USER MANUAL FOR IBM SKILLSGUILD

Written By Calvin Dantis (ZMZF47),
William Farmer (HDTK63),
Chung Hei Isaac So (DCMC71),
Ki Yan Nathan Lam (QWSK85),
Nathaniel Oppenheimer (XXCS76),
Liam McCarthy (BCHH34).

Software Engineering Group 20:
07/03/2024

Table of Contents

1 - Overview.....2
  1.1 Background.....2
  1.2 Our Solution: SkillsGuild.....2
  1.3 Potential Impacts on Ethics and Society.....2
    1.3.1 Current & Future Ethical Impacts.....2
    1.3.2 Current & Future Societal Impacts.....3
2 - Users' Guide: How to learn with SkillsGuild.....4
  2.1 Dependencies & Set Up.....4
  2.2 Starting the Web App.....4
  2.3 Navigating the Web App.....4
  2.4 Signing Up & Logging In.....6
  2.5 Understanding the Profile.....7
  2.6 Using the Chatbot.....8
    2.6.1 Accessing the Chatbot.....8
    2.6.2 Interacting with the Chatbot.....8
    2.6.3 Get SkillsBuild Recommendations.....9
    2.6.3.1 Career Recommendations.....9
    2.6.3.2 Education Recommendations.....10
    2.6.3.3 Interest Recommendations.....11
    2.6.4 Get Information.....12
    2.6.4.1 Information About SkillsBuild Course.....12
    2.6.4.2 Information About Career.....13
    2.6.5 Other Assistance.....13
  2.7 Completing Courses.....14
  2.8 Understanding the Inventory and User Equipment.....15
  2.9 Troubleshooting, Feedback & Contact Details.....16
3 - Developers' Guide: System Maintenance & Future Developments.....17
  3.1 Set up the Development Environment.....17
  3.2 System API.....17
    3.2.1 GET Web Pages.....17
    3.2.2 Backend Requests.....19
  3.3 Backend Functionality.....22
  3.4 Technical Walkthrough.....26
    3.4.1 Navigating the Web App.....26
    3.4.2 Signing Up & Logging In.....28
    3.4.3 Understanding the Profile.....30
    3.4.4 Using the Chatbot.....30
    3.4.5 Completing Courses.....31
    3.4.6 Understanding the Inventory and User Equipment.....32
  3.5 Database.....35
  3.6 UI.....37
  3.7 Development Process.....37
  3.8 Maintenance.....38

- 3.9 Future Developments........43
  - 3.9.1 Unimplemented Features........43
  - 3.9.2 Suggestions for Future Developments........44

4 - Requirements & Resources........45

- 4.1 Status of User Requirements........45
  - 4.1.1 Functional Requirements........45
  - 4.1.2 Non-functional Requirements........48
- 4.2 Resources & References........50

1 - Overview

This is the User Manual for IBM SkillsGuild Web Application (also named as SkillsGuild). It is divided into 4 sections. The first section summarises the background and our solution to tackle the problems. Section 2 provides a comprehensive guide for learners on how to use the application and assist their learnings on SkillsBuild. All functions are outlined, ensuring clarity and ease of understanding for all. Section 3 contains technical information on product development and system maintenance, ensuring the client is able to maintain and improve the web application following our product handover. Section 4 describes the potential impacts of SkillsGuild, followed by detailed status of user requirements and resources for references.

1.1 Background

IBM SkillsBuild is an extensive learning platform covering artificial intelligence, data science, cyber security, and various tech fields. However, this platform does not support consumable online learning currently. Learners might also get lost and confused in the vast number of courses. Our product, SkillsGuild is designed to help the situation and enhance the online learning experience of IBM SkillsBuild platform, so that potential learners could better equip new skills and prepare for new careers and opportunities. We believe a more consumable and enjoyable learning experience could help learners engage in the courses.

1.2 Our Solution: SkillsGuild

Our product, SkillsGuild is a mobile-friendly web application supporting the learning experience on IBM SkillsBuild. Our initial user base is Level 1 Computer Science students, however, it could be extended to all students at later stages. SkillsGuild combines AI chatbot and gamification to assist learners and enhance their learning experience.

Learners may find IBM SkillsBuild hard to navigate and inconvenient to search for suitable courses. To tackle this issue, SkillsGuild is going to take requirements from learners and identify suitable courses and software from IBM SkillsBuild for them. SkillsGuild is integrated with IBM Watson AI chatbot to streamline communication and refine recommendation. This creates a tailor-made experience for learners who aim to improve their technology skills and advance in their career paths. As for learners' engagement, SkillsGuild is equipped with gamification elements to encourage learners engaging in their courses. Avatars are assigned to learners and they are awarded items for completing the recommended courses. To promote competitiveness among peers, a leaderboard is introduced to SkillsGuild, encouraging learners to spend time on the courses with their friends and peers within the same institution.

1.3 Potential Impacts on Ethics and Society

This section outlines the potential impacts SkillsGuild could have on society, considering the different aspects of the web app and how the system interactions could affect the users. Various stakeholders have been considered and the effect SkillsGuild would have on them.

1.3.1 Current & Future Ethical Impacts

Firstly, elements of gamification might harm personal growth of learners when they become addictive. While these elements, such as leaderboard, rewards and avatars, aim at increasing learners' engagement and motivation by fostering senses of accomplishment, satisfaction and competition, there is a potential for these elements to induce addictive behaviours, where learners prioritise earning those rewards or climbing leaderboard over acquiring the course knowledge and skills. It might undermine the primary goal of learning or even personal growth in the long term.

To avoid such unhealthy obsessions, in the future SkillsGuild can implement a daily experience point limit for learners to control their time spent on the platform. In case the experience point limit is reached, the learner can no longer earn rewards and points, or continue to climb the leaderboard until the next day. This measure can balance between engagement and personal growth of the users.

Secondly, information of learners collected from SkillsGuild might raise privacy concerns. Positively, this feature helps SkillsGuild provide more relevant course recommendations and tailor the learning experience to each of their needs and aspirations. They can be vital for determining career path and learning journey, so that SkillsGuild is able to customise recommendations carefully. However, if the data is not handled securely or shared with third parties without consent, there would be privacy concerns. Such data may be useful for analysis on certain institutions or areas for development or advertisement purposes. For example, if a Durham student aspires to become a software engineer, more technology advertisements will appear around them. Research companies can also take a large sum of data for analysis purposes. SkillsGuild aims to minimise these risks by not sharing information with any third parties unless it is absolutely necessary, e.g. storing the database online.

When SkillsGuild is further developed, a tick box should be added on the registration page to acknowledge users that the data would be collected solely for SkillsGuild chatbot use. This would lower privacy concerns when the web app is published to the public. On the other hand, data encryption will be applied to the database, ensuring no third parties can access the personal information of learners. It can help prevent data breaches and potential identity theft of the learners. We acknowledge that it is of paramount importance to keep any collected data safe and free from people or organisations without appropriate consent. Any changes of consent and the privacy policy will be informed to all users within a short period of time. Such can enhance the trustworthiness of SkillsGuild and protect all users from potential harm.

1.3.2 Current & Future Societal Impacts

Firstly, SkillsGuild increases the affordability and accessibility of quality education for learners from various backgrounds, locations and education levels. IBM SkillsBuild is a well-structured and content-rich platform for learners to explore technology. However, its poor navigation and engagement system create a barrier for learners from accessing and enjoying the courses. SkillsGuild is capable of bringing a solution to help learners realise their dreams through brushing up their knowledge. By recommending suitable courses and software from IBM SkillsBuild, SkillsGuild could help ambitious learners acquire relevant and in-demand skills that could improve their employability and career prospects.

In the future, when SkillsGuild is easily accessible as a website on the Internet, numerous learners all around the world could potentially benefit from SkillsGuild and its course recommendation, allowing them to enjoy quality education from the platform for free. Learners who live in remote or rural areas, or who face financial or social barriers, can access these courses and software with tailor-made instructions on how they start their learning journey, allowing them to improve their skills and advance in their career paths.

Secondly, SkillsGuild brings a trend of transforming learning experience through artificial intelligence (AI). AI was integrated into SkillsGuild to understand learners' needs and streamline communication. Though AI is not highly matured and accurate, it can help categorise different requirements and suggest learning paths according to the categories. Simplifying the hugely complicated human language is a key to allow systems to respond more accurately. In the foreseeable future it can be applied everywhere so that learning experiences across all aspects can be improved.

In the future, gamification is also going to be vital in learning experience. The gamification of the education experience provides new motivation for learning, especially popular amongst younger children. The innate competitiveness of humans means that leaderboard-tracked learning systems have also become popular. SkillsGuild expands on this societal shift, taking SkillsBuild courses and encouraging users to tackle them in exchange for items and other rewards. Other companies, such as Duolingo with 'streaks', have already implemented this on a large scale. Not only does this encourage users to start learning (and using the platform), but also motivates them to return to the platform, and learn more as a result.

Thirdly, a trend of integrating artificial intelligence (AI) into websites and systems could diminish valuable human-to-human interactions. These interactions are vital for making the help and support of software kind and friendly to all users of all ages and backgrounds. However, in an AI-powered chatbot, most dialogues are systematic and plain. Although there is minimal error or grammatical mistakes in these phrases, it is hard for the users to feel the love without human language.

In the future, an excessive dependence on AI can lead to jobs being replaced by AI. Due to the low cost of running chatbots, more businesses are adopting chatbots on their website and offer immediate help to users and customers. When the chatbots become advanced, potentially they can replace human instructors and mentors on software, leading to job losses in the sector. This is concerning when the role of humans is downgraded by computers. Instead of letting AI replace an essential part, we should utilise it and preserve the importance of humans between these necessary interactions.

2 - Users' Guide: How to learn with SkillsGuild

This section outlines a basic guide for non-technical users to navigate and use the functionalities of the SkillsGuild web app. Included is a guide on how to start, use the login system, navigate the app, descriptions of key information, how to interact with the chatbot and how the game systems function. This section aims to provide a simple, surface level overview on the basic operationality of the system.

2.1 Dependencies & Set Up

The web app uses python and as a result requires python 3.9 or above to operate, there are several guides outlining how to download and set up python on your device. Python can be downloaded from the official python website (https://www.python.org/downloads/), they also provide a beginner's guide to installing python on a variety of operating systems (https://wiki.python.org/moin/BeginnersGuide/Download).

A web browser is also required to load the web pages the app is composed of and is required to interact with the system. The web app is compatible with all major web browsers so the specific browser is subject to user preference. We recommend using a Windows based machine for this application to ensure functionality.

An internet connection is also required to interact with the main feature of the web app so ensure the device the web app will run on has a stable internet connection.

To install all dependencies required to run the application, navigate to the 'program' directory in a terminal window and run

```
pip install -r requirements.txt
```

This will install all of the required packages needed to run SkillsGuild.

The web app uses several python packages to operate and these packages will have to be installed as a result, there are several guides outlining how to install python packages such as the python packaging user guide (https://packaging.python.org/en/latest/tutorials/installing-packages/). The packages SkillsGuild requires are:

- flask (https://pypi.org/project/Flask/)
- pandas (https://pypi.org/project/pandas/)
- ibm_watson (https://pypi.org/project/ibm-watson/)

To run the web app, ensure the source code has been downloaded and extracted, ensure all the dependencies have been resolved in the source code's location. If all these have been satisfied then the web app should be ready for start up and operation.

2.2 Starting the Web App

Once the web app has been set up with all its dependencies resolved, the web app can be started by executing the web_app.py file in the source code folder, this can be done from the command line terminal by navigating to the source code directory from the terminal and entering the command:

```
python web_app.py
```

It can also be executed by opening web_app.py using an IDE such as VSCode (https://code.visualstudio.com/download) and executing it from within the IDE.

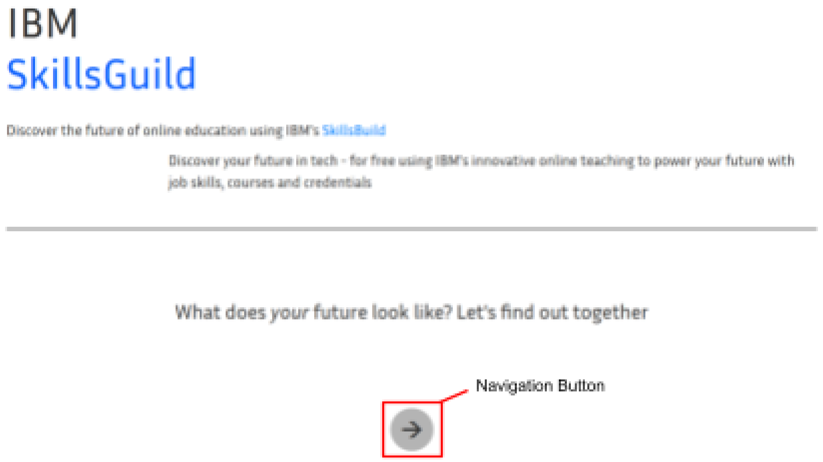
It will take a few seconds for the web app to start up but there will be a message in the terminal stating that the system is running when it finishes starting up. There will be a message in the terminal stating:

```
Running on http://127.0.0.1:5000
```

Once this message is displayed, open a tab in a web browser and input `http://127.0.0.1:5000` into the URL search bar, upon entering this URL, the web app will be loaded with the landing page being displayed in the browser. The web app is now running and is operational.

2.3 Navigating the Web App

Each page of the app contains at least one circular button with an arrow, these are the main mode of navigation for the web app, they are typically found at the bottom of the page. If there are multiple possible pages that can be navigated to, these buttons are labelled, denoting what page interacting with the button will load. If a button is unlabelled, it denotes that interacting with the button is the only way to progress through the web app from the current page. Every page in the web app is accessible through these buttons.



Discover the future of online education using IBM's SkillsBuild

Discover your future in tech - for free using IBM's innovative online teaching to power your future with job skills, courses and credentials

What does your future look like? Let's find out together

Image of the web app's landing page, highlighting the navigation button

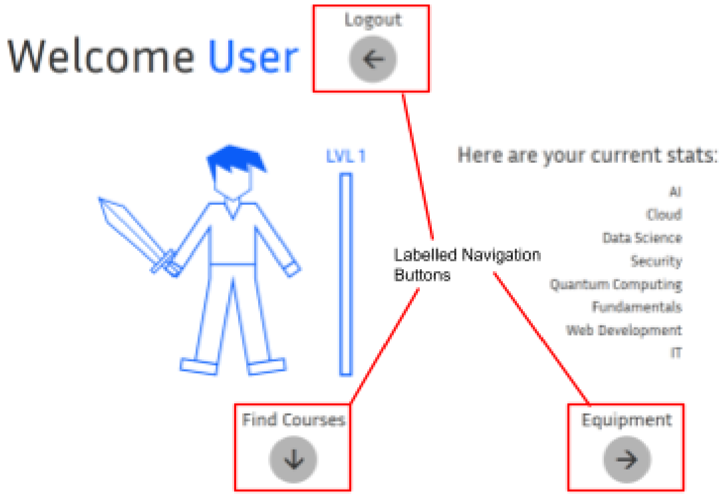


Image of the web app's profile page, highlighting the multiple labelled navigation buttons

The only exceptions to this circular button navigation system are the login and sign up pages, they both contain circular buttons allowing for navigation from the login page to the sign up page and vice versa, however to navigate to the main web app pages, the user must enter their login details or sign up details and interact with the corresponding 'Login' and 'Sign Up' buttons. This will redirect the user to the relevant user profile page or the sign up sequence.

