WolfPal 2.0: Feedback and recommender system for course selection

Neel Kapadia
North Carolina State University
Raleigh, NC, US
ntkapadi@ncsu.edu

Rohan Chandavarkar
North Carolina State University
Raleigh, NC, US
rgchanda@ncsu.edu

Sainag Shetty
North Carolina State University
Raleigh, NC, US
sgshetty@ncsu.edu

Rohit Naik
North Carolina State University
Raleigh, NC, US
rtnaik@ncsu.edu

ABSTRACT

In today's competitive world students need to take certain decisions in their academic life which seem to be easy but have a major impact on their future. Choosing the right courses in their degree program at the right time is one such decision making task which is of great relevance to a student's academic and professional future. Apart from contributing to the grade point average and shaping up the resume, the course selection also affects the qualitative factors like the student's interest in the degree program. Since the decision of coming out with a proper course plan directly affects the career of the student, this decision should be a well informed one. The students rely on various sources and use multiple techniques to find out the best courses for them which fit in their credit-based study. We thought of contributing to one such platform which brings together all such resources and help the students in making this course plan.

The platform has the functionalities of interacting with peers and discuss important points like professor reviews, average grades, course structure. We added new functionalities in the platform using latest techniques like natural language processing so that the factor of student's interests and skill-sets is considered more strongly in the course planning. Moreover, we extracted course information from NCSU Course Catalog which replaced the earlier static database. Another important feature we added in the web application was the file upload functionality so that users can add syllabus of respective courses.

Keywords

Course Feedback, Course Recommendation, Keyword Extraction

1. INTRODUCTION

The task of planning the courses is one of the most important tasks in a student's life. Students rely on a lot of resources which are scattered all around and not all resources give the students a perfect strategy to choose the right courses for them. The task of course selection is taken seriously by students all over the globe and it is necessary that all the resources are utilized in conjunction to give a course plan to the students which will be beneficial for them.

There are hundreds of universities in North America and huge number of students pursuing advanced degrees in these universities. During the application phase, the students select certain universities based on the area of specialization and the courses offered in that field. So it would be unfair for the students if they do not get the right courses after taking admission due to lack of resources or lack of communication. Wolfpal, a course planning assistant, is a platform which aims to resolve these shortcomings and come up with a good course plan for the students. We have added new functionalities to this novel approach of course planning by applying latest techniques available.

1.1 Previous Functionalities

Wolfpal 1.0 was a highly efficient platform which had important functionalities like chatting with peers about the courses, average grades of previous batches, course structure. These parameters were judged accurately by experience of similar people and a chatting platform for the same serves the purpose. Moreover, the platform also had a bot functionality where in the students can get answers to certain queries related to planning of courses. The system also provided a list to the students for the course plan and also displayed course information from the course catalog to the students.

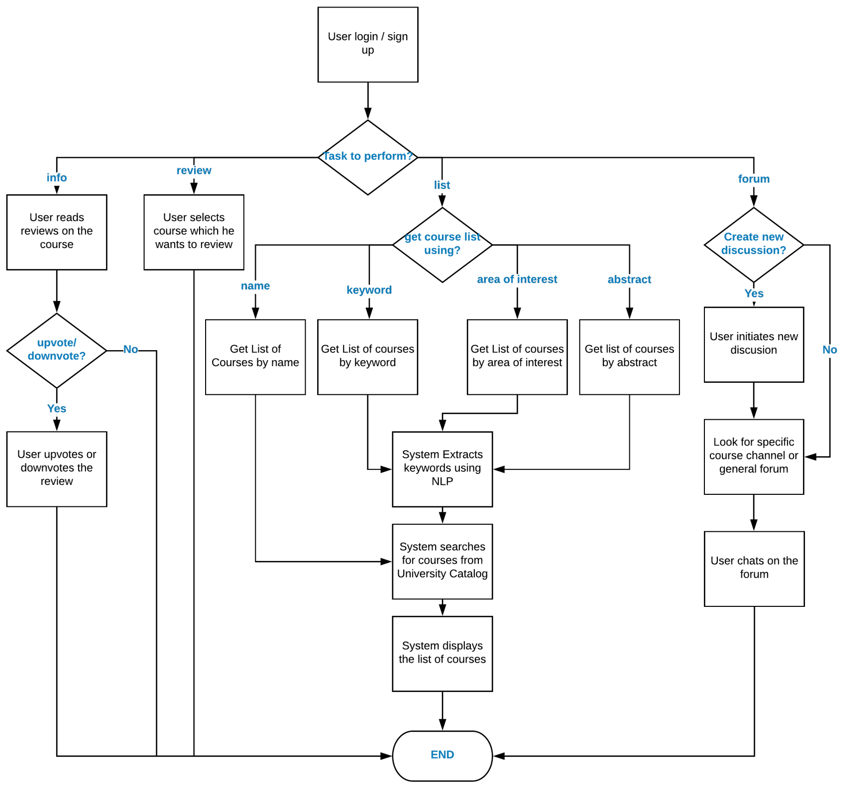
1.2 New Functionalities

The main aim of our new functionalities was to capture maximum students' interests, requirements and skills to suggest them a course plan.

1. We added a functionality of suggesting courses by mapping course descriptions with keywords. We used Natural Language Processing techniques for the same.

2. The course details are now extracted directly from the NCSU Course Catalog, thus replacing the earlier static database which was designed by manually entering the course details.

3. Another functionality we added was a file uploader so that students could add syllabus of the courses for



User login / sign up
Task to perform?
info
review
list
forum
User reads reviews on the course
User selects course which he wants to review
get course list using?
name
keyword
area of interest
abstract
Create new discussion?
Yes
No
upvote/ downvote?
Yes
No
Get List of Courses by name
Get List of courses by keyword
Get List of courses by area of interest
Get list of courses by abstract
User initiates new discusion
User upvotes or downvotes the review
System Extracts keywords using NLP
Look for specific course channel or general forum
System searches for courses from University Catalog
User chats on the forum
System displays the list of courses
END

Figure 1: Activity Diagram

other users to view thereby giving the students more information about the courses.

2. MOTIVATION

The dilemma of deciding which course to take is a major problem faced by every student. With a plethora of options and not a very reliable source of information, this task only becomes more difficult. Take the case of North Carolina State University, specifically the Computer Science stream, there are around 125 different courses. A student needs to study and find out which amongst these courses best suits his/her interest, which is a lot of work. Assuming it takes about 15 minutes to find out what the course has to offer, doing that for all of the possible courses takes 31.25 hours. Also, this is based on the condition that, all of the information about that course is readily available.

