FYP Final Report

Goal Quest

Final Year Project Report

by

Bilal Waheed

Malik Muhammad Idrees

Muhammad Nafees Tariq

In Partial Fulfillment

Of the Requirements for the degree

Bachelors of Engineering in Software Engineering (BESE)

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan

(2016)

DECLARATION

We hereby declare that this project report entitled "GOAL QUEST" submitted to the "SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (SEECS)", is a record of an original work done by us under the guidance of Supervisor "DR. ARSALAN AHMAD" and "DR. ALI TAHIR" and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

Team Members	**Signature**
Bilal Waheed	______________
Malik Muhammad Idrees	______________
Muhammad Nafees Tariq	______________

Supervisor:	**Signature**
Dr. Arsalan Ahmad	______________
Dr. Ali Tahir	______________

Date:

30th May 2022

Place:

SEECS, NUST

TABLE OF CONTENTS

ABSTRACT

A lot of people struggle when it comes to finding the right information that can help them to achieve their professional goals. They don’t know the right skills they need to learn to land their dream job. Or even if they know the skills, it is very difficult to find the right content that they can learn in a reasonable amount of time. This is because there is tons and tons of information on the internet, but finding the right information is like finding a needle in a haystack (which might be possible, but the amount of time and energy that will be spent on doing so is probably not worth it).

We aim to provide a system/platform where users can enter the IT-related skill that they want to learn or a job that they want to land. The system will generate a roadmap for the users that they can simply follow to achieve that goal. The system will take in the user’s information like what is the goal that they want to achieve right now, and whether they’re just learning a skill for the sake of learning or are learning to land a job. This information will be used to generate the content for the roadmap. Users will also tell about their expertise level related to that skill and the amount of time that they have to achieve that goal. This information will be used to personalize the content of the roadmap for each user, thus providing them with a roadmap that is specifically designed for them. Such a personalized roadmap will help them to get to that goal in a very reasonable amount of time.

The user will provide a keyword for the job that they want to get like the title of the job. That keyword will be used to get the list of related skills from job portals related to that job. This list will be generated by scrapping job posts from job portals and then using Natural Language Processing (NLP) on those scrapped job posts to get those keywords. And then the system will get content against those keywords from the internet.

Our application can be used by any of the more than 7 billion people who want to learn an IT-related skill or land an IT-related job. There is 1 existing system out there that does this, but the content that they provide is hard-coded, rather than automatically generated from the internet. Our system on the other hand generates content from the already existing content on the internet, which just needs expert

approval to be cleared for use by our users. This makes our system highly scalable and robust. Right now, our system only provides roadmaps for IT-related fields, but since our system is very scalable, we can use this model to generate roadmaps for other fields as well.

Chapter 1

INTRODUCTION

Access to the right information is very necessary to make any decision. Whether it's about a career or any other decisions in life, the more information that they have, the better the decision they'll be able to make. Especially when it comes to career, career counselling is very important. This lets people decide (and often helps to find out) what they love and want to do for the rest of their life or for some portion of their foreseeable future. While on one hand career counselling is important because it helps you determine what you would want to do professionally, there is still another very important thing that is often overlooked.

1.1 UNMET NEED OR PROBLEM:

A lot of people often struggle when it comes to finding the right information that they need to achieve their career goals. Career counselling helps to narrow down or decide what people would like to do for a professional career. But people still need a lot of information about that career path. Some of this information is provided by career counselling departments while some of it is expected to be learnt by the people themselves by watching YouTube videos or by just Googling about it. Since there are so many resources available out there and most of them recommend different/slightly different/different versions of the same thing, it becomes difficult for beginners to navigate through that information and find out exactly what they need to focus on. They don't know the skills that they need to land their dream job. Even if they know, it is not very easy to find the right content to learn those skills in a reasonable amount of time.

The number of internet users have grown in the world from 413 million in 2000 to 4.7 billion in 2021. The boom of the internet has brought about a lot of changes and lots and lots of benefits. Surely it has made learning much easier and accessible. But this has raised a new problem. Now that the majority of the people have access to the internet, anyone can upload almost anything on the internet. This makes it very difficult for people to be sure of the correctness of any piece of information. That's why, there is a very dire need for a system where people can find all the answers in one place related to one specific thing i.e., in this case, the skills that they need to learn and the content to learn those skills in a reasonable amount of time.

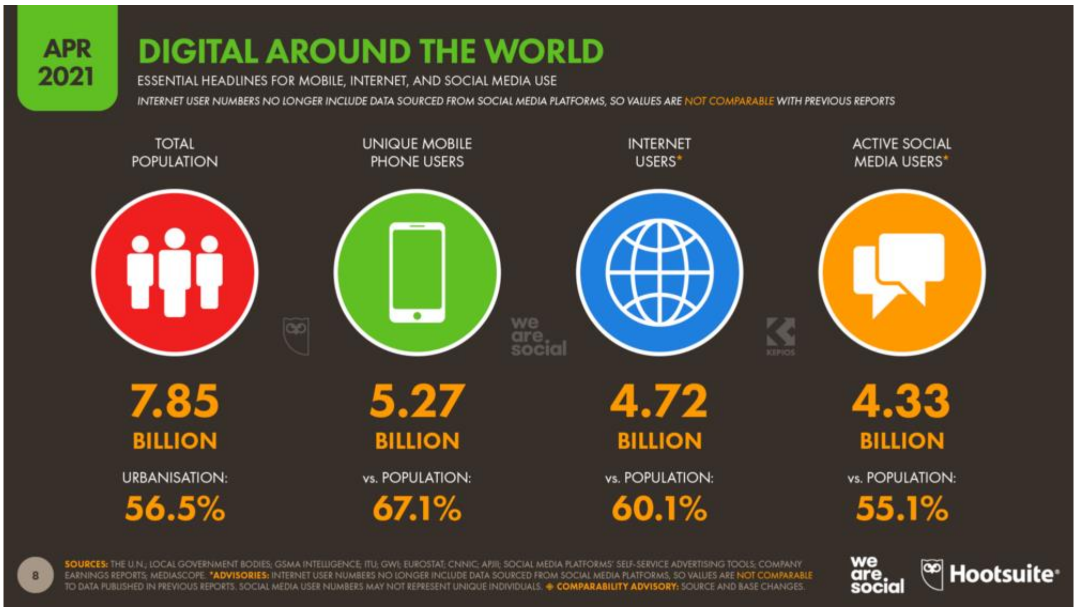


Figure 1: Digital Around the World

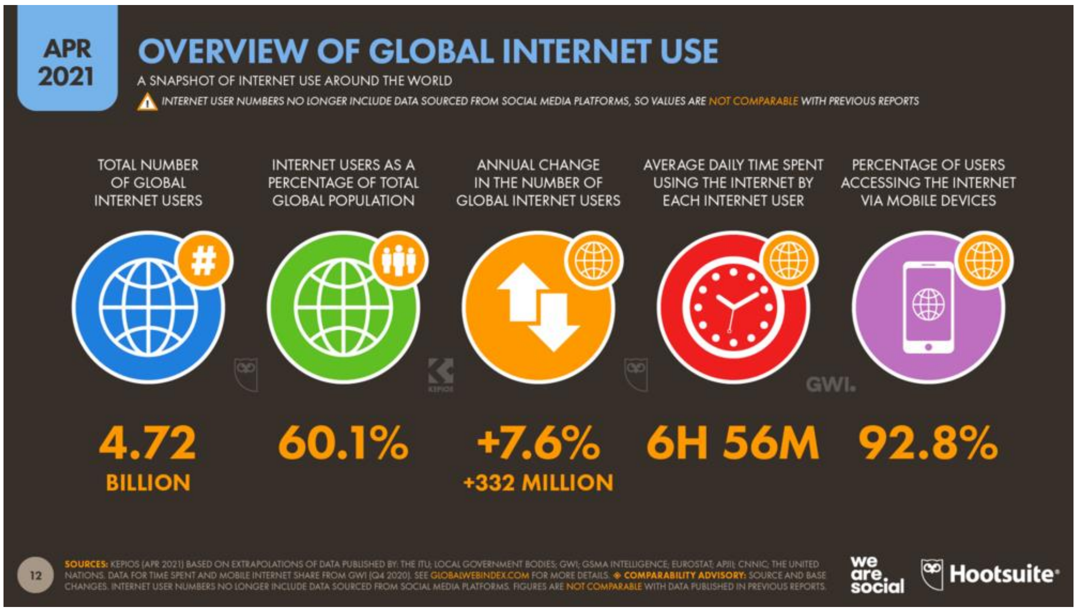


Figure 2: Overview of the Global Internet Use

1.2 WHO NEEDS IT:

Anyone of the 7.9 billion people in the world who has a career goal that he / she wants to achieve or a skill that they want to learn, they can use our application to do so. The base of internet users has greatly expanded over the last 20 years. As of April 2021, 60.1% of the world population has access to the internet. Indeed, not all of them use the internet for training or learning purposes, but it is enough to mention that the worldwide market of online learning is expected to exceed 243 billion U.S. dollars by 2022 compared to 46.67 billion U.S. dollars back in 2016. This can give us an insight into the potentials of learning online and how huge the market is.

Chapter 2

LITERATURE REVIEW

Two major modules of our system are to gather the data from different sources, and to process that data to produce useful content for the users. Our research and study were primarily focused on these two things.

Collection of useful data for machine learning systems can be a tedious task because these systems require large amounts of data for training, often in order of thousands. The more training data we provide to the system, the higher is its performance and accuracy. Web scraping can be employed to collect these large amounts of data. From our research, we found that the data can be gathered through web scraping from websites that are available for public for free. Since different websites have different underlying structures, it is still quite a challenge to come up with a scraping system that is generalized. According to other articles, we can make an automated and a generalized web crawler that would extract the useful data from websites. Navigation of the website is the key in such systems as it is different for websites.

The system would be focused on gathering data that is most relevant for the users, thus saving his time. So, we can use some criteria to determine this criterion. According to a paper, this criterion can be the number of shares of a particular article on social media platforms.

Second major module of the system provides useful content that is also gathered from different sources. The content that is provided to the users must be relevant for each one of them. This relevance of data can be determined using a multi-class classification model of machine learning. The gathered data can be fed to the model, and it determines how relevant it is for a user. As the users can have different levels of expertise in a specific field, the system can provide data according to that level. The system will then use API calls to populate that date on our website.

Chapter 3

PROBLEM DEFINITION

There are many people who want to learn one thing or another. This can be because they want to land a job and that job requires that skill or maybe they just want to learn that skill for the sake of learning. Whatever the reason might be, we can safely say that almost everyone has something to learn.

To learn something, first people need to know what they need to learn. If somebody wants to land a job and they want to learn the required skills for that job, they first need to know which skills would be required for that job. For this, they can do a Google search but finding this information on Google is very time consuming. We asked different people about this in a questionnaire and 71.7% of the respondents said that finding the right information on the internet is difficult or time-consuming. This is one part of the problem.

Now, let's suppose if the people know what skill they need to learn, it is very difficult to find the right content to learn that skill. Some of the content can be found on YouTube, Google, or some course website like Udemy or Coursera or LinkedIn Learning but using these options to learn something properly still take a lot of time. Time is a very important factor here. If somebody is learning something to get a job, it is very likely that they will not have a lot of time to spend just on learning that thing. Or even if they have the time, why spend more time on something that can be done in significantly less time?

So, these are the two main problems that we're trying to solve here:

1. Find and provide the list of all the skills that the user needs to learn in one place so that they don't have to scour the internet just to find what they need to learn.
2. Once the user knows what skills they need to learn, provide them the best top-notch content so that they can learn in the best possible way.
3. Help users to learn these skills in the best possible time by providing personalized courses for them.

Chapter 4

METHODOLOGY

The system was developed using a combination of both waterfall and agile development methods. Functional and non – functional requirements i.e., the features of the system were finalized in the early stages and were not changed later. However, system architecture and design were somewhat changed later.

4.1 TOOLS AND TECHNIQUES

As the system has three major modules, namely frontend, backend and the content gathering and processing modules, these are developed separately, using different libraries and programming languages.

Frontend is developed using React, a frontend library. React is one of the most widely used frontend library because of its powerful features such as code reusability and reactive webpages, providing exceptional loading speeds.

Backend is developed using Django, a python framework. Django is a specialized framework for web development, based on the model-template-view pattern. Django makes it much more efficient and easier for developers to create secure and maintainable websites.

Content gathering module, incorporating NLP, is written using Selenium and Spacy, both of which are python based. Selenium is an automation framework that can be used for web scraping as well as automating repetitive web tasks on the web. It is a very powerful tool and well suited for our web scraping requirements. Spacy is an NLP library that comes with some pretrained models for NLP tasks and supports pipelines to process the data as we wish. It has an extensive range of functions that help in performing NLP tasks such lemmatization, stop word removal, entity recognition, feature extraction and so on.

The Selenium and Spacy part of our solution works as follows. We have developed Selenium web scrappers that pick-up job descriptions from different job search platforms. The next step is the extraction of useful information from these job descriptions. This information is then stored and used once again from different scrappers. These scrappers are made for YouTube and Google to gather learning resources for the users. The resources are then shaped into a course which is then stored in the database against the user for which it was created. This course is personalized differently for different users, based on their goals and experience.

4.2 WORKFLOW/USAGE OF THE SYSTEM

The usage or workflow of the system corresponds to the above-mentioned modules of the system.

1. In the very first step, the user registers with the system and to use the system he logs in the application.
2. The next step is the creation of the user profile where the user enters some basic information about himself.
3. Next, to create a new course, the user enters some details in the form and submits it. These details include his goal, meaning what he wants to learn, his expertise level, whether he is beginner or an expert, whether his aim is to get a job, and lastly the total time he has to complete the course.
4. After the submission of the details, the backend modules run and the course for that particular user is displayed to the user so he can start learning. More about the backend process is discussed in the system design section.
5. The user can track his progress by marking the topics as “completed”. Moreover, the user can take notes for every topic.

This complete process can be visualized from the diagram below.

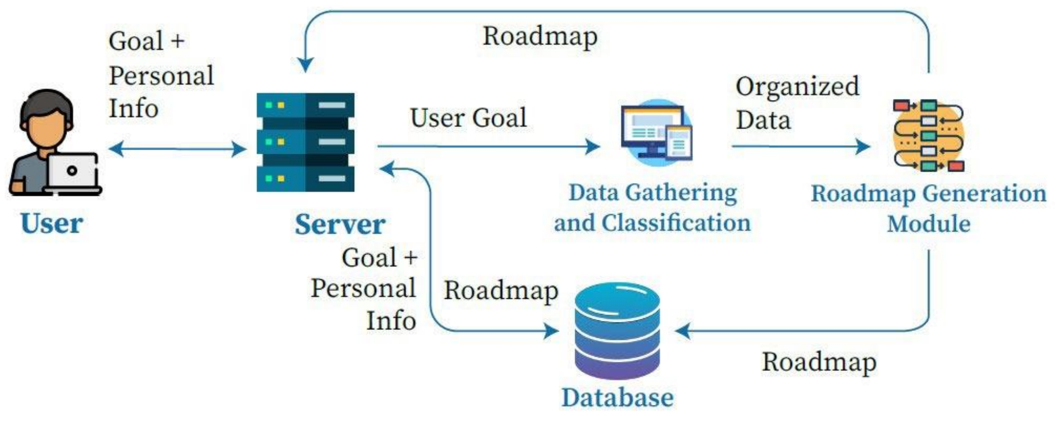


Figure 3: Workflow of the system

Chapter 5

DETAILED DESIGN AND ARCHITECTURE

5.1 ARCHITECTURE

5.1.1 Architecture Design Approach

The major responsibilities that the system must undertake include user interaction i.e., the user must be able to input the required data into the system, and also to receive data back from the system, that is generated based on the user's input. For this an easy-to-use interface is required in the frontend**.**

Secondly, we need a server side as well, that will process the user input and store the necessary information in the database to make it persistent. The server side will be responsible for handling all the user requests and their responses.

Thirdly, we need to collect the data from web using web scrapping techniques and then it must be processed using natural language processing to generate useful content.

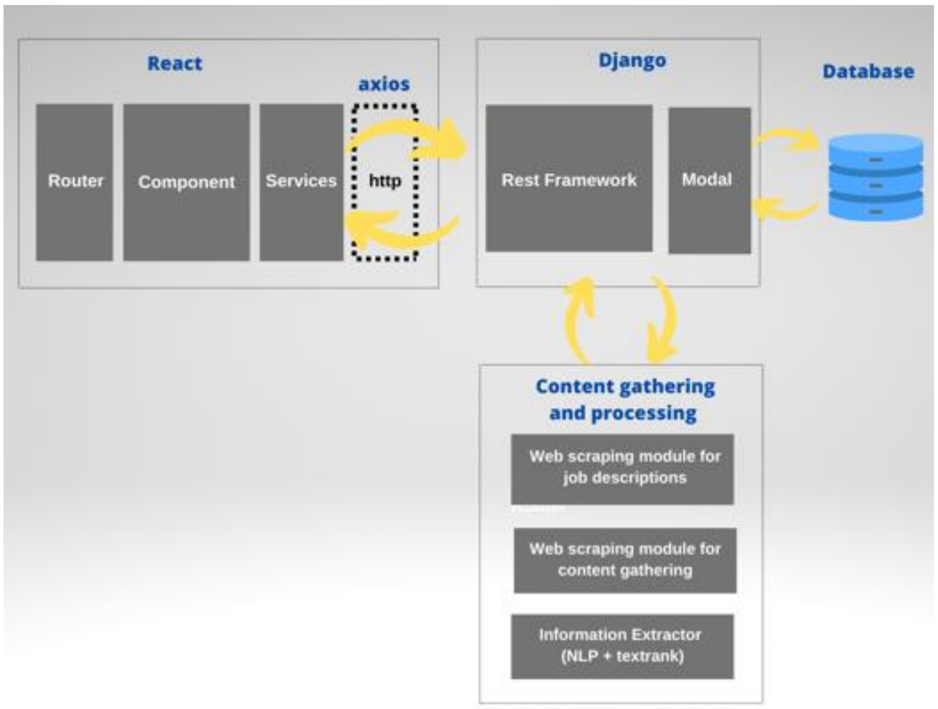

React
axios
Router
Component
Services
http
Django
Rest Framework
Modal
Database
Content gathering and processing
Web scraping module for job descriptions
Web scraping module for content gathering
Information Extractor (NLP + textrank)

Figure 4: System Architecture

5.1.2 Architecture Design

Keeping the roles and responsibilities of the system mentioned above in mind, the system was broken down into three top level components.

One component is the **frontend**, that is responsible for user interaction, that is taking user input and providing the user with the necessary response in a very elegantly design user interface.

The second component is the **backend**, comprising of the server and a database. The backend will deal with the server requests and responses along with triggering the third module and receiving the data from it and storing it in the database.

Thirdly, we need a **component for data gathering and processing of the gathered data**. This module is divided into two modules: one module for the gathering of the data that will comprise of web scrappers and the other module for processing this data to extract the useful information using natural language processing (NLP).

5.1.3 Subsystem Architecture

It might be easier to explain the relationship between the components and the collaboration among them in the sequence of operation of the system.

Firstly, the user will interact with the system using the frontend. The user will provide the data using forms.

Then the frontend will send that data to the backend using API requests. At the backend, the necessary of the request will be stored in the database. For example, if the user is signing up for the first time, then his data will be stored in the database at this point.

The backend will interact with the third module mentioned above by providing the data from the user including his goals and experience etc.

The gathering and processing module will send the data back to the backend which will store it in the database and send it back to the user at frontend.

5.1.3.1 Components in each subsystem

The three subsystems aforementioned can be decomposed further as discussed here with the help of diagrams.

1. Frontend subsystem
    - ❖ Landing/Main
    - ❖ User Registration and Login
    - ❖ Dashboard
    - ❖ Roadmap/Course
2. Backend subsystem
    - ❖ User Registration and login
    - ❖ User Profile
    - ❖ Courses/Dashboard
    - ❖ Course Details
3. Content gathering and processing subsystem
    - ❖ Web scrapper for job descriptions
    - ❖ Information extractor
    - ❖ Web scrapper for content gathering

5.1.3.2 Data Flow Diagrams (DFDS) for Subsystems

1. Frontend Subsystem

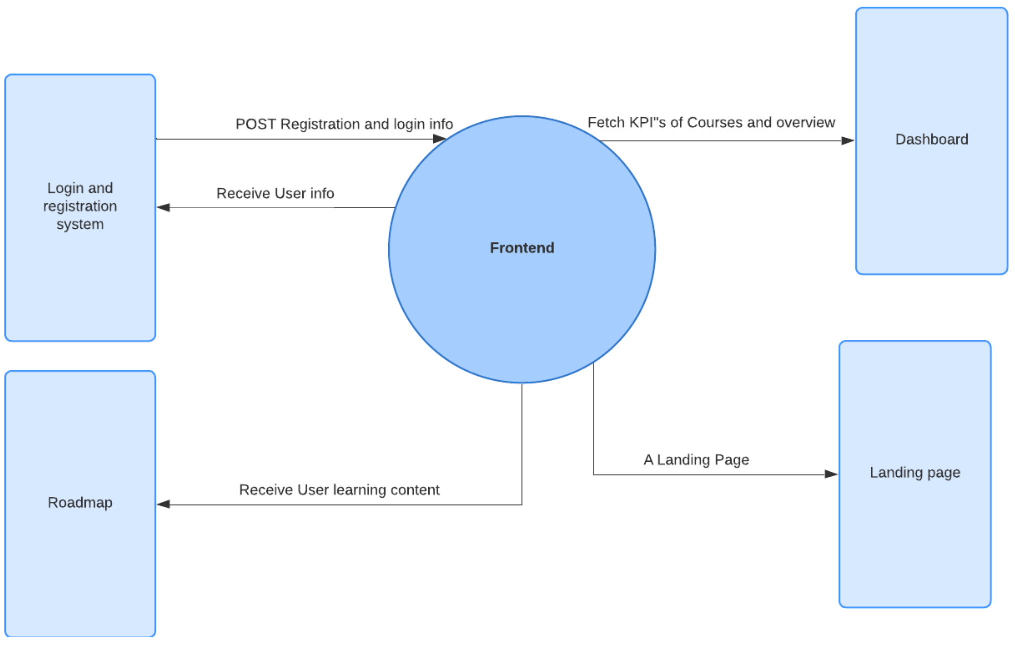

Login and registration system
POST Registration and login info
Receive User info
Frontend
Fetch KPI"s of Courses and overview
Dashboard
Roadmap
Receive User learning content
A Landing Page
Landing page

Figure 5: Frontend DFD

2. Backend Subsystem

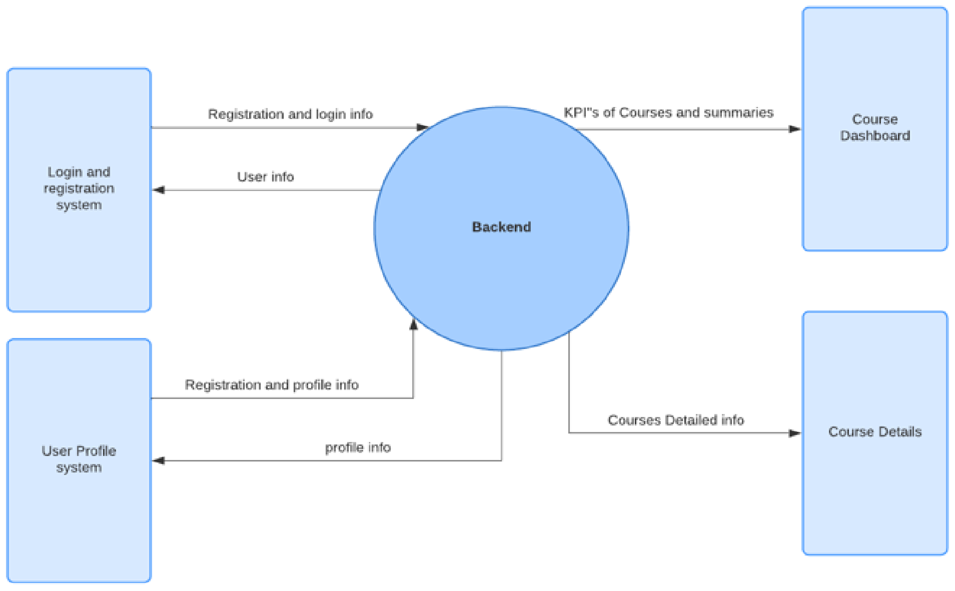

Login and registration system
Registration and login info
User info
Backend
KPI"s of Courses and summaries
Course Dashboard
User Profile system
Registration and profile info
profile info
Courses Detailed info
Course Details

Figure 6: Backend DFD

3. Content gathering and processing Subsystem

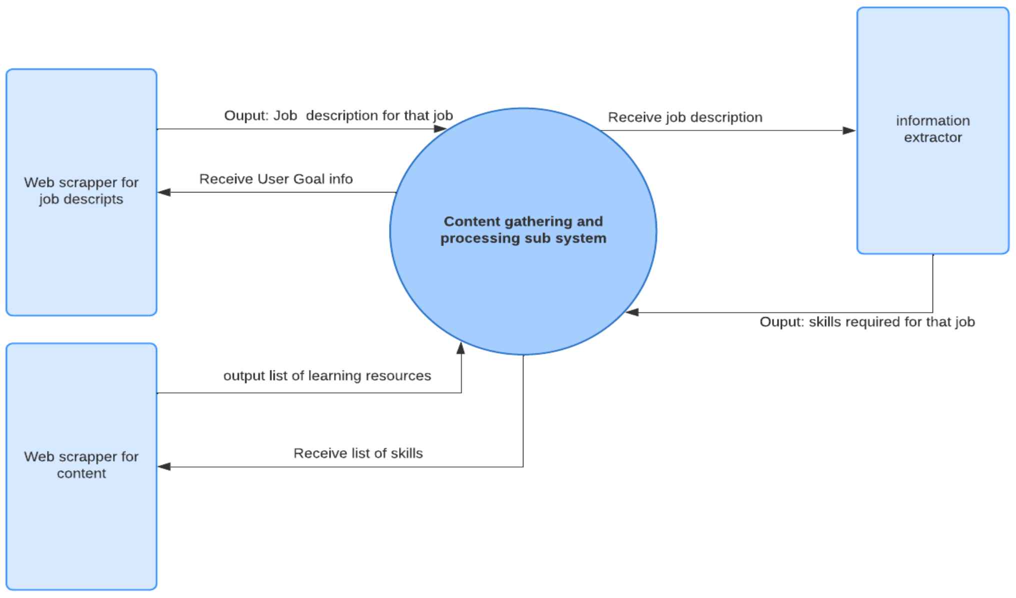

Web scrapper for job descripts
Ouput: Job description for that job
Receive User Goal info
Content gathering and processing sub system
Receive job description
information extractor
Ouput: skills required for that job
Web scrapper for content
output list of learning resources
Receive list of skills

Figure 7: Content Extraction and Processing DFD

5.1.3.3. Rationale for this decomposition

This decomposition enabled our team to develop and test the individual subsystems separately. This made the process more efficient, smooth, and easy to maintain. This decomposition results in separation of concerns as well. It becomes much easier to integrate the subsystems and test the system end-to-end.

5.2 DETAILED SYSTEM DESIGN

Following section further divides the Architectural components into more details and adds a detailed explanation of everything included in each of the subsystems mentioned above.

5.2.1 Frontend Design

5.2.1.1 Classification

This component is a sub-system.

5.2.1.2 Definition

This is the front-end of the application. It is part of the view model of the application.

5.2.1.3 Responsibilities

This component is responsible for whatever the user sees on the website. In this component, we take care of the UI/UX of the application, and we try to make sure that the system looks as attractive as possible.

This sub-system is responsible for:

1. Presenting information in an attractive way
2. Presenting beautiful landing page
3. Presenting beautiful and user-friendly courses dashboard page
4. Presenting simple and beautiful user’s profile page
5. Making user stay on the website as long as possible

6. Attracting new users

5.2.1.4 Constraints

1. Every user who visits the website has their own idea of what a good website should look like. It is very difficult to make a website that is equally pleasing for everyone. So, we have to go with the design that is likely to appeal to the majority of our intended audience.
2. Browser compatibility can be a big constraint. Although most modern browsers would perfectly support our browsers, it might not run perfectly fine on older browsers.

5.2.1.5 Composition

This sub-system is composed of 4 different pages:

1. Landing Page
2. User Profile Page
3. Course Dashboard Page
4. Course Page

5.2.1.6 Uses/Interactions

This component is responsible for the look and feel of the website. For a user, front-end is the most important part. How the information is presented, how optimized the user-experience is, how good does it look on different browsers and different devices and how does it make the learning journey easier by providing a pleasing design is very important for users.

This sub-system only interacts with:

1. Users
2. Back-end
3. Database

5.2.1.7 Resources

This sub-system needs internet connection, a device such as computer/laptop or mobile/tablet. It also needs to be able to connect to the back-end and database to be able to provide interactivity and content to the users.

5.2.1.8 Processing

This module works by getting the content from the back-end servers and database. This content is then displayed on the browser in users' device.

5.2.1.9 Interfaces Exports

Landing Page

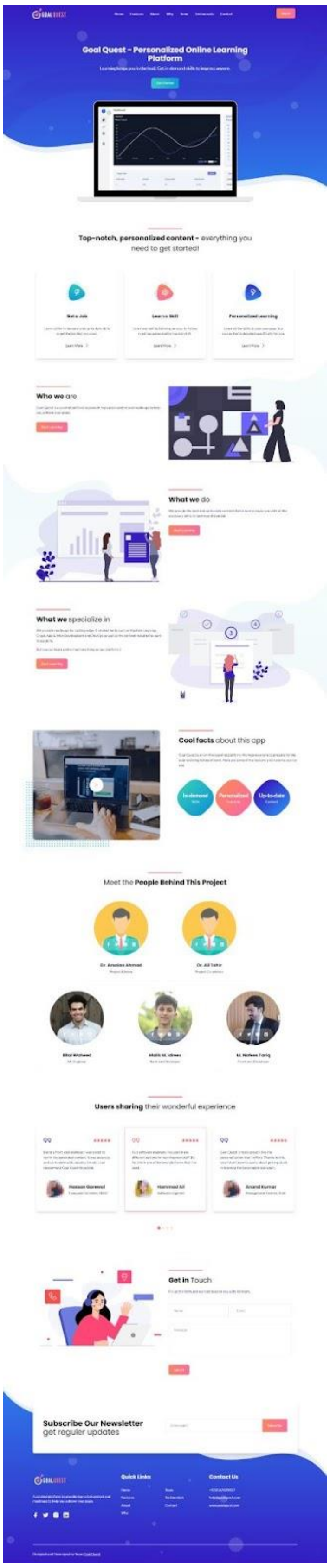


Figure 8: Landing Page Design

Sign Up Page

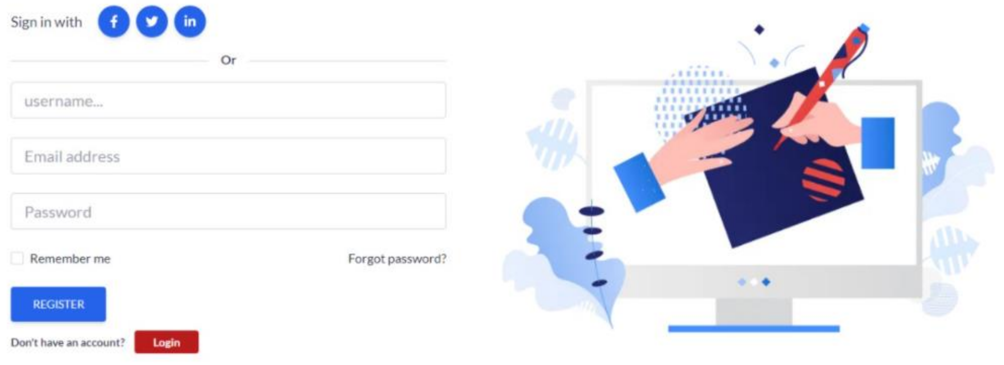


Figure 9: Sign Up Page Design

Sign In Page

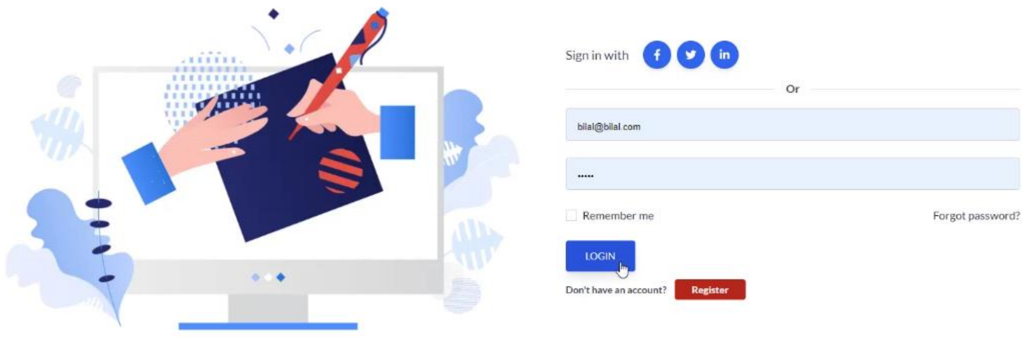


Figure 10: Sign In Page Design

Profile page

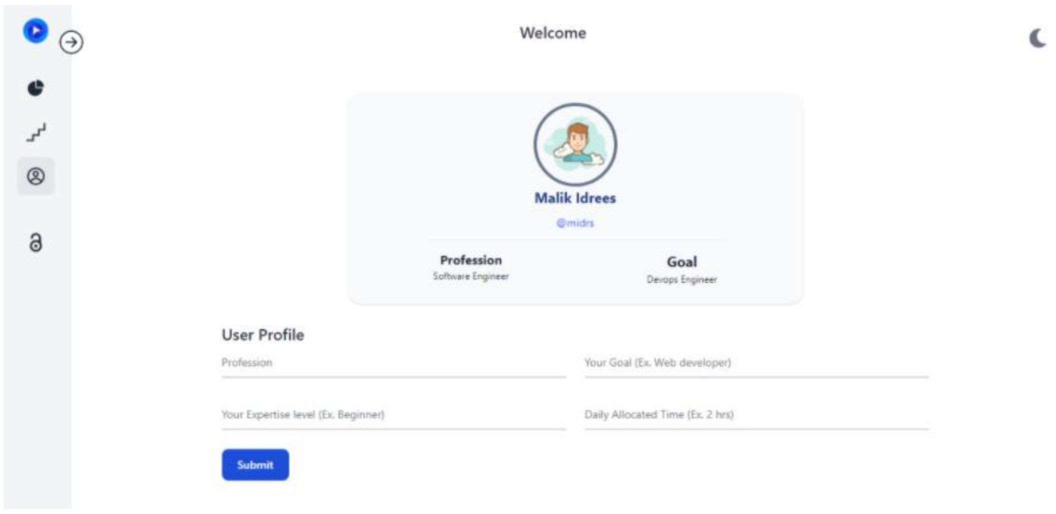


Figure 11: User Profile Page Design

Courses Dashboard Page

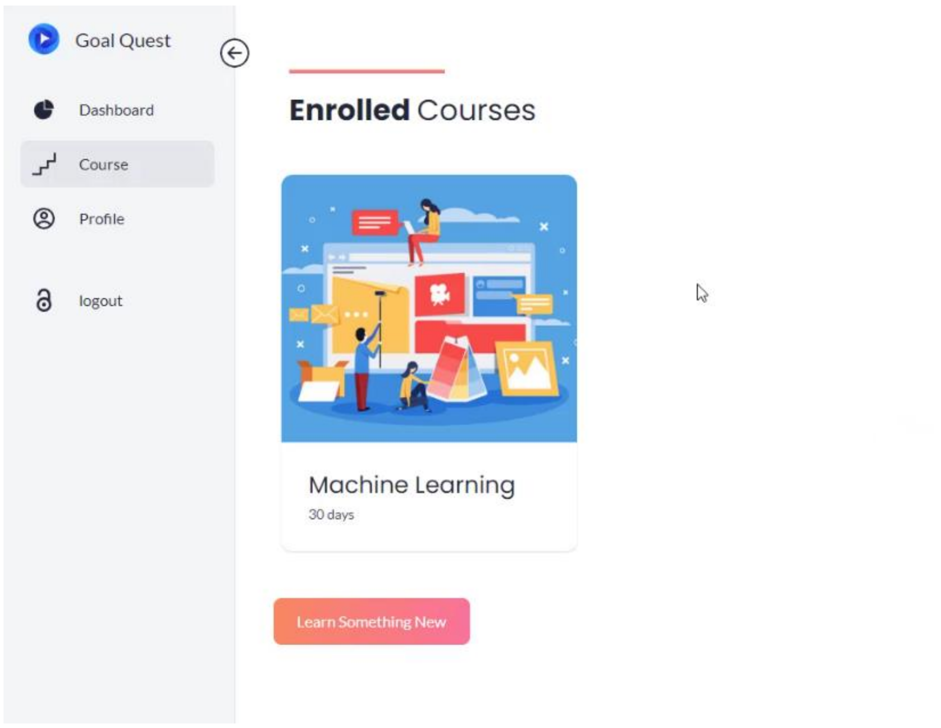
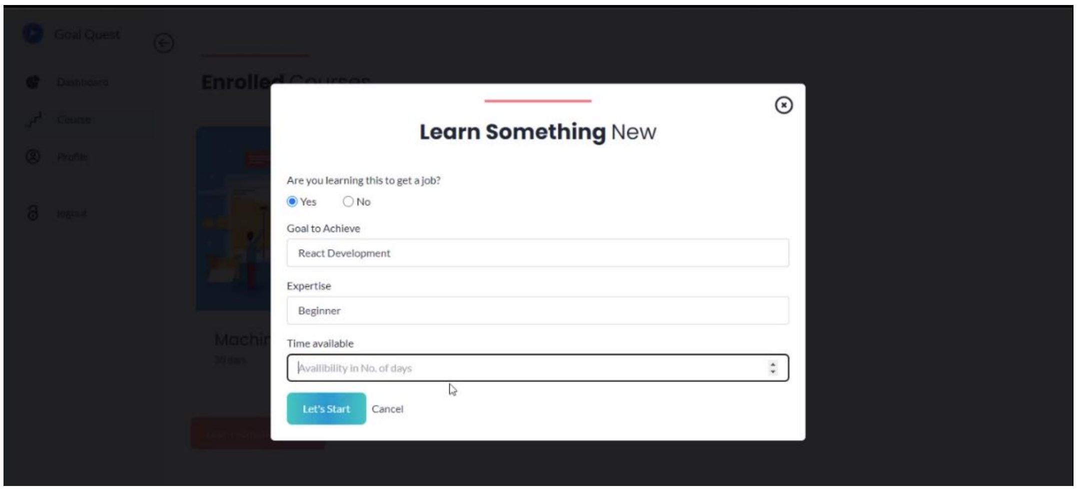


Figure 12: Courses Dashboard Page Design

Course Sign up Modal



Learn Something New

Are you learning this to get a job?

[✓ Yes] No

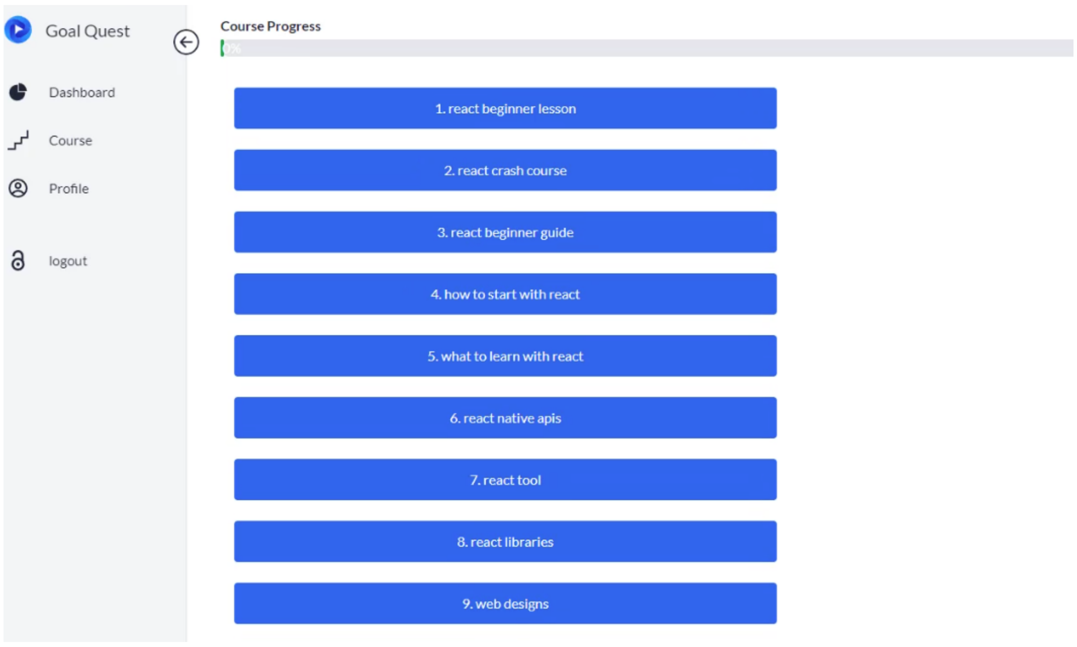
Goal to Achieve: React Development

Expertise: Beginner

Time available: Availibility in No. of days

Figure 13: Course Creation Modal Design

Courses Details Page



1. react beginner lesson
2. react crash course
3. react beginner guide
4. how to start with react
5. what to learn with react
6. react native apis
7. react tool
8. react libraries
9. web designs

Figure 14: Course Page

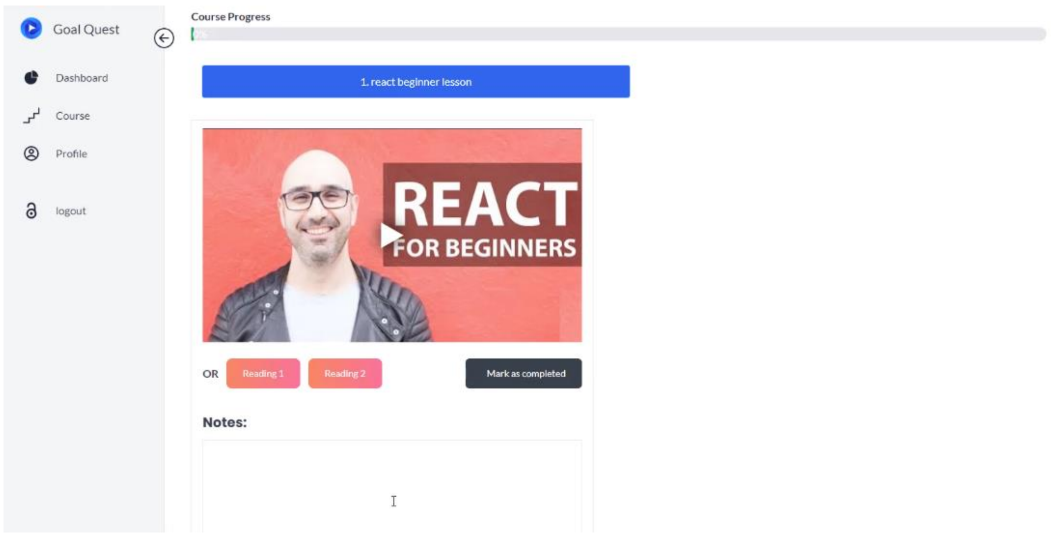


Figure 15: Course Content

5.2.1.10 Detailed Subsystem Design

Subsystem design of this component has been mentioned under the "Subsystem Architecture" heading.

5.2.2. Backend Design

5.2.2.1 Classification

This is a subsystem representing the complete backend of the application.

5.2.2.2 Definition

This is where our system's logic is defined, and all the data related queries are handled. Be it providing it to frontend or passing as well as receiving it from the NLP interface.

5.2.2.3 Responsibilities

Operations that are the responsibility of this component are:

- Provide data when queried from frontend.

- User management

- Password hashing

- Access to admin dashboard

- Generate content for user with help of NLP interface

5.2.2.4 Constraints

This component has the following constraints:

- An active internet connection.

- The database in operational form

- Operational NLP module.

- IP (Internet Protocol) address rotation

5.2.2.5 Composition / Detailed Subsystem Design

The API is developed using Django Rest Framework. All the endpoints receive and send data in json format. We have protected routes in our system, and we have utilized JWT (JSON Web Tokens) to verify authentication. As an added security measure, we have limited the lifetime of access tokens issued using JWT and provide a way to refresh it through refresh tokens to prevent masquerade as well as man in the middle attacks.

5.2.2.6 Uses/Interactions

Our frontend which is based on React makes API calls to our backend and displays the data. All the communication between frontend and backend in done through API calls

5.2.2.7 Resources

A python based open-source framework called Django and Django Rest Framework. Postman for our API testing. An SQLite database during test development.

5.2.2.8 Processing

The moment a user tries to login our system becomes active. The data is received through an API call and our system generates appropriate responses.

5.2.2.9 Interface/Exports

This subsystem is where we have defined the logic of our system. It handles communication between all components namely frontend and NLP module. It accepts requests from frontend and responds accordingly. The internal working of a Django API can be understood from these components below:

Django MVT Architecture

Model-View-Template (MVT) design pattern is used by Django. As the name implies it consists of three parts: Models, Views, and Templates. Allowing us to separate responsibilities and thus creating flexibility.

Models: It is an interface to data. It defines our database schema and its relations. It comprises data-related essential fields and behaviours.

Views: They receive a request and return the response. They interact with databases through a model and return data. We define our business logic in these views.

Templates: is a presentation layer responsible for displaying static as well as dynamic content in rendered HTML files.

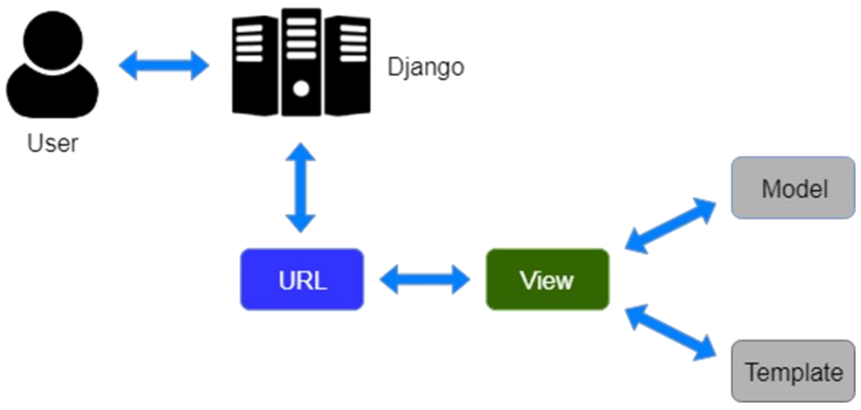


Figure 16: Django working

Serializers in Django REST

In Goal Quest, UI is handled with React — frontend, and Django is used to handle the data — backend. And the two are connected using REST API.

As we are using React to render our interface, we are not utilizing the Template part. And to provide JSON formatted data we changed the view to Django REST Framework

Now it becomes a Model-View-Serializer.

Serializer: It is responsible to render the data to a format that can make the data accessible at our frontend. We are using them to make our data available in JSON format.

5.2.2.10 Detailed Subsystem Design

This system has further four modules:

1. User Registration/Login
2. User Profile
3. Dashboard / Courses
4. Course

These components and their interactions have been discussed in the architecture section.

The detailed design of Django backend along with its working is depicted in the diagram below.

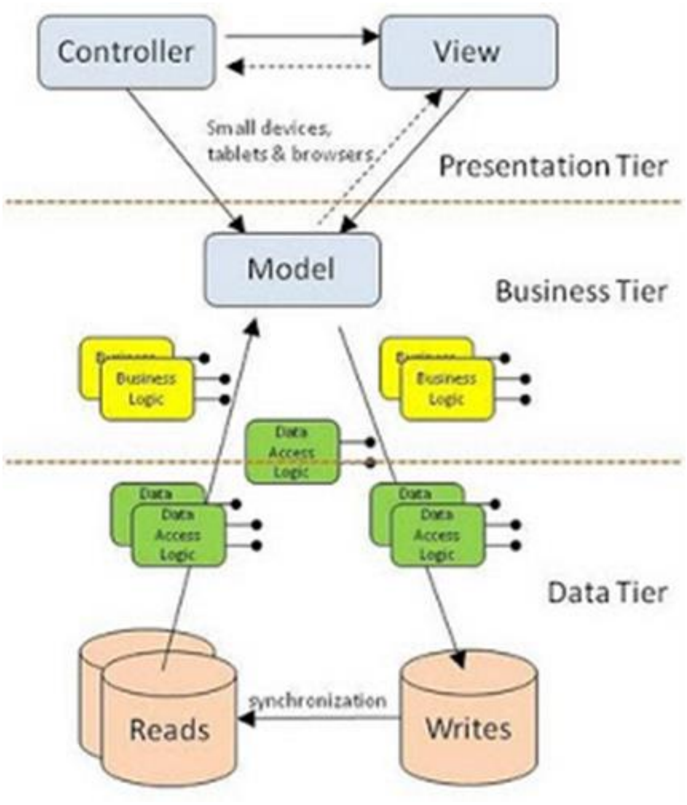


Figure 17: From making a request to receiving a response

5.2.3 Content gathering and processing Design

5.2.3.1 Web scrapper for job descriptions

5.2.3.1.1 Classification

This component is a *module*.

5.2.3.1.2 Definition

This is a web scraping module that collects job descriptions from Indeed.com and Monster.com both of which are job-search platforms widely used around the globe.

5.2.3.1.3 Responsibilities

This module provides the search query to the search bar in the webpages based on the user input and collects the job descriptions from the search results. This is an automated process. It collects and stores 15-20 job descriptions for further processing.

5.2.3.1.4 Constraints

The input to this module must come directly from the “goal” field in the user input. This input is of string type.

The output is of an array of strings where each string is a complete job description. This array is then passed as it is, to the next module for processing.

A constraint on this module is that the websites should be up otherwise no results will be gathered. It is assumed that if the websites are up, results will always be provided for valid queries.

5.2.3.1.5 Composition

This module is composed of two functions, one for Monster.com and one for Indeed.com. Both of them work exactly the same way with same inputs and outputs.

5.2.3.1.6 Uses/Interactions

The first interaction is the triggering of this module from the server. The server passes the search parameters for the module to operate.

The second interaction is between this module and the information extractor module. This module will pass the array of job description to the extractor module.

5.2.3.1.7 Resources

This module only needs memory and processor. The web scrapper has some processor requirements. The array of strings will be stored in memory until used by the next module. There are no database requirements.

5.2.3.1.8 Processing

The way that this module operates is as follows.

Firstly, from the parameters passed by the user, the module opens up indeed.com and monster.com and searches for job descriptions using those parameters. The resulted job descriptions are scrapped and stored in an array. Before storing in and array, irrelevant job descriptions are filtered out.

5.2.3.1.9 Interfaces Exports

This function is the scraper for indeed.com. It is a subsystem of this module. Scrapper for monster.com has the same structure.

```
# scrapper for indeed.com

# this scrapper takes the user's query and location and as parameters

# and returns the array of job descriptions as output

def indeed_scrapper(query, location):



    PATH = 'C:\Program Files (x86)\chromedriver.exe'

    url = 'https://pk.indeed.com/'



    USER_LOCATION = location

    chrome_options = Options()

    chrome_options.add_argument("--headless")

    driver = webdriver.Chrome(PATH, options=chrome_options)

    driver.get(url)
```

```
 try:
     searchBar = WebDriverWait(driver, 10).until(
         EC.presence_of_element_located((By.ID, 'text-input-what'))
     )
     searchBar.send_keys(query)

     locationBar = WebDriverWait(driver, 10).until(
         EC.presence_of_element_located((By.ID, 'text-input-where'))
     )

     locationBar.send_keys(USER_LOCATION)
     locationBar.send_keys(Keys.RETURN)

     # to navigate to us indeed
     if USER_LOCATION != 'pakistan':
         nav_link = WebDriverWait(driver, 10).until(
                                                  EC.presence_of_element_located((By.XPATH,
'/html/body/table[2]/tbody/tr/td/table/tbody/tr/td/table/tbody/tr/td[1]/div[2]/div/p/a')
)
         )
         nav_link.click()

     jobCardContainer = WebDriverWait(driver, 10).until(
         EC.presence_of_element_located((By.ID, 'mosaic-provider-jobcards'))
     )

     jobCardsFiltered = []
     jobCards = jobCardContainer.find_elements_by_class_name('job_seen_beacon')
     for jobCard in jobCards:
         jobCardsFiltered.append(jobCard.text)

     print('Collecting text from job descriptions ...')

     #collecting and saving the data
     jobDescriptions = []
     for i in range(len(jobCards)):
         driver.implicitly_wait(1)
         try:
             jobCards[i].click()
             jobDescIframe = WebDriverWait(driver, 10).until(
                 EC.presence_of_element_located((By.ID, 'vjs-container-iframe'))
             )
             driver.switch_to.frame(jobDescIframe)
             jobDescription = driver.find_element(By.ID, 'jobDescriptionText')
             if query in jobDescription.text.lower():
                 jobDescriptions.append(jobDescription.text)
             driver.switch_to.default_content()
         except Exception as e:
             print(e)
             continue

     #writing to file
     json_object = json.dumps(jobDescriptions, indent = 4)
     with open(f'./data/{location}_indeed_data.json', 'w') as file:
         file.write(json_object)

     driver.quit()

 except Exception as e:
     print(e)
     print("Something went wrong!")
     driver.quit()

 return jobDescriptions
```

5.2.3.1.10 Detailed Subsystem Design

This module has 2 similar components. One of indeed.com and one for monster.com. These have been discussed above in detail.

5.2.3.2 Information Extraction and Keyword Generation Module

5.2.3.2.1 Classification

This component is a *module*.

5.2.3.2.2 Definition

This module extracts information from the job descriptions and processes it to produce key words / required job skills.

5.2.3.2.3 Responsibilities

This module is responsible for processing the job descriptions to get the required keywords using NLP and Textrank algorithm, which are then used in the next module in the pipeline.

5.2.3.2.4 Constraints

The input to this module is an array of job descriptions. These job descriptions are iteratively read and processed. This data comes from the previous module, i.e., scrapper for job descriptions.

The output of the module is an array of strings. The strings are filtered keywords extracted and processed by this module. This array is then passed to the next modules that run in parallel to get the resources from the web.

5.2.3.2.5 Composition

This module is composed of two sub-modules or functions.

One function generates the **cluster of relevant sentences** from job descriptions. This function uses NLP in its operation to extract the sentences based on some pre-defined rules and specified text patterns.

The second function applies the **Textrank algorithm** on this cluster.

5.2.3.2.6 Uses/Interactions

The first interaction is the triggering of this module from the previous scrapper module. The previous module passes the search parameters for this module to operate.

The second interaction is between this module and the next scrapper module that gathers the resources. This module passes the array of filtered keywords to the next module.

5.2.3.2.7 Resources

This module only needs memory and processor. The NLP function as well as the Textrank algorithm has some processor requirements because it processes a lot of text-based data. The array of strings will be stored in memory until used by the next module. There are no database requirements as there is no persistence.

5.2.3.2.8 Processing

The approach taken by this module is as follows. Firstly, this module looks for relevant sentences that potentially contain relevant information and extracts those sentences. Those sentences are then clustered and Textrank algorithm is applied to get the keywords as output. The keywords are once again passed through some filtering functions to get the final and filtered output. The output keywords are already arranged in the order of importance by the Textrank algorithm.

5.2.3.2.9 Interfaces Exports

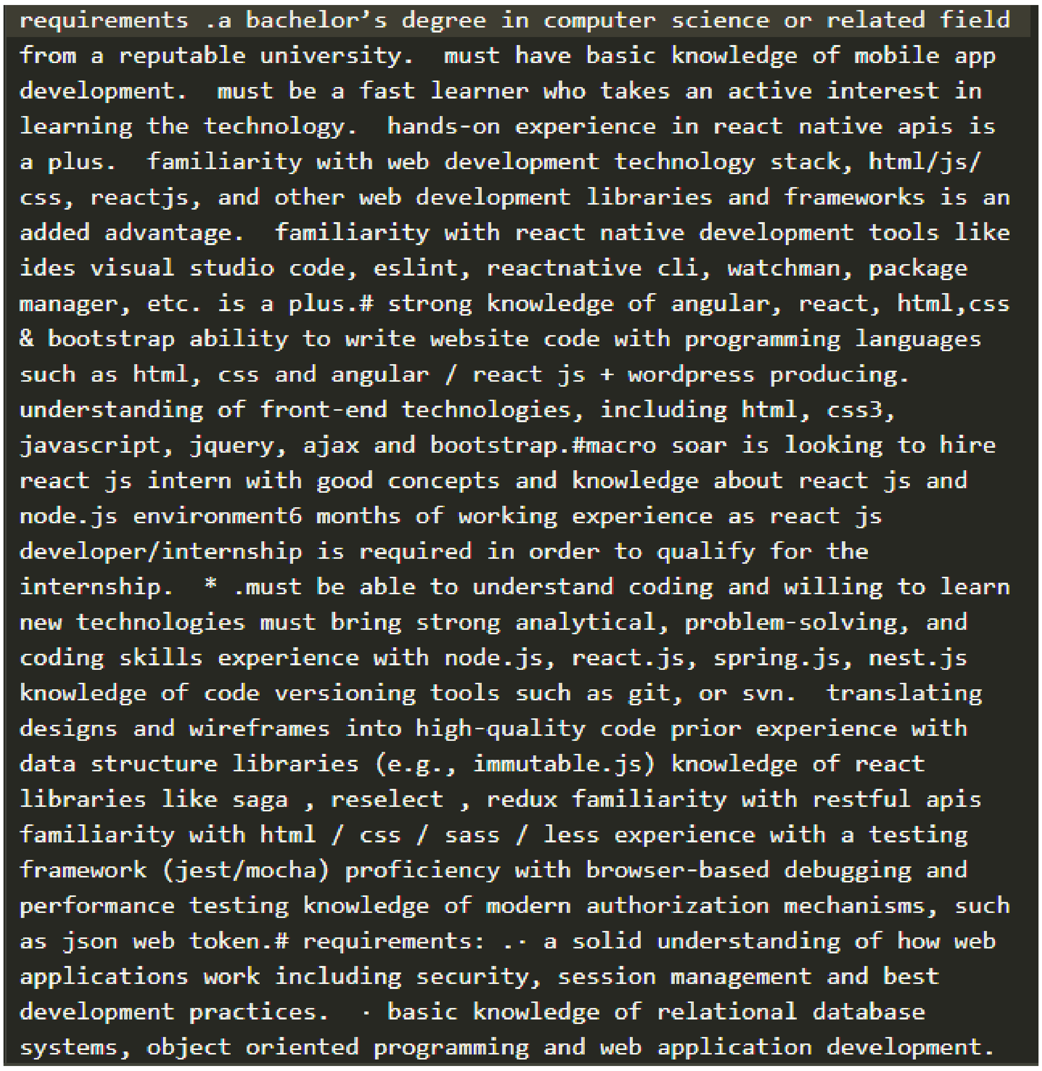
Sentence extraction component

```
#starting NLP on the jobDescription from here

    phraseMatcher = PhraseMatcher(nlp.vocab)



    #these are the phrases for "requirements"

    phrases = ["must have", "must be", "expertise in", "experience in", "experience
with", "experience as", "familiarity with"

     "should have", "responsibilities include", "understanding of", "knowledge of",
"requirements", "familiarity with"

    ]
```

```python
data = []
for i in jsonObj:
    jobDesc = i
    headings = re.findall('\n.{1,24}\n', jobDesc)

    for i in headings:
        jobDesc = jobDesc.replace(i, f'.{i}.')

    jobDesc = jobDesc.lower()
    patterns = [nlp(text) for text in phrases]
    phraseMatcher.add('keys', patterns)
    doc = nlp(jobDesc)

    text_to_write = []
    for sent in doc.sents:
        sentence = sent.text.replace('\n', ' ')

        for match_id, start, end in phraseMatcher(nlp(sentence)):
            if nlp.vocab.strings[match_id] in ["keys"]:
                if sentence not in text_to_write:
                    # print('->', sentence)
                    text_to_write.append(sentence)
```

```
text_string = ' '.join(text_to_write)

text_string = text_string.replace('\n', ' ')

data.append(text_string)
```

The output of this component is a cluster of sentences extracted from all the job descriptions. This is shown in the image below. All these sentences potentially contain the items that is of importance to us.



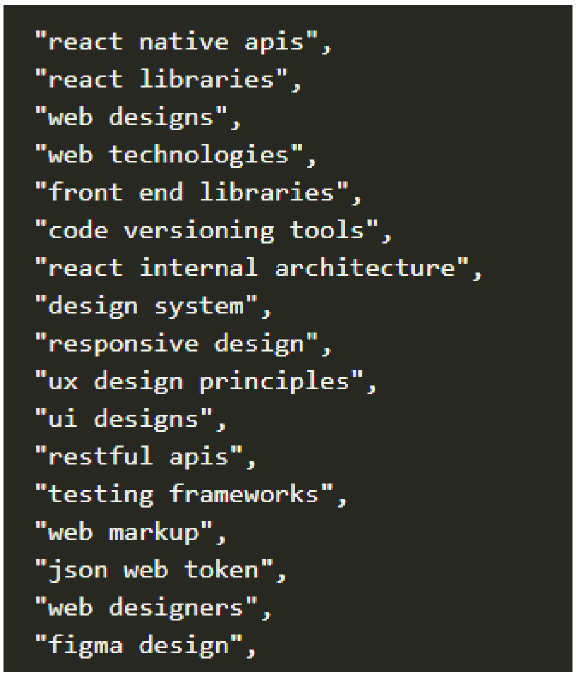
```
requirements .a bachelor’s degree in computer science or related field
from a reputable university.  must have basic knowledge of mobile app
development.  must be a fast learner who takes an active interest in
learning the technology.  hands-on experience in react native apis is
a plus.  familiarity with web development technology stack, html/js/
css, reactjs, and other web development libraries and frameworks is an
added advantage.  familiarity with react native development tools like
ides visual studio code, eslint, reactnative cli, watchman, package
manager, etc. is a plus.# strong knowledge of angular, react, html,css
& bootstrap ability to write website code with programming languages
such as html, css and angular / react js + wordpress producing.
understanding of front-end technologies, including html, css3,
javascript, jquery, ajax and bootstrap.#macro soar is looking to hire
react js intern with good concepts and knowledge about react js and
node.js environment6 months of working experience as react js
developer/internship is required in order to qualify for the
internship.  * .must be able to understand coding and willing to learn
new technologies must bring strong analytical, problem-solving, and
coding skills experience with node.js, react.js, spring.js, nest.js
knowledge of code versioning tools such as git, or svn.  translating
designs and wireframes into high-quality code prior experience with
data structure libraries (e.g., immutable.js) knowledge of react
libraries like saga , reselect , redux familiarity with restful apis
familiarity with html / css / sass / less experience with a testing
framework (jest/mocha) proficiency with browser-based debugging and
performance testing knowledge of modern authorization mechanisms, such
as json web token.# requirements: .· a solid understanding of how web
applications work including security, session management and best
development practices.  · basic knowledge of relational database
systems, object oriented programming and web application development.
```

Textrank algorithm

```
def textrank():
    nlp = spacy.load('en_core_web_lg')
    nlp.add_pipe('textrank')


    with open('./data/summary.txt', 'r', encoding="utf-8") as file:
        data = file.read()
        doc = data.replace('#', ' ')
        doc_arr=[]
        doc_arr.append(doc)
    keywords = []
    k=[]


    for doc in doc_arr:
        #doc is an array here
        doc = doc.replace('\n', ' ')
        total_words = doc.split()
        clean_words = [w for w in total_words if not w in stop_words]
        clean_doc = ' '.join(clean_words)


        doc = nlp(doc)


        tr = doc._.textrank
```

```
for combination in doc._.phrases:
    skip = False
    for i in garbage_words:
        if i in combination.text:
            skip = True
            continue
    if combination.rank > 0.035 and not skip:
        obj = {}
        obj['text'] = combination.text
        obj['rank'] = combination.rank
        keywords.append(obj)
        k.append(combination.text)
```

The output of the Textrank algorithm is shown in the image below. These are all the keywords/ key phrases extracted from the output shown above.


```
"react native apis",
"react libraries",
"web designs",
"web technologies",
"front end libraries",
"code versioning tools",
"react internal architecture",
"design system",
"responsive design",
"ux design principles",
"ui designs",
"restful apis",
"testing frameworks",
"web markup",
"json web token",
"web designers",
"figma design",
```

5.2.3.2.10 Detailed Subsystem Design

This module has two subsystems as depicted in the diagram below.

1. Sentence extraction module
2. Textrank Algorithm

Both these modules, along with their outputs have been discussed above.

5.2.3.3 Web scrapper for content gathering from YouTube

5.2.3.3.1 Classification

This component is a *module*.

5.2.3.3.2 Definition

This is a web scraping module that collects content from YouTube.

5.2.3.3.3 Responsibilities

This module uses the keywords from the previous module and creates a query and searches that on YouTube to gather the learning material.

5.2.3.3.4 Constraints

The input to this module must come directly from the previous module i.e., the Information Extraction and Keyword Generation Module. This input is an array of strings.

The output is of an array of objects where each object contains the video link, title, views, and runtime. This array is then used by the server where it is stored in database.

A constraint on this module is that the YouTube should be up and running otherwise no results will be gathered. It is assumed that if the websites are up, results will always be provided for valid queries.

5.2.3.3.5 Composition

This module is composed of a single function and runs a web scrapper for YouTube.

5.2.3.3.6 Uses/Interactions

The first interaction is the triggering of this module from the previous module. The array of strings is passed to this module as a parameter.

The second interaction is the sending and storing of gathered results in the database.

5.2.3.3.7 Resources

This module only needs memory, processor, and database for persisting the results. The web scrapper has some processor requirements. The results will be stored in memory until saved in the database.

5.2.3.3.8 Processing

The way that this module operates is as follows.

Firstly, the array of keyword is passed to this module, the module opens up YouTube page and searches those keywords/queries iteratively. The resulted videos are scrapped and stored in an array.

5.2.3.3.9 Interfaces Exports

This section contains the source code along with the comments for this module.

This function/module takes in the array of keywords as a parameter and returns the scrapped YouTube data.

```
# Scrapper for YouTube

# This function takes in the keywords array as input

def yt_scrapper(keywords_arr):



    queries = keywords_arr

    scrapped_data = []



    # setting up some variables for Selenium scrapper

    URL = 'https://www.youtube.com/'

    PATH = 'C:\Program Files (x86)\chromedriver.exe'

    chrome_options = Options()

    chrome_options.add_argument("--headless")

    driver = webdriver.Chrome(PATH, options=chrome_options)

    driver.get(URL)

    for i in queries:

        try:

            searchBar = WebDriverWait(driver, 10).until(

                        EC.presence_of_element_located((By.XPATH, '/html/body/ytd-
app/div/div/ytd-masthead/div[3]/div[2]/ytd-searchbox/form/div[1]/div[1]/input'))

            )
            searchBar.clear()
            searchBar.send_keys(i)
            searchButton = WebDriverWait(driver, 10).until(
                        EC.presence_of_element_located((By.XPATH, '/html/body/ytd-app/div[1]/div/ytd-masthead/div[3]/div[2]/ytd-searchbox/button'))
            )
            time.sleep(1)
            searchButton.click()

            #scraping and saving the video results in the form of an array of objects
            time.sleep(2)
            results = driver.find_elements_by_class_name('style-scope ytd-video-renderer')
            videos = []
            for r in range(5):
                        data = {}
                        try:
                            data['title'] = results[r].find_element_by_tag_name('h3').text
                            data['views'] = results[r].find_element_by_xpath('.//*[@id="metadata-line"]/span[1]').text
                            data['runtime'] = results[r].find_element_by_id('text').text
                            href = results[r].find_element_by_id('thumbnail').get_attribute('href')
                            data['href'] = href
                            videos.append(data)
                        except Exception as e:
                            pass
            obj={
                        'topic': i,
                        'data': videos
            }
            scrapped_data.append(obj)

        except Exception as e:
            print(e)
            print("Something went wrong!")

#writing the data to file
json_object = json.dumps(scrapped_data, indent = 4)
with open('./data/yt_data.json','w', encoding="utf-16", errors="surrogatepass") as file:
    file.write(json_object)
driver.quit()
return scrapped_data
```

5.2.3.3.10 Detailed Subsystem Design

This module does not have any further sub-components.

5.2.3.4 Web scrapper for content gathering from Google

5.2.3.4.1 Classification

This component is a *module*.

5.2.3.4.2 Definition

This is a web scraping module that collects content from Google.

5.2.3.4.3 Responsibilities

This module uses the keywords from the previous module and creates a query and searches that on Google to gather the learning material in the form of links from different websites.

5.2.3.4.4 Constraints

The input to this module must come directly from the previous module i.e., the Information Extraction and Keyword Generation Module. This input is an array of strings.

The output is of an array of objects where each object contains the title and the link. This array is then used by the server where it is stored in database.

A constraint on this module is that the Google should be up and running otherwise no results will be gathered. It is assumed that if the websites are up, results will always be provided for valid queries.

5.2.3.4.5 Composition

This module is composed of a single function and runs a web scrapper for Google.

5.2.3.4.6 Uses/Interactions

The first interaction is the triggering of this module from the previous module. The array of strings is passed to this module as a parameter.

The second interaction is the sending and storing of gathered results in the database.

5.2.3.4.7 Resources

This module only needs memory, processor, and database for persisting the results. The web scrapper has some processor requirements. The results will be stored in memory until saved in the database.

5.2.3.4.8 Processing

The way that this module operates is as follows.

Firstly, the array of keyword is passed to this module, the module opens up Google page and searches those keywords/queries iteratively. The resulted links are scrapped and stored in an array. This data is returned from the function.

5.2.3.4.9 Interfaces Exports

This section contains the source code along with the comments for this module.

This function/module takes in the array of keywords as a parameter and returns the scrapped Google data.

```
# Scrapper for Google

# This function takes in the keywords array as input

def google_scrapper(keyword_arr):



    queries = keyword_arr

    links_arr = []



    # setting up some variables for Selenium scrapper

    url = 'https://www.google.com/search?q=.'

    PATH = 'C:\Program Files (x86)\chromedriver.exe'

    chrome_options = Options()

    chrome_options.add_argument("--headless")
```

```
driver = webdriver.Chrome(PATH, options=chrome_options)

driver.get(url)


for query in queries:

    try:

        searchBar = WebDriverWait(driver, 10).until(

                                                    EC.presence_of_element_located((By.XPATH,
'/html/body/div[4]/div[2]/form/div[1]/div[1]/div[2]/div/div[2]/input'))

        )

        searchBar.clear()

        searchBar.send_keys(query)

        searchBar.send_keys(Keys.RETURN)

        time.sleep(1)


        links = WebDriverWait(driver, 10).until(

            EC.presence_of_all_elements_located((By.CLASS_NAME, 'yuRUbf'))

        )


        #scraping and saving the video results in the form of an array of objects

        data = []

        for i in range(9):

            try:

                link = links[i].find_element_by_tag_name('a').get_attribute('href')

                title = links[i].find_element_by_tag_name('h3').text
                if title == '':
                    continue
                obj = {}
                obj['link'] = link
                obj['title'] = title
                data.append(obj)
            except Exception as e:
                pass

        link_obj = {
            'topic': query,
            'links': data
        }

        links_arr.append(link_obj)

    except Exception as e:
        print(e)
        print ("Something went wrong!")

driver.quit()

#writing the data to file
```

```
json_object = json.dumps(links_arr, indent = 4)

with open('./data/google_data.json', 'w') as file:

    file.write(json_object)

print('data written')

return links_arr
```

5.2.3.4.10 Detailed Subsystem Design

This module does not have any further sub-components.

5.3 CLASS DIAGRAM

Below is the class diagram for the system.

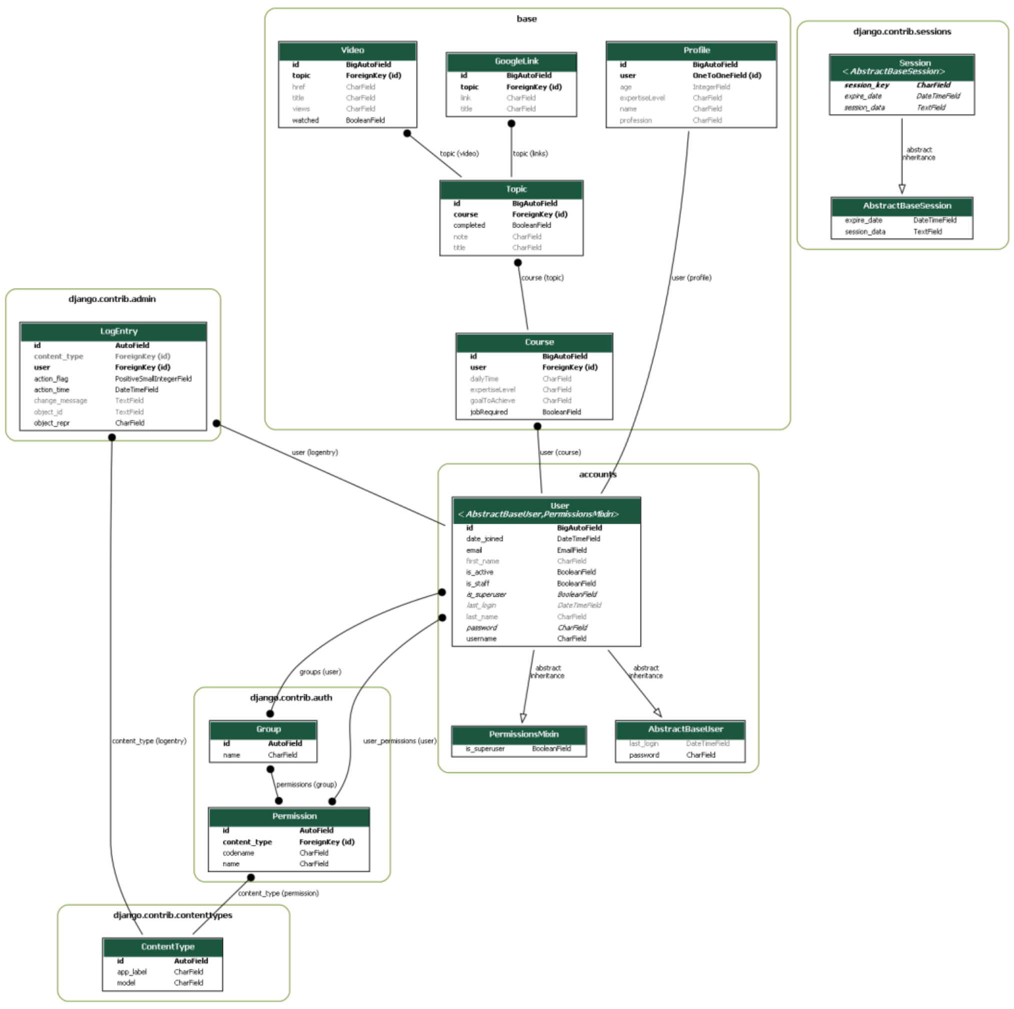


Figure 18: Class Diagram

5.4 ER DIAGRAM

Below is the Entity Relation (ER) diagram of the system.

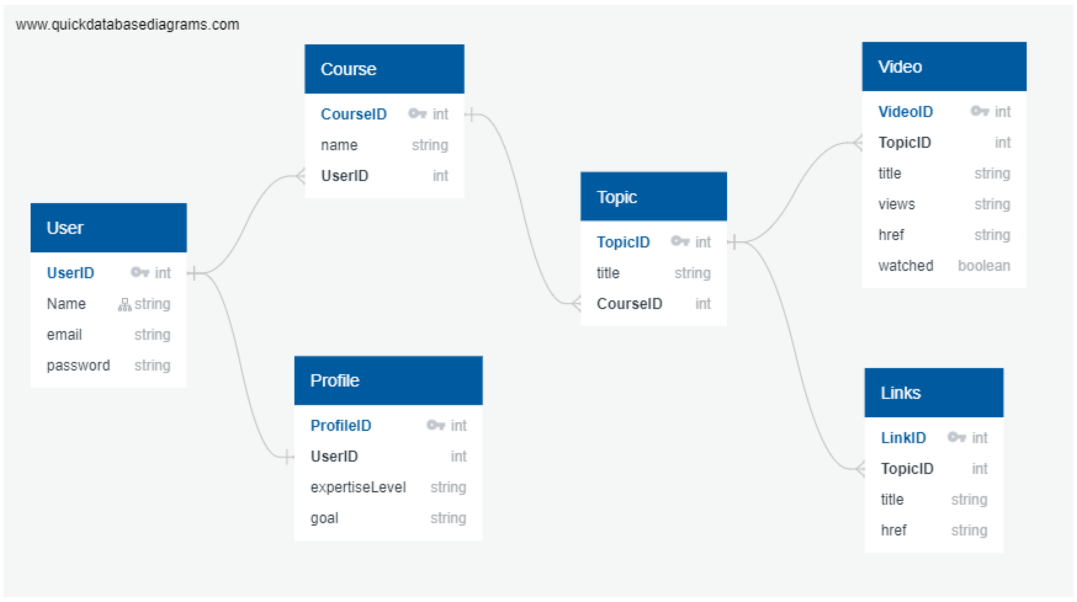


Figure 19: ER Diagram

Chapter 6

IMPLEMENTATION AND TESTING

The system was developed using a combination of both waterfall and agile development methods. Functional and non – functional requirements i.e., the features of the system were finalized in the early stages and were not changed later. However, system architecture and design were somewhat changed later.

6.1 TOOLS AND TECHNIQUES

As the system has three major modules, namely frontend, backend and the content gathering and processing modules, these are developed separately, using different libraries and programming languages. Consequently, the testing was also done using appropriate techniques and testing frameworks.

6.2 TESTING

The three major subsystems were developed separately and were individually tested by their respective developers. Different testing techniques were incorporated. Both black box white box testing was done at different stages. Each of the subcomponent in backend and content processing module was tested in detail using white box testing, whereas the inputs and outputs (only in the backend) were also tested as black boxes. Frontend was testing using a frontend specific testing framework i.e., Mocha and the API calls and their responses were tested as a black box.

The subsystems were then integrated, and the system was tested again using “end-to-end” testing. No framework was used here. The system was tested manually, and it was verified that all the features were completed, and requirements were verified to a maximum.

6.3 TESTING OF THE NLP MODULE

NLP module involved a lot of research and development as well as a lot of experimentation. All the intermediate outputs were tested and verified manually, and adjustments were made wherever necessary. For example, the outputs of the scripts for these parts that are added in the design section were tested manually to get the accuracy results mentioned in the results section.

6.4 EVALUATION OF THE SOFTWARE

The implemented system was also compared with the functional and non-functional requirements i.e., the features and the quality factors. All the features were implemented as per the project plan and as described the architecture and design sections.

Let’s take a look at the quality attributes i.e., non-functional requirements first. The requirements of speed in the context of page load time are satisfied. The page load time is less than 3 seconds (PR-1). The registration process is easily completed in under 30 seconds (PR-3), given that the network is working properly. Login time is less than 15 seconds (PR-4). The process of content creation takes somewhere between 3 to 5 minutes, significantly lower than 12 minutes (PR-5).

Security requirements are also satisfied. Database entries are protected, and passwords are hashed before storing.

Requirement of ease-of-use is also satisfied. Users are provided with a concise manual that explains the usage. The usage of the system itself is pretty simple.

Rest of the requirements have not been tested because the system has not been deployed in such a production environment.

The results of the system are very accurate when compared with the initial idea. The accuracy is discussed in the “results” section. In short, the accuracy of the system serves to solve the problem addressed by the project.

Chapter 7

RESULTS AND DISCUSSION

The system is prepared with the aim of achieving maximum accuracy for the generated content because the effectiveness of the whole system depends on it. We've used all the best possible tools and techniques to ensure that the accuracy of the generated content is up to the mark.

In-depth research of all the algorithms was conducted. We looked at all the possible options we had for the design of the system so that we can figure out the best and most efficient one for our system. We looked at different algorithms options that we could use for our system. We looked at Machine Learning (ML) and Natural Language Processing (NLP). First, we wanted to go with Machine Learning but there wasn't enough data to train a model that could give good accuracy. To train a model like that, we needed thousands of entries. That was not possible because we were scrapping the data in real-time, and it would have been very difficult and time-consuming to get that much data.

Keeping the above limitation in view, we looked at the other available option, which was NLP. We did some test runs and it proved to be very efficient and accurate, so we decided to go with it. We were gathering content using Web Scraping using Selenium from job posting websites like LinkedIn and monster.com. After that, the data was analyzed and clustered by the NLP module which gave us a list of keywords to work on. These keywords are the skills that are needed to be learned.

Once we have the keywords, we use the data that we gathered from our survey that we conducted among university students. According to this data, a lot of students prefer to learn through video lectures while at the same time, many other students prefer to learn by reading it. So, we decided to add both of those things to our project. W used the keywords provided by the NLP module to again scrape the content from the internet. We used YouTube to get videos and then we used Google to get reading material for the users who prefer to learn by reading.

The tricky part of our solution is the gathering and processing of the data. The job descriptions that we collect do not have a standard format, and there are limited useful job descriptions that can be gathered. These factors inhibit the results of the system, i.e., the generated content. These results can be improved if the job descriptions have some standard format, because it becomes easier to identify and extract information. Secondly, if a large of number of job descriptions can be gathered, it can become easier to train some useful machine learning models like word2vec that can give some useful outputs. Still, our solution is still efficient in terms of the output it can generate.

Following were the results of our work:

1. From the job posts that we scrapped, we found out that our system was able to extract **85 - 90%** of the keywords.

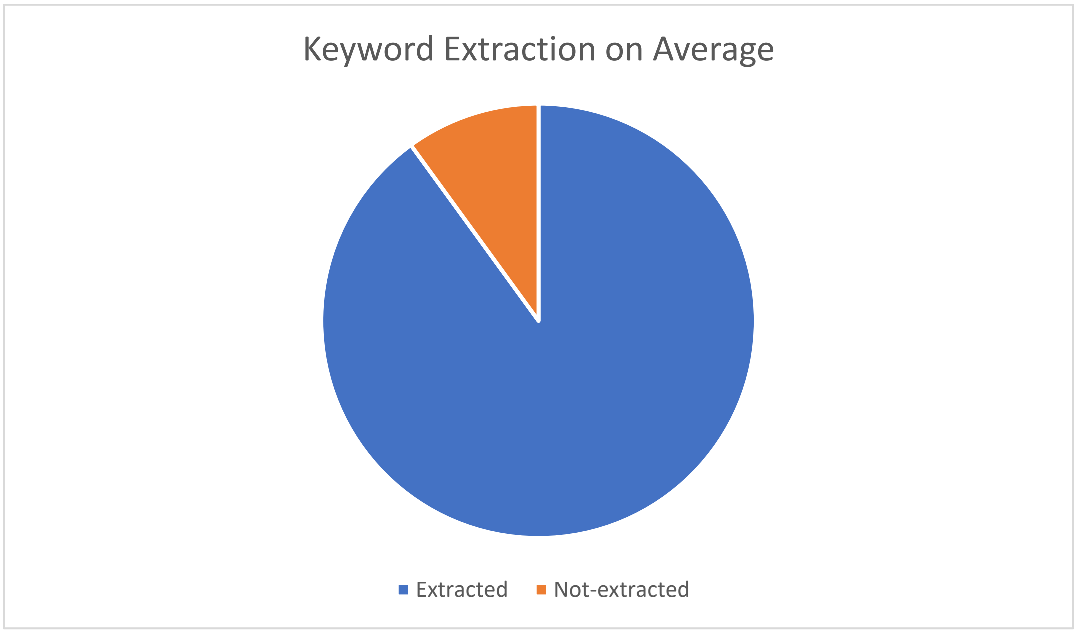


Figure 20: Keyword Extraction Graph

2. Similarly, using these keywords, the resources that are gathered have an accuracy of around **70-75%.** These figures tell that on average, 75% of gathered resources are actually relevant to a particular field. The **25-30%** includes resources that are very general in nature i.e., that can be related to many fields, and some resources are not at all relevant to the field. This can be considered a result of the system working without human interaction.

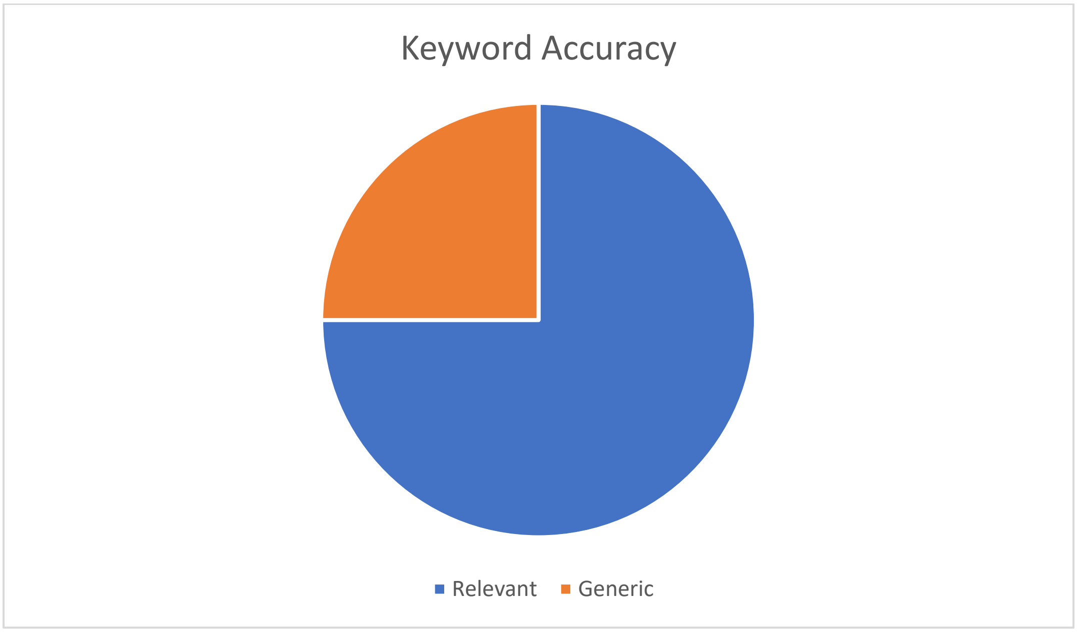


Figure 21: Keyword Accuracy Graph

From these numbers, we can already see that the content generated by the system is highly accurate. All this accuracy is thanks to our extensive research and trial and error. As already mentioned, there was not a lot of literature about this, so we had to learn by trial and error. We tried different available options to technologies for different things and saw which worked better and provided more accuracy. We also tried different combinations of options to find the most optimal solution.

CHALLENGES FACED

Following changes were faced during the making of this project:

1. There hasn’t been much work done on this specific application where users can get roadmaps to achieve their goals. Therefore, there was very limited literature that we could find related to this. We had to learn a lot of things through trial and error. We tried different things, compared the results and then chose the ones that suited us best.
2. Initially, we had planned to do this using Machine Learning by training our own models. For getting a decent accuracy from those models, we needed dataset with thousands of entries. Getting such large dataset through web scrapping was very time consuming. Plus, there weren’t thousands of active job posts related to a particular post. Since we couldn’t’ get the data to train our models, we have to change our approach and switch from using Machine Learning to using NLP. It was a great learning experience.
3. Finding and removing irrelevant content was difficult. Since we were using NLP, we would sometimes get keywords that we not part of the skillset required for a particular job. But the system would still extract those keywords and their relevant content because the NLP module generated those keywords. It was very challenging for us to remove such extra keywords.

LIMITATIONS

Following are the limitations of our system right now:

1. The content generated by the system is not 100% accurate. We have to get help from field experts to get that content verified before we can make it available for users. This really limits the number of fields for which we can provide roadmaps.
2. Our system is getting the content from the internet. Therefore, the quality of the system can only be as good as the content from the internet.
3. Right now, the content roadmaps can be generated only for limited IT fields.

Chapter 8

CONCLUSION AND FUTURE WORK

Although there are many different types of resources available on the internet that can help anyone to achieve their career goals, there is still a need for a platform like this where users can simply enter the information about their goals and the system generates easy-to-follow roadmaps for them. In this modern age of technology, everyone has access to information, thanks to the internet. But it is important to get the right information in a reasonable amount of time for that information to matter.

We conducted a survey and where we asked university students and fresh graduates if they would like to use a platform where all the information related to their career goals can be found in one place and 69.8% of respondents said that yes they would really like to use such a platform and that it will be a life-saver. This clearly shows that there is a market gap. Our application "Goal Quest" caters to that need. Plus, not just our application solves that problem, it provides personalized learning using which can potentially solve a lot of time for people.

We have solved the basic problem and provided some very useful additional features like personalization of roadmaps for every user according to their requirements as well as some utility features like saving and tracking your progress, taking notes and providing every user the ability to achieve those goals at their own pace. But still, there are some things that can be added to our application to make it even better and more useful for more people.

Our developed solution is by no means perfect and can be improved to a great extent. Following are the three main things that can added to our project in future:

1. User LinkedIn Profile Integration

Right now, each user who wants to generate a roadmap to achieve some kind of goal, has to create their profile and add all the information manually be filling up a form on the website. This can be made easier by giving user the ability to sign up using with their LinkedIn profile and then the basic information can be taken from their LinkedIn profile.

2. Add More Fields

As of yet, our system provides roadmaps for IT-related fields only because we need field experts to verify the roadmaps generated for those fields. In future, we can get experts from more fields on board and generate roadmaps for various other fields as well.

3. CV Integration

We can add the option where users can upload their CV and the application will automatically get as much information from that CV as possible. This will allow the user to get started in the least amount of time because now he doesn’t have to fill long forms.

4. Remove the need for content verification:

Right now, the content roadmaps generated by our system need to be verified by our experts to make sure that there is no redundant content. In future, we can try to make our system more accurate up to a point where we no longer need to get the content verified by field experts and we can be sure of its accuracy and authenticity of generated roadmaps.

Chapter 9

REFERENCES

[1] SpaCy, "PyTextRank," [Online]. Available: https://spacy.io/universe/project/spacy-pytextrank/.

[2] TierNok, "Automated Keyword Extraction – TF-IDF, RAKE, and TextRank," [Online]. Available: http://www.tiernok.com/posts/automated-keyword-extraction-tf-idf-rake-and-textrank.html.

[3] SpaCy, "PhraseMatcher," [Online]. Available: https://spacy.io/api/phrasematcher.

[4] SpaCy, "Language Processing Pipelines," [Online]. Available: https://spacy.io/usage/processing-pipelines.

[5] Real Python, "https://realpython.com/natural-language-processing-spacy-python/," [Online]. Available: Natural Language Processing With spaCy in Python.

[6] Gensim, "Word2vec embeddings," [Online]. Available: https://radimrehurek.com/gensim/models/word2vec.html.

[7] Towards Data Science, "How to Use Selenium to Web-Scrape with Example," [Online]. Available: https://towardsdatascience.com/how-to-use-selenium-to-web-scrape-with-example-80f9b23a843a.

[8] Monkey Learn, "Understanding TF-ID: A Simple Introduction," [Online]. Available: https://monkeylearn.com/blog/what-is-tf-idf/.

[9] Towards Data Science, "Word Embedding Techniques: Word2Vec and TF-IDF Explained," [Online]. Available: https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08.

[10] FinTechExplained, "NLP: How To Evaluate The Model Performance," [Online]. Available: https://medium.com/fintechexplained/nlp-how-to-evaluate-the-model-performance-7e1820cdb759.

[11] Knowledge Officer, "Homepage," [Online]. Available: https://www.knowledgeofficer.com/.

[12] Ed App, [Online]. Available: https://www.edapp.com/.

[13] Django, "Django documentation," [Online]. Available: https://docs.djangoproject.com/en/4.0/.

[14] Tailwind, "Get started with Tailwind CSS," [Online]. Available: https://tailwindcss.com/docs/installation.