indicate the return values for respective functions and terms inside the square brackets indicate conditions, that need to be fulfilled, before a message can be activated. Sequence diagrams representing some of the major functionalities are shown in Figure 3.6, 3.7 and 3.8:

* + - 1. **User / Admin Log-In**

**X – AXIS (Objects)**

:DBAccess

UserID

DataTable

:PrResultSummary

:DBAccess

:LogIn

|  |  |
| --- | --- |
| :LogInInfoCalc | |
| rName, | FillLoc DataTab string |

LoginButton\_Click(object sender, EventArgs e)

alTable(

GetUserInfoRecord(string strUse

string strpass)

le tblName, query)

[UserId = ‘Admin’] Response.Redirect("~/AdminPanel.aspx")

[UserId NotEqual ‘Admin’]

**Y – AXIS(Time)**

GetPrev\_Selection(int id)

DataTable

FillLocalTable(DataTable tblName,string query)

DataTable

[DataTable = NULL] Response.Redirect("~/Programing.aspx")

[DataTable NotEqual NULL] Response.Redirect("~/ResultSummary.aspx")

**Figure 3.6** Sequence diagram: User / admin log-in

* + - 1. **Add Programming Expertise**

**X – AXIS (Objects)**

:UserProfileSummary

btnNext\_Click(object sender, EventArgs e)

SaveUserInputRecord(string T

BusinessLayer.AssociateModul

InsertUserInputRecord(int user\_ int AA\_ID,int Level)

tring q

**Y – AXIS (Time)**

|  |  |
| --- | --- |
| :DataServices | |
| blName, eInfo m)  id, | Execute Query(S |

|  |  |  |
| --- | --- | --- |
| :DBAccess | | |
| uery) |  | |
|  |  |  |
|  |  | |

**Figure 3.7** Sequence diagram: Adding programming expertise

* + - 1. **Calculate Result**

**X – AXIS (Objects)**

:Result

Page\_Load (object sender, EventArgs e)

:CalculateResult

DataSet

DataSet

CalculateResults(DataTable dtTools, DataTable dtMath,DataTable dtResearch ,DataTable dtPrograming )

GetModuleInfo()

GetModuleReqInfo()

**Y – AXIS (Time)**

SortSmester1Data()

checkExpertise(DataRow[] drreq, DataTable dtConMat, DataRow drModule, DataTable dtTools, DataTable dtMath, DataTable dtResearch, DataTable dtPrograming)

SortSmester2Data()

**Figure 3.8** Sequence Diagram: Calculate result

* + 1. **Class Diagram (Part -1)**

Class diagram having data access layer class (marked in green) and business logic layer classes (marked in red) are shown in Figure 3.9. In terms of object oriented design, each class of the business layer has “simple association” relationship with *DbAccess* class, represented by straight line. It means that objects from the two classes have no intrinsic relationship (composition, aggregation or inheritance) with each other and the relationship only represents a reference by which one class can interact with another by declaring its object in one’s scope. In terms of cardinality, each of the business layer class has one-one relationship with *DbAccess* class i.e. one object of any of the business layer class can invoke exactly one object of *DbAccess* class. *DbAccess* class implementation is generic and has a universal mechanism for opening database connection, executing SQL Command and closing the connection. Actual query required to be executed and return values / rows / error code are handled by respective business layer class, thus following the principle of “separation of concerns”. Explanation for individual classes and their respective functions is discussed in Section 4.4.2 and 4.4.3.

**EditModuleReq**

-SQLStr : string

-rowCount : int

+GetModuleInfo()

+GetModuleRequirement()

+DeleteModuleReqRecord()

+InsertModuleReqRecord()

1

**LogInInfoCalc**

-newTable : DataTable

-SQLStr : string

-userID : int

-rowCount : int

+InsertRecord()

+GetUserInfoRecord()

+DuplicateUserInfoRec()

1

**DbAccess**

-dbConn : SqlConnection

-dbCommand : SqlCommand

-strConnString : string

1 1 -noOfAttempts : int

+CreateConn()

+FillLocalTable()

+ExecuteQuery()

**DataServices**

-newTable : DataTable

-rowCount : int

-SQLStr : string

1 1 +GetToolAcademicInfo()

+GetProgAcademicInfo()

+DeleteUserInputRec()

+InsertUserInputRec()

1

1

**PrResultSummary**

-dtProg : DataTable

-dtTool : DataTable

-dtMaths : DataTable

-dtResearch : DataTable

+GetProgSelection()

+GetToolSelection()

+GetMathematicSelection()

+GetResearchSelection()

1

1

**CalculateResult**

-totalCreditHour : int

-level : int

-truePositive : float

-falseNegative : float

+GetModuleInfo()

+GetModuleReqInfo()

+CalculateResults()

+CompareData()

+CheckExpertise()

**Figure 3.9** Class diagram: Business and data access layer

* + 1. **Class Diagram (Part - 2)**

Class diagram having presentation layer classes (marked in blue) are shown in Figure 3.10. Class details from data access layer (marked in green) and business layer (marked in red) have been reduced for the sake of brevity. These presentation layer classes have “simple association“ relationship with their respective business layer classes with one-one cardinality. Following the 3-tiered architecture discussed in Section 3.2.1, classes from presentation layer only deal with the front end functionality and invoke objects of business layer classes for logic implementation, which subsequently talk to data layer class for data services. E.g. *Log-In* class instantiates an object of *LogInfoCalc* class, and pass user credentials to it as a *String,* which further validates them against user database by invoking a static function of *DbAccess* class. Similarly, during second iteration, *AdminPanel* and *AdminEditReq* classes are implemented to allow amendments in module requirements. These classes invoke *EditModuleReq* class, which retrieves existing requirements and submit changes to database by invoking *DbAccess* class. Individual classes and their respective functions are discussed in Section 4.4.1.

**AdminPanel**

-info : editModuleReq

-newTable : DataTable

+Page\_Load()

+ChangeModuleReq\_Click() 1

**RegisterUser**

**AdminEditReq**

-info : editModuleReq

1 -expertiseName : string

-level : int

+CheckStateOnGrid()

+SetCheckState()

+SaveModuleReq\_Click()

1 1

-info : LogInInfoCal

-noOfRecord : int

-userInfoList : List <string>

+Page\_Load()

+RegisterUser()

1 1

**LogInInfoCalc**

-----------

**EditModuleReq**

------

----------

**DbAccess**

-------

------

1

**PrResultSummary**

---

------

1

1

1

---

**CalculateResult**

----

---------

1 1

1

**Result**

**LogIn**

-info : LogInInfoCal

+dtProg : DataTable

+dtTool : DataTable

+dtMath : DataTable

+dtResearch : DataTable

+Page\_Load()

+LogInButton\_Click()

**ResultSummary**

-userID : int

-sessionTable : DataTable

-result [ ] : DataRow

1 +ShowToolSummary()

+ShowProgrammingSummary()

+ShowMatematicSummary()

+ShowResearchSummary()

-creditHr : int

-saveObj : calculateResult

-dsSem1 : DataSet

-dsSem2 : DataSet

+Add\_falseNegative()

+RemoveSem1ModuleRec()

+RemoveSem2ModuleRec()

+ValidateSem1CreditHr()

+ValidateSem2CreditHr()

**Figure 3.10** Class diagram: Presentation layer (log-in, modify module requirements, result calculation)

* + 1. **Class Diagram (Part - 3)**

Class diagram having remaining presentation layer classes (marked in blue) are shown in Figure 3.10. Similar to their counterparts in Section 3.2.6, these classes also have “simple association“ relationship with their respective business layer classes with one-one cardinality. To retrieve already saved user profile information against existing user, *PrReusltSummary* class invokes *DbAccess* class and pass it “*User ID*” and “*Attribute Domain ID*” in the form of SQL query. The class returns the required data in form of *DataTable* object. Moreover, *Programming*, *MathematicsResearch* and *Tool* classes do not submit respective user profile data into database directly, and rather keep it in respective session variables for efficient data handling within a single session. The data is submitted, once the user finally reviews it through the interface, implemented by *UserProfileSummary* class. Individual classes and their respective functions are discussed in Section 4.4.1.

**Programming**

-dbTable : DataTable

-sessionTable : DataTable

-newTableColumn : DataColumn

+Page\_Load() 1

+OnPageIndexChanging()

+OnSelectedIndexChanged()

+OnRowDataBound()

**Tool**

-info : DataServices

-sessionTable : DataTable

1 -newTable : DataTable

+Page\_Load()

+Next\_Click()

**LogInInfoCalc**

**EditModuleReq**

**DbAccess**

1

**DataServices**

+OnPageIndexChanging()

+SetLoadStateData()

+LoadSelectedState() 1

**PrResultSummary**

**CalculateResult** 1 1 1

1 1 1

**MathematicsResearch**

-info : DataServices

-dbTableMaths : DataTable

-dbTableRes : DataTable

-sessionTableMaths : DataTable

-sessionTableRes : DataTable

+Page\_Load()

+LoadMathematicSelectedState()

+SetMathematicData()

+LoadResearchSelectedState()

+SetResearchData()

+OnPageIndexChanging()

**UserProfileSummary**

-userID : int

-info : DataServices

-sessionTable : DataTable

-result[] : DataRow

+Page\_Load()

+ShowToolSummary()

+ShowProgrammingSummary()

+ShowMatematicSummary()

+ShowResearchSummary()

+SaveUsedrInputRecord()

+Next\_Click()