For a student, all of this is a lot of work given there is hardly any guidance. The aim of WolfPal is to reduce the effort put in by the student to do this work. The application integrates the resources provided by the university with the remarks of the students on a course, who had taken the course earlier, which will be ideal for the tackling the problem of lack of information. Generally, students are aware of certain keywords or domains in their field of study and not exactly the course titles. This observation prompted us to add the functionalities of suggesting courses based on keywords and domains. Before enrolling into a course, a student is also interested in knowing about the work load and grading system of the course. By adding the file upload functionality to upload syllabus, we attempt to address this requirement of the student.

3. LITERATURE REVIEW

Taking good courses is a very important necessity in the career of every student. Despite steady work done related to course recommendation and feedback system, course selection remains a challenging process for any graduate student. In the following section we will discuss the previous works related to course recommendation and feedback system along with their shortcomings.

3.1 WolfPal 1.0

WolfPal 1.0[1] an interactive platform that aims toward helping students with selecting courses for their graduation. It tries to integrate the resources provided by the university with the remarks of the students on a course, who had taken the course earlier which will be ideal for the tackling the problem of lack of information. Thus, it tries to give the student an idea of how and what to expect from a particular course. The system will provide a way for the student to connect to other students for sharing any information or queries about courses. The system also gives a chatbot for course recommendation.

The shortcomings from this current version of the application can be described as follows:

- Chatbot integrated into a Web GUI isn't intuitive:

  The application helps in course recommendation using a chatbot that recommends courses. But the problem here is that whilst using a Web Based GUI having a bot which does a QA is very intuitive.

- The keyword filtering of the courses is not effective:

  While the system does have the feature of filtering out courses by putting in keywords, this filtering technique is not guaranteed to produce ideal results as also noticed during testing.

3.2 Course feedback system

"Course feedback system"[2] provides three features. One is a forum that can share short reviews for each course with like and dislike button that can increase its reliability. The second feature is a feedback system that senior students could answer eight questions for each course, such as numbers of project, number of assignment, course knowledge and so on. The third feature is a suggestion system that helps student search courses by 8 filters based on the data from second feature. We think this system is a more reliable and a more feedback friendly system than the previous one, because of its separate review and feedback, as well as grading criteria (questions) by multiple choice to prevent typing errors.

3.3 An Automated Recommender System for Course Selection

This paper[3] presents a collaborative recommender system that recommends university elective courses to students by exploiting courses that other similar students had taken. The system employs an association rules mining algorithm as an underlying technique to discover patterns between courses. The system tries to recommend elective courses to students based on what other similar students have taken. It finds similar students and then apply association rule mining algorithm on their courses to create courses association rules. Discovered courses association rules are used to get recommendation.

4. CHANGES FROM EXISTING SYSTEM

We added the following new features to the application -

4.1 Course suggestions based on keyword mapping

Introduction: This feature particularly targets the improvement of the keyword search functionality in the web application.

Motivation: The main reason for adding this functionality was the results of the evaluation of WolPal 1.0. The course search based on keywords was an important functionality and we wanted to at least improve if not perfect how it operates. Also, according to our initial discussions with our mentor, Ken Tu, we finalized that this was a great feature where we could implement along with the use of Natural Language Processing techniques. Also, sometimes the name of the course is misleading or under representative of what the course actually is. It requires more than the course name for a student to make a decision about a course. Hence, searching based only on the course title won't be of help to the students.

Solution: From the course description that we fetched using the web scraping, we removed all the possible stop words and other irrelevant words. After which the words were stemmed and lemmatized using the NLTK library in Python, thereby fetching keywords for a courses. Repeating this process for all courses available, we then mapped these keywords onto a bucket of keywords that segregated them into topics using Topic Modelling approach. Also, we extracted keywords from the user's interest description. Then we matched those keywords with the topic map and display the corresponding matching courses with all the details of those courses. This will ensure all round information for the user.

4.2 Upload syllabus functionality

Introduction: This functionality refers to an upload button and a back end architecture to handle file uploads from the users of the system and at the same time allowing them to download syllabus of uploaded subjects.

Motivation: Currently, the data in the system was uploaded by the developers themselves, which won't work in the longer term. It might cause unnecessary overhead on the developers' end, of maintaining the syllabus files on the portal and adding additional files. This idea won't be practical and the time spent by developers can instead be used for developing other important features.

Solution: Hence, we have implemented a file upload functionality in the web application so that students can upload syllabus files for others to view. We have used the refile gem for adding this functionality. By using the gem instead of developing the functionality from scratch, we get multiple benefits such as - configurable backends, convenient integration with ORMs, on the fly manipulation of images and other files, streaming IO for fast and memory friendly uploads, effortless direct uploads, support for multiple file uploads, and many more. Thus, we get a fully secure, robust and versatile feature for our application and can also get to spend more time on other features.

4.3 Adding courses dynamically from NCSU Course Catalog

Introduction: This feature aims at making the display of courses cleaner in the backend.

Motivation: It was one of the sought after features from the previous evaluations of WolfPal 1.0. Dynamic course content retrieval could lead to opening many doors for the project scope such as easily adding courses from other departments and expanding the user base of the application. Further, this also leads to saving of the developers' time to manually get course descriptions from the catalog. Also, it ensures that the descriptions that are updated or new courses which are added, get updated in the system without extra effort.

Solution: To implement this, there were a few challenges that we faced. The course catalog page from which the descriptions were to be extracted was not a page with all the details given directly. The page had a dialog box, in which we needed to enter parameters which then led us to the next page having the course descriptions. To implement this, we wrote a selenium script which entered the parameters in the dialogue box and took us to the next page which was then used to extract the course content. The same procedure was done for all the different departments to get courses from all departments.

4.4 Fetching courses by domain

Introduction: This feature aims at giving a list view of courses according to the domain they belong to such as Development, Systems, Data Science, Networking, Electronics, Robotics, etc.

Motivation: Many times students want to know what courses does the university has to offer in a particular field of study which happens to be their liking. In such cases, it becomes very helpful for the students to have such a list of courses in front of them.

Solution: With the help of topic modeling, we create a mapping between courses and topics. When a user requests for a domain, the domain name is mapped with topic and corresponding courses are returned.

5. IMPLEMENTATION PLAN

5.1 Previous Modules

5.1.1 Graduate Plan Formation:

This features enables students to easily look for important details of the course like course description, syllabus, schedule, instructor, workload, grade distribution, projects, fieldwork, and also student can access the dedicated forum of each course to view the thread of discussions or raise relevant query which can be answered by the students who has already enrolled in that course. Moreover, students can use the combination of filters to search the course on the basis of the keywords, semester, project, core or fieldwork. This feature also includes an option for students to add or remove courses to the the plan. With the help of this feature, a new student can be provided with a proper blueprint of his entire university schedule till he/she graduates. Having access to such a significant resource before even stepping foot on university premises will show a marked upturn in a student's academic performance.

5.1.2 PalTalk: student communication

The application provides a platform for students to communicate with relevant peers in the university which can help new students to discuss any queries related to course selection. Moreover, this platform provides an option to send a private message to any peer. We have tried to make this platform as generic as possible to promote communication by providing a public platform to interact with all students. This will help resolve any doubts a student possesses. A university-wide open forum promotes communication between the students to a previously unprecedented level. The academic study will be helped with department-specific and course-specific threads on this forum, to provide an open platform for communication. Additionally, this feature keeps the students more involved in campus life by keeping them informed about the upcoming events in the university. Specific threads of communication can also be created for this.

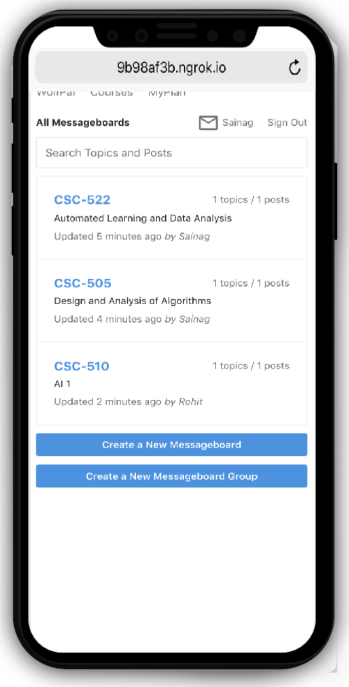
5.2 Use Cases

1. login/signup

   This is the authentication module where the users create a new account or login to their existing accounts.

2. get list of courses by name

   The users search for a particular course by the name of the course or the course number. A direct mapping


9b98af3b.ngrok.io

All Messageboards Sainag Sign Out

Search Topics and Posts

CSC-522 1 topics / 1 posts
Automated Learning and Data Analysis
Updated 5 minutes ago *by Sainag*

CSC-505 1 topics / 1 posts
Design and Analysis of Algorithms
Updated 4 minutes ago *by Sainag*

CSC-510 1 topics / 1 posts
AI 1
Updated 2 minutes ago *by Rohit*

Create a New Messageboard

Create a New Messageboard Group

Figure 2: PalTalk

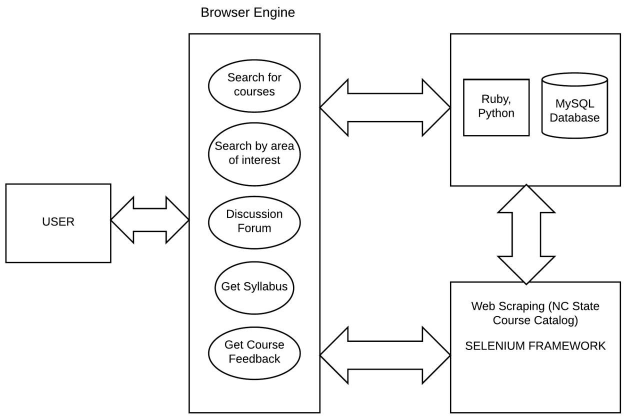


Figure 3: Architecture Diagram

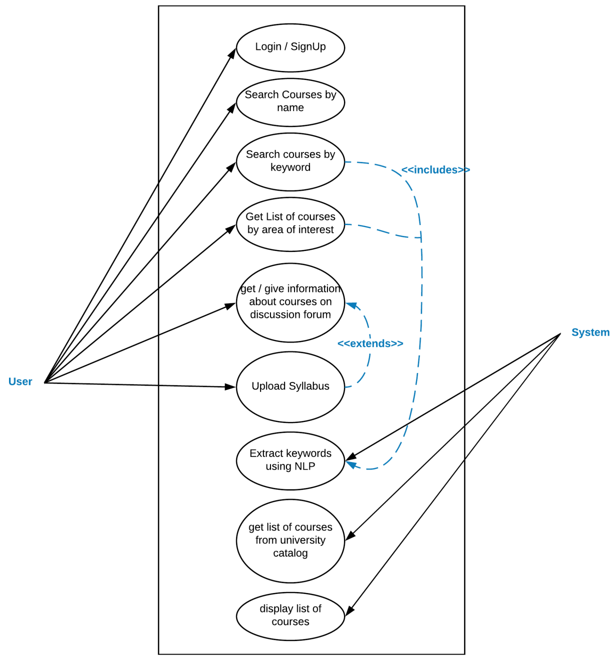
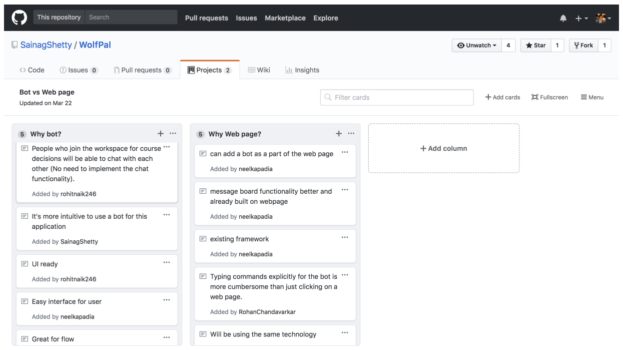

Login / SignUp
Search Courses by name
Search courses by keyword
Get List of courses by area of interest
get / give information about courses on discussion forum
Upload Syllabus
Extract keywords using NLP
get list of courses from university catalog
display list of courses
<<includes>>
<<extends>>
User
System

Figure 4: Use Case Diagram



Bot vs Web page
Updated on Mar 22

5 Why bot?

People who join the workspace for course decisions will be able to chat with each other (No need to implement the chat functionality).
Added by rohitnaik246

It's more intuitive to use a bot for this application
Added by SainagShetty

UI ready
Added by rohitnaik246

Easy interface for user
Added by neelkapadia

Great for flow

5 Why Web page?

can add a bot as a part of the web page
Added by neelkapadia

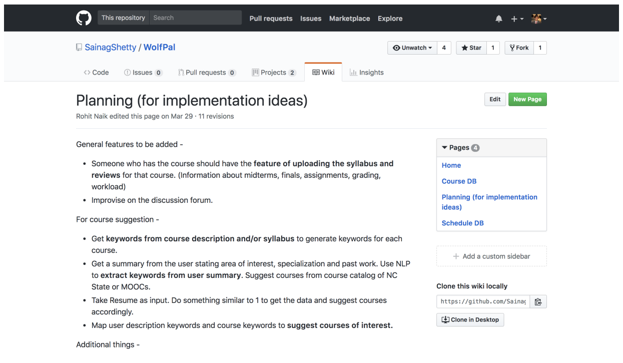
message board functionality better and already built on webpage
Added by neelkapadia

existing framework
Added by neelkapadia

Typing commands explicitly for the bot is more cumbersome than just clicking on a web page.
Added by RohanChandavarkar

Will be using the same technology

Figure 5: Using columns in Github Projects for planning



Planning (for implementation ideas)

Rohit Naik edited this page on Mar 29 · 11 revisions

General features to be added -

- Someone who has the course should have the **feature of uploading the syllabus and reviews** for that course. (Information about midterms, finals, assignments, grading, workload)
- Improvise on the discussion forum.

For course suggestion -

- Get **keywords from course description and/or syllabus** to generate keywords for each course.
- Get a summary from the user stating area of interest, specialization and past work. Use NLP to **extract keywords from user summary**. Suggest courses from course catalog of NC State or MOOCs.
- Take Resume as input. Do something similar to 1 to get the data and suggest courses accordingly.
- Map user description keywords and course keywords to **suggest courses of interest.**

Additional things -

Figure 6: Using Wiki in Github for planning

between the search and the course name/number is done and the course(s) is/are returned.

3. get list of courses by keyword

   The users search for courses by keyword. For example, if the user enters the keyword "computer science" then all courses related to that keyword will be displayed. It involves a topic modeling approach which maps each course to topic(s).

4. get list of courses according to area of interest

   The users search for courses by their area of interest. For example, if the user enters the keyword "i am interested in data science" then related courses to the domain of data science will be displayed. Using NLP, the search term is broken down and mapped to the course map, which contains the topic and the course.

5. get/give information about courses on discussion forum

   The users can provide information on the courses they attended or get information on the courses they wish to attend on the discussion forum. Basically it is a chat-based application where the users can interact and share information

6. upload syllabus of course

   Students can enter the pdf file of the syllabus of the courses they took. Once a user uploads a syllabus this is available to all users of the application.

7. extract keywords from NLP

   The system extracts keywords from the abstract the user provides. Using the Stemmer and Lemmatizer of the nltk library, the course description is broken down into keywords after removing the stopwords.

8. get courses from university course catalog

   The system get the list of courses from the university course catalog. Using the NCSU course catalog, the selenium webdriver scrapes it to dynamically load the courses into the database.

9. display list of courses

   System displays the list of courses after mapping the keywords with the courses

5.3 Software Engineering Lifecycle and Practices

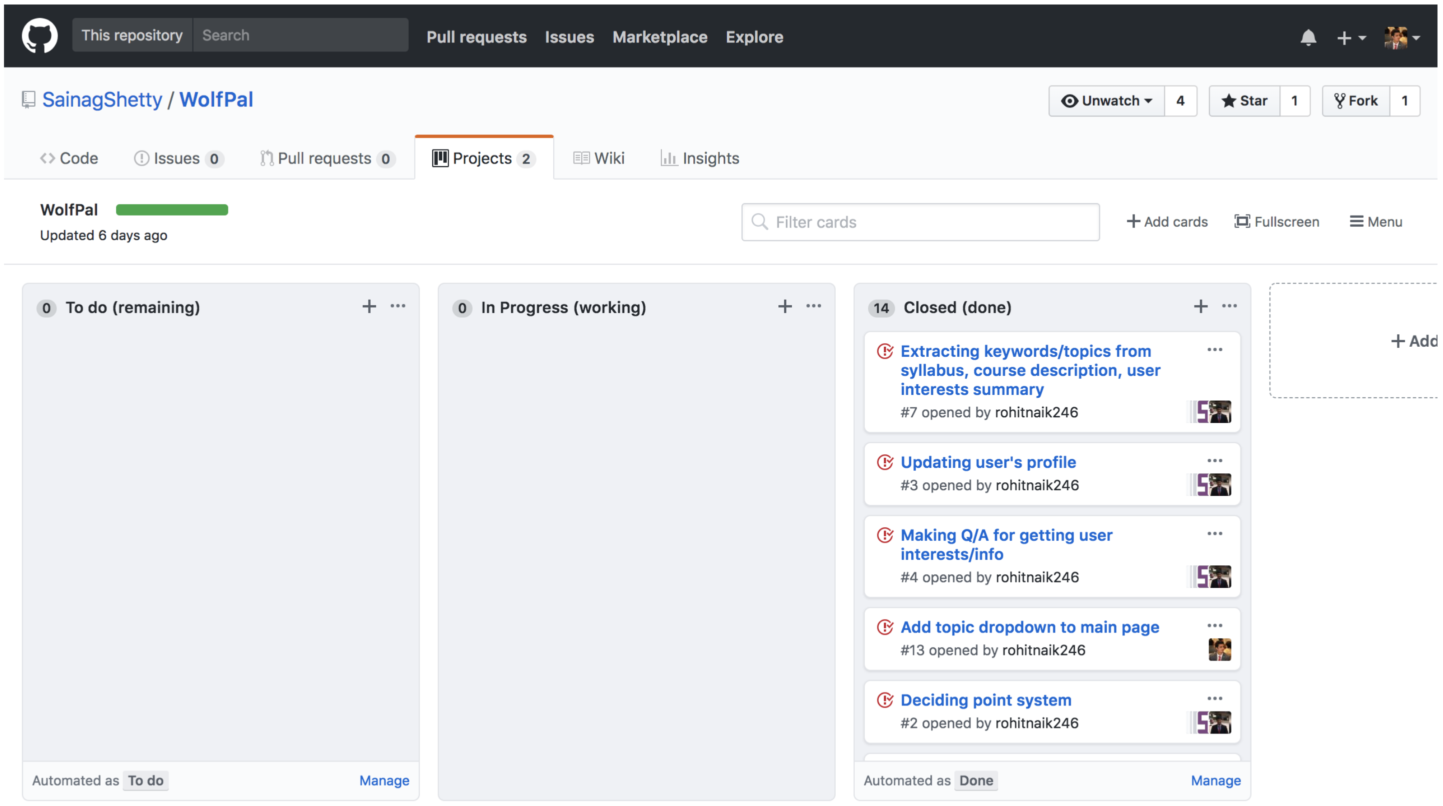
5.3.1 *Software Engineering Lifecycle*

We followed Agile methodologies for implementing the application. We met twice every week meetings in which we deployed basic implementations/improvements of the application and discussed the issues, ideas and future work.

The steps of SDLC we followed:

1. **Requirement Analysis**

   In the initial week of the project, we worked on deciding the scope of the project. We estimated the time and resource requirement for the project. Deciding the scope of the project was important because we didn't


This repository Search Pull requests Issues Marketplace Explore

SainagShetty / WolfPal

Unwatch 4 Star 1 Fork 1

Code Issues 0 Pull requests 0 Projects 2 Wiki Insights

WolfPal
Updated 6 days ago

Filter cards + Add cards Fullscreen Menu

0 To do (remaining)
Automated as To do Manage

0 In Progress (working)

14 Closed (done)
Extracting keywords/topics from syllabus, course description, user interests summary
#7 opened by rohitnaik246
Updating user's profile
#3 opened by rohitnaik246
Making Q/A for getting user interests/info
#4 opened by rohitnaik246
Add topic dropdown to main page
#13 opened by rohitnaik246
Deciding point system
#2 opened by rohitnaik246
Automated as Done Manage

+ Add

Figure 7: Story-board

want to set requirements which are impossible to implement. Also, we had to make some decisions such as whether to have a bot or continue focusing on the web page, which we were able to make easily by making columns on the Project section of Github and listing down pros and cons as shown in Figure 5. Also, we made a wiki page for ease of planning the features 6

2. **Design**

   After deciding the scope, we worked on technologies to be used, use case diagram, activity diagram and sequence diagram of the application. We also created a Gantt chart to plan the work throughout our project. We spent proper time on planning and designing the system we had thought of before going on to the implementation.

3. **Implementation**

   The implementation phase followed the design phase. After all the planning was completed, we could seamlessly move to the implementation. We kept on reporting all the progress information through regular team meetings.

4. **Testing**

   Since the application is focused for student use, we performed Beta Testing and invited 20 students to evaluate our application. We received some great inputs from the testers and worked on the changes required.

5.3.2 *Software Engineering Practices*

We used the following Software Engineering Practices for our project:

1. **Agile Methodology**

   We used Agile[4] methodology for development of our application. So the features were built initially as small incremental features, and then the final one was built on top of those. We are making use of the storyboard feature provided by Github to form issues and then resolve them as you can see in Figure 7.

2. **Code Review**

   Code Review[5] practices are followed by creating different branches on Github other than master and working on those before pushing on master. In this way, we try to reduce bugs and error in our code. Everyone's code was reviewed by the all other team members before committing any file to the master branch.

3. **Pair Programming**

   Pair programming[6] is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently. While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This is intended to free the driver to

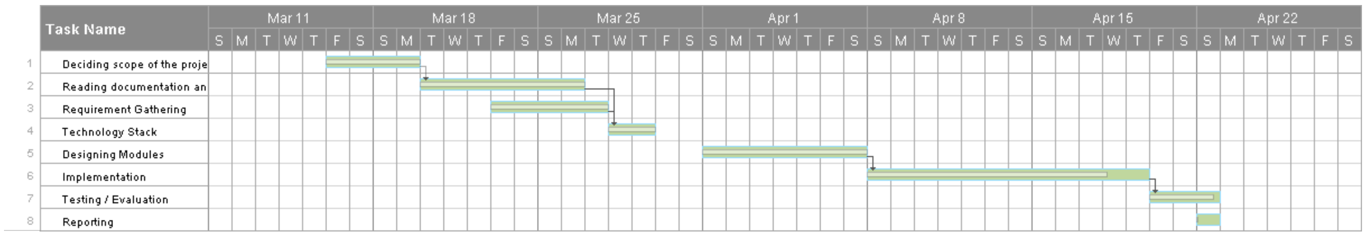


Figure 8: Gantt Chart

focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide. We, being a group of 4, formed 2 groups and did pair programming for the core functionality of the system. This helped debug errors faster and because of it more than 1 member now knows how the module functions. Thus, if 1 of the programmers is not available, then there is always a backup for urgent changes or issues.

5.4 Technology Stack

1. **Python(nltk)**
   - Python is powerful and fast, plays well with others, runs everywhere, is friendly & easy to learn, is open source[7]. Python provides plethora of libraries which can be used to solve almost every problem. Also, it is an easy to use and easy to learn language.
   - The primary reasons for selecting python to develop the core algorithm were:
     (a) suitable libraries available
     (b) team comfort with python
   - We used this language for natural language processing using the python nltk library. The Natural Language Toolkit (NLTK)[8] is an open source Python library for Natural Language Processing.
2. **Javascript**
   - We used javascript as a scripting language between the front-end and back-end.
   - Javascript is used for validation of forms.
   - JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content engineering.[9]
3. **Ruby on Rails**
   - Our system uses Ruby on Rails[10] under the hood which provides ease of development using MVC architecture.
   - The vast available list of Ruby gems (similar to Java packages) helps with the development of a lot of features such as Authentication and Forum, with ease.
4. **Bootstrap**
   - Bootstrap is a free front-end framework for faster and easier web development.
   - Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, models, image carousels and many other, as well as optional JavaScript plugins[11]
   - Bootstrap also gives you the ability to easily create responsive designs.
   - We used bootstrap templates for styling our webpage. Bootstrap templates provide many easy-to-use style for various functionalities such as forms, buttons, tables etc.
5. **SQLite**
   - SQLite is a relational database management system contained in a C programming library.
   - In contrast to many other database management systems, SQLite is not a client server database engine.
   - We used SQLite3[12] because it is the default database for Rails applications.
6. **Selenium WebDriver**
   - Selenium WebDriver[13] accepts commands (sent in Selenese, or via a Client API) and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser, and retrieves results.
   - We use the Selenium WebDriver to extract course information from NCSU Course Catalog.
   - For the purpose of our application, we use the Chrome Driver as the browser driver.

5.5 Challenges Faced

1. **New Technologies**
   The project was implemented in Ruby and CoffeeScript. It was a challenge to learn these technologies and implement the project in such a short span of time. Eventually, we were able to overcome the challenge in the limited amount of time.
2. **SQLite Database**
   The data we had for the project was unstructured and hence more suited for MongoDB NoSQL format. Since the existing system used SQLite database, we first needed to learn about it and also it was difficult to convert our data into a structured format.

3. **Time Management**

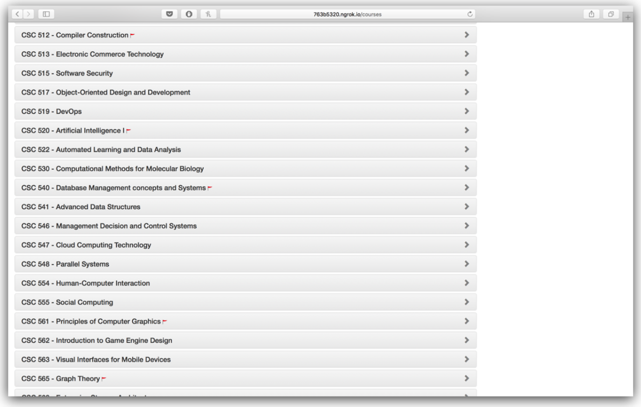
   There was a phase in the project when we were not getting enough time for meetings due to varying schedules. In such a situation, we used Google Hangouts for conducting such meetings.

6. RESULTS

WolfPal 2.0 is an improvement from the initial iteration. Based on the reviews obtained and the future scope[1] of the previous version, this iteration has some key areas which has been improved. This new iteration implements 4 key improvements:

1. **Dynamic course retrieval:**

   The application uses the selenium webdriver to scrap courses from the NCSU Course Catalog. Unlike the first version in which the courses were hardcoded into the application file. Now, it dynamically fetches all the courses and their details. This feature not only gets more details of the courses but also gives way for easy future expansion of the project.



763b5320.ngrok.io/courses

CSC 512 - Compiler Construction
CSC 513 - Electronic Commerce Technology
CSC 515 - Software Security
CSC 517 - Object-Oriented Design and Development
CSC 519 - DevOps
CSC 520 - Artificial Intelligence I
CSC 522 - Automated Learning and Data Analysis
CSC 530 - Computational Methods for Molecular Biology
CSC 540 - Database Management concepts and Systems
CSC 541 - Advanced Data Structures
CSC 546 - Management Decision and Control Systems
CSC 547 - Cloud Computing Technology
CSC 548 - Parallel Systems
CSC 554 - Human-Computer Interaction
CSC 555 - Social Computing
CSC 561 - Principles of Computer Graphics
CSC 562 - Introduction to Game Engine Design
CSC 563 - Visual Interfaces for Mobile Devices
CSC 565 - Graph Theory

Figure 9: Displaying list of courses

2. **Upload syllabus functionality:**

   This feature allowed previous takers of the course to add the syllabus of that course. This is an important feature because it helps others in deciding whether or not to take a course.

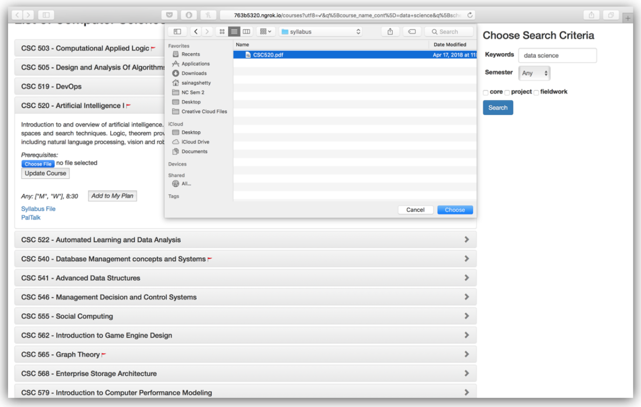
3. **Content-based Keyword search:**

   Unlike the previous iteration of this feature, now the keyword is more content-based i.e. it maps the the user entry to a topic modeling map(course map) using Natural Language Processing.

4. **Fetching courses by domain:**

   The application allows to search and filter courses by domain like Software Development, Data Science, Machine Learning, among others.

   The application involves assigns topics to each course by breaking the description into keywords and then mapping it onto a bucket of keywords which essentially contains relevant keywords along with the topic. This helps in assigning topics to courses.



763b5320.ngrok.io/courses?utf8=✓&q%5Bcourse_name_cont%5D=data+science&q%5Bsch

CSC 503 - Computational Applied Logic
CSC 505 - Design and Analysis Of Algorithms
CSC 519 - DevOps
CSC 520 - Artificial Intelligence I

Introduction to and overview of artificial intelligence.
spaces and search techniques. Logic, theorem prov
including natural language processing, vision and rob

Prerequisites:

Choose File no file selected

Update Course

Any: ["M", "W"], 8:30 *Add to My Plan*

Syllabus File
PalTalk

syllabus

Name	Date Modified
CSC520.pdf	Apr 17, 2018 at 11

Cancel Choose

CSC 522 - Automated Learning and Data Analysis
CSC 540 - Database Management concepts and Systems
CSC 541 - Advanced Data Structures
CSC 546 - Management Decision and Control Systems
CSC 555 - Social Computing
CSC 562 - Introduction to Game Engine Design
CSC 565 - Graph Theory
CSC 568 - Enterprise Storage Architecture
CSC 579 - Introduction to Computer Performance Modeling

Choose Search Criteria

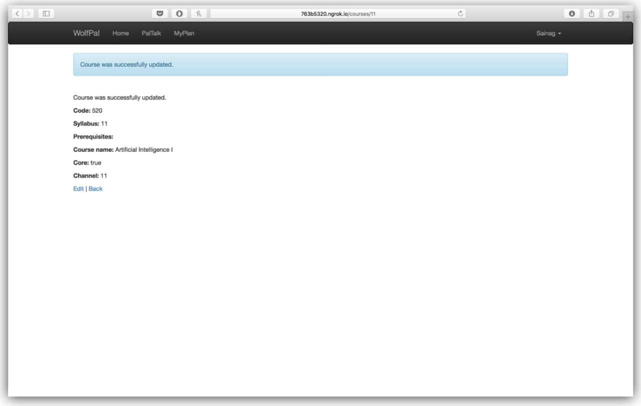
Keywords data science

Semester Any

core project fieldwork

Search

Figure 10: User gets an option to upload syllabus

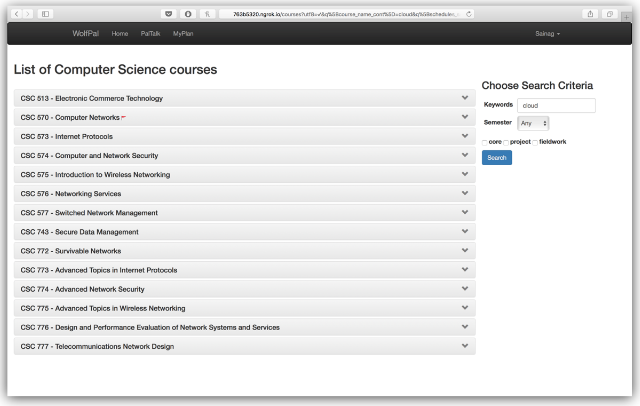


763b5320.ngrok.io/courses/11

Course was successfully updated.

Course was successfully updated.
Code: 520
Syllabus: 11
Prerequisites:
Course name: Artificial Intelligence I
Core: true
Channel: 11
Edit | Back

Figure 11: Syllabus upload confirmation



763b5320.ngrok.io/courses?utf8=✓&q%5Bcourse_name_cont%5D=cloud&q%5Bschedules_s

List of Computer Science courses

CSC 513 - Electronic Commerce Technology
CSC 570 - Computer Networks
CSC 573 - Internet Protocols
CSC 574 - Computer and Network Security
CSC 575 - Introduction to Wireless Networking
CSC 576 - Networking Services
CSC 577 - Switched Network Management
CSC 743 - Secure Data Management
CSC 772 - Survivable Networks
CSC 773 - Advanced Topics in Internet Protocols
CSC 774 - Advanced Network Security
CSC 775 - Advanced Topics in Wireless Networking
CSC 776 - Design and Performance Evaluation of Network Systems and Services
CSC 777 - Telecommunications Network Design

Choose Search Criteria

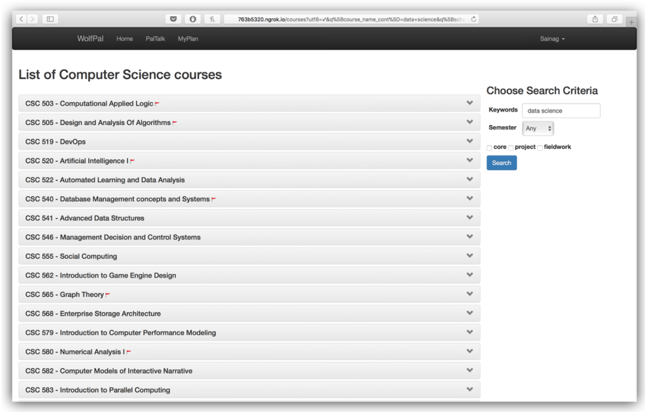
Keywords cloud

Semester Any

core project fieldwork

Search

Figure 12: User searches by keyword


763b5320.ngrok.io/courses?utf8=✓&q%5Bcourse_name_cont%5D=data+science&q%5Bsch
WolfPal
Home
PalTalk
MyPlan
Sainag

List of Computer Science courses

CSC 503 - Computational Applied Logic
CSC 505 - Design and Analysis Of Algorithms
CSC 519 - DevOps
CSC 520 - Artificial Intelligence I
CSC 522 - Automated Learning and Data Analysis
CSC 540 - Database Management concepts and Systems
CSC 541 - Advanced Data Structures
CSC 546 - Management Decision and Control Systems
CSC 555 - Social Computing
CSC 562 - Introduction to Game Engine Design
CSC 565 - Graph Theory
CSC 568 - Enterprise Storage Architecture
CSC 579 - Introduction to Computer Performance Modeling
CSC 580 - Numerical Analysis I
CSC 582 - Computer Models of Interactive Narrative
CSC 583 - Introduction to Parallel Computing

Choose Search Criteria

Keywords: data science
Semester: Any
core project fieldwork

Figure 13: User searches by domain

7. EVALUATION

7.1 Evaluation Method

We used Beta Testing[14] as an evaluation measure for our system. Beta testing is also sometimes referred to as user acceptance testing (UAT) or end user testing. In this phase of software development, applications are subjected to real world testing by the intended audience for the software. The experiences of the early users are forwarded back to the developers who make final changes before releasing the software commercially. We invited users to test our application. They were given a specific set of instructions about what the new functionalities were and how they Once they completed testing, testers filled the evaluation form which asked some basic questions about the functioning of the application. Questions were based on ease of use, correctness of results, understandability of the README etc. The evaluation form and responses are covered in detail in the next section.

According to the feedback from the teaching staff about the last report, we decided to modify the type of questions we ask the users while they evaluate our system. We focused on making a questionnaire that does not guide the users of the system to any particular response and keeps all options equally tempting, which ensures that we get honest feedback.

7.2 Questions and Results

The questions which we asked for our evaluation are as follows:

1. **How much would you rate the new Keyword Search module (after integrating text analysis)?**

   The users of the system were given a brief idea about what the module does and then were asked to use the module. Then we asked them to rate their experience using this new search module in terms of the quality of implementation, smoothness of flow, results obtained, etc. We found that almost everyone liked the new search module after integrating text analysis.

2. **How much would you rate WolfPal 2.0?**

   We asked the users to rate how they found WolfPal 2.0 according to ease of use and functionalities.

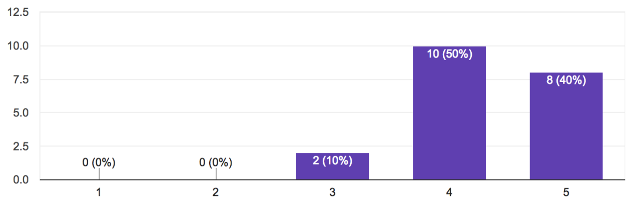


Figure 14: Response for Question 1

The results showed that more than half the users rated it 5 out of 5. Even though this were true, there were 7 out of the 20 evaluators who liked the system but did not find it perfect and hence gave a 4 out of 5. We can see this in a little more detail in the following sections.

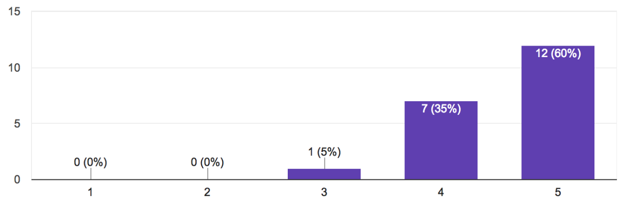


Figure 15: Response for Question 2

3. **How did the new Upload Syllabus functionality work for you?**

   Here, we asked the users to rate the new upload syllabus functionality that we added to the existing system. The options kind of gave a variety of options and possible reactions users could have towards the new feature - ranging from ”Great addition, well executed!” to ”Bad implementation, bad idea!”.

   Fortunately no one thought it was a bad idea and a bad implementation. People actually liked the idea of adding syllabus files. Some of them felt the implementation could have been better.

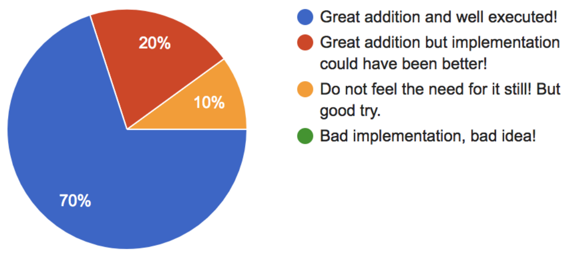


Figure 16: Response for Question 3

4. **How did you find the course retrieval from NCSU Course Catalog feature?**

   This question aimed at understanding how the dynamic course retrieval feature fared among the users. Again the options were made as varied as possible to make sure there was no bias in the questions and the options too.

To our surprise, 100% of the users were really happy seeing additional courses and data on the platform

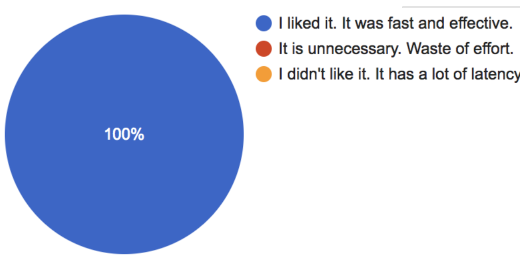


Figure 17: Response for Question 4

5. **Consider yourself as an incoming student. Do you think this platform can be used to connect with relevant peers?**

   This was kind of the concluding question to get a general overview about the system and leaving thoughts about the general sentiment about the platform in the mind of the user.

   We found that there was no one from the 20 evaluators that had an average or below average feeling about the system.

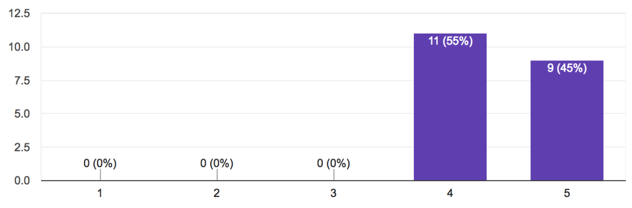


Figure 18: Response for Question 5

7.3 Improvements Suggested After Evaluation

The following are some of the valuable improvements which were suggested by users who evaluated our web application:

1. **The file upload functionality allows uploading of pdf files only. It should also support other file types**

   - The primary motivation to add the file upload functionality in our system was to provide the students an option to share course syllabus with others.
   - As the course syllabus are generally shared as pdf files, we limited our functionality to pdf files.
   - If needed, the functionality can be extended to other file types in future.

2. **Give the users a button so that they can fetch the latest course details from NCSU Course Catalog whenever they want to while using the application.**

   - Ideally, the web application should contain the functionality, but the time taken for the script to fetch the details refrained us from integrating that feature in our module.
   - The NCSU Course Catalog is designed in such a way that there is a popup for each course and the details are present in the popup. So the script needs to click on every popup to retrieve all the details, which takes up a lot of time.
   - By adding a button to extract the features, the user will have to wait for a long time to get results after clicking the button, which is undesirable.

3. **Do not restrict the courses only to CSC department. Students can choose courses from multiple departments.**

   - Our system can be easily expanded to cover all courses of the University. It is a one of the useful future scopes of this project.
   - We restricted the courses to CSC department due to time constraint as our primary objective was to expand the system functionalities across the depth, and not breadth.

8. FUTURE SCOPE

- **Increase the scope of the project in terms of courses covered**

  The application now caters to students of the CSC Department of North Carolina State University. It could be expanded so as to include more departments within NCSU and also to more universities. The application could also include MOOCs from popular platforms such as Coursera and Udacity.

- **Fetching more personalized course recommendation**

  It course recommendation system could be made more personalized. It is a known fact that Resume is a document which tells what the person's interests are. It gives an idea as to what skillsets the person possesses. So if an module is made in WolfPal which parses a person's resume, it can then be used to fetch the person's skill sets among others which can then be used to map it along with the Topic Modelling map, thereby suggesting courses which are more relevant to her/him.

- **Adding professor as the user**

  The application can have a specialized view for the professors using which they can update the course details. They can also get an estimate of students who are planning to enroll in the course and this estimate can be used to plan the allotted rooms based on the space required and sessions to the course. The professors would also be able to help other user through the forum. An interactive forum which includes the professor as a user can be a boon to all students.

- **Including the syllabus to build a better Topic Modeling algorithm**

The issue that we faced is that we could not use more sophisticated topic modeling algorithms like latent Dirichlet allocation algorithm because it requires more data. It wasn't readily available to us because we used the course descriptions. However, if syllabus for each course was available, then LDA could be used, since the amount of text would be large. Thus, having a more efficient and robust topic modeling technique.

- **Integration with Moodle and MyPack**

  Now the application scrapes the course catalog of NCSU, which doesn't give a comprehensive information regarding the course. If a way could be found to access the Moodle and MyPack system then the application would have access to more details of the course such as the course description, past courses of the student, assignments, among other academic related information.

- **Rating and Feedback**

  Allowing the users to rate and give feedback to the courses they have taken, this will allow potential takers make better judgements of whether or not they want to take that course.

9. CONCLUSION

WolfPal is a single go-to application for student who want to get information on courses offered by NC State University. WolfPal provides a variety of options for the student to decide which courses he/she should take. Student can get course information based on name, keyword search. Additionally, if the student does not know the courses offered by the university, he/she can search by keyword, and WolfPal will give a list of courses which will fit his/her area of interest and also satisfy the breadth requirements for the major. The keyword search is more advanced with a context-based approach, which uses a Topic Modeling approach that makes it more efficient. Thus, WolfPal is a resourceful standalone application for students who face difficulties selecting courses.

10. REFERENCES

[1] WolfPal Report *WolfPal CSC 520* Retrieved from: `https://github.com/ragarwa7/WolfPal/blob/master/Reports/team-k_wolfpal_mar_report.pdf` [Accessed: 21-Mar-2018].

[2] R. Bhatt, D. Desai, L. Shi, and C. Zhao. *Course feedback system. Software Engineering, 2017.* Retrieved from: `https://github.com/Rushi-Bhatt/SE17-Team-K-CourseFeedbackSystem/blob/master/Final_Report_MAY.pdf` [Accessed: 21-Mar-2018].

[3] Al-Badarenah, Amer, and Jamal Alsakran. *An Automated Recommender System for Course Selection.* International Journal of Advanced Computer Science and Applications 7.3 (2016): 1166-175.

[4] Agile Methodology *What is Agile?* Retrieved from: `https://www.cprime.com/resources/what-is-agile-what-is-scrum/`[Accessed:21-Mar-2018]

[5] Code Review Practices *Code Review Wikipedia* Retrieved from: `https://en.wikipedia.org/wiki/Code_review`[Accessed:21-Mar-2018]

[6] Pair Programming *Pair Programming Wikipedia* Retrieved from: `https://en.wikipedia.org/wiki/Pair_programming`[Accessed:21-Mar-2018]

[7] Python Documentation *Python* Retrieved from: `https://www.python.org/about/` [Accessed: 26-Mar-2018]

[8] NLTK Documentation *NLTK* Retrieved from: `https://www.nltk.org/install.html` [Accessed: 26-Mar-2018]

[9] JavaScript Documentation *JavaScript Documentation* Retrieved from: `https://en.wikipedia.org/wiki/JavaScript` [Accessed: 26-Mar-2018]

[10] Ruby on Rails Tutorial *Ruby on Rails Guides* Retrieved from: `http://guides.rubyonrails.org/`[Accessed:26-Mar-2018]

[11] Bootstrap Tutorial *Bootstrap W3Schools* Retrieved from: `https://www.w3schools.com/bootstrap/bootstrap_get_started.asp`[Accessed:26-Mar-2018]

[12] SQlite3 Documnetation *SQlite3 Documentation* Retrieved from: `https://www.sqlite.org/docs.html`[Accessed:2-Apr-2018]

[13] Selenium Wikipedia Information *Selenium Wikipedia* Retrieved from: `https://en.wikipedia.org/wiki/Selenium_(software)`[Accessed:1-Apr-2018]

[14] TechTarget *What is Beta Testing?* Retrieved from: `http://whatis.techtarget.com/definition/beta-test` [Accessed: 21-Mar-2018]

11. CHITS

1. GTT
2. GLO
3. KQE
4. SFV
5. VRS
6. LHT
7. SMJ
8. STW
9. MZA
10. KHZ
11. RXH
12. XUC
13. ZEC
14. PPN
15. BHC
16. VJK
17. HGW
18. SUZ
19. MOS
20. JBS