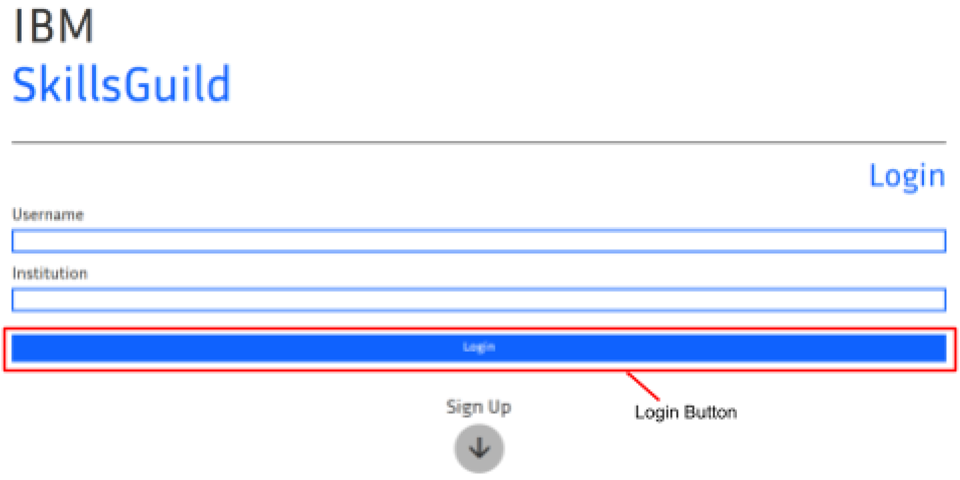
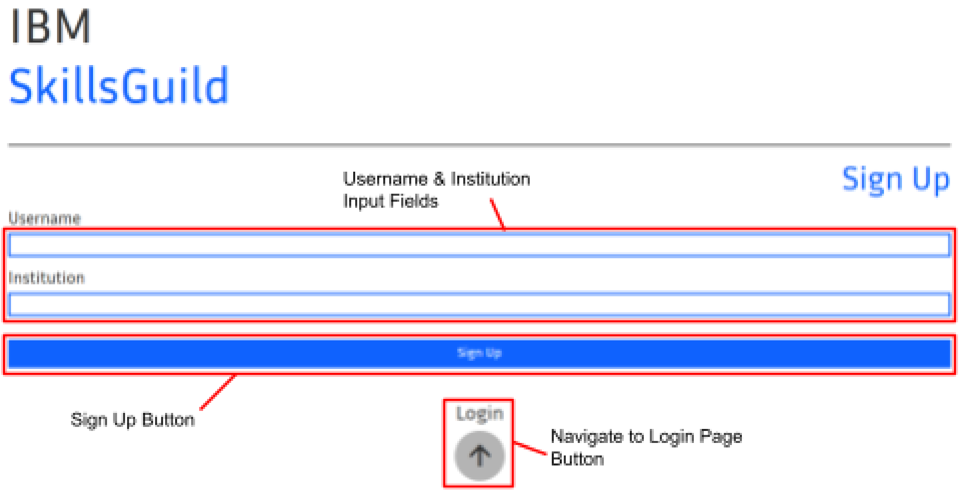


Image of the web app's login page, highlighting the exceptional login button

2.4 Signing Up & Logging In

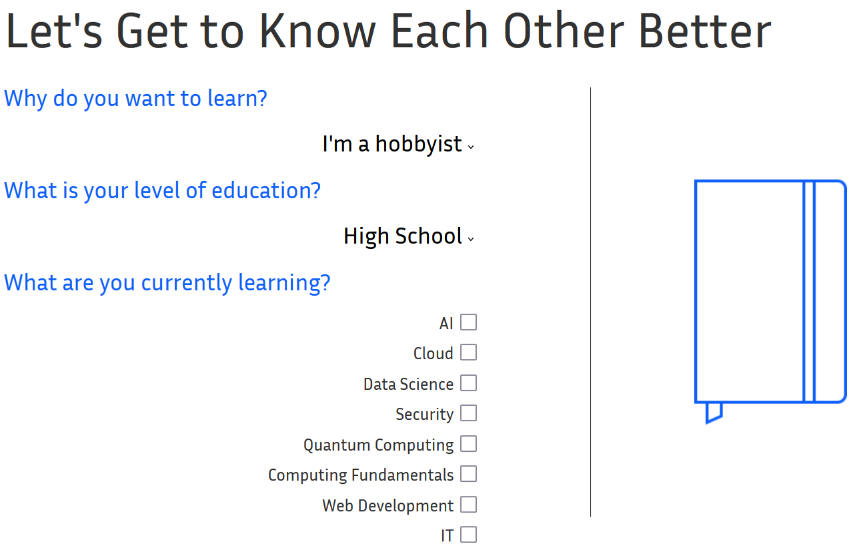
Usage of the web app's main functionalities requires the user to be logged in with an account, this is done via either the login or sign up pages. Interacting with the navigation button from the landing page will redirect the user to the login page, the sign up page can be accessed through interaction with the 'Sign Up' navigation button at the bottom of the login page. The login page can be returned to from the sign up page by interacting with the 'Login' navigation button at the bottom of the sign up page.

Signing up is for users who don't already possess a SkillsGuild account or wish to create a new one. This is done by entering the new account's username into the 'Username' box and the new account's institution into the 'Institution' box of the sign up page. If these details don't already match with an existing account's details, then the user will be redirected to the sign up sequence page; however if they do, the user will not be redirected and a warning message will be displayed above the sign up details boxes expressing that the details already belong to an existing account.



Annotated image of the sign up page

The sign up sequence page consists of 4 simple questions for the system to better understand the user, the first 2 require the user to select their education aspiration and their level of education from a drop-down box, the final 2 require the user to select the specific topics they are currently learning and wish to learn by interacting with several checkboxes, each corresponding to a different topic. Once the user has completed answering the question, they can finalise signing up using the navigation button at the bottom of the page, this creates the account and redirects the user to their new associated profile page.



Let's Get to Know Each Other Better

Why do you want to learn?

I'm a hobbyist ⌄

What is your level of education?

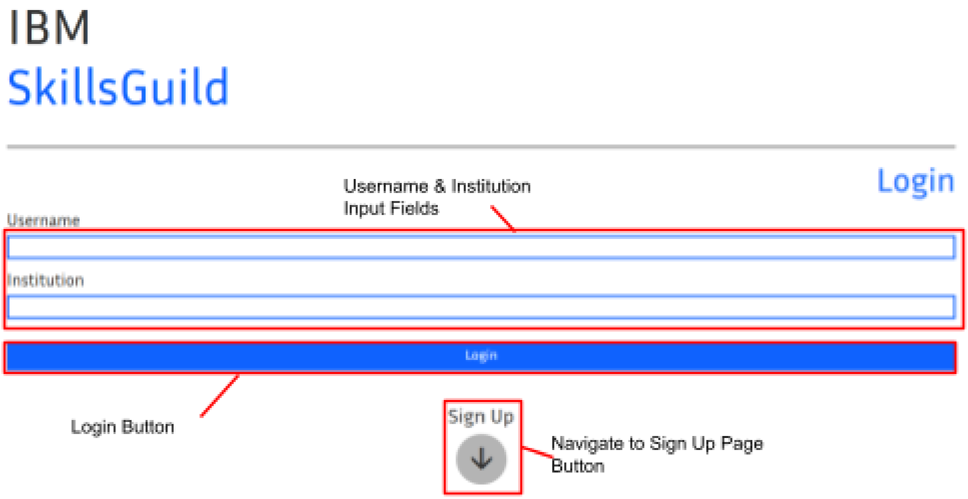
High School ⌄

What are you currently learning?

AI ☐
Cloud ☐
Data Science ☐
Security ☐
Quantum Computing ☐
Computing Fundamentals ☐
Web Development ☐
IT ☐

Image of the sign up sequence page

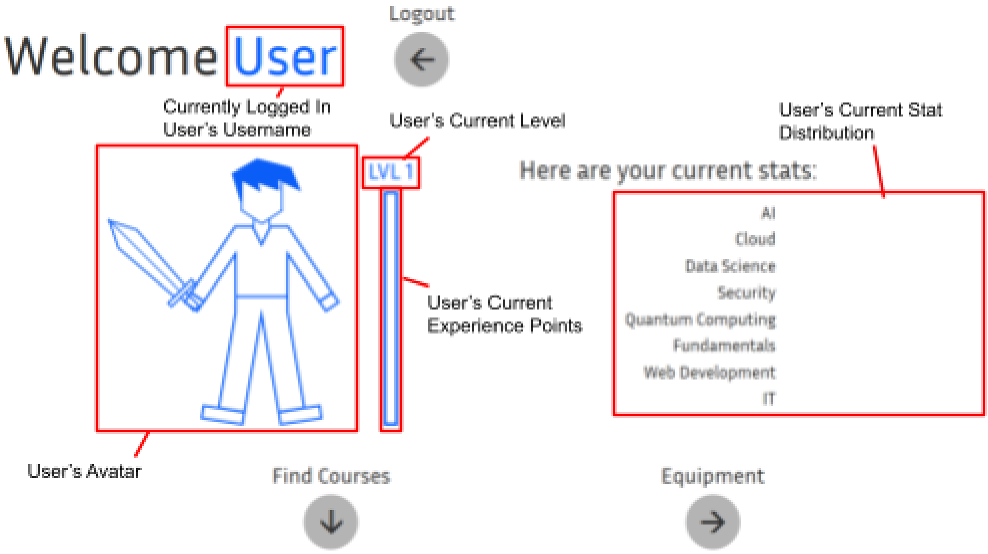
Logging in is for users who already possess a SkillsGuild account and wish to access the web app using this pre-existing account. This is done by entering the desired account's username into the 'Username' box and the account's institution into the 'Institution' box of the login page. These details should both match the details used to create the account. Once input, interacting with the ‘Login’ button will redirect the user to the associated profile page if the details are correct and match an existing account’s details; however if they don’t, the user will not be redirected and a warning message will be displayed above the login details boxes expressing that the details were incorrect.


IBM
SkillsGuild
Login
Username & Institution Input Fields
Username
Institution
Login
Login Button
Sign Up
Navigate to Sign Up Page Button

Annotated image of the login page

2.5 Understanding the Profile

The profile is the main page of the web app, it outlines the user’s progress and acts as a central navigation page to access the web app’s various functionalities. It contains an image of the user’s avatar, a virtual representation of the user in the SkillsGuild world, it also displays a bar with the user’s current level at the top. The bar shows how much experience the currently has and how much they need to reach the next level, the area filled blue denotes the amount of experience the user currently has and the white area show how much more experience they need to completely fill the bar, thus reaching the next level: resetting the user’s experience and incrementing their level by 1. There are various containers displaying the user’s stats, there are 8 bars, each representing a different topic and each stat has a bar next to it, the length of this bar denotes the amount of points the user has is each topic, this us a representation of the user’s progress and ability in each of these aspects. When a user creates a new account, they start at level 1 with 0 experience and 0 stat points in each stat.



Annotated image of a new user’s profile page

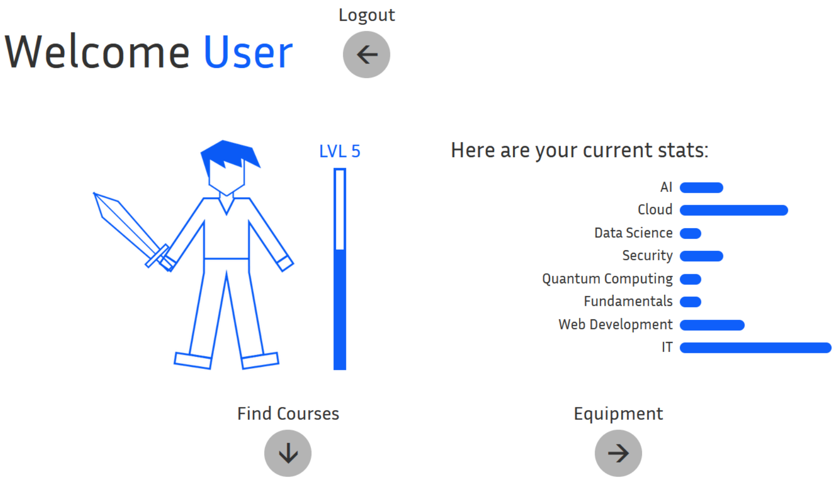


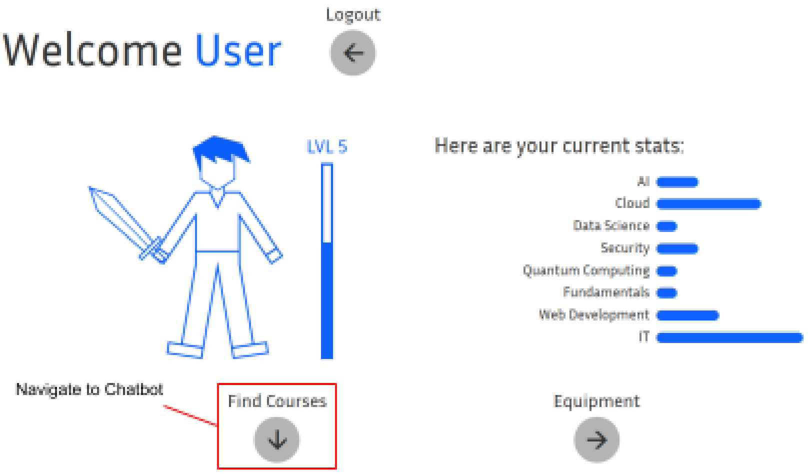
Image of an experienced user's profile page

2.6 Using the Chatbot

The chatbot offers a way to get course recommendations and learn about the courses offered from IBM SkillsBuild. You can also learn about careers and what skills are used to help you decide what you want to learn about. This section will go into depth about the chatbot, its functionalities and how to use it.

2.6.1 Accessing the Chatbot

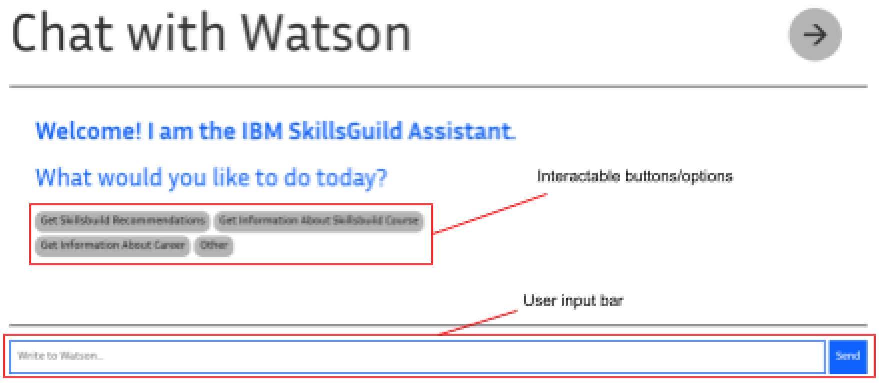
Once the user has logged in or signed up they will then be able to get recommendations. The chatbot helps the user find IBM SkillsBuild courses that are suitable to their requirements. To access the chatbot, the user can select the 'Find Courses' button to be redirected to the chatbot.



Annotated image of how to access the chatbot.

2.6.2 Interacting with the Chatbot

When the page loads, the user is greeted with a welcome message and the communication with the chatbot can begin. The user can choose to type their response or they can select one of the prompt buttons provided by the chatbot. The chatbot will try to guide the user to a solution through these buttons and interactions however the user may choose to enter their question or command directly.



Chat with Watson

Welcome! I am the IBM SkillsGuild Assistant.

What would you like to do today?

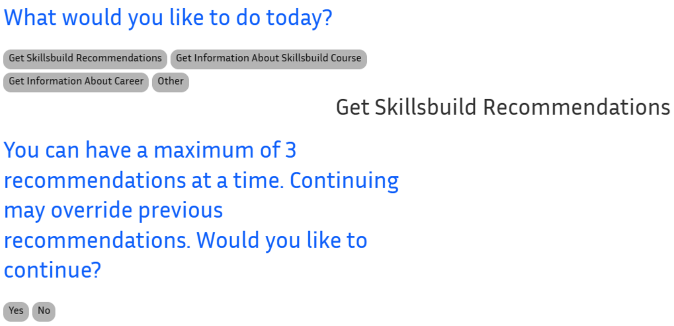
Get Skillsbuild Recommendations
Get Information About Skillsbuild Course
Get Information About Career
Other
Interactable buttons/options
User input bar
Write to Watson...
Send

Annotated image of how to interact with the chatbot.

This following section breaks down each option provided by the assistant and what is performed. The welcome message contains four options: '**Get SkillsBuild Recommendations**', '**Get Information About SkillsBuild Course**', '**Get Information About Career**', and '**Other**'. These options are the current features provided by our assistant.

2.6.3 Get SkillsBuild Recommendations

If you select the '**Get SkillsBuild Recommendations**' button, you will then get a warning message informing you that you may override recommended courses. If this is your first time using the assistant then you have nothing to worry about. However, if you have recommendations already then you may overwrite these. You can only have 3 recommendations at a time to prevent you from assigning all the SkillsBuild courses straight away.



What would you like to do today?

Get Skillsbuild Recommendations
Get Information About Skillsbuild Course
Get Information About Career
Other

Get Skillsbuild Recommendations

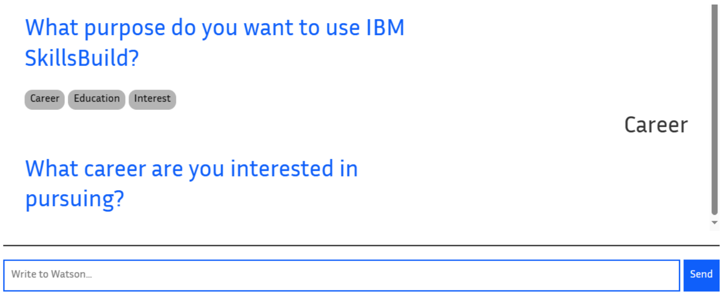
You can have a maximum of 3 recommendations at a time. Continuing may override previous recommendations. Would you like to continue?

Image of the chatbot warning the user that this action may override their current recommendations.

You can choose to proceed or cancel. If you select '**No**', then you will be greeted again and can perform a different task. If you select '**Yes**' the chatbot will continue to help you with recommending courses. The chatbot will ask what is your purpose for using IBM Skills Guild. This will help provide suitable courses to you. The options include: '**Career**', '**Education**', and '**Interest**'.

2.6.3.1 Career Recommendations

If you select '**Career**', you have the option to enter the career you would like to pursue. If this is recognised then it will provide recommendations to you and assign them to your profile. These are presented as hyperlinks which will direct the user to the relevant course.

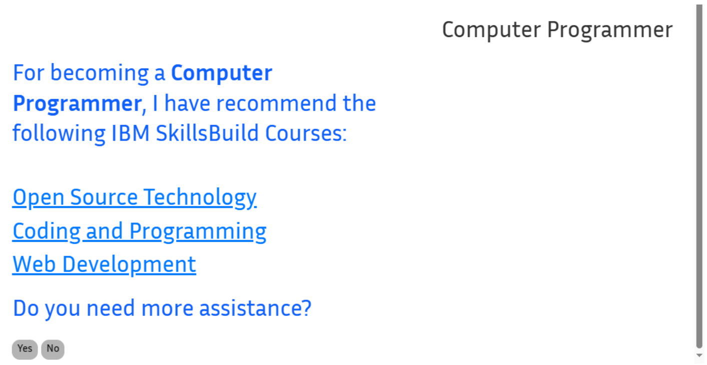


What purpose do you want to use IBM SkillsBuild?

Career

What career are you interested in pursuing?

Image of the chatbot when the user selects 'Career'.



Computer Programmer

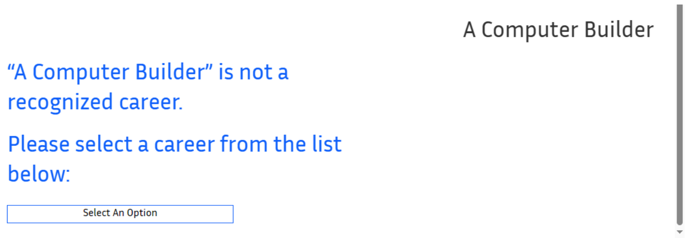
For becoming a **Computer Programmer**, I have recommend the following IBM SkillsBuild Courses:

Open Source Technology
Coding and Programming
Web Development

Do you need more assistance?

Image of the chatbot providing recommendations for a 'Computer Programmer'.

Alternatively if the input is not recognised, either due to a typo or not being supported, a drop down menu is provided to allow the user to select the most relevant course. They can click one of the provided options and this will then recommend the courses for the selected career.



A Computer Builder

"A Computer Builder" is not a recognized career.

Please select a career from the list below:

Image of the chatbot not recognising a career.

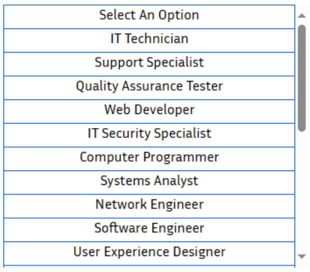
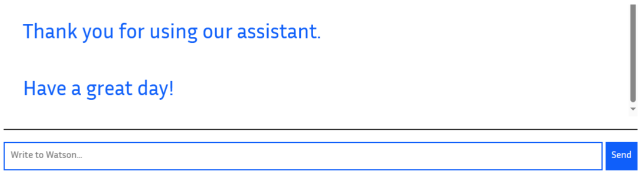


Image of the expanded chatbot and some of the offered courses.

When courses are recommended the chatbot asks if the user requires more assistance. If they select '**Yes**', the chatbot loops back to the welcome message. If the user selects '**No**', the user receives the following message:



Thank you for using our assistant.

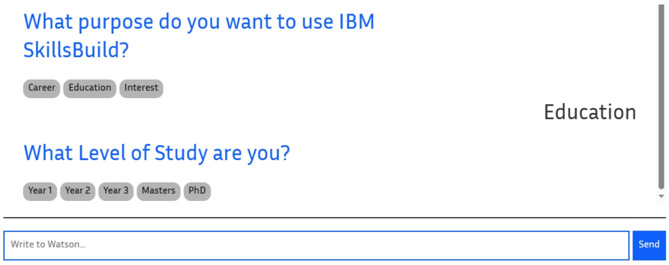
Have a great day!

Image of the chatbot ending the interaction.

The user input bar is still visible since the user can still resume their interaction with the chatbot if they require more assistance.

2.6.3.2 Education Recommendations

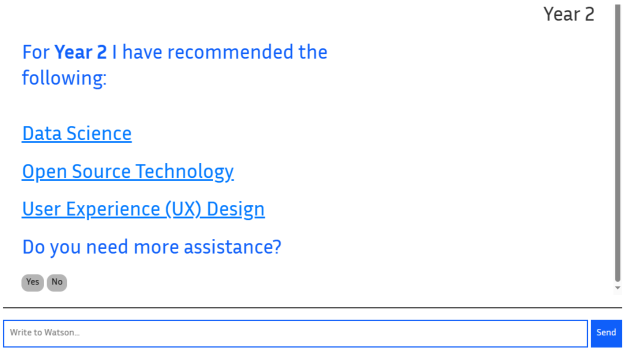
If the user would like to get recommendations relevant to their current year of study in Computer Science then this section will help recommend courses for them. If the user selects '**Education**', the chatbot will prompt the user to select their current level of study. If the user selects one of these prompts or types in their response then they will receive recommendations for suitable courses.


What purpose do you want to use IBM SkillsBuild?

Education

What Level of Study are you?

Image of the chatbot when the user selects 'Education'.


Year 2

For **Year 2** I have recommended the following:

[Data Science]

[Open Source Technology]

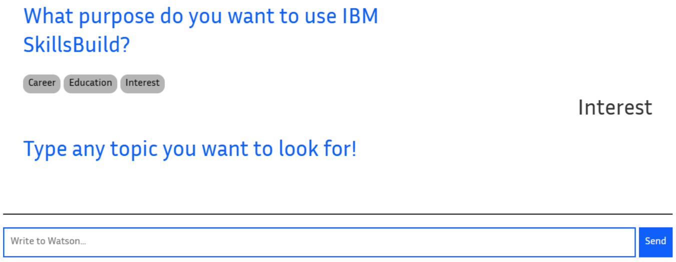
[User Experience (UX) Design]

Do you need more assistance?

Image of the chatbot when the user selects 'Year 2'.

2.6.3.3 Interest Recommendations

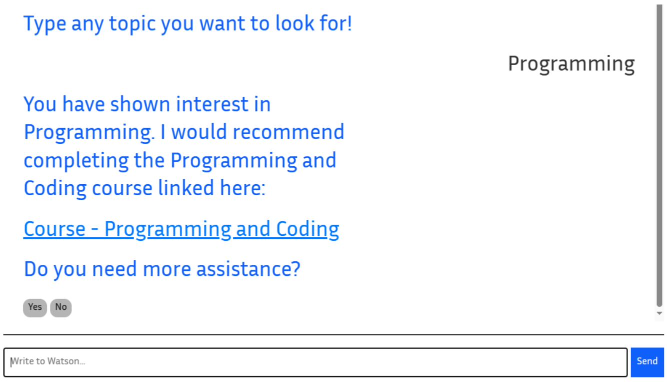
This section will help users with a specific interest in a computer science topic. If the user selects '**Interest**', the chatbot will expect a written user input to search for the topic the user is interested in learning about. If a course matches then a recommendation is made for the matched course. If not, then the user is informed.


What purpose do you want to use IBM SkillsBuild?

Interest

Type any topic you want to look for!

Image of the chatbot when the user selects 'Interest'.


Type any topic you want to look for!

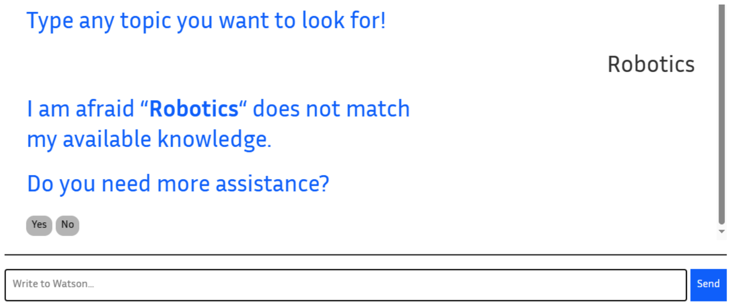
Programming

You have shown interest in Programming. I would recommend completing the Programming and Coding course linked here:

Course - Programming and Coding

Do you need more assistance?

Image with an example of the user being recommended a course based on an interest in programming.



Type any topic you want to look for!

Robotics

I am afraid “**Robotics**“ does not match my available knowledge.

Do you need more assistance?

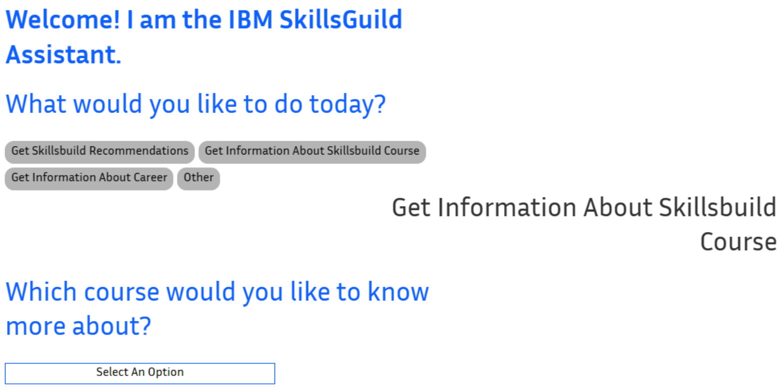
Image with an example of the user’s interest not being recognised.

2.6.4 Get Information

Beside making recommendations the chatbot has also had some additional functionality implemented which can help users learn more about careers including what they entail and the related salary but also information about IBM Skills Build Courses. These features are implemented to offer functionality that will not override their current recommendations. These options are provided in the greeting message.

2.6.4.1 Information About SkillsBuild Course

A user can select the ‘**Get Information About SkillsBuild Course**’ button in the welcome message to receive information about specific IBM SkillsBuild Courses. They are provided with a drop down menu that lists all the currently implemented courses. The user can select one of these and that will then provide information to the user.



Welcome! I am the IBM SkillsGuild Assistant.

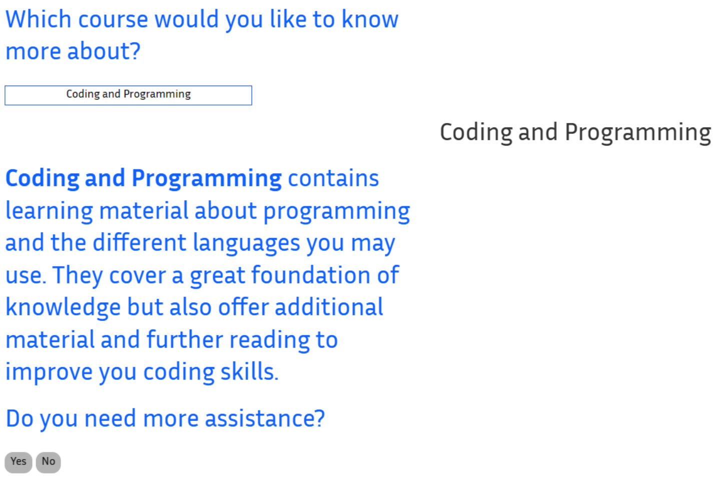
What would you like to do today?

Get Skillsbuild Recommendations
Get Information About Skillsbuild Course
Get Information About Career
Other

Get Information About Skillsbuild Course

Which course would you like to know more about?

Image of the Chatbot Responding to ‘Get Information About Skills Build Course’.



Which course would you like to know more about?

Coding and Programming

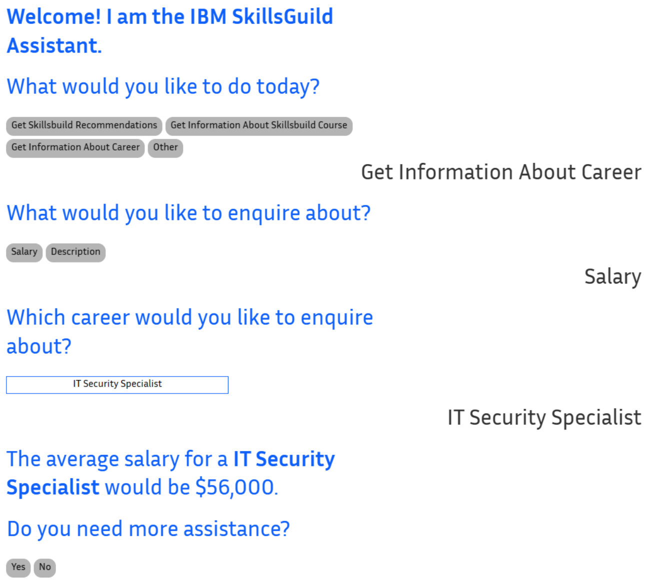
Coding and Programming contains learning material about programming and the different languages you may use. They cover a great foundation of knowledge but also offer additional material and further reading to improve you coding skills.

Do you need more assistance?

Image of the Chatbot Providing Information about Coding and Programming.

2.6.4.2 Information About Career

A user can select the ‘**Get Information About Career**’ button in the welcome message to receive information about a specific career. The chatbot can provide the average salary for a specific career but can also provide a description of the career. This is structured very similarly to section 2.6.4.1 but includes one additional question.



Welcome! I am the IBM SkillsGuild Assistant.

What would you like to do today?

Get Skillsbuild Recommendations
Get Information About Skillsbuild Course
Get Information About Career
Other

Get Information About Career

What would you like to enquire about?

Salary

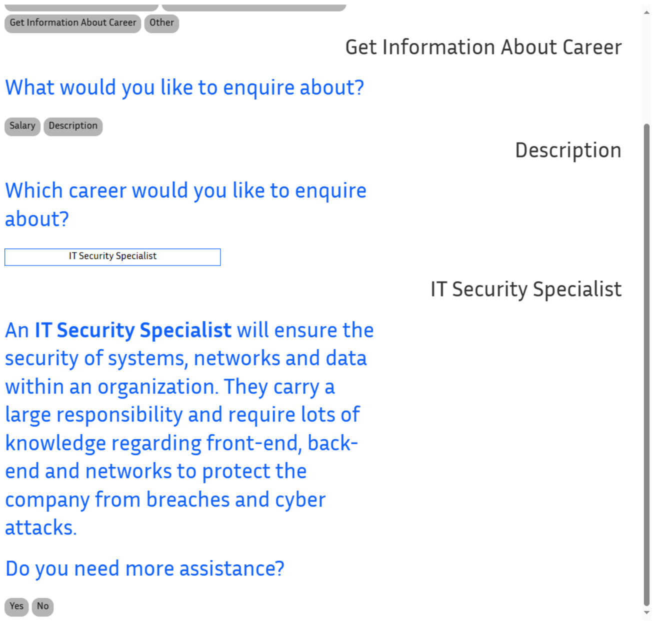
Which career would you like to enquire about?

IT Security Specialist

The average salary for a **IT Security Specialist** would be $56,000.

Do you need more assistance?

Image of the chatbot providing the salary for an IT Security Specialist.



Get Information About Career

What would you like to enquire about?

Description

Which career would you like to enquire about?

IT Security Specialist

An **IT Security Specialist** will ensure the security of systems, networks and data within an organization. They carry a large responsibility and require lots of knowledge regarding front-end, back-end and networks to protect the company from breaches and cyber attacks.

Do you need more assistance?

Image of the chatbot providing the description of an IT Security Specialist.

2.6.5 Other Assistance

Additionally to the recommended options from the chatbot, if the user would like to skip this and ask a specific question then they can do so. This can be done at any point even if they have been using the buttons. Alternatively, selecting ‘**Other**’ will allow them to feel more in control.

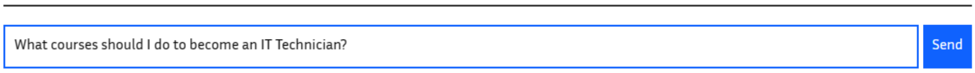
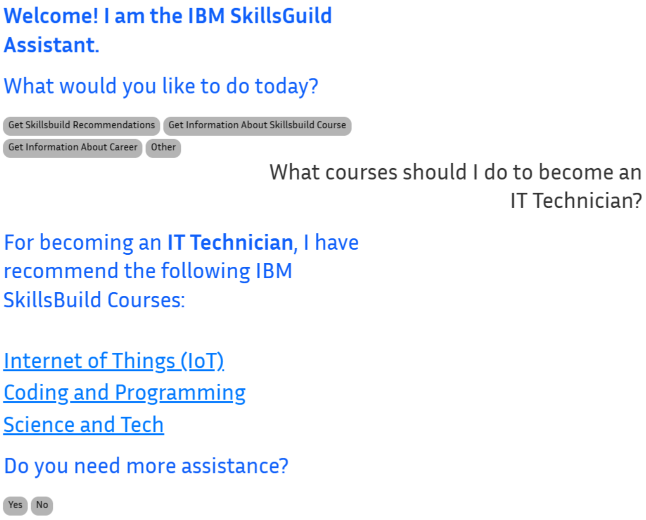

What courses should I do to become an IT Technician?

Image of the user input during the chat.


Welcome! I am the IBM SkillsGuild Assistant.

What would you like to do today?

Get Skillsbuild Recommendations
Get Information About Skillsbuild Course
Get Information About Career
Other

What courses should I do to become an IT Technician?

For becoming an **IT Technician**, I have recommend the following IBM SkillsBuild Courses:

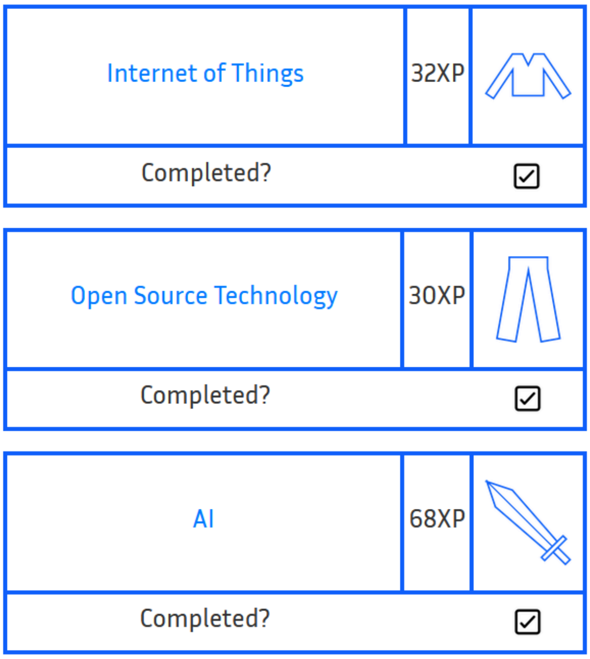
Internet of Things (IoT)
Coding and Programming
Science and Tech

Do you need more assistance?

Image of the output provided by the chatbot.

2.7 Completing Courses

Once the user has some courses recommended to them by the chatbot, they will be able to view their currently recommended courses from the courses page. The courses page is navigable to from the equipment page. There it displays a list of boxes, each denoting a different SkillsBuild course, they display the name of the course, the amount of experience completing the course would award the user and the type of equipment they'll get if they complete it. The names of each course within each course box are interactable and link to their corresponding SkillsBuild course.


Internet of Things 32XP
Completed? ☑

Open Source Technology 30XP
Completed? ☑

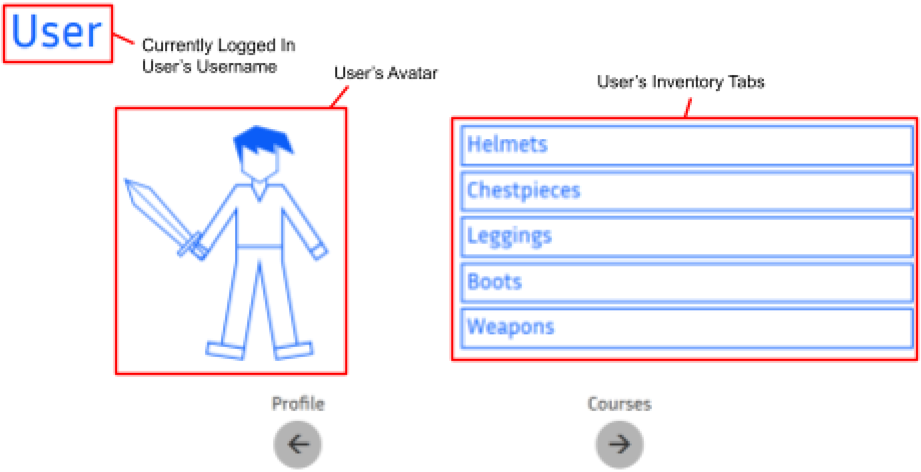
AI 68XP
Completed? ☑

List of the user's recommended courses in the courses page

Each course box has a checkbox at the bottom that, when interacted with, will mark the course as complete; when a course has been marked as complete, it will disappear from the courses page, its assigned experience will be added to the user's current experience level, if the user has enough experience (100 or more), their level will increase, the course's assigned stats will be added to the corresponding user's stats and a random item of the assigned type that the user doesn't currently have in their inventory will be added to their inventory. These changes can be checked from the equipment and profile pages. More recommendations can be gotten from further interactions with the chatbot.

2.8 Understanding the Inventory and User Equipment

The equipment page is where the user can view their inventory of items they have gained from completing courses and equip them onto their avatar. The page displays the user's username, avatar as well as a list of tabs denoting the different equipment types that can be viewed and equipped. Interacting with one of these tabs opens a drop down list listing all the items the user currently has of that type. Interacting with a different tab closes the currently open tab if there is one and opens the new tab the user has interacted with. If the user has no items of a given type, that type's tab will open and state that the user has no items of that type.



Annotated image of the equipment page

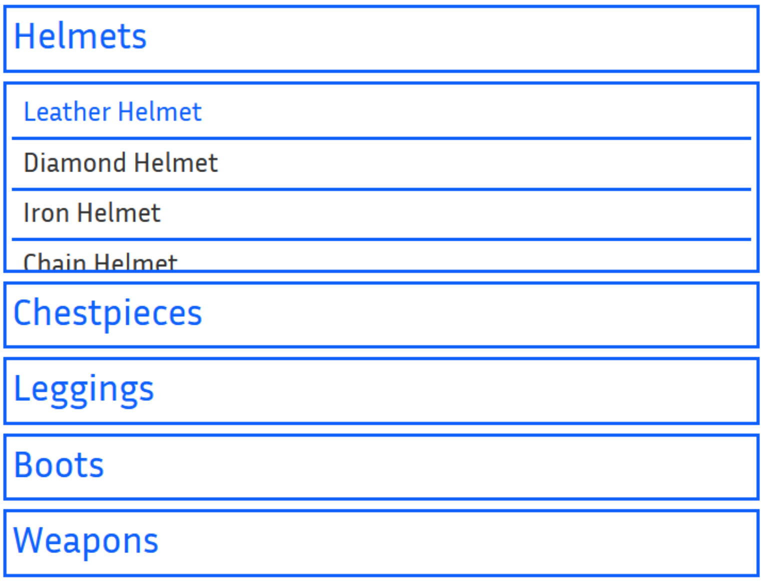


Image of the user's inventory with the helmets tab open

The user's currently equipped items are denoted in blue, unequipped items are black. Items can be equipped by interacting with the item's name from the inventory, if no items of that type are currently equipped then the chosen item will be equipped and turn blue. If an item of the given type is already equipped and a different item of the same type is interacted with, the currently equipped item of the given type will be unequipped and turn black while the newly chosen item will be equipped and turn blue.

2.9 Troubleshooting, Feedback & Contact Details

If you encounter problems or have any questions regarding using our application, the team is available to help. Below are the contact details for each member of our team, along with the sections they worked on. If you can direct your query to the relevant member the problem can be solved more efficiently and effectively. If you are unsure of who to contact, do not worry. Each member of the team will be happy to assist and direct you to the most suitable person for the problem.

Non-Technical Support:

Nathan Lam - qwsk85@durham.ac.uk

Technical Support - Chatbot:

Liam McCarthy - bchh34@durham.ac.uk

Isaac So - dcmc71@durham.ac.uk

Technical Support - Database:

Will Farmer - hdtk63@durham.ac.uk

Nate Oppy - xxcs76@durham.ac.uk

Technical Support - Application:

Calvin Dantis - zmzf47@durham.ac.uk

Should you try to troubleshoot on your own, the easiest method of troubleshooting involves creating a new user and then using the well-documented backend functions to ensure that all pathways are operational. For example, if you want to check that the correct stats are being applied to a user's profile upon completing a course you could navigate to the *completion* function.

If you are noticing issues with Watson then refer to section 3.8 which discusses hosting Watson Chatbot. Here you can view the structure of the chatbot. Try and notice where you are encountering issues and find the relevant area in the menu that will contain the problem. Here you can break down the problem and attempt to fix this. If you need additional help, please contact Liam or Issac.

3 - Developers’ Guide: System Maintenance & Future Developments

3.1 Set up the Development Environment

The web app uses python and as a result requires python 3.9 or above to operate, a web browser is also required to load the web pages the app is composed of and is required to interact with the system and the web app has been tested with, and is compatible with, all major web browsers. It should be noted that testing on macOS has not been possible. We recommend using a Windows based machine for this application to ensure functionality.

An internet connection is also required to interact with the main feature of the web app so ensure the device the web app will run on has a stable internet connection.

The web app uses several python packages to operate and these packages will have to be installed as a result. The packages SkillsGuild requires are:

- flask (https://pypi.org/project/Flask/)
- pandas (https://pypi.org/project/pandas/)
- ibm_watson (https://pypi.org/project/ibm-watson/)

To run the web app, ensure the source code has been downloaded and extracted, and ensure all the dependencies have been resolved in the source code’s location. If all these have been satisfied then the web app should be ready for start up and operation.

The source code contains all the major components of the web app with the exclusion of the chatbot’s prompt and route engineering as this belongs on a proprietary IBM platform. The source code consists of: the web app entry point and page router webapp.py, the backend functionalities that queries and updates the database backend.py, a templates directory with all the HTML5 pages the web app uses, a static directory containing all the static assets such as JavaScript scripts, CSS stylesheets and images the web app uses, and a database directory containing all the database files.

The web app can be run by executing the webapp.py file from the source code directory and navigating to the URL: `http://127.0.0.1:5000` in a web browser.

3.2 System API

SkillsGuild offers several request routes to interact with the system and this section outlines each path and offers a description of the request and its requirements.

3.2.1 GET Web Pages

These requests outline the retrieval of the

Request	GET /
Route	http://127.0.0.1:5000/
Returns	login.html
Description	Returns the initial landing page for the web app as a HTML5 page.

Request	GET /login
Route	http://127.0.0.1:5000/login
Description	Returns the login page for the web app as a HTML5 page.

Request	GET /signup
Returns	signup.html

Route	http://127.0.0.1:5000/signup
Description	Returns the sign up page for the web app as a HTML5 page.

Request	GET /create
Route	http://127.0.0.1:5000/create
Returns	**User Logged In:** create.html **User Not Logged In:** GET /signup
Description	Returns the account creation page for the web app as a HTML5 page, requires details to be input and submitted on the sign up page first before this page can be returned. If new account details have not been submitted via the sign up page before accessing the account creation page, a redirection to the sign up page will be made via GET /signup along with a flash message specifying that the user needs to sign up first before accessing this feature.

Request	GET /profile
Route	http://127.0.0.1:5000/profile
Returns	**User Logged In:** profile.html **User Not Logged In:** GET /login
Description	Requires a user to be currently logged in via the login or sign up page, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature. The userID of the currently logged in user will be used to query this user's stats, level, experience amount, and name, these are then used to update the profile page before returning it as a HTML5 page.

Request	GET /chatbot
Route	http://127.0.0.1:5000/chatbot
Returns	**User Logged In:** chatbot.html **User Not Logged In:** GET /login
Description	Requires a user to be currently logged in via the login or sign up page, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature. Returns the chatbot page for the web app as a HTML5 page.

Request	GET /recommendations
Route	http://127.0.0.1:5000/recommendations
Returns	**User Logged In:** recommendations.html **User Not Logged In:** GET /login
Description	Requires a user to be currently logged in via the login or sign up page, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature. The userID of the currently logged in user will be used to query this user's currently recommended courses and their relevant information: name, experience amount, item type, and

	URL. These are used to update the recommended page before returning it as a HTML5 page.

Request	GET /equipment
Route	http://127.0.0.1:5000/equipment
Returns	**User Logged In:** equipment.html **User Not Logged In:** GET /login
Description	Requires a user to be currently logged in via the login or sign up page, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature. The userID of the currently logged in user will be used to query this user's name, the items currently in their inventory, and the items they currently have equipped, these are then used to update the equipment page before returning it as a HTML5 page.

Request	GET /courses
Route	http://127.0.0.1:5000/courses
Returns	**User Logged In:** courses.html **User Not Logged In:** GET /login
Description	Requires a user to be currently logged in via the login or sign up page, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature. The userID of the currently logged in user will be used to query this user's currently recommended courses and their relevant information: name, experience amount, item type, and URL. These are used to update the courses page before returning it as a HTML5 page.

3.2.2 Backend Requests

Request	GET /logout
Route	http://127.0.0.1:5000/logout
Returns	GET /login
Description	Resets the currently logged in user to null and returns the login page for the web app as a HTML5 page.

Request	POST /trysignup
Route	http://127.0.0.1:5000/trysignup
Body	form-data:
Returns	**Success:** GET /create **Failure:** GET /signup

Body:

```
{
     "username":"<account username>",
     "password":"<account password>"
}
```

Description	Requires a body in the form of form data defining a username and password, these are then checked against the database of existing users and their login details. If this combination of username and password already exists then a redirection to the sign up page will be made via GET /signup along with a flash message signifying that this combination of details already belongs to a user. Otherwise these details will be temporarily saved and a redirection will be made to the creation page via GET /create.

Request	POST /trylogin
Route	http://127.0.0.1:5000/trylogin

Body	form-data:
	`{`
	`"username":"<account username>",`
	`"password":"<account password>"`
	`}`

Returns	**Success:** GET /profile **Failure:** GET /login
Description	Requires a body in the form of form data defining a username and password, these are then checked against the database of existing users and their login details. If this combination of username and password don't already exist then a redirection to the login page will be made via GET /login along with a flash message signifying that this combination of details doesn't belong to a user. Otherwise, these details will be used to save the userID associated with these details as the currently logged in user and a redirection to the profile page via GET /profile will be made.

Request	POST /submitprof
Route	http://127.0.0.1:5000/submitprof
Returns	GET /profile
Description	Uses the temporary account details saved by POST /trysignup to create a new account with the associated details and uses these to save the userID associated with these details as the currently logged in user and a redirection to the profile page via GET /profile will be made. The temporary details are also deleted.

Request	POST /session_start
Route	http://127.0.0.1:5000/session_start
Returns	sessionID
Description	Starts a session with the IBM Watson chatbot and returns the sessionID it generates to store locally in the opened page. This is recalled every time the chatbot page is reloaded.

Request	POST /assistant_message
Route	http://127.0.0.1:5000/assistant_message

Body	form-data:
	`{`
	`"user_input":"<user message>",`
	`"sessionID":"<ID of the chatbot session>"`

```
}
```

Returns

```
//A list of dictionaries where each dictionary could be either of the 3:
messages = [
     {“u_content”:<user message>},
     {“b_content”:<chatbot message>},
     {“option_flag”:True}
]
```

Description

This function requires a session ID and the user_input to be sent as parameters. This data is then posted to IBM Watson using Assistant V2 API. This is dependent on predefined values stored in webapp.py. A large dictionary is returned which is then processed to determine what data is being received. More specifically there are 2 types, ‘text’ and ‘option’. Text is a simple message which is the main reply offered from watson. These can be information or recommendations. Option is when a prompt is suggested from the assistant and these are handled differently to be able to display them as buttons. Once all the response has been processed these are then appended to a dictionary which declares whether it is a normal message or an option message for future processing prior to displaying. This array is then returned once all of the response has been handled. If a recommendation is detected in a text content message, then the addRecommendedCourses function is called to handle the recommendation. This will check that a user is logged in and then call the function passing the ID of the current user and the ID of the course being recommended. This can handle multiple courses being recommended.

Request	POST /equip
Route	http://127.0.0.1:5000/equip

Body

raw JSON:

```
{
     “item_id”:”<ID of the desired item>”,
     “item_type”:”<Type of the desired item>”
}
```

Returns

Success:

```
{
     “status”:”success”,
     “message”:”Item <itemID> equipped successfully”
}, 200
```

Failure:

```
{
     “status”:”error”,
     “message”:”Item ID and Type is required”
}, 400
```

User Not Logged In: GET /login

Description

Requires a body in the form of JSON defining the ID and type of the item to equip, after checking if these have been provided, a check is done to see if there is a user currently logged in and if there is, the item associated with the input itemID is equipped to the currently logged in user and a JSON outlining a successful request is returned, otherwise a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature.

Request	POST /comp
Route	http://127.0.0.1:5000/comp
Body	raw JSON:

```
{
     “course”:”<ID of the completed course>”,
     “course_type”:”<Item type associated with the course>”
}
```

Returns

Success:

```
{
     “status”:”success”,
     “message”:”Course <course> successfully completed”
}, 200
```

Failure:

```
{
     “status”:”error”,
     “message”:”Course ID is required”
}, 400
```

User Not Logged In: GET /login

Description	Requires a body in the form of JSON defining the ID and type of the item to equip, after checking if these have been provided, a check is done to see if there is a user currently logged in and if there is, the stats and experience associated with the course of the input courseID is added to the currently logged in user’s stats and experience, a random item the currently logged in user doesn’t own of the specified course type is added to the currently logged in user’s inventory and a JSON outlining a successful request is returned. If the currently logged in user already has all the items of the given type then nothing is added to their inventory. If there is no currently logged in user: a redirection to the login page will be made via GET /login along with a flash message specifying that the user needs to be logged in first before accessing this feature.

3.3 Backend Functionality

Name	readDF
Arguments	filepath (str): The path to the desired table in the database colNames (list of strings): The names of each column to be read in the specified table
Returns	pandas Dataframe: The table read from the database, stored as a Dataframe
Summary	Reads a specified table from the database and stores it as a pandas Dataframe for manipulation

Name	writeCSV
Arguments	df (pandas Dataframe): The updated table to replace the current one in the database filepath (str): The path to the desired table in the database
Returns	N/A
Summary	The procedure writes the new table directly to a CSV file, overwriting the old table. Hence, nothing is returned.

Name	userExist

Arguments	username (str): The given username of the user password (str): The given password of the user
Returns	userID (int or bool): The userID of the user if they exist (int), or False if the user does not exist (bool)
Summary	Checks if the combination of username and password exists for login

Name	getRecommended
Arguments	userID (int): The value of the primary key for a record in the users table
Returns	response (array of dictionaries): An array with a dictionary of values about the recommended courses
Summary	Get all courses that are recommended to the user, as well as their stats and associated xp gain and equipment reward type for completion

Name	getName
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	username (str): The user’s username
Summary	Gets a user’s username from their userID

Name	getStats
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	stats (dict): A dictionary of all the user’s stats
Summary	Retrieves a user’s stats from their userID and returns them for display

Name	getLevel
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	level (int): The user’s current level
Summary	Retrieves a user’s level from their userID and returns it for display

Name	getXP

Arguments	userID (int): The value of the primary key for the record in the users table
Returns	exp (int): The user’s current experience value
Summary	Retrieves a user’s current experience value from their userID and returns it for display

Name	getInventoryItems
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	inventory_items (list of strings): The name of all items in the user’s inventory
Summary	Retrieves a user’s inventory from their userID and returns all items present for equipping/unequipping items

Name	getInventory
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	inventory_dict(dictionary of lists of strings): Dictionary of item types, where each item type is a key with values of the items of that type in the *user’s inventory*. E.g. “weapon”:“sword” is one such key-value pair.
Summary	Retrieves all item types and items in a user’s inventory from their userID and returns them for equipping/unequipping items

Name	getEquipment
Arguments	userID (int): The value of the primary key for the record in the users table
Returns	equip_dict (dictionary of lists of strings): A dictionary containing *all of the user’s items* sorted by their item type. e.g. “weapon”:”sword” is one such key-value pair. NOTE: an item can be in a user’s inventory and still be included in this dictionary.
Summary	Retrieves all item types and items that a user owns from their userID and returns them for equipping/unequipping items. It is also used to check if a user already owns an item so that they are never rewarded with an item they already possess.

Name	getRandomItem
Arguments	userID (int): The value of the primary key for the record in the users table itemType (string): The type of item the user will be receiving, E.g. weapon. In this case the user’s random item will always be a weapon they do not already have.
Returns	randomItem (str): The name of the random item they will receive. An empty string is returned if all items of this type are owned by the user.
Summary	Generates a random item not owned by the user and gives it to them as reward for completing course

Name	equipItem
Arguments	userID (int): The value of the primary key for the record in the users table itemName (str): The name of the item to be equipped itemType (str): The type of item to be equipped
Returns	N/A
Summary	Receives information about an item and equips it. The information is written directly to the users table so nothing is returned.

Name	addNewItem
Arguments	userID (int): The value of the primary key for the record in the users table itemType (str): The type of the new item
Returns	flag (int): 0 for successful completion
Summary	Uses getRandomItem to generate a new item and adds it to their collection.

Name	addCourseStats
Arguments	userID (int): The value of the primary key for the record in the users table courseID (int): The value of the primary key for the record in the courses table
Returns	N/A
Summary	Retrieves all stat rewards from completing a course and applies it to the user’s profile. All experience points are added and the level of the user is updated if the experience totals over 100. In this case the experience total decrements by 100. The data is written directly to the users table so nothing is returned.

Name	completeCourse
Arguments	userID (int): The value of the primary key for the record in the users table courseId (int): the value of the primary key for the record in the courses table
Returns	N/A
Summary	Adds a completed course to the completed course table. If the course was in the recommended courses table for the same user then it is removed so users aren’t recommended courses they have already completed.

Name	addRecommendedCourse

Arguments	userID (int): The value of the primary key for the record in the users table courseId (int): the value of the primary key for the record in the courses table
Returns	N/A
Summary	Adds a course to the user's recommended list by writing directly to the recommended_courses table. Hence, nothing is returned.

Name	addNewUser
Arguments	newUserData (list of strings and integers): The stats, username, password, inventory and items of a new user.
Returns	df (pandas Dataframe): The users table newUserID (int): userID (int): The value of the primary key for the new record in the users table
Summary	Receives default data to assign to the newly created user and generates a new userID for them

Name	signUp
Arguments	username (str): The username of the user being created password (str): The password of the user being created
Returns	userID: The value of the primary key for the new record in the users table
Summary	Allows a user to sign up to SkillsGuild, using the AddNewUser function to generate a new userID and writing the new user's information to the users table.

3.4 Technical Walkthrough

This section outlines a technical walkthrough for users with a greater knowledge of computer systems to navigate and use the technical functionalities of the SkillsGuild web app. The specific processes and their purposes for each aspect of the web app are outlined in this section.

3.4.1 Navigating the Web App

Every page of the web app contains a circular button with an arrow typically at the bottom of the page, this is the web app's main method of navigation. These are hyperlink elements that contain a hyperlink to a specific route defined in webapp.py.

Of the pages that inherit from the layout.html page (index.html, login.html, signup.html, profile.html, equipment.html and courses.html), the route each button points to is defined in webapp.py as it is passed into the page via the Jinja templating engine. Pages that have these default navigation buttons have a dictionary passed into them as they are rendered via flask, this dictionary called nav_items contains several keys which are each named according to the label that should be displayed above the button. The value assigned to each label (key) is a tuple of 2 strings, the first is a single arrow character defining what character should be displayed within the button, the second is the route the button's hyperlink should point to.

```
@app.route('/login')
def login():
    return render_template('login.html', nav_items={'Sign Up':('□', 'signup')})
```

Example navigation item parameter passing into page rendering

When a page is rendered, the number of keys in the nav_items dictionary is calculated and this determines how many button objects should be created through Jinja, the key of each button is used to label the button by filling in the content of a label span with the key's name, the content of the button is replaced by the arrow character in the tuple and the hyperlink 'href' is replaced by the route from the tuple.

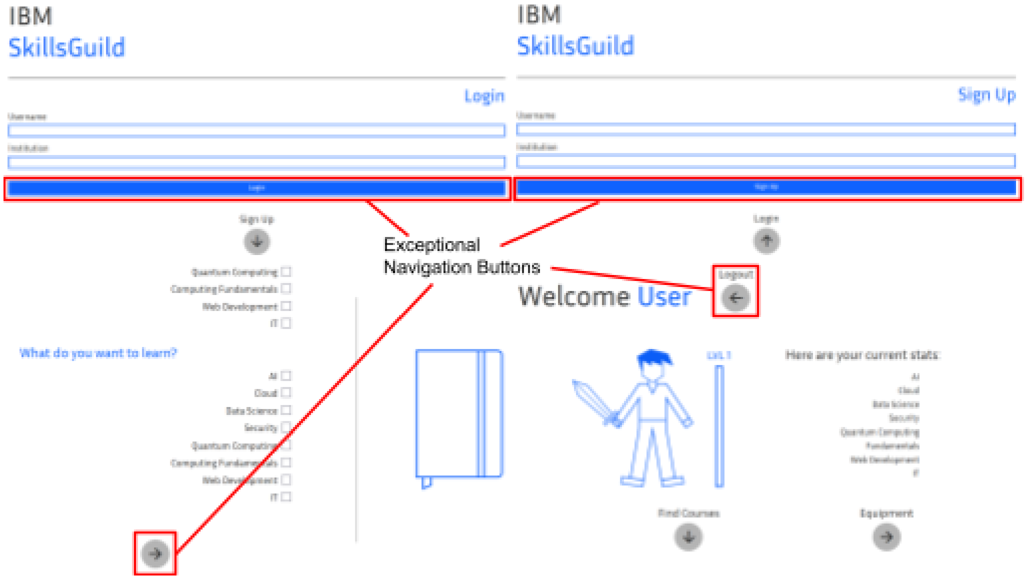
```
{% for key, item in nav_items.items() %}
            <div class="grow">
                <span class="label">{{ key }}</span>
                <a class="circle-button" href="{{ item[1] }}">
                    <span>{{ item[0] }}</span>
                </a>
            </div>
{% endfor %}
```

The iteration and replacement functionality for navigation items using Jinja

The other pages inherit from base.html and don't use the default navigation button system, they specify their own circular buttons with hyperlink references manually within the html.

The exceptions to this are the login and sign up buttons on the login and sign up pages and the circular button the account creation page. These are all form submission buttons that route to a location as the form is submitted and this route redirects to another page once the form has been processed. As a result the routes for these are not defined in the buttons themselves but the form definitions and the post request routes they point to. Other exceptions are the logout button on the profile page which also has a hardcoded route to /logout, the navigation button on the chatbot page which has a hardcoded route to the recommendations page and the navigation button on the recommendations page with a hardcoded route to the profile page.

The only exceptions to the circular button navigation system are the login and sign up pages, they both contain circular buttons allowing for navigation from the login page to the sign up page and vice versa, however to navigate to the main web app pages, the user must enter their login details or sign up details and interact with the corresponding 'Login' and 'Sign Up' buttons. This will redirect the user to the relevant user profile page or the sign up sequence.


IBM
SkillsGuild
Login
IBM
SkillsGuild
Sign Up
Exceptional Navigation Buttons
Welcome User
Logout
What do you want to learn?
Here are your current stats:
Find Courses
Equipment

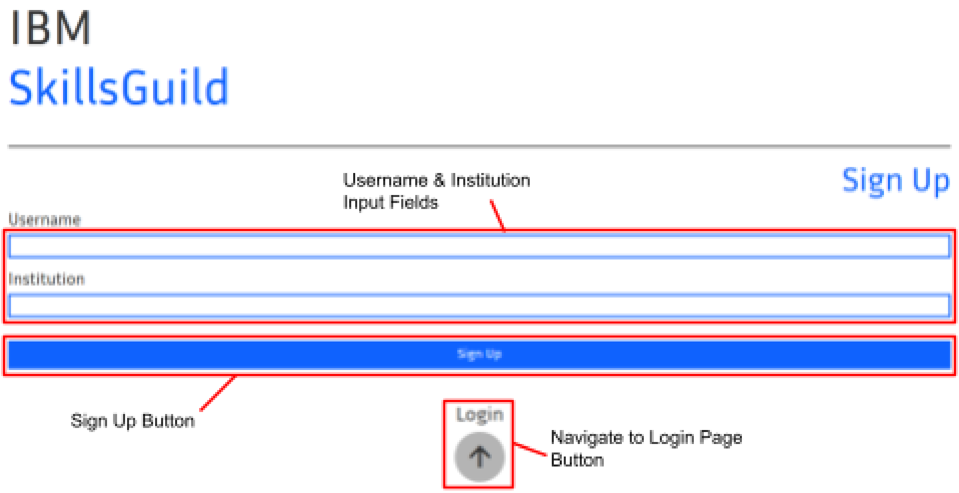
Annotated image of the login, sign up, account creation, and profile page highlighting the exceptional navigation buttons

Button	Route
Index page navigation button	/login
Sign up navigation button on login page	/signup
Login navigation button on sign up page	/login
Login form button	/trylogin

Sign up form button	/trysignup
Account creation navigation button	/submitprof
Logout button on profile page	/logout
Chatbot button on profile page	/chatbot
Equipment button on profile page	/equipment
Chatbot page navigation button	/recommendations
Recommendations page navigation button	/profile
Profile button on equipment page	/profile
Courses button on Equipment page	/courses
Equipment button on courses page	/equipment

3.4.2 Signing Up & Logging In

Signing up and logging in are done through a form on the sign up and login pages respectively, each of these forms consist of 2 text fields and a submit button, the top input field is for the username and bottom is for the institution which functions as a password. These are required fields and must contain information to be submittable via their respective sign up and login submit buttons. When the fields have been filled and the submit button has been interacted with, the page will create a /trylogin or /trysignup request for login or sign up respectively.

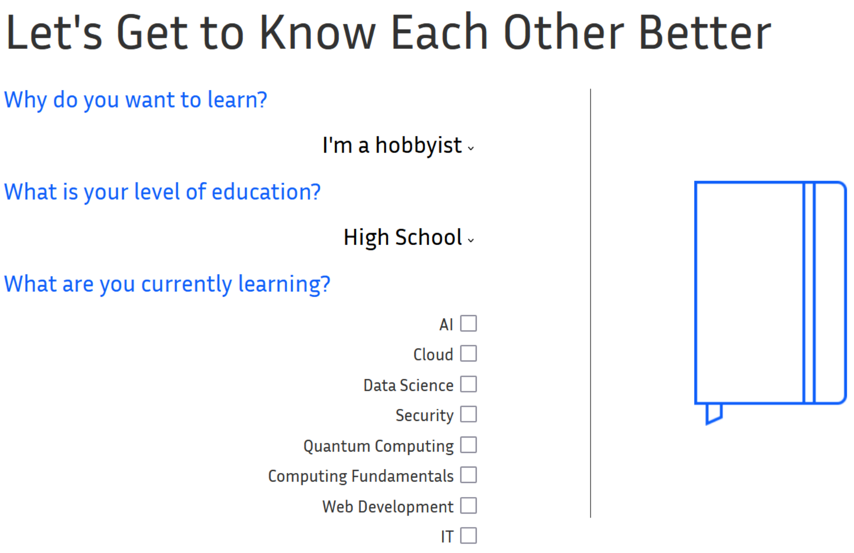


Annotated image of the sign up page

/trylogin is a POST request that takes the form details of the input username and password and uses them to make a call to a backend function: userExist, by passing in the form's username and password. This function returns whether or not the combination of username and password already belongs to an entry in the user.txt database file, if the details already exist and belong to a user then the corresponding userID is returned whereas false means it doesn't exist. If the details don't belong to an existing user then the login has failed and a redirection is made to /login along with a danger flash message specifying that the login failed and that the details need to be checked, this flash message is then rendered to the login page above the input form when the html is rendered. If a user with the specified details does exist and a userID is returned by userExist then this userID is stored to the global CURRENT_USER variable to denote that this is the userID of the currently logged in user and a redirection is made to /profile.

/trysignup is similar in that it also uses userExist with the signup form's username and password fields to check whether a user with the corresponding details already exists in the users.txt database. If they do then the sign up has failed and a redirection is made to /signup along with a danger flash message specifying that the sign up failed and that the details already exist, this flash message is then rendered to the sign up page above the input form when the html is rendered. If the details don't belong to an existing user then the username and password from the form is saved in the global NEW_USER variable to denote a temporary storage of the details as the account is being created, this is stored as a tuple of (username, password). Then a redirection is made to /create to initiate the account creation process along with the new temporary user details.

The account creation page consists of a form with 4 questions and input fields, the first 2 are dropdown boxes and the final 2 are a series of 8 checkboxes each. The circular navigation button at the bottom functions as a form submission button which will post the form when interacted with, making a POST request to /submitprof. This uses the details expressed in the NEW_USER tuple and calls the signUp function using them which creates a new entry in the user.txt database file with the input details specified in the temporary details NEW_USER variable. signUp also returns the userID of the newly created account and this is stored in CURRENT_USER to denote that the newly created account is the currently logged in user and the NEW_USER variable is reset to null to prevent additional account creation errors. Finally, a redirection is made to /profile.



Let's Get to Know Each Other Better

Why do you want to learn?

I'm a hobbyist ⌄

What is your level of education?

High School ⌄

What are you currently learning?

AI ☐
Cloud ☐
Data Science ☐
Security ☐
Quantum Computing ☐
Computing Fundamentals ☐
Web Development ☐
IT ☐

Image of the sign up sequence page

Several pages (profile, chatbot, recommendations, equipment and courses) require the CURRENT_USER variable to assigned a valid userID so for all of these routes, if an attempt is made to access them while CURRENT_USER is null, a redirection will instead be made to /login with a danger flash message expressing that the user must be logged in first. A similar check is made for /create however it checks if NEW_USER is empty instead as an account can't be created if no details have been specified.

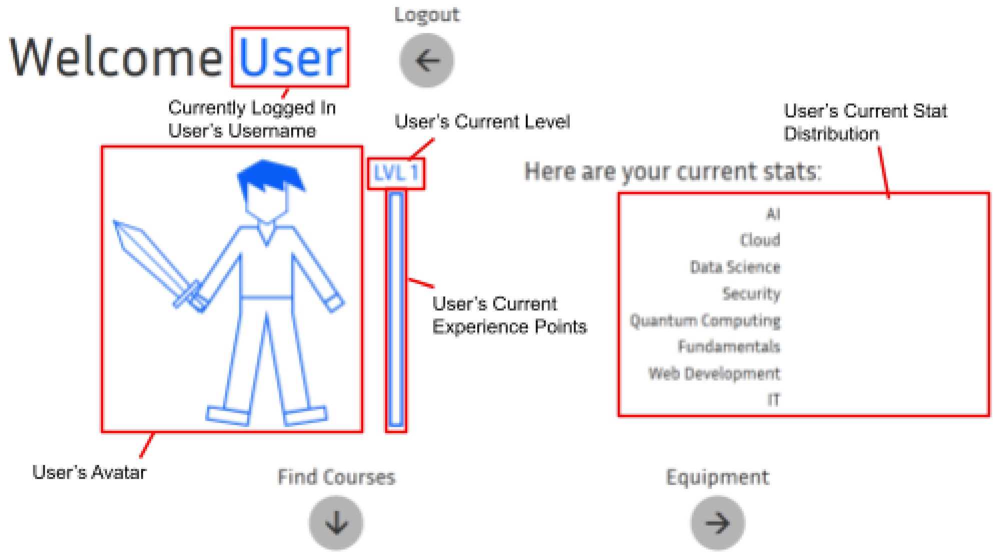
```
global CURRENT_USER
    if not CURRENT_USER:
        flash(('You need to login first before accessing SkillsGuild', 'danger'))
        return redirect(url_for('login'))
```

Example of a check for a currently logged in user

3.4.3 Understanding the Profile

When /profile is requested it first checks if a user is currently logged in, if they are then a series of calls to backend functions are made to retrieve user information to be rendered to the page. These include: getStats, getLevel, getXP and getName. Each of these functions are passed CURRENT_USER to retrieve the information for this userID and they are all saved in unique variables. These variables are then passed into the rendering of the profile.html page to update it with the user's information.

Stats is a dictionary of 8 elements with each key corresponding to a different stat and each stat's value is a number, these numbers are used to determine the width of the stat bars using inline CSS styling and replacing the width as a percentage of the full bar (e.g. an AI stat of 60 would cause the AI stat bar to have a width of 60%). The level value is a number and is used to update the number next to the 'LVL' text above the user's experience bar, denoting their current level. The experience value is a number between 0 and 100, this is used to replace the inline CSS styling height value of the experience bar to be the specified percentage of the full experience bar. The name is a string of the user's associated username and this is used to replace the name displayed at the top of the page next to 'Welcome'.



Annotated image of a new user's profile page

This page also contains 3 navigation buttons: logout, chatbot and equipment. Logout routes to /logout and resets the CURRENT_USER value to null to denote there are no currently logged in users and then it redirects to /login. Both the chatbot and equipment are standard navigation buttons.

3.4.4 Using the Chatbot

When you are redirected to the chatbot page. A few processes will automatically occur. The first being the calling of /session_start. This will generate a sessionID that is stored locally on the user's device. This is used to maintain the conversation with the chatbot. Due to financial limitations the sessions are limited to 5 minutes before expiring. With this in mind, the chatbot has been designed to be as efficient at performing recommendations as possible. This will be discussed in more detail shortly.

Upon succession of the sessionID being received, the chatbot will call /assistant_message with a blank value for the user_input. This will trigger the greeting message provided by our Watson Assistant, and begin the conversation. The response format from watson is a dictionary with many smaller dictionaries embedded within. This means that the specific keys are used to represent different types of responses from the chatbot. This is where we are able to process the output and display it in a way that improves user experience and encourages the user to be satisfied before the session expires.

Text responses are basic and just appended to the messages array ready to be output. Options can come in one of two formats. If there are five or less options then these will be displayed as interactable buttons that will post the textual contents straight back to /assistant_message. This reduces user involvement and in turn improves efficiency and reduces the likelihood of user errors preventing them from achieving their required action. When there are more than 5 options, they are no longer provided as buttons but instead a dropdown which can be scrolled through. Both the buttons and the drop down will post the value of the option to the /assistant_message making the system simpler.

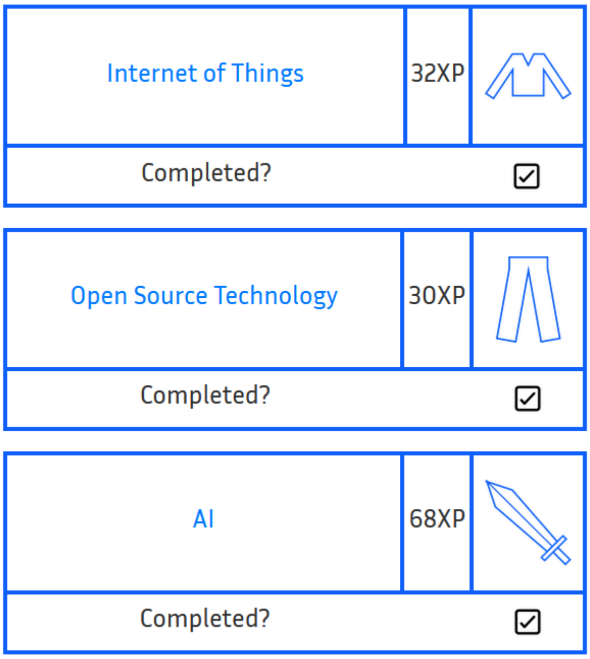
Recommendations are identified through the use of links in the responses. When the response is received if it is a textual message and the link is within the message then it then calls the /addrecommendedcourse function and passes the current user and the courseID to be appended to the database. This all happens automatically. In the future it would be better to allow the user to have a menu appear for removing, swapping and adding recommended courses that are provided by the chatbot.

The chatbot itself has an embedded menu built into it that will indefinitely loop to ensure that the user can keep receiving help without refreshing the page. This structure performs subtraction on the user's behalf to understand what the user is wanting to do and to be able to perform the necessary task. It is completely optional for the user to follow this structure. The Watson Assistant is trained to handle interrupt requests such as when the user wants to perform a specific action. Upon finishing the interrupt task, the chatbot returns to the menu loop to offer more services if required. Each action offered by Watson Assistant has training values to help determine when it is required. So you can be in the middle of getting information about a SkillsBuild course then switch to getting recommendations for an IT Technican.

3.4.5 Completing Courses

When /courses is routed to, it first checks if there is a currently logged in user and if there is then it calls getRecommended using CURRENT_USER to get a list of the currently logged in user's recommended courses in the form of a list of dictionaries where each dictionary specifies the: name, experience value, item type, URL, and courseID of each of the recommended courses. This list is then passed into the rendering of the courses page.

If the html page uses a check via Jinja to see if the list of recommended courses is empty (i.e. the user has no recommended courses), if it is then the page will display a large message stating that there are no recommendations and that the chatbot should be used to get recommendations. If the list isn't empty and the user has recommendations, then for each course in the list, a table is rendered in the centre of the courses page with the corresponding courseID stored as a data value for the table. Within this table are 2 rows, the first row has 3 columns, the first states the name of the course and uses a hyperlink to make the name interactive where interacting with it will link to the corresponding SkillsBuild course page. This is done by replacing the element content with the name of the course stored in the dictionary and assigning it a 'href' value according to the course's URL in the dictionary. The second column displays the course's experience value next to an 'XP' label. The third column displays an image relevant to the item type of the course, this is done using if else statements in Jinja to check the value assigned to the course's type in the dictionary and the relevant image is chosen from the static assets image directory if a type match is found.



Internet of Things | 32XP
Completed? ☑

Open Source Technology | 30XP
Completed? ☑

AI | 68XP
Completed? ☑

List of the user's recommended courses in the courses page

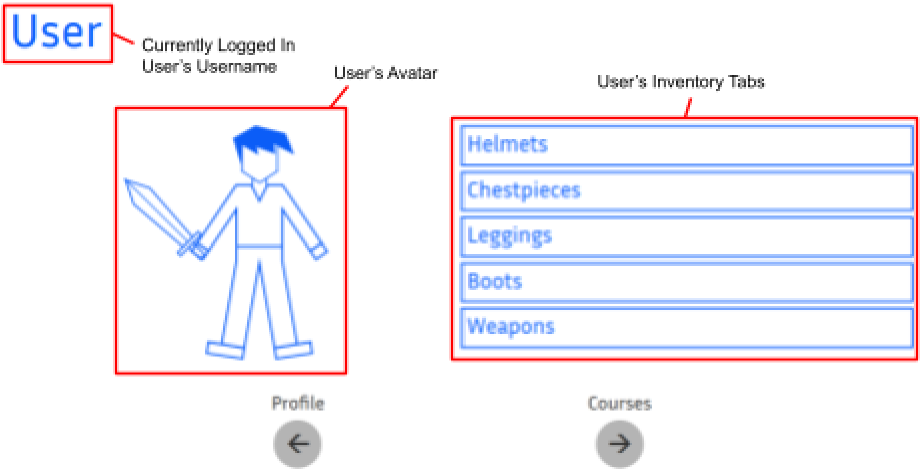
The second row consists only of a label and a checkbox, this is used to determine the completion of a course. The checkbox is an image type input element that displays an image of a checkbox, it also stores data of the corresponding course's item type and ID. The script.js JavaScript file in the static assets js directory has a function that runs when this checkbox is clicked, it takes the ID and type stored as data in the completion box and uses them to make an AJAX request to /comp in webapp.py, sending a JSON body with the course's ID and type with it. /comp then retrieves these arguments and uses them to make calls to backend functions relevant to course completion. It first checks if there is a currently logged in user and uses this to call addCourseStats, addNewItem and completeCourse.

These add the course's stats to the currently logged in user's stats and update their database entry accordingly, it also randomly selects an item of the specified course type from the equipment database file that is not currently in the inventory of the currently logged in user as is specified in their entry of the users database file. The user's experience is also updated and if it exceeds 100, the user's level is incremented and the user's experience is reduced by 100 to denote a level up. The completeCourse function moves the courseID from the user's recommended courses list as is stored in the recommended database file to the completed courses list in the same file, this is done by taking the input courseID and parsing the database list, the courseID is then removed from the courses list and appended to the completed list which is then stored back in a string format. Several checks are implemented surrounding how pandas manages databases.

After all these calls are made, a return JSON is sent denoting whether or not this request was successful, if an error has occurred then this is output to the console, if it was successful however, a JavaScript function with jQuery is run that selects the table containing the course that was just marked as complete and removes it from the page's html, removing it from view.

3.4.6 Understanding the Inventory and User Equipment

The user's inventory is stored in their entry of the users.txt file in the database, this is a list of item names separated by '-'s, the user's currently equipped items for each item type is also stored here. There are 5 item types that an item can be categorised into and be equipped: weapon, helmet, chestpiece, leggings, and boots. When /equipment is requested, a check is done to see if there is currently a logged in user, if there is then 3 backend function calls are made using the CURRENT_USER userID: getName, getInventory, and getEquipment. These are used to retrieve the name, inventory and equipment associated with the currently logged in user's userID. Both the inventory and equipment is retrieved as dictionaries using the item types as keys. These variables are then passed into the rendering of the equipment page.



Annotated image of the equipment page

The name is used to display at the top of the page, for each item type, the corresponding item tab table is populated with rows where each row corresponds to a unique item in the user's inventory of that type. Each of these table data elements also store the corresponding itemID and item type as data values. As each item type has a defined tab and table in the html, every table is populated with the relevant inventory items. As the tables are being filled with items, they are each checked via Jinja to see if they are an item the user has currently equipped, if they are then they are given an additional class which makes them appear blue, denoting that they are equipped.

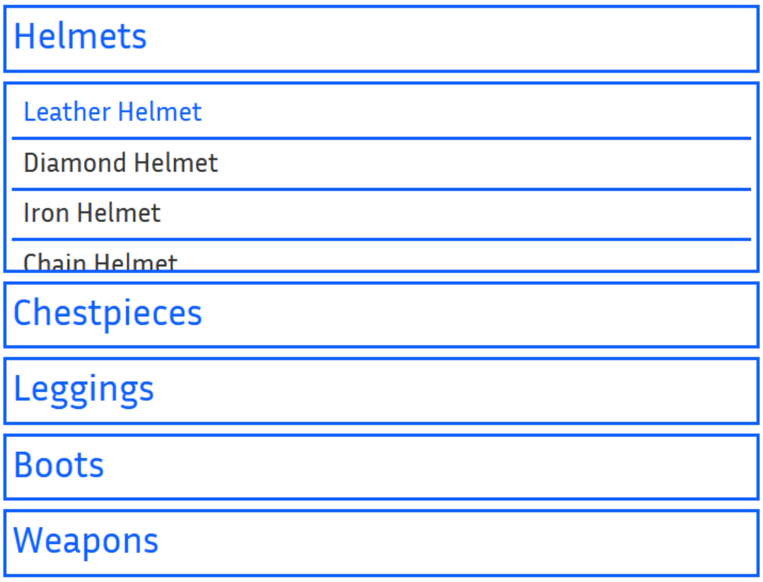


Image of the user's inventory with the helmets tab open

```
{% if inventory["helmet"] %}
      {% for helm in inventory["helmet"] %}
             <tr>
                 <td class="item {% if helm == equipment["helmet"] %}blue{% endif %}" data-item-id="{{ helm }}" data-item-type="helmet">{{ helm }}</td>
             </tr>
      {% endfor %}
{% else %}
      <tr>
             <td>No Helmets</td>
      </tr>
{% endif %}
```

HTML outlining helmet inventory table population using Jinja

Each of the item tabs are radio buttons that are styled to appear as tabs, this ensures that only 1 can be active at any given time. CSS is defined in the stylesheet that determines the visibility of the inventory tables that correspond to their item type tab, if a tab radio button is marked as checked then the tab content is made visible through ‘display:block’, otherwise the tables have their ‘display’ set to ‘none’, making them invisible. Each of the inventory tables have a set height and can be scrolled through ensuring every inventory is accessible. If the user has no items of a given type, the corresponding inventory table will have a single entry outlining that the user has no items of this type.

```
.tab input[type="radio"]:checked + label + .content {
    display: block;
}
.tab .content {
    display: none;
    padding: 5px;
    border: 3px solid #0f62fe;
    margin-bottom: .5rem;
}
```

CSS stylesheets entries determining inventory table visibility

Every item can be equipped, the user can only have a single item of each type equipped at any given time. Equipped items are denoted by their blue text which is assigned to them through the ‘blue’ class they each share. The script.js file defines functionality when an item in the inventory is interacted with, the itemID and type is retrieved from the data fields of the interacted item and these are used to make an AJAX request to /equip. /equip then extracts the itemID and type from the request, these are checked to see if they are valid, if they aren’t then an error is raised where a JSON is returned expressing the error. If the arguments are valid then a check is made to see if there is a currently logged in user, if there is then the item arguments and the CURRENT_USER userID are used to make a call to the equipItem backend function which updates the database to change the currently stored item of the given type in the database to the new itemID that has been provided. When this completes, a successful JSON is returned informing the AJAX request that it has been successfully executed. When the AJAX request receives the success message, an additional function is run that selects all the elements with the newly equipped item’s type and removes their ‘blue’ class (visually unequipping them), the element with the newly equipped item’s ID is then assigned the ‘blue’ class (visually equipping it).

3.5 Database

Table Name: Users	
Attribute Name	**Attribute Summary**
userID	The primary key for a record in the table. It is a unique identifier, so no records will have the same userID.
username	The username for a user. It is used when logging in and generated when signing up for the first time.
password	The password for a user. It is used when logging in and generated when signing up for the first time.
AI	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
dataScience	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
cloud	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
security	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
quantum	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
fundamentals	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
webDev	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
IT	One of the user's stats. It starts at 0, but completing certain courses can increase the stat.
weapon	The currently equipped weapon. This is shown in the user's profile.
helmet	The currently equipped helmet. This is shown in the user's profile
chestplate	The currently equipped chestplate. This is shown in the user's profile
leggings	The currently equipped leggings. This is shown in the user's profile
boots	The currently equipped boots. This is shown in the user's profile
inventory	All of the items currently owned by the user. It is a concatenated string of the names of each item separated by a hyphen.
level	The current level of the user. It is displayed in the user's

	profile.
exp	The current exp value of the user. Once the exp value reaches 100 it is reset to 0 and the user's level increments by 1.

The Users table is by far the largest and most used table in the database. We have userID as our primary key as it is a unique identifier for a record (in this case a user) in the database, so we can be certain which user's information and stats are being updated. The username and password are only used in the login and sign-up process, with the username also being displayed in the profile page. The stats (AI, dataScience, etc) are split up individually for atomicity and also to allow easier updating of stats after course completion. The same can be said for the equipped equipment (helmet, chestplate, etc): splitting them up ensures atomicity and also makes it easier to equip items, as this way we can ensure that a user can only equip a helmet into the helmet slot. However, to get inventory to function as intended we had to forgo atomicity. Our functions getInventory, equipItem, addNewItem and others all rely on the inventory being a singular string of names. These functions then split the names up into a list and interact with the Equipment table to identify their equipment type. The level and exp are integer values, with level having a minimum value of 1 and exp having a minimum value of 0 and a maximum value of 99.

Table name: Equipment	
Attribute Name	**Attribute Summary**
equipmentName	The primary key for a record in the Equipment table. It is a unique identifier for an item for lookup purposes. It is the name given to an item and is displayed in the inventory page.
type	The type of the item, one of either weapon, helmet, chestplate, leggings, or boots. This is used when equipping items or when giving a new item to the player.

The Equipment table is the smallest table in the database, containing only two attributes, but it is immensely necessary. The equipmentName is a primary key that also features as a foreign key in the users table. This creates a relation that ensures the user can not only equip the item but equip it into the correct slot, thanks to the type attribute. This is used when generating a random item to give to the user, as the backend function getRandomItem checks that the item is of the same type as that which has been requested.

Table name: Courses	
Attribute Name	**Attribute Summary**
courseID	The primary key for a record in the Courses table. It is a unique identifier for lookup and allows relations to be made with the Users table to help recommend courses and identify which courses have been completed.
courseName	The name of the course
AI	The stat bonus given to the user upon completing the course
dataScience	The stat bonus given to the user upon completing the course
cloud	The stat bonus given to the user upon completing the course
security	The stat bonus given to the user upon completing the course
quantum	The stat bonus given to the user upon completing the course
fundamentals	The stat bonus given to the user upon completing the course
webDev	The stat bonus given to the user upon completing the course

IT	The stat bonus given to the user upon completing the course
exp	The experience points granted to the user upon completing the course
equipmentType	The type of equipment the user receives upon completing the course
url	The URL to the course on SkillsBuild.

Sometimes courses have very similar names, and so we decided to add a courseID and a courseName even though there is a dependency between. This has been done to limit errors on the backend by potential spelling mistakes directing users to the wrong courses. Once again the stats have been split up to ensure atomicity and to make it easier to grant these boosts to the user's correct stats. The course's experience bonus is added to the user upon completion, as is a random item of the type stored in equipmentType. The URL is included to direct users to the course in SkillsBuild.

Table name: Recommended Courses	
Attribute Name	**Attribute Summary**
userID	The primary key for a record in the table. It is directly related to the user who possesses this userID in the Users table.
courseID	The courseID(s) of the course(s) recommended to the user
completedID	The courseID(s) of the course(s) completed by the user.

The Recommended Courses table acts as the main link between all other tables in the database, ensuring referential integrity and keeping it as normalised as possible. The userID is the primary key in this table as well as it keeps track of the courses recommended to the users and the courses completed by the users. courseID is a string of the courseIDs of all of the user's completed courses separated by a hyphen for easier lookup and addition with the addRecommendedCourse and getRecommendedCourses functions. The completed courseIDs are stored as a hyphen-separated string in the attribute completedID to ensure that a user does not get recommended a course that they have already completed.

3.6 UI

Each page of the web app has its own HTML5 file that defines it, the web app uses the Jinja templating engine to create its pages and so there are 2 basic template pages that all the other pages inherit from: a 'base' page outlining the metadata of the web app and all the additional files the pages use while defining a section for the page content, and a 'layout' page that inherits from the base template and includes a customisable navbar at the bottom containing the circular navigation buttons. Every page inherits from either of these and defines unique content in the 'content' block of the page inheritance.

The style of pages uses a combination of a CSS stylesheet defined in the static assets directory and inline style definitions; inline style definitions re used for the elements that use unique style and aren't used elsewhere in the app however more common styles are stored under a unique class name in the stylesheet, the name is representative of the type and purpose of the element it is styling. There are also definitions in the stylesheet on styles that are affected by the size of the screen and uses media queries to adapt to different screen sizes. 5 different screen size categories have been outlined: XS (screen width under 600 pixels), S (screen width between 600 and 768 pixels), M (screen width between 768 and 992 pixels), L (screen width between 992 and 1300 pixels), and XL (screen width over 1300 pixels). These media queries define the screen compatibility of the web app.

3.7 Development Process

In constructing the IBM Watson component, an extensive research process was executed. Information about various courses were collected, with direct links to their respective websites. Meanwhile, a list of careers and another one of education levels were sorted and strategically associated to different courses, the purpose of this was to facilitate personalised recommendations for users based on their career goals or education level, based on user's preferences. In order to enrich the functionality of IBM Watson, data on the average salaries corresponding to these careers in the UK was also compiled. These datasets serve as foundational training material for IBM Watson, enabling it to provide users with well-informed advice regarding suitable courses and career paths.

To cater to the needs of different users, a user-friendly chatbot with four different routes was produced. In the careers route, users have the flexibility to input their desired occupation and the chatbot responds by recommending courses suitable for that career path. This is implemented by IBM Watson checking whether the user's input matches with some career. The education route allows users to select their academic level, ranging from first year university students to PhD, and the chatbot provides course suggestions accordingly. With the interest route, users can enter any topic they are interested in, the chatbot will quickly show the information of the course if the input matches the name of any course. Our chatbot additionally includes a career query channel where users are able to inquire about the typical pay and work description for any career that is listed in the system. This approach of incorporating IBM Watson into the chatbot not only makes interactions more dynamic and user-focused, but also enhances the system's functionality, making it a flexible platform which adopts the needs of users with a range of preferences.

3.8 Maintenance

The main problem in maintaining SkillsGuild is ensuring all new courses added to SkillsBuild are also added to SkillsGuild. This system has been implemented manually so far, with one of the team reviewing the course, identifying its difficulty, duration, and relevance to different topics, assigning the stats given accordingly. However, continuing to do so for the considerable future may prove difficult, and so some automation may be involved in maintaining this in the future.

Similarly, new equipment and items would need to be added to itempool, as eventually some users will have collected every item and will no longer have motivation to continue using SkillsBuild as they are not seeing any tangible reward for their work. While these would likely have to be created manually, the process of doing so is easy: simply make a new entry in the 'equipment' table, give it a name and a type (weapon, helmet, chestplate, leggings, or boots) and it is immediately available to be obtained from completion rewards due to the nature of the backend code.

The chatbot is hosted through IBM Watson Assistant and this uses a plan that is limited to 10,000 API calls and deletion after 30 days of inactivity. Should this occur an IBM Cloud account would be required to host the chatbot again. An account can be created using this link (https://cloud.ibm.com/registration). Upon signing in, you are redirected to a dashboard. Selecting the '**Create Resource**' button will load a new page with lots of products.

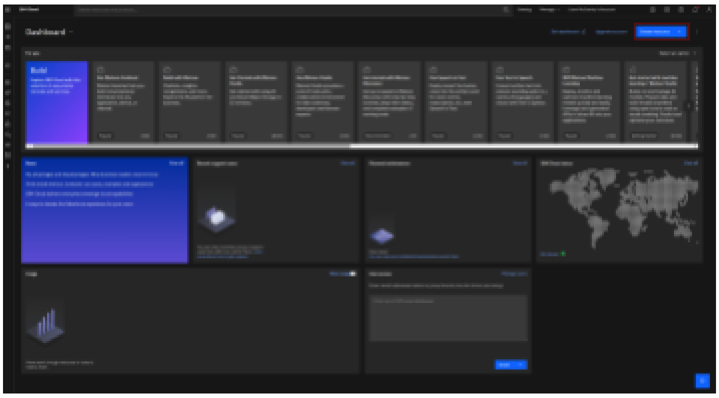


Image of IBM Cloud once signed in. With a highlight around the 'Create Resource' button.

Using the filters on the left hand side of the webpage you can select '**AI / Machine Learning (22)**' which will display products using AI and machine learning.

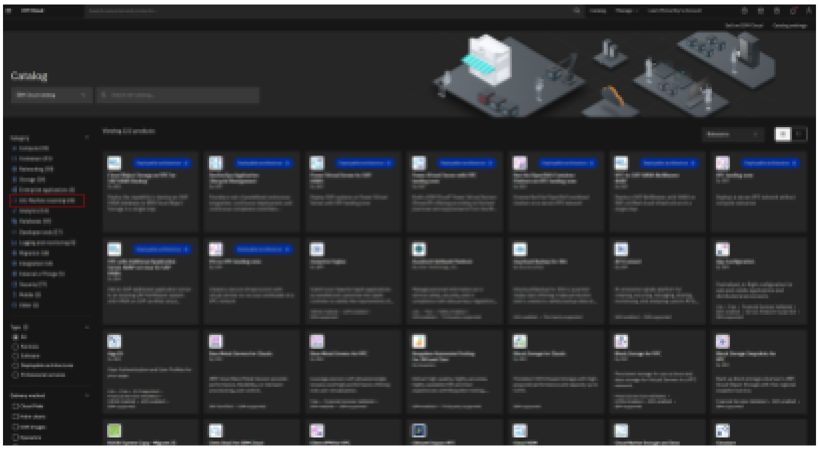


Image of IBM Cloud products page, with a highlight around the 'AI / Machine Learning (22)' filter.

You can then select '**Watson Assistant**' by IBM. This will begin creating the new resource and request some information.

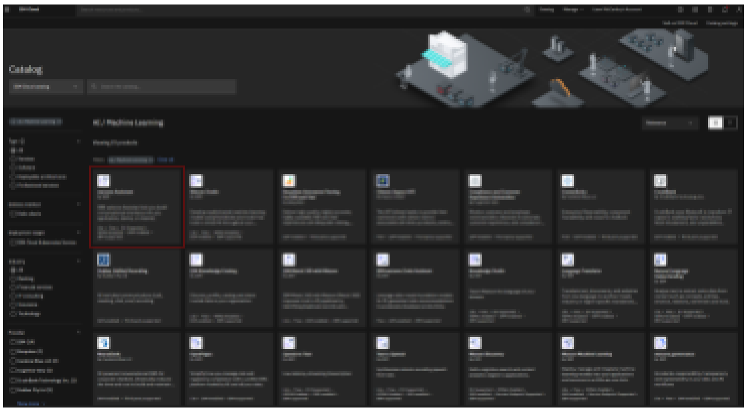


Image of IBM Cloud AI and Machine Learning Products. With a Highlight around Watson Assistant.

You will then have to fill out some information so to reduce the amount of changes required select these values:
Location: London (eu-gb)
Plan: Lite
Name: SkillsGuild

When these are selected/input read and agree to the licence agreement and then click '**Create**' in the bottom right. You will be redirected to a new page. While here it is a good idea to make a copy of the **API Key** provided as this is required soon. To begin uploading the chatbot select '**Launch Watsonx Assistant**'.

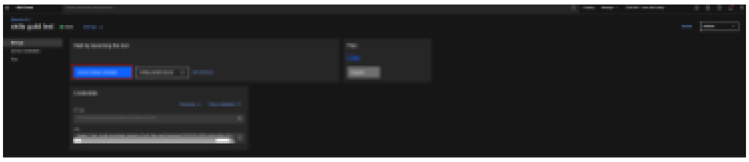


Image of IBM Watson Service Page. With a Highlight around the 'Launch Watsonx Assistant' button.

To start you will be required to create your first assistant. The information you enter here will not affect the functionality or visual display of the chatbot in our applicaiton. So continue until you see the preview page. Click '**Create**' to initialise the assistant.

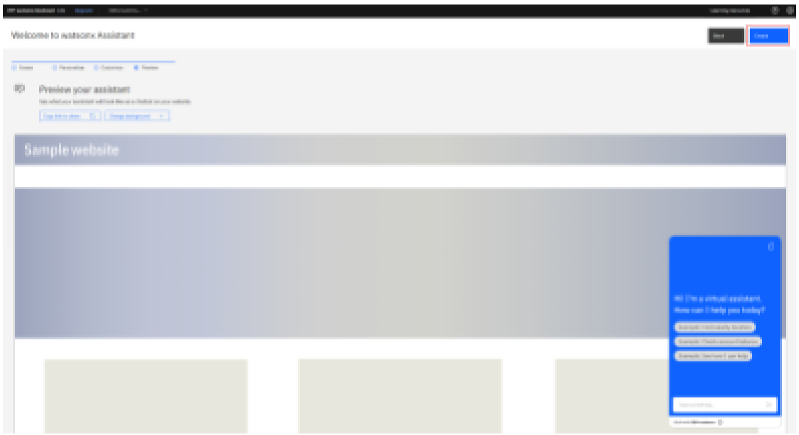


Image of IBM Watson Assistant Preview Page with Highlighted 'Create' button.

The page will then load steps to get started on your new chatbot. Since you are going to upload the one we have created, these can be ignored and you can use the left hand menu to select '**Assistant Settings**'.

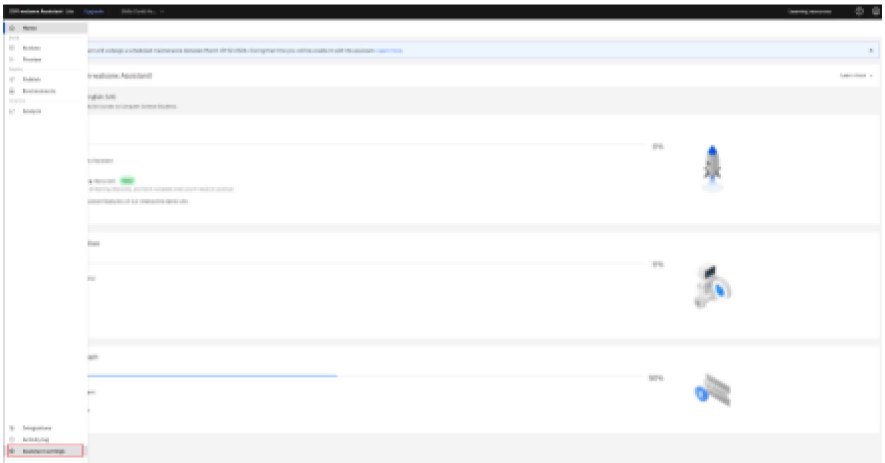
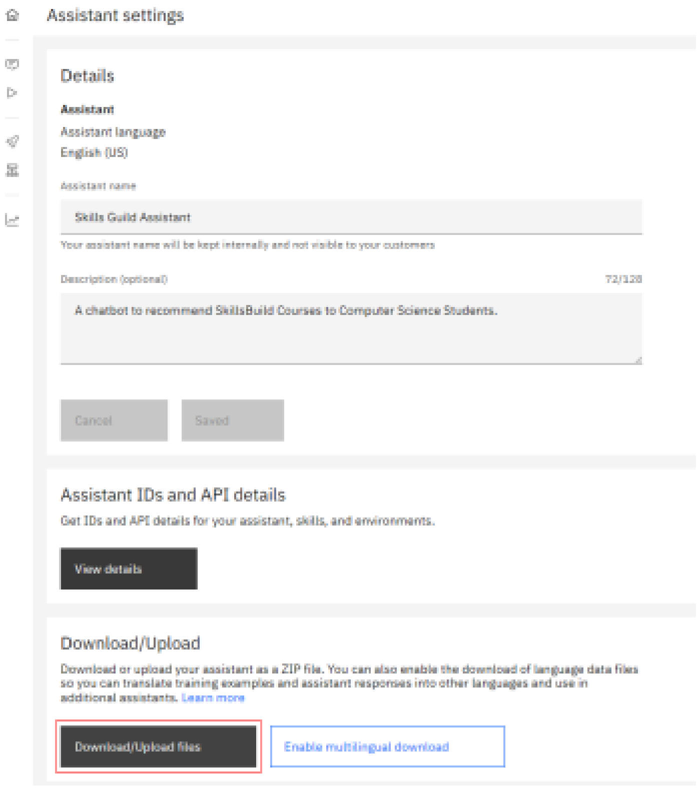


Image of IBM Watson Assistant's Menu with a highlight on 'Assistant Settings'.

You can then select the '**Download/Upload Files**' button to upload our chatbot. The chatbot is in the submitted code in a folder called 'Watson Backup'. Within this folder is a zipped copy of the chatbot named 'IBM-WA-6a083171-47d1-4a77-9dd3-2fa33be2cca0-V5.zip' which can simply be uploaded to the page. When this is uploaded you will be warned it will replace your existing draft content so click '**Upload and Replace**'.



Assistant settings

Details

Assistant

Assistant language
English (US)

Assistant name
Skills Guild Assistant
Your assistant name will be kept internally and not visible to your customers

Description (optional) 72/128
A chatbot to recommend SkillsBuild Courses to Computer Science Students.

Assistant IDs and API details
Get IDs and API details for your assistant, skills, and environments.

Download/Upload
Download or upload your assistant as a ZIP file. You can also enable the download of language data files so you can translate training examples and assistant responses into other languages and use in additional assistants. Learn more

Image of Assistant Settings with a Highlighted 'Download/Upload Files' Button.

This will be initialised and you will get a success message when it is done. The assistant is then viewable in the action menu. You may notice at the top right it will say 'system is training'. This is a process where the assistant learns the actions provided. After a few seconds this should be done and are then required to publish the assistant into a live version. Navigate to the '**Publish**' button in the left hand menu. You will see all the unpublished content in the chatbot. Selecting '**Publish**' on this page will allow you to upload this new content.

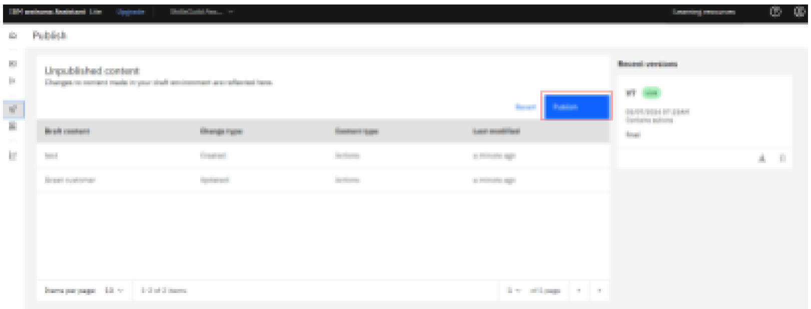