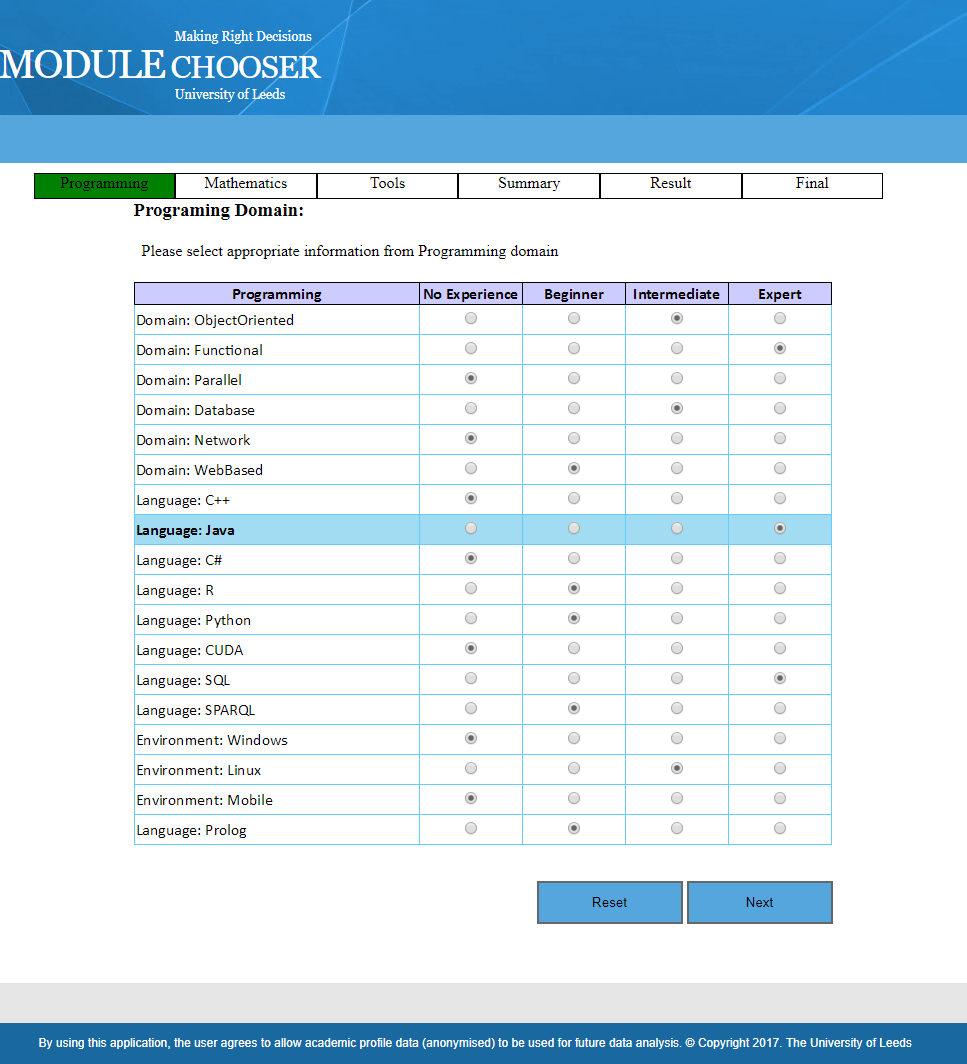
**Figure 3.11** Class diagram: Presentation layer (configuring user profile)

* + 1. **Graphical User Interface (GUI) Design**

GUI was developed using built-in design support provided within visual studio 2010 platform. Following paragraphs (Section 3.2.8.1, 3.2.8.2 and 3.2.8.3) show final GUI design for some of the application functionalities, as a result of second iteration. Corresponding GUI of the same functionalities.

* + - 1. **Adding Programming Expertise**

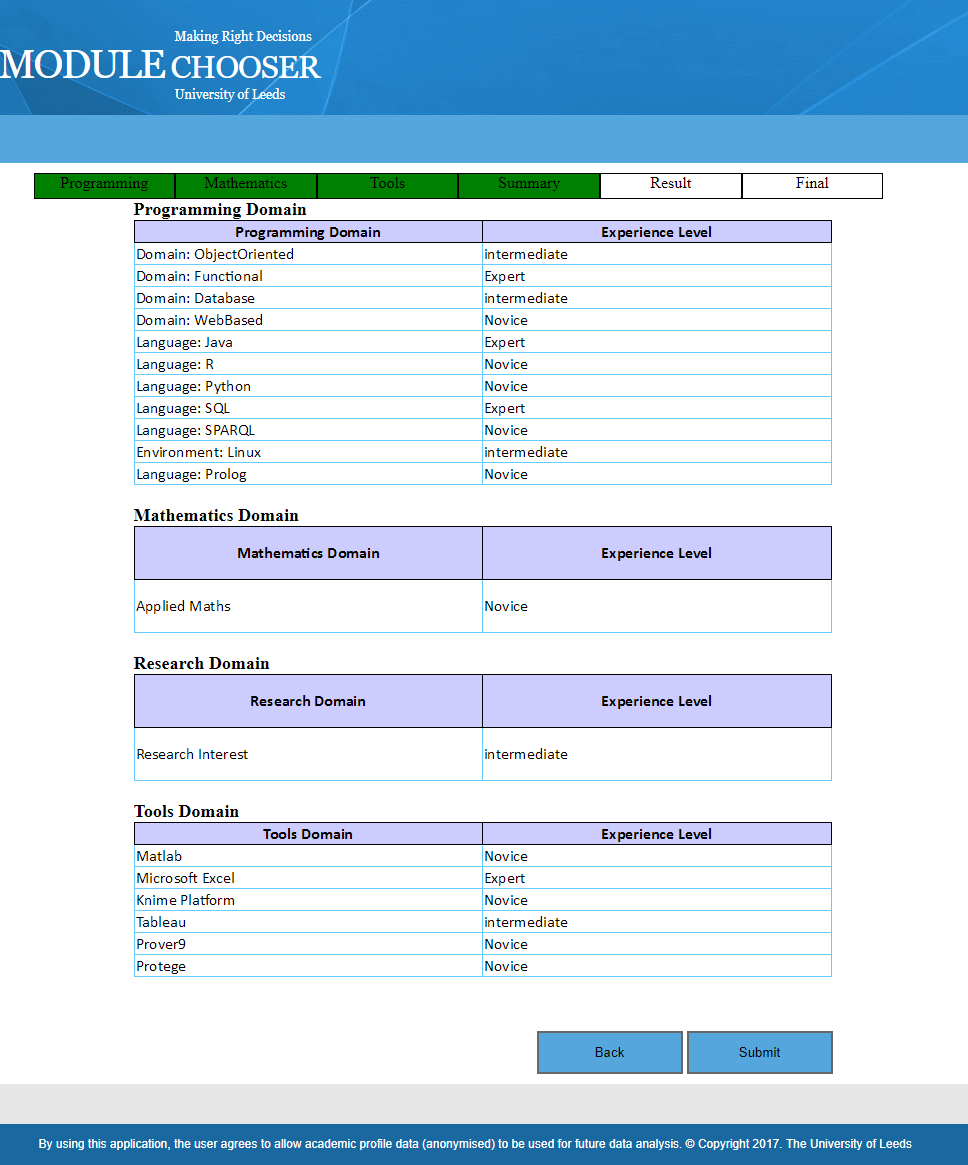
The graphical interface (Figure 3.12) is used to add user’s expertise in programming domain with help of attributes listed in first column. Relevant experience level in each attribute is configured by selecting respective radio button controls placed in next four columns. Default

**Figure 3.12** GUI: Adding programming expertise

setting for each attribute is ‘*No Experience’* option. *Reset* and *Next* buttons at the bottom are used to reset the whole page to default options and for moving forward to next configuration steps respectively. The page also contains a progress bar beneath application title, indicating overall progress and next steps in the configuration process. GUI design for configuring other expertise domains (Mathematics, Research, Tools) follows the same structure and theme.

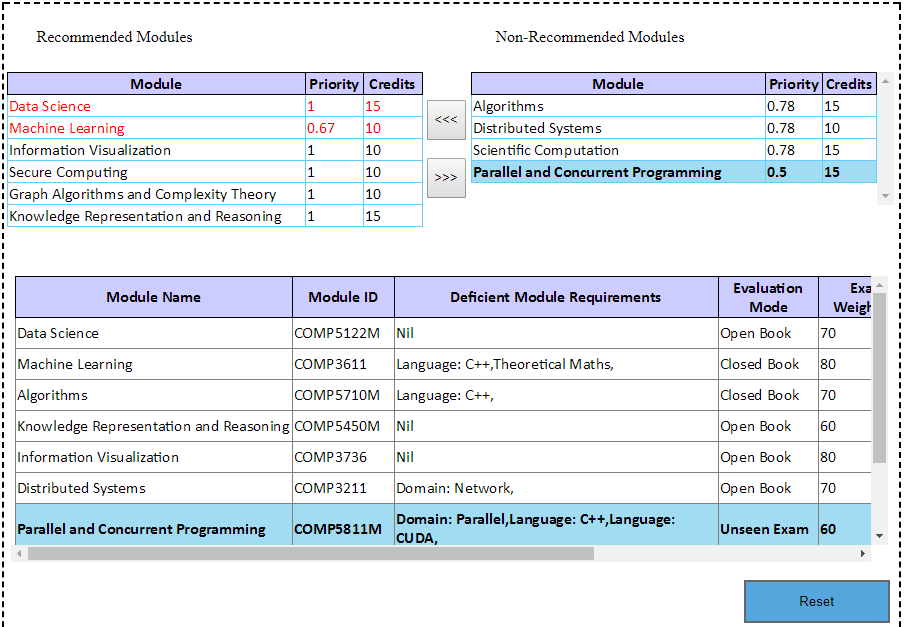
* + - 1. **User Profile Summary**

The graphical interface (Figure 3.13) displays the overall configuration summary in all the four domains, as done by logged-in user. It further gives an option of either submitting it to database through *Submit* button or going back to configuration settings through *Back* button.

**Figure 3.13** GUI: User profile summary

* + - 1. **Recommended Modules**

The graphical interface (Figure 3.14) shows set of three tables about semester 1, having system recommended, non-recommended modules and other relavant inforamtion that can help in decision making process. It includes module ranking, credit hours, reaosn for system recommendations in terms of user’s deficiencies against each module, evaluation mode, weightage of practical and theoretical part and link to online catalogue. The interface also gives an option of making manual changes through arrow buttons between two table, and resetting it back to original system recommendations through *Reset* button. In the same way, semester 2 would also have same information regarding its own set of modules.



**Figure 3.14** GUI: System recommended modules

**3.3 Conclusion**

The chapter deals with the requirement analysis and software design phase of the project. It starts with deliberating upon different URDs such as functional / non-functional requirements, use case diagrams, application workflow and relevant business domain models. In the light of these requirements, later part of the chapter explains different design documents such as application architecture, conceptual / logical database design, sequential / class diagrams and GUI design. The next chapter would highlight the actual implementation process, done in-line with the above referred documents.

**Chapter 4 System Implementation**

Implementation defines a systematically structured approach to translate user requirements and database / application design documents (discussed in chapter 3), into a workable software product that meets desired business objectives.

The chapter highlights the system implementation process in detail. Section 4.1 describes the development environment and technologies employed, whereas Section 4.2 and 4.3 explains underlying business logic being used and its implementation. Section 4.4 deliberates over application implementation details and discusses key classes and functions by categorizing them into the 3 - tiered architecture. The chapter concludes in Section 4.5 by briefly touching upon the major functionalities achieved in first and second iteration, during the course of development.

* 1. **Development Environment**

The application has been developed using Microsoft web application framework; *ASP.Net Web Forms*, within Windows 10 Pro environment. Language C# is used in the “*code-behind*” file, whereas ASP controls are used for “*aspx”* file. The “*aspx”* file is processed on ASP.Net Developer Server (a server used to run ASP.Net web pages in local host scenario), generating HTML for user’s web browser. Microsoft ADO.Net is used as a data access technology to communicate between front end controls and underlying relational database, residing in Microsoft SQL Server 2008 R2 (SQLEXPRESS). Visual Studio 2010 *Ultimate* is used as an integrated development environment for application development.

Appropriate development technologies for the implementation of this project has been deliberated upon in Section 2.4.

* 1. **Business Logic**

Business logic has been derived on the basis of *knowledge based approach*. Related discussion on this approach and the reasons to adopt it, have been explained in Section 2.2.3 and 2.2.5 respectively. The logic involves three fundamental components of *Module Chooser* application.

* + 1. **Academic Module Model**

*Academic Module Model* is used to represent module requirements for each MTech module in terms of domains / attributes finalized during requirement analysis phase and are discussed in Section 3.1.6.1 and 3.1.6.2.

For the sake of mathematical implementation, the table below indicates numerical scale of possible values used for each attribute.

**Table 4.1** Numerical scale for possible values of an academic attribute

|  |  |
| --- | --- |
| **Possible Value** | **Numerical Scale** |
| No Knowledge Required | 1 |
| Novice | 2 |
| Intermediate | 3 |
| Expert | 4 |

* + 1. **User Profile Model**

*User Profile Model* is used to capture individual students’ expertise level against each attribute, already referred in Section 4.2.1 for formulating *Academic Module Model*, with one of the four possible options i-e *No knowledge available, Novice, Intermediate, Expert.* Same numerical scale (Table 4.1) as that of *Academic Module Model* is used for *User Profile Model* as well.

* + 1. **Mathematical Algorithm**

The algorithm is primarily based upon two concepts; *confusion matrix* and *fuzzy logic*, that are used to compare and relate *Academic Module Model* for each module, against individual student’s *User Profile Model*. Use of confusion matrix is based upon the fact that the student credentials should match the academic module requirements for that module to be recommended to the specific student. As a result of this match-making and comparison, a *Recall* value for each academic module is calculated, which is subsequently used to prioritize and recommend modules by the system. Salient features of the underlying mathematical algorithm are briefly explained in the following paragraphs.

The confusion matrix adapted for this application can be represented as shown in Table 4.2:

**Table 4.2** Confusion Matrix adaptation: Current application

|  |  |  |
| --- | --- | --- |
|  | **Module Requirement - Yes** | **Module Requirement - No** |
| **Student Expertise Level - Yes** | True Positive (T.P) | False Positive (F.P) |
| **Student Expertise Level - No** | False Negative (F.N) | True Negative (T.N) |

While considering four possible values for each attribute and their respective numerical scale representation (Table 4.1), confusion matrix for this application can be calculated as shown in Table 4.3.

**Table 4.3** Confusion matrix calculation: Current application

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Module Requirements (Academic Module Model)** | | | | |
| **Student Expertise Level (User Profile Model)** |  | **1** | **2** | **3** | **4** |
| **1** | T.N | F.N | F.N | F.N |
| **2** | F.P | T.P | F.N | F.N |
| **3** | F.P | T.P | T.P | F.N |
| **4** | F.P | T.P | T.P | T.P |

The calculations derived in Table 4.3 are further elaborated as:

* + - * If “student expertise level” value is equal to or greater than that of “module requirements” for any attribute then the two values would be considered as same and case may be treated as True Positive (T.P), as marked with green colour in Table 4.3.
      * If “student expertise level” value is 1 and “module requirements” value is 2,3, or 4 for any attribute, then the two values would be considered as different and the case may be treated as False Negative (F.N), as marked with red colour in Table 4.3.
      * If “student expertise level” value is 1 and “module requirements” value is also 1 for any attribute, then the two values would be considered as same and the case may be treated as True Negative (T.N), as marked with orange colour in Table 4.3.
      * If “student expertise level” value is 2,3 or 4 and “module requirements” value is 1 for any attribute, then the two values would be considered as different and the case may be treated as False Positive (F.P), as marked with blue colour in Table 4.3.
      * The tabular area marked with grey colour in Table 4.3, indicates the cases that require fuzzy logic implementation. The logic is being used in those cases, where it is required to incorporate ‘degree of truth’ or ‘partial truth’, rather than representing ‘completely true’ or ‘completely false’ facts. For example, there may come a situation, when value of student expertise level is novice (numerical scale - 2) or intermediate (numeric scale - 3), whereas module requirement against that specific attribute is expert (numeric scale - 4). These cases, neither fall into criterion of being True Positives as discussed in above paras, nor they should be treated as False Negatives; considering the student at least has some of the required expertise level. Thus, fuzzy logic comes into play to implement these partial truths. The logic is implemented as follows:
        + If “student expertise level” is 2 and “module requirement” value is 3 for any attribute, then the case may be treated as “1/2 (True Positive)” i-e 1/2\*(1) = 0.5

may be added to the total number of True Positives instead of 1, while calculating True Positives for that module.

* + - * + If “student expertise level” is 2 and “module requirement” value is 4 for any attribute, then the case may be treated as “1/3 (True Positive)” i-e 1/3\*(1) = 0.3333 may be added to the total number of True Positives instead of 1, while calculating True Positives for that module.
        + If “student expertise level” is 3 and “module requirement” value is 4 for any attribute, then the case may be treated as “2/3 (True Positive)” i-e 2/3\*(1) = 0.6666 may be added to the total number of True Positives instead of 1, while calculating True Positives for that module.

Subsequently, the recall value is calculated as:

𝑅𝑒𝑐𝑎𝑙𝑙 𝑉𝑎𝑙𝑢𝑒 =

𝑇𝑟𝑢𝑒 𝑃𝑜𝑠𝑖𝑡𝑖𝑣𝑒𝑠

𝑇𝑟𝑢𝑒 𝑃𝑜𝑠𝑖𝑡𝑖𝑣𝑒𝑠 + 𝐹𝑎𝑙𝑠𝑒 𝑁𝑒𝑔𝑎𝑡𝑖𝑣𝑒𝑠

* 1. **Business Logic Implementation**

Business logic is implemented in a number of sequential steps, listed below:

* Calculate confusion matrix for “student profile” data set inserted by user against each academic module within programme, as explained in Section 4.2.3.
* Calculate *Recall* value for each academic module, with reference to given user profile.
* Split, all the academic modules on the basis of semester (semester 1 and 2), in which they are being offered.
* Rank all academic modules in each group on the basis of *Recall* value, in descending order.
* In case of more than one module having recall value equal to 1, ranking logic may prioritize the module with greater Number of True Positives, as higher in the list.
* In case of more than one module having recall value equal to 1 and same number of True Positives, ranking logic may consider them on equal priority.
* Select appropriate number of modules sequentially for each semester, in such a way that:
  + Each semester group must has compulsory modules as indicated by university module catalogue, irrespective of its recall value.
  + Total credit hours for each semester may be equal to or greater than 50, and lesser than or equal to 70, with 60 credit hours as preferred value.
  + Total credit hours for two semesters may preferably be equal to 120, but in no case can be greater than 125 credit hours.
  1. **Implementation Details**

Major classes and functions used to achieve business logic functionality discussed in Section 4.2 and 4.3 are grouped according to 3-tiered application architecture, and are explained in the following paragraphs.

* + 1. **Presentation Layer**

Presentation layer functionality is mainly achieved through following classes:

* + - 1. ***Programming* Class**

The class is implemented to collect and display user profile data in terms of programming expertise from logged-in user. The functions used in this class are described in Table 4.4.

**Table 4.4** Functions used in ‘*Programming’* class

|  |  |
| --- | --- |
| Page\_Load() | The function is called automatically at the time of page load. It starts by instantiating an object of type ‘*DataServices*’ (business layer class). The object is used to invoke business layer class function, which further calls another data access layer function to retrieve programming attributes from application database. The function is also used to invoke *LoadSelectedState()* function, if session variable is not null and page is being loaded for the first time i-e (*IsPostBack* = false). |
| LoadSelectedState() | The function is used to retrieve one’s programming expertise data from appropriate session variable and type cast it to an object of type *DataTable*. Subsequently, same type casted data is presented to the user, by using appropriate ‘*GridViewRow’* and ‘*RadioButton’* controls. |
| SetLoadStateData() | The function is used to retrieve user specified expertise categories like ‘No Experience’, Novice, Intermediate or Expert from each row of the ‘*GridViewRow*’ control, representing different programming attributes and saves it to an object of type ‘*DataTable*’ (in-memory representation of a single database table in ADO.NET library). The ‘*DataTable’* object is subsequently stored in appropriate session variable, against specific user for future use. |
| btnReset\_Click () | It is an event based function, used to reset option for all programming attributes to ‘No Experience’, irrespective of their previous selections. |

* + - 1. ***MathematicsResearch* Class**

The class is implemented to collect and display user profile data in terms of two domains i-e Mathematics aptitude and Research expertise, from logged-in user. The functions used in this class are described in Table 4.5.

**Table 4.5** Functions used in ‘*MathematicsResearch’* class.

|  |  |
| --- | --- |
| Page\_Load() | The function is called automatically at the time of page load. It starts by instantiating an object of type ‘*DataServices’* (business layer class). The object is used to invoke two functions form business layer class, which further calls another data access layer function to retrieve mathematics and research domain attributes from application database. The function is also used to invoke *LoadMathsSelState()* and *LoadResSelState()* functions, if respective session variables are not null and the page is being loaded for the first time (*IsPostBack* = false). |
| LoadMathsSelState() | The function is used to retrieve one’s mathematics aptitude data from appropriate session variable and type cast it to an object of type ‘*DataTable’*. Subsequently, same type casted data is presented to the user, by using appropriate ‘*GridViewRow’* and ‘*RadioButton’* controls. |
| LoadResSelState() | The function is used to retrieve one’s research expertise data from appropriate session variable and type cast it to an object of type ‘*DataTable’*. Subsequently, same type casted data is presented to the user, by using appropriate ‘*GridViewRow’* and ‘*RadioButton’* controls. |
| SetMathsData() | The function is used to retrieve user specified expertise categories like ‘No Experience’, Novice, Intermediate or Expert from each row of the ‘GridViewRow’ control, representing different mathematics attributes and saves it to an object of type ‘*DataTable*’. The ‘DataTable’ object is subsequently stored in appropriate session variable, against specific user for future use. |
| SetResExpData() | The function is used to retrieve user specified expertise categories like ‘No Experience’, Novice, Intermediate or Expert from each row of the ‘GridViewRow’ control, representing different research attributes and saves it to an object of type ‘*DataTable*’. The is subsequently stored in appropriate session variable, against specific user for future use. ‘DataTable’ object |
| btnNext\_Click () | It is an event based function, used to invoke SetMathsData() and SetResExpData() functions, and navigates the user to next page in the application. |
| btnPre\_Click () | It is an event based function, used to invoke SetMathsData() and SetResExpData() functions, and navigates the user to previous page in the application. |

* + - 1. ***Tool* Class**

The class is implemented to collect and display user profile data in different tools’ domain with associated degree of expertise, from logged-in user. Because of obvious functional similarities between *‘Tool’* class and *‘Programming’* and *‘MathematicsResearch’* classes discussed in Section 4.4.1.1 and 4.4.1.2 respectively, functions’ implementation in *‘Tool’* class is not much different; and is not being explained further to avoid unnecessary repetition.

* + - 1. ***UserProfileSummary* Class**

The class is implemented to display summary of complete user profile in domains of programming, mathematics, research experience and tools with their associated expertise levels, as configured by logged-in user. Moreover, the class also adds additional functionality of saving user profile data in application database. The functions used in this class are described in Table 4.6.

**Table 4.6** Functions used in ‘*UserProfileSummary’* class.

|  |  |  |
| --- | --- | --- |
| Page\_Load() | The function is called automatically at the time of page load. It simply invokes 04 functions namely; *ShowToolSummary()*, *ShowProgSummary()*,*ShowMathsSummary()* and *ShowResSummary()*, provided the page is being loaded for the first time. (*IsPostBack* = false). | |
| ShowProgSummary() ShowToolSummary() ShowMathsSummary() ShowResSummary() | These 04 functions are used to display the summary of user profile data in programming, tool, mathematics and research domains respectively with relevant expertise level. The functions retrieve relevant data from appropriate session variables and typecast it to the objects of type *‘DataTable’*. After excluding user profile attributes / rows that have ‘No Experience’ option selected; as they are of no use in the given functions, the *‘DataTable’* objects are bound to appropriate *‘GridViewRow’* controls for display. | |
| SaveUserInputRec() | | The function takes two parameters as an input namely; table name of respective expertise domain (programming, tools etc.) and an object of business class. It ensures that the respective session variable is not null and then retrieves associated table name and user ID data to typecast it to *‘DataTable’* object and *‘int’* variable respectively. Subsequently, this function invokes business layer class function that further calls data access layer class to save configured user profile data in the application database. |
| btnNext\_Click() | | It is an event based function, that first instantiates an object of business layer class and then invokes *SaveUserInputRec()* method 04 times, one each for the relevant expertise domains. Moreover, before doing so, it deletes any already configured expertise data, saved in the application database against the logged-in user. Lastly it navigates the user to next page in the application. |

* + - 1. ***Result* Class**

The class is implemented to display recommended and non-recommended academic modules for each semester, associated academic details for each module and provision to make manual changes in module recommendations. The class also provides functionality to save finalized academic modules in application database.

**Table 4.7** Functions used in ‘*Result’* class.

|  |  |
| --- | --- |
| Page\_Load() | The function is called automatically at the time of page load. It retrieves user expertise information (programming, tools etc.) from related session variables and typecast it to relevant objects of type *‘DataTable’*. These *‘DataTable’* objects are passed on to business layer class function as an argument, which implements the business logic, responsible to compare configured user profile against module requirements for each academic module and prioritizes them. The function is also used to invoke functions, required to display associated academic info for each module and sort out modules on the basis of recommended, non- recommended, semester 1 and semester 2 groups and then binds them to appropriate *‘GridViewRow’* controls for presentation. |
| RemSem1ModRec() RemSem2ModRec() | The functions are used to allow manual shifting of selected module from recommended to non-recommended set in case of semester 1 and semester 2 respectively, if desired by user. The functions also ensure that that the module being shifted is not compulsory in nature, and does not allow subject change, if it is the case. |
| RemSem1ModNonRec() RemSem2ModNonRec() | The functions implement manual shifting of selected module from non-recommended to recommended set in case of semester 1 and semester 2 respectively, if desired by user. The function also ensures that that the manual change by user does not violate maximum credit hour limit for single semester (50 ≤ creditHr ≤ 70) or for the whole programme (120 ≤ creditHr ≤ 125), and does not allow the change, if it is the case. |
| btnSubmit\_Click() | The function retrieves user ID information from appropriate session variable and use it to save finalized module selections to the database against logged-in user, by invoking a function of business layer class. The function also checks the credit hour requirements for the whole program, before saving data. To maintain data integrity at application level, the function also deletes any module information already stored in the database against same user. Before exiting, the function navigates the user to next page. |
| btnResetSemester1\_Click() btnResetSemester2\_Click() | The two event based functions are used to reset module recommendations for semester 1 and semester 2 respectively, according to the priorities calculated by the application logic. |

* + - 1. ***LogIn* Class**

The class is used to implement log-in functionality for the existing as well as new users. The functions used in this class are described in Table 4.8.

**Table 4.8** Functions used in ‘*LogIn’* class.

|  |  |
| --- | --- |
| Page\_Load() | The function is called automatically at the time of page load. It simply gives hyperlink to ‘register page’ for new users’ registration. It also clears all items that have been added to the session object's contents collection by invoking *RemoveAll()* method. |
| LoginButton\_Click() | This even based function invokes a method from business layer class, which is used to authenticate user credentials against user data already saved in the database. In case of successful authentication,  the function adds these user details to current session for subsequent use, throughout the application lifecycle. Furthermore, the function also retrieves already configured user profile data from application database, and displays it for further processing.  However if the user is from admin category, then the function redirects the user to appropriate application interface for amending academic module requirements for each module. |

* + 1. **Business Layer**

Business layer functionality is mainly achieved through following classes:

* + - 1. ***DataServices* Class**

The class implements part of business logic, which is responsible for data retrieval and submission to the application database. The functions used in this class are described in Table 4.9.

**Table 4.9** Functions used in ‘*DataServices’* class.

|  |  |
| --- | --- |
| GetProgAcademicInfo() GetMathAcademicInfo() GetToolAcademicInfo() GetResAcademicInfo() | These 04 methods retrieve user profile attributes from programming, mathematics, tools and research domains respectively, by running SQL Query on database table *‘AcademicInfo’* and using appropriate values of column ‘CA\_ID’ as where clause condition. The SQL query is executed by invoking static function of data access layer, which takes two arguments; first one is *‘DataTable’* object passed by reference and second one is SQL query. |
| DeleteUserInputRec() | The function takes user ID as an argument and is invoked by presentation layer function, before submitting newly configured user profile data to the application database. It deletes previous user profile data of an existing user by running SQL query on database table ‘*UserProfile’* while using user ID as where clause condition. The SQL query is actually executed by invoking static class function of data access layer. |
| InsertUserInputRec() | The function takes user ID, AcademicInfo ID and Expertise level as an argument and is invoked by presentation layer function. It submits newly configured user profile data of an existing user by running SQL query on database table ‘*UserProfile’*. The SQL query is actually  executed by invoking static class function of data access layer, which takes this query as an string type argument. |

* + - 1. ***CalculateResult* Class**

Core business logic of the application as explained in section 4.2 and 4.3 has been implemented in this class. Salient functions comprising the class has been described in Table 4.10.

**Table 4.10** Functions used in ‘*CalculateResult’* class.

|  |  |
| --- | --- |
| CheckExpertise() | The function takes user profile info as an argument in the form of *‘DataTable’* objects representing programming, mathematics, tools and research domain and compares it against module requirements of each module. This comparison process calculates True Positives, False Negatives and Recall value for each academic module, while using confusion matrix and fuzzy logic; as discussed in detail in Section 4.2. The information is stored in ‘class scope’ object of type ‘*DataTable’* for subsequent use. This function is invoked by *CalculateResults()* method of same class, which subsequently is invoked by *Page\_Load()* function of presentation layer class. |
| SemModSelection() | The function takes semester No, and min and max credit hours for each semester / program as an argument. The function selects most suitable modules for each semester on the basis of their *Recall* value, while also taking into account compulsory modules and credit hours limitations. |
| GetModReqInfo() | The function retrieves academic module requirements for each module by using SQL query against database table ‘*ModuleRequirement’*. The query is being executed through static function of data access layer class, which takes two parameters; ‘*DataTable’* object passed as reference and SQL query. |
| GetModuleInfo() | The function is used to retrieve details for each academic module from database table ‘*AcademicModule’* by running database query. The function is invoked by *CalculateResults()* method of same class. |

* + - 1. ***LogInInfoCalc* Class**

The class implements part of business logic, that is responsible for user data management and log-in functionality. Major functions used in this class are described in Table 4.11.

**Table 4.11** Functions used in ‘*LogInInfoCalc’* class.

|  |  |
| --- | --- |
| InsertRecord() | The function is used to insert user data into database table ‘*User*’ by executing SQL query through a function of data access layer class. *InsertRecord()* function is invoked by ‘*RegisterUser’* class of presentation layer. |
| GetUserInfoRecord() | The function is used to validate user credentials by retrieving user data from database table ‘*User*’. The function executes SQL query through a function of data access layer class and returns user ID as a return value. This function is invoked from number of presentation layer classes namely; *ChangePassword, RegisterUser and LogIn* class. |
| DuplicateUserInfoRec() | The function is used to avoid duplication of user name during registration process and is invoked by ‘*RegisterUser’* class of presentation layer. |

* + - 1. ***EditModuleReq* Class**

The class implements part of business logic, responsible for amendments in academic module requirements, normally handled by admin user. Major functions used in this class are described in Table 4.12.

**Table 4.12** Functions used in ‘*EditModuleReq’* class.

|  |  |
| --- | --- |
| GetModuleReq() | The function takes Module ID as an argument and retrieves academic module requirements against that module by executing SQL query through static function of data access layer class. |
| DeleteModuleReq() | The function takes Module ID as an argument and deletes academic module requirements against that module by executing SQL query through static function of data access layer class. |
| InsertModuleReq() | The function takes Module ID, AcademicInfo ID and Expertise Level as an argument and inserts new module requirement row in database table ‘ModuleRequirement’, by executing SQL query through static function of data access layer class. |

* + 1. **Data Access Layer**

Data access layer functionality is mainly achieved through following class:

* + - 1. ***DbAccess* Class**

The class implements functionality, that manages connection with underlying database and is responsible to communicate (data retrieval and submission) with it. The class acts as a gateway between business layer and application database. Major functions used in this class are described in Table 4.13.

**Table 4.13** Functions used in ‘*DbAccess’* class.

|  |  |
| --- | --- |
| CreateConn() | The function uses an instance of *SqlConnection* class ‘*DbConn*’ with class level scope, and sets its ‘*ConnectionString’* property, to required user credentials and other keyword values like *Timeout*, *Data Source*, *Initial Catalog*, and *Security Info* etc. Subsequently, *DbConn.Open* method is used to open connection with application database by using ‘*ConnectionString’* property settings. |
| FillLocalTable() | The function initiates two objects; an object of ‘*SqlDataAdapter’* class to provide communication between *‘DataSet’* objects and SQL database and an object of *‘SqlCommand’* class to represent SQL query. The function invokes *CreateConn()* method to open connection to the application database, if it is not already opened and then update *‘DataTable’* object (passed as reference to this function) according to given SQL query. At the end, the function releases all resources by invoking *Dispose()* method. |
| ExecuteQuery() | The function instantiates an object of *‘SqlCommand’* class, sets is relevant properties like *Connection*, *CommandText* and *CommandType,* and then invokes *ExecuteNonQuery()* method to insert, update or delete data rows in application database according to given SQL query. |

* 1. **Source Code**

**Programming Class**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Data;**

**using System.Drawing;**

**namespace Data\_analytic**

**{**

**public partial class Programming : System.Web.UI.Page**

**{**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**BusinessLayer.DataServices m = new BusinessLayer.DataServices();**

**DataTable dtb = m.GetProgAcademicInfo();**

**grdPrograming.DataSource = dtb;**

**grdPrograming.DataBind();**

**if (Session["Programing"] != null && !IsPostBack)**

**{**

**LoadSelectedState();**

**}**

**}**

**protected void OnRowDataBound(object sender, System.Web.UI.WebControls.GridViewRowEventArgs e)**

**{**

**if (e.Row.RowType == DataControlRowType.DataRow)**

**{**

**e.Row.Attributes["onclick"] = Page.ClientScript.GetPostBackClientHyperlink(grdPrograming, "Select$" + e.Row.RowIndex);**

**e.Row.ToolTip = "Click to select this row.";**

**}**

**}**

**protected void OnSelectedIndexChanged(object sender, EventArgs e)**

**{**

**foreach (GridViewRow row in grdPrograming.Rows)**

**{**

**if (row.RowIndex == grdPrograming.SelectedIndex)**

**{**

**row.ToolTip = string.Empty;**

**}**

**else**

**{**

**row.BackColor = ColorTranslator.FromHtml("#FFFFFF");**

**row.ToolTip = "Click to select this row.";**

**}**

**}**

**}**

**//To load previous selections of the student, during same session**

**void LoadSelectedState()**

**{**

**DataTable dt = (DataTable)Session["programing"];**

**int i = 0;**

**foreach (DataRow dr in dt.Rows)**

**{**

**GridViewRow gdrow = grdPrograming.Rows[i];**

**RadioButton rbt = (RadioButton)gdrow.FindControl("RowSelector1");**

**RadioButton rbt1 = (RadioButton)gdrow.FindControl("RowSelector");**

**RadioButton rbt3 = (RadioButton)gdrow.FindControl("RowSelector2");**

**RadioButton rbt4 = (RadioButton)gdrow.FindControl("RowSelector3");**

**if (dr["Available"].ToString() == "1")**

**{**

**rbt.Checked = true;**

**}**

**else if (dr["NotAvailable"].ToString() == "1")**

**{**

**rbt1.Checked = true;**

**}**

**else if (dr["intermediate"].ToString() == "1")**

**{**

**rbt3.Checked = true;**

**}**

**else**

**{**

**rbt4.Checked = true;**

**}**

**i++;**

**}**

**}**

**// To maintain the user selections within session variable for subsequent use**

**void SetLoadStateData()**

**{**

**DataTable dt = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "data";**

**dt.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "AcademicInfo\_ID";**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Available", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("NotAvailable", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("intermediate", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Expert", typeof(int));**

**dt.Columns.Add(dc);**

**foreach (GridViewRow row in grdPrograming.Rows)**

**{**

**DataRow dr = dt.NewRow();**

**dr["data"] = row.Cells[0].Text;**

**dr["AcademicInfo\_ID"] = row.Cells[1].Text;**

**RadioButton r = (RadioButton)row.FindControl("RowSelector1");**

**if (r.Checked)**

**dr["Available"] = 1;**

**else**

**dr["Available"] = 0;**

**RadioButton r1 = (RadioButton)row.FindControl("RowSelector");**

**if (r1.Checked)**

**dr["NotAvailable"] = 1;**

**else**

**dr["NotAvailable"] = 0;**

**RadioButton r2 = (RadioButton)row.FindControl("RowSelector2");**

**if (r2.Checked)**

**dr["intermediate"] = 1;**

**else**

**dr["intermediate"] = 0;**

**RadioButton r3 = (RadioButton)row.FindControl("RowSelector3");**

**if (r3.Checked)**

**dr["Expert"] = 1;**

**else**

**dr["Expert"] = 0;**

**dt.Rows.Add(dr);**

**}**

**Session["Programing"] = dt;**

**}**

**// Go to next page**

**protected void btnNext\_Click(object sender, EventArgs e)**

**{**

**SetLoadStateData();**

**Response.Redirect("~/MathematicsResearch.aspx");**

**}**

**// To reset all options to default state - "No Experience"**

**protected void btnReset\_Click(object sender, EventArgs e)**

**{**

**BusinessLayer.DataServices m = new BusinessLayer.DataServices();**

**DataTable dtb = m.GetProgAcademicInfo();**

**grdPrograming.DataSource = dtb;**

**grdPrograming.DataBind();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Data;**

**using System.Drawing;**

**namespace Data\_analytic**

**{**

**public partial class MathematicsResearch : System.Web.UI.Page**

**{**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**BusinessLayer.DataServices m = new BusinessLayer.DataServices();**

**DataTable dtMath = m.GetMathAcademicInfo();**

**grdMath.DataSource = dtMath;**

**grdMath.DataBind();**

**DataTable dtResearch = m.GetResAcademicInfo();**

**grdResearch.DataSource = dtResearch;**

**grdResearch.DataBind();**

**if (!IsPostBack)**

**{**

**if(Session["Mathmetic"] != null)**

**LoadMathsSelState();**

**if (Session["ResearchExp"] != null)**

**LoadResSelState();**

**}**

**}**

**protected void OnRowDataBound(object sender, System.Web.UI.WebControls.GridViewRowEventArgs e)**

**{**

**if (e.Row.RowType == DataControlRowType.DataRow)**

**{**

**e.Row.Attributes["onclick"] = Page.ClientScript.GetPostBackClientHyperlink(grdMath, "Select$" + e.Row.RowIndex);**

**e.Row.ToolTip = "Click to select this row.";**

**}**

**}**

**protected void OnSelectedIndexChanged(object sender, EventArgs e)**

**{**

**foreach (GridViewRow row in grdMath.Rows)**

**{**

**if (row.RowIndex == grdMath.SelectedIndex)**

**{**

**row.ToolTip = string.Empty;**

**}**

**else**

**{**

**row.BackColor = ColorTranslator.FromHtml("#FFFFFF");**

**row.ToolTip = "Click to select this row.";**

**}**

**}**

**}**

**protected void OnRowDataBound1(object sender, System.Web.UI.WebControls.GridViewRowEventArgs e)**

**{**

**if (e.Row.RowType == DataControlRowType.DataRow)**

**{**

**e.Row.Attributes["onclick"] = Page.ClientScript.GetPostBackClientHyperlink(grdResearch, "Select$" + e.Row.RowIndex);**

**e.Row.ToolTip = "Click to select this row.";**

**}**

**}**

**protected void OnSelectedIndexChanged1(object sender, EventArgs e)**

**{**

**foreach (GridViewRow row in grdResearch.Rows)**

**{**

**if (row.RowIndex == grdResearch.SelectedIndex)**

**{**

**row.ToolTip = string.Empty;**

**}**

**else**

**{**

**row.BackColor = ColorTranslator.FromHtml("#FFFFFF");**

**row.ToolTip = "Click to select this row.";**

**}**

**}**

**}**

**//To load previous selections of the student, during same session (Maths domain)**

**void LoadMathsSelState()**

**{**

**DataTable dt = (DataTable)Session["Mathmetic"];**

**int i = 0;**

**foreach (DataRow dr in dt.Rows)**

**{**

**GridViewRow gdrow = grdMath.Rows[i];**

**RadioButton rbt = (RadioButton)gdrow.FindControl("RowSelector1");**

**RadioButton rbt1 = (RadioButton)gdrow.FindControl("RowSelector");**

**RadioButton rbt3 = (RadioButton)gdrow.FindControl("RowSelector2");**

**RadioButton rbt4 = (RadioButton)gdrow.FindControl("RowSelector3");**

**if (dr["Available"].ToString() == "1")**

**{**

**rbt.Checked = true;**

**}**

**else if (dr["NotAvailable"].ToString() == "1")**

**{**

**rbt1.Checked = true;**

**}**

**else if (dr["intermediate"].ToString() == "1")**

**{**

**rbt3.Checked = true;**

**}**

**else**

**{**

**rbt4.Checked = true;**

**}**

**i++;**

**}**

**}**

**// To maintain the user selections within session variable for subsequent use (Maths domain)**

**void SetMathsData()**

**{**

**DataTable dt = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "data";**

**dt.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "AcademicInfo\_ID";**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Available", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("NotAvailable", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("intermediate", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Expert", typeof(int));**

**dt.Columns.Add(dc);**

**foreach (GridViewRow row in grdMath.Rows)**

**{**

**DataRow dr = dt.NewRow();**

**dr["data"] = row.Cells[0].Text;**

**dr["AcademicInfo\_ID"] = row.Cells[1].Text;**

**RadioButton r = (RadioButton)row.FindControl("RowSelector1");**

**if (r.Checked)**

**dr["Available"] = 1;**

**else**

**dr["Available"] = 0;**

**RadioButton r1 = (RadioButton)row.FindControl("RowSelector");**

**if (r1.Checked)**

**dr["NotAvailable"] = 1;**

**else**

**dr["NotAvailable"] = 0;**

**RadioButton r2 = (RadioButton)row.FindControl("RowSelector2");**

**if (r2.Checked)**

**dr["intermediate"] = 1;**

**else**

**dr["intermediate"] = 0;**

**RadioButton r3 = (RadioButton)row.FindControl("RowSelector3");**

**if (r3.Checked)**

**dr["Expert"] = 1;**

**else**

**dr["Expert"] = 0;**

**dt.Rows.Add(dr);**

**}**

**Session["Mathmetic"] = dt;**

**}**

**//To load previous selections of the student, during same session (Research domain)**

**void LoadResSelState()**

**{**

**DataTable dt = (DataTable)Session["ResearchExp"];**

**int i = 0;**

**foreach (DataRow dr in dt.Rows)**

**{**

**GridViewRow gdrow = grdResearch.Rows[i];**

**RadioButton rbt = (RadioButton)gdrow.FindControl("RowSelector5");**

**RadioButton rbt1 = (RadioButton)gdrow.FindControl("RowSelector4");**

**RadioButton rbt3 = (RadioButton)gdrow.FindControl("RowSelector6");**

**RadioButton rbt4 = (RadioButton)gdrow.FindControl("RowSelector7");**

**if (dr["Available"].ToString() == "1")**

**{**

**rbt.Checked = true;**

**}**

**else if (dr["NotAvailable"].ToString() == "1")**

**{**

**rbt1.Checked = true;**

**}**

**else if (dr["intermediate"].ToString() == "1")**

**{**

**rbt3.Checked = true;**

**}**

**else**

**{**

**rbt4.Checked = true;**

**}**

**i++;**

**}**

**}**

**// To maintain the user selections within session variable for subsequent use (Research domain)**

**void SetResExpData()**

**{**

**DataTable dt = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "data";**

**dt.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "AcademicInfo\_ID";**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Available", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("NotAvailable", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("intermediate", typeof(int));**

**dt.Columns.Add(dc);**

**dc = new DataColumn("Expert", typeof(int));**

**dt.Columns.Add(dc);**

**foreach (GridViewRow row in grdResearch.Rows)**

**{**

**DataRow dr = dt.NewRow();**

**dr["data"] = row.Cells[0].Text;**

**dr["AcademicInfo\_ID"] = row.Cells[1].Text;**

**RadioButton r = (RadioButton)row.FindControl("RowSelector5");**

**if (r.Checked)**

**dr["Available"] = 1;**

**else**

**dr["Available"] = 0;**

**RadioButton r1 = (RadioButton)row.FindControl("RowSelector4");**

**if (r1.Checked)**

**dr["NotAvailable"] = 1;**

**else**

**dr["NotAvailable"] = 0;**

**RadioButton r2 = (RadioButton)row.FindControl("RowSelector6");**

**if (r2.Checked)**

**dr["intermediate"] = 1;**

**else**

**dr["intermediate"] = 0;**

**RadioButton r3 = (RadioButton)row.FindControl("RowSelector7");**

**if (r3.Checked)**

**dr["Expert"] = 1;**

**else**

**dr["Expert"] = 0;**

**dt.Rows.Add(dr);**

**}**

**Session["ResearchExp"] = dt;**

**}**

**// Go to next page**

**protected void btnNext\_Click(object sender, EventArgs e)**

**{**

**SetMathsData();**

**SetResExpData();**

**Response.Redirect("~/Tool.aspx");**

**}**

**// Go to previous page**

**protected void btnPre\_Click(object sender, EventArgs e)**

**{**

**SetMathsData();**

**SetResExpData();**

**Response.Redirect("~/Programming.aspx");**

**}**

**// Reset to default options**

**protected void btnReset\_Click(object sender, EventArgs e)**

**{**

**BusinessLayer.DataServices m = new BusinessLayer.DataServices();**

**DataTable dtMath = m.GetMathAcademicInfo();**

**grdMath.DataSource = dtMath;**

**grdMath.DataBind();**

**DataTable dtResearch = m.GetResAcademicInfo();**

**grdResearch.DataSource = dtResearch;**

**grdResearch.DataBind();**

**}**

**}**

**}**

**UserProfileSummary Class**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Data;**

**namespace Data\_analytic**

**{**

**public partial class UserProfileSummary : System.Web.UI.Page**

**{**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**if (!IsPostBack)**

**{**

**ShowToolSummary();**

**ShowProgSummary();**

**ShowMathsSummary();**

**ShowResSummary();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to programing domain**

**/// </summary>**

**void ShowProgSummary()**

**{**

**if (Session["Programing"] != null)**

**{**

**DataTable dtPrograming = (DataTable)Session["Programing"];**

**DataRow[] result = dtPrograming.Select("NotAvailable=0");**

**DataTable dtSelectPrograming = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectPrograming.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectPrograming.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectPrograming.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectPrograming.Rows.Add(drn);**

**}**

**grdPrograming.DataSource = dtSelectPrograming;**

**grdPrograming.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Maths domain**

**/// </summary>**

**void ShowMathsSummary()**

**{**

**if (Session["Mathmetic"] != null)**

**{**

**DataTable dtMath = (DataTable)Session["Mathmetic"];**

**DataRow[] result = dtMath.Select("NotAvailable=0");**

**DataTable dtSelectionMath = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionMath.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionMath.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectionMath.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**//Expert**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionMath.Rows.Add(drn);**

**}**

**grdMath.DataSource = dtSelectionMath;**

**grdMath.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Research domain**

**/// </summary>**

**void ShowResSummary()**

**{**

**if (Session["ResearchExp"] != null)**

**{**

**DataTable dtResearch = (DataTable)Session["ResearchExp"];**

**DataRow[] result = dtResearch.Select("NotAvailable=0");**

**DataTable dtSelectionResearch = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionResearch.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionResearch.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectionResearch.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**//Expert**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionResearch.Rows.Add(drn);**

**}**

**grdResearch.DataSource = dtSelectionResearch;**

**grdResearch.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Tools domain**

**/// </summary>**

**void ShowToolSummary()**

**{**

**if (Session["Tool"] != null)**

**{**

**DataTable dtTool = (DataTable)Session["Tool"];**

**DataTable dtSelectionTools = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionTools.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionTools.Columns.Add(dc);**

**foreach (DataRow dr in dtTool.Rows)**

**{**

**if (dr["NotAvailable"].ToString() == "0")**

**{**

**DataRow drn = dtSelectionTools.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionTools.Rows.Add(drn);**

**}**

**}**

**grdTools.DataSource = dtSelectionTools;**

**grdTools.DataBind();**

**}**

**}**

**/// <summary>**

**/// To save user profile to application database**

**/// </summary>**

**void SaveUserInputRec(string TblName, BusinessLayer.DataServices m)**

**{**

**if (Session[TblName] != null)**

**{**

**DataTable dt = (DataTable)Session[TblName];**

**int User\_ID = (int)Session["UserID"];**

**foreach (DataRow dr in dt.Rows)**

**{**

**if (dr["NotAvailable"].ToString() == "0")**

**{**

**int AcademicInfo\_ID= int.Parse(dr["AcademicInfo\_ID"].ToString());**

**int Level = 1;**

**if (dr["intermediate"].ToString() == "1")**

**{**

**Level = 3;**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**Level = 4;**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**Level = 2;**

**}**

**m.InsertUserInputRec(User\_ID, AcademicInfo\_ID, Level);**

**}**

**}**

**}**

**}**

**// To delete any previous data in database related to student and call SaveUserInputRec() for each expertise domain**

**protected void btnNext\_Click(object sender, EventArgs e)**

**{**

**BusinessLayer.DataServices m = new BusinessLayer.DataServices();**

**int User\_ID = (int)Session["UserID"];**

**m.DeleteUserInputRec(User\_ID);**

**for (int i = 0; i < 4; i++)**

**{**

**if (i == 0)**

**{**

**SaveUserInputRec("Programing", m);**

**}**

**else if (i == 1)**

**{**

**SaveUserInputRec("Tool", m);**

**}**

**else if (i == 2)**

**{**

**SaveUserInputRec("Mathmetic", m);**

**}**

**else if (i == 3)**

**{**

**SaveUserInputRec("ResearchExp", m);**

**}**

**}**

**Session["CallFromResult"] = "NotModify";**

**Response.Redirect("~/Result.aspx");**

**}**

**//To modify Tool selections - Go to previous page**

**protected void btnPre\_Click(object sender, EventArgs e)**

**{**

**Response.Redirect("~/Tool.aspx");**

**}**

**}**

**}**

**ResultSummary**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Data;**

**namespace Data\_analytic**

**{**

**public partial class ResultSummary : System.Web.UI.Page**

**{**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**if (!IsPostBack)**

**{**

**showToolSummary();**

**showProgramingSummary();**

**showMathmeticSummary();**

**showResearchSummary();**

**BusinessLayer.CalcualteResult obj = new BusinessLayer.CalcualteResult();**

**int User\_ID =(int)Session["UserID"];**

**DataTable dtModuleSemester2= obj.GetPreRecordSM2(User\_ID);**

**grdSugSemester2.DataSource = dtModuleSemester2;**

**grdSugSemester2.DataBind();**

**DataTable dtModuleSemester1 = obj.GetPreRecordSM1(User\_ID);**

**grdSugSemester1.DataSource = dtModuleSemester1;**

**grdSugSemester1.DataBind();**

**}**

**}**

**// To implement web link column for module details grid**

**protected string SetUrl(object o)**

**{**

**if (o.ToString().Trim() == "")**

**{**

**return "";**

**}**

**else**

**{**

**return "WebLink";**

**}**

**}**

**// To implement web link column for module details grid**

**protected string GetUrl(object id)**

**{**

**return "http://" + id;**

**}**

**/// <summary>**

**//Show user profile selections associated to Programming domain**

**/// </summary>**

**void showProgramingSummary()**

**{**

**if (Session["Programing"] != null)**

**{**

**DataTable dtPrograming = (DataTable)Session["Programing"];**

**DataRow[] result = dtPrograming.Select("NotAvailable=0");**

**DataTable dtSelectPrograming = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectPrograming.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectPrograming.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectPrograming.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectPrograming.Rows.Add(drn);**

**}**

**grdPrograming.DataSource = dtSelectPrograming;**

**grdPrograming.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Maths domain**

**/// </summary>**

**void showMathmeticSummary()**

**{**

**if (Session["Mathmetic"] != null)**

**{**

**DataTable dtMath = (DataTable)Session["Mathmetic"];**

**DataRow[] result = dtMath.Select("NotAvailable=0");**

**DataTable dtSelectionMath = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionMath.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionMath.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectionMath.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**//Expert**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionMath.Rows.Add(drn);**

**}**

**grdMath.DataSource = dtSelectionMath;**

**grdMath.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Research domain**

**/// </summary>**

**void showResearchSummary()**

**{**

**if (Session["ResearchExp"] != null)**

**{**

**DataTable dtResearch = (DataTable)Session["ResearchExp"];**

**DataRow[] result = dtResearch.Select("NotAvailable=0");**

**DataTable dtSelectionResearch = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionResearch.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionResearch.Columns.Add(dc);**

**foreach (DataRow dr in result)**

**{**

**DataRow drn = dtSelectionResearch.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**//Expert**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionResearch.Rows.Add(drn);**

**}**

**grdResearch.DataSource = dtSelectionResearch;**

**grdResearch.DataBind();**

**}**

**}**

**/// <summary>**

**//Show user profile selections associated to Tools domain**

**/// </summary>**

**void showToolSummary()**

**{**

**if (Session["Tool"] != null)**

**{**

**DataTable dtTool = (DataTable)Session["Tool"];**

**DataTable dtSelectionTools = new DataTable();**

**DataColumn dc = new DataColumn();**

**dc.ColumnName = "Data";**

**dtSelectionTools.Columns.Add(dc);**

**dc = new DataColumn();**

**dc.ColumnName = "Expertise";**

**dtSelectionTools.Columns.Add(dc);**

**foreach (DataRow dr in dtTool.Rows)**

**{**

**if (dr["NotAvailable"].ToString() == "0")**

**{**

**DataRow drn = dtSelectionTools.NewRow();**

**drn["Data"] = dr["Data"];**

**string str = "";**

**if (dr["intermediate"].ToString() == "1")**

**{**

**str = "intermediate";**

**}**

**else if (dr["Expert"].ToString() == "1")**

**{**

**str = "Expert";**

**}**

**else if (dr["Available"].ToString() == "1")**

**{**

**str = "Novice";**

**}**

**drn["Expertise"] = str;**

**dtSelectionTools.Rows.Add(drn);**

**}**

**}**

**grdTools.DataSource = dtSelectionTools;**

**grdTools.DataBind();**

**}**

**}**

**//Go to Programming page for any modificatiosn in expertise and recalculate modules of semester 1 and semester 2**

**protected void btnModifyExpertise\_Click(object sender, EventArgs e)**

**{**

**Response.Redirect("~/Programming.aspx");**

**}**

**// Go to result page for readjusting the semester 1 and semester 2 modules**

**protected void btnModifyModule\_Click(object sender, EventArgs e)**

**{**

**Session["CallFromResult"] = "Modify";**

**Response.Redirect("~/Result.aspx");**

**}**

**}**

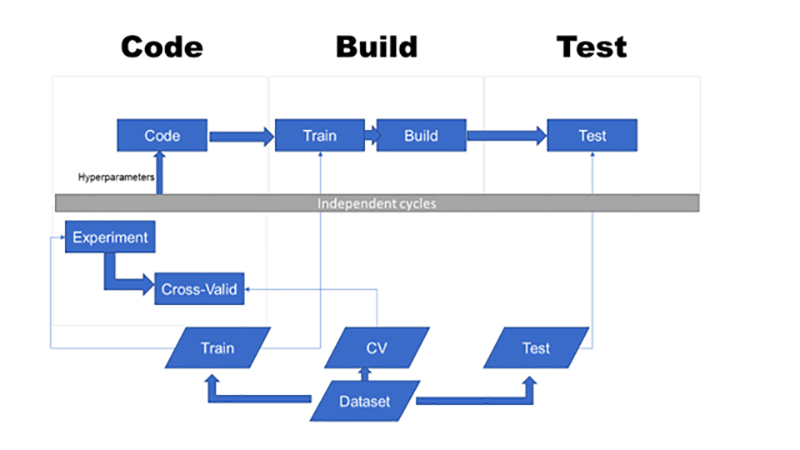
**}**

**4.6 Conclusion**

The chapter explains the business logic implementation, system development process and working of major classes / functions on the basis of 3-tiered architecture. In the end, versioning process through GitHub has been discussed briefly, along with application functionalities achieved in each of the two iterations. The next chapter would test and evaluate these functionalities against project scope and objective. Current MTech students’ feedback on application performance and design would also be discussed in the coming chapter.

**Chapter 5 Testing and Evaluation**

**5.1 Testing Machine Learning**

****

Testing machine learning and AI algorithms is hard. In fact, testing scientific software in general is hard, and there has been some literature on this more general topic. As said in the introduction of Carver et al (2017):

“The development of scientific software differs significantly from the development of more traditional business information systems, from which many software engineering best practices and tools have been drawn. These differences appear at various phases of the software lifecycle”

These differences appear throughout the following stages of the software lifecycle: requirement, design, coding, validation and verification, and deployment. In particular, the difficulties one usually faces in validating and verifying scientific software are, according to Carver et al:

* Results are often unknown when exploring novel science or engineering areas and algorithms
* Popular software engineering tools often do not work on the architectures used in computational science and engineering

Indeed, machine learning and AI algorithms are often the result of explorations by data scientists and researchers. When the resulting algorithms are subsequently implemented by engineers, some very important practices in software quality assurance are often difficult implement, due to the differences in working culture between the two roles.

## 5.2 The oracle problem

One of them is the **oracle problem**, as was discussed in Weyuker (1982) extensively. In software testing, an oracle refers to a mechanism which can tell you whether a program is working correctly. For the simplest case, an oracle could be a direct comparison of the output of the program with the correct answer. More complicated oracles may involve running another program to determine if the output of the target program is correct.

For machine learning and AI algorithms, the problem is that often there is no oracle without human intervention. Take image recognition for example. A common way of solving image recognition problem is by supervised machine learning. This usually begins by curating a dataset of correctly labelled image for training and validating, which is itself a way of introducing an oracle for the software model in development by human intervention. This implies that the range of test conducted to the model will be limited by human efforts, and expanding the test cases, for example increasing the sampling size and varieties significantly with online testing, would be a problem.

Several methods have been proposed to mitigate the oracle problem, including the use of pseudo-oracle and metamorphic testing, which we will discuss later in this article.

In the following we will discuss metamorphic testing which is proposed by Chen et al (1998). We will start by introducing the concept of metamorphic testing as a general purpose toolkit in software testing, followed by explaining some research regarding applying metamorphic testing to machine learning software.

## 5.3 Metamorphic testing

## 

Metamorphic testing is a software testing methodology to mitigate the oracle problem. The idea is simple: even if we do not know the correct output of a single input, we might still know the relations between the outputs of multiple inputs, especially if the inputs are themselves related. We can check the software for these relations, called **metamorphic relation**. If they don’t hold, it is a sure sign that the software has some defects.

An example given by Chen et al (1998) is to test whether a given path is the shortest path in an undirected weighted graph. This is a typical case in software testing where resorting to an oracle would be expensive when the graph is complicated. If (x, v1, …, vN, y) is the proposed shortest path between x and y with length p, Chen et al suggest a few follow-up tests to conduct on the algorithm, among them:

* Find the shortest paths between y and x (in reverse direction), and verify that the length is also p.
* Find the shortest paths between x and vK, and between vK and y. Verify that the sum of the lengths of the two paths is also p, for any K between 1 and N.

To put the methodology formally, let’s say f is the software we are testing and we have an input x with the output f(x). We don’t have an oracle for f(x)so verifying this single test case would be difficult. However, a known transformation T, which is based on a metamorphic relation, can be applied to x to generate the follow-up test case T(x). We can calculate f(T(x)), and then verify it with an oracle T'(f(x)) constructed from the known f(x) by using the metamorphic relation. In the above example of the shortest path problem, the known resulting length p becomes an oracle for the two categories of follow-up test cases.

## 5.4 Test Cases

## 5.4.1 Machine Learning Test Cases

|  |  |
| --- | --- |
| TC-01 | Permutation of class labels: Change the class label of testing data for each iteration |
| TC-02 | Permutation of the attribute: Change the attributes of testing data for each iteration |
| TC-03 | Addition of uninformative attributes: Addition of random information attributes in data |
| TC-04 | Addition of informative attributes: Addition of logical informational attributes |
| TC-05 | Consistence with re-prediction: Predict the similar value repeatedly to check the consistency |
| TC-06 | Additional training sample: Addition of more training data sample |
| TC-09 | Removal of classes: Removing classes in data sample |
| TC-10 | Removal of samples: Removing the sample data |

## 5.4.1 UI Test Cases

|  |  |
| --- | --- |
| TC-01 | Testing of font style and colour theme applied in the module. |
| TC-02 | Amount of module recommendation information visible on the screen. |
| TC-03 | Making manual changes to module selections on *Result Page.* |
| TC-04 | System does not allow manual changes to the module selections in violation of credit hour limitations |
| TC-05 | User to go back and make any changes in user profile or module selections, once they have been added to the database |
| TC-06 | Re-displaying list of system recommended modules on *Result Page*, once they have been shuffled by the user |
| TC-09 | Testing of configuration process involves almost 30 academic attributes with four possible radio button options. Testing of all modules |

* 1. **Conclusion**

Being the final chapter of the report, it elaborates on system testing and evaluation activities carried out, either on self-assessment basis or with the help of project supervisor and current M. Tech students. While doing so, the chapter critically analyses different aspects of the project such as development methodology, and its aims and objectives viz-a-viz the software product developed. It also discusses different observations raised on working prototype and actions taken to address them. The chapter concludes by highlighting recommendations for future work and personal reflection on the whole process.

* 1. **Future Work**

Background research on possible implementation approaches (Section 2.2) and appropriate mathematical algorithms (Section 2.3) for recommendation systems’ implementation highlighted new dimensions in the same domain, which can be explored further to exploit their true potential. Some of these areas are discussed below:

* + - While implementation on the basis of knowledge based approach in the context of this project has its own advantages (Section 2.2.5), but merging all feasible approaches in one solution through well-defined mechanism would be overwhelming. The resultant hybrid approach may take into account:
      * Collaborative approach: “What is popular among my peers”
      * Content based: “More of the same”
      * Knowledge based: “What fits my needs”

For collaborative approach implementation, the application would use existing database of students’ user profiles and their respective module selections and recommend modules to new students on the basis of academic expertise similarities between them and already saved user profiles. Similarly, for content based approach, keywords representing main concepts and topics for each module can be used as an ‘item features’ and can be used to suggest a module on the basis of feature similarity level between previously selected modules by the student and new modules.

* + - The application can take into account other factors contributing in module selection process such as future career goals, academic scores (CGPA, Grade etc.), career prospects and popularity level of a certain module or its leader on the basis of past student reviews. Additionally, a user-configurable weightage parameter can be added to each factor, so as to produce recommendation results according to one’s personal priorities.
    - The application design and its implementation is generic in nature and can be used for other M.Tech and undergraduate programmes. However academic module requirements data against new programme would be required to be added to the database.
    - Looking into possibility of using other mathematical algorithms (Section 2.3) for identifying similarity levels between ‘items’ and ‘features’ or among themselves is an interesting area of research. In the same context, role of machine learning methods in educational recommender systems can also be investigated, by training and testing the algorithms with the help of historical data records.
    - Validating and evaluating the performance of educational recommender systems is a worth considering research area. Learning is an extended and qualitative affair, having numerous direct and indirect contributory factors associated with it, thus it is difficult to judge whether a particular recommendation actually worked for user or not in the longer run. Moreover user preference, prediction accuracy, system coverage, serendipity, diversity and risk calculation etc. are critical concepts used to gauge performance of recommender systems [11]. Their implementation in an educational domain could give another view to system assessment.

**List of References**

1. <https://www.brainyquote.com/quotes/topics/topic_education.htmlQuote>.
2. <http://preston-uni.blogspot.co.uk/2012/10/importance-of-selecting-right-degree.html>.
3. Connor H, Burton R, Pearson R, Pollard E, Regan J. “Making the Right Choice How Students Choose Universities and Colleges”. Universities UK Group. [http://intouniversity.org/sites/all/files/userfiles/files/IES%20%20Making%20the%20Righ t%20Choice%20(summary).pdf](http://intouniversity.org/sites/all/files/userfiles/files/IES%20%20Making%20the%20Right%20Choice%20(summary).pdf). Published June, 1999..
4. <https://soc-dm.leeds.ac.uk/module/62/view>.
5. <http://vulms.vu.edu.pk/Courses/CS605/Downloads/SoftwareEngineeringII.pdf>. CS605. Software Engineering-II.
6. <http://vulms.vu.edu.pk/Courses/CS504/Downloads/SoftwareEngineeringI.pdf>. CS504. Software Engineering-I..
7. <https://en.wikipedia.org/wiki/Collaborative_filtering>.
8. Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich. “Recommender Systems: An Introduction”. Cambridge University Press. 2011
9. [https://www.fi.muni.cz/~xpelanek/PV254/slides/other-techniques.pdf](https://www.fi.muni.cz/%7Expelanek/PV254/slides/other-techniques.pdf)
10. <https://www.techopedia.com/definition/28556/filter-bubble>
11. Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B Kantor. “Recommender Systems Handbook”. Springer London, 2011
12. Bo Xiao, Izak Benbasat. “E-Commerce Product Recommendation Agents: Use, Characteristics, And Impact”. MIS Quarterly Vol. 31 No. 1, pp. 137-209/March 2007.
13. [https://www.quora.com/Which-companies-use-recommender-recommendation- systems](https://www.quora.com/Which-companies-use-recommender-recommendation-systems)
14. Olga C. Santos, Jesus G. Boticario. Spanish National University for Distance Education. “Educational Recommender Systems and Technologies: Practices and Challenges” December, 2011. DOI: 10.4018/978-1-61350-489-5.
15. Parameswaran, A, Venetis, P & Garcia-Molina, H 2011, “Recommendation systems with complex constraints: A course recommendation perspective” ACM Transactions on Information Systems, volume 29, No. 4, 2037665. DOI: [10.1145/2037661.2037665.](http://dx.doi.org/10.1145/2037661.2037665)
16. <http://infolab.stanford.edu/db_pages/courserank/courserank.html>.
17. Grewal DS, Kaur K (2016). “Developing an Intelligent Recommendation System for Course Selection by Students for Graduate Courses”. Business and Economics Journal. Volume 7. Issue 2. doi:10.4172/2151-6219.1000209.
18. Joe Manley Gnanasekaran, Ramkishore Swaminathan, Swetha Krishnakumar. “University Recommender System for Graduate Studies in USA”. [https://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/026.pdf](https://cseweb.ucsd.edu/%7Ejmcauley/cse255/reports/fa15/026.pdf).
19. Rob Koper, Bas Giesbers, Peter van Rosmalen, Peter Sloep, Jan van Bruggen, Colin Tattersall, Hubert Vogten, Francis Brouns. “A Design Model for Lifelong Learning Networks”. Interactive Learning Environments. Vol. 13, No. 1-2, April-August 2005, pp. 71- 92. DOI: 10.1080/10494820500173656.
20. Marije De Heus. “Design and Evaluation of a Recommender System for High School Courses in The Netherlands”. December, 2013. [http://essay.utwente.nl/65029/1/Masters%20Thesis%20Marije%20de%20Heus.pdf](http://essay.utwente.nl/65029/1/MSc%20Thesis%20Marije%20de%20Heus.pdf).
21. <http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>.
22. Guanwen Yao, Lifeng Cai. “User-Based and Item-Based Collaborative Filtering Recommendation Algorithms Design”. [https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Guanwen%20Yao\_Lifeng\_Ca i.pdf](https://cseweb.ucsd.edu/%7Ejmcauley/cse255/reports/wi15/Guanwen%20Yao_Lifeng_Cai.pdf).
23. <https://docs.microsoft.com/en-us/aspnet/overview>.
24. <https://docs.microsoft.com/en-us/aspnet/web-forms/what-is-web-forms>.
25. <https://www.asp.net/web-pages>.
26. [https://stackoverflow.com/questions/7674102/advantages-and-disadvantages-of-asp- net-webforms-vs-asp-net-mvc-some-points](https://stackoverflow.com/questions/7674102/advantages-and-disadvantages-of-asp-net-webforms-vs-asp-net-mvc-some-points).
27. [https://stackoverflow.com/questions/102558/biggest-advantage-to-using-asp-net-mvc- vs-web-forms](https://stackoverflow.com/questions/102558/biggest-advantage-to-using-asp-net-mvc-vs-web-forms).
28. <https://en.wikipedia.org/wiki/ASP.NET#State_management>.
29. <https://www.tutorialspoint.com/asp.net/asp.net_managing_state.htm>.
30. [https://msdn.microsoft.com/en-us/library/0wxwcakt(v=vs.71).aspx](https://msdn.microsoft.com/en-us/library/0wxwcakt(v%3Dvs.71).aspx)
31. <http://www.enqos.com/blog/key-things-need-know-asp-net-core/>.
32. <http://vulms.vu.edu.pk/Courses/CS403/Downloads/CS403%20Handouts.pdf>.
33. <https://en.oxforddictionaries.com/definition/gold_standard>.
34. [https://docs.microsoft.com/en-us/aspnet/web-forms/overview/getting-started/creating- a-basic-web-forms-page](https://docs.microsoft.com/en-us/aspnet/web-forms/overview/getting-started/creating-a-basic-web-forms-page).
35. <https://minerva.leeds.ac.uk/webapps/blackboard/execute/content/blankPage?cmd=vie>w&content\_id=\_4213014\_1&course\_id=\_427823\_1
36. [https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/authentication-in- sql-server](https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/authentication-in-sql-server)
37. <http://whatis.techtarget.com/definition/fuzzy-logic>.
38. [http://webprod3.leeds.ac.uk/catalogue/modulesearch.asp?Y=201718&T=S&L=TP](http://webprod3.leeds.ac.uk/catalogue/modulesearch.asp?Y=201718&amp;T=S&amp;L=TP).
39. [http://webprod3.leeds.ac.uk/catalogue/modulesearch.asp?Y=201718&T=S&L=UG](http://webprod3.leeds.ac.uk/catalogue/modulesearch.asp?Y=201718&amp;T=S&amp;L=UG).
40. <http://www.conceptdraw.com/How-To-Guide/uml-flowchart-symbols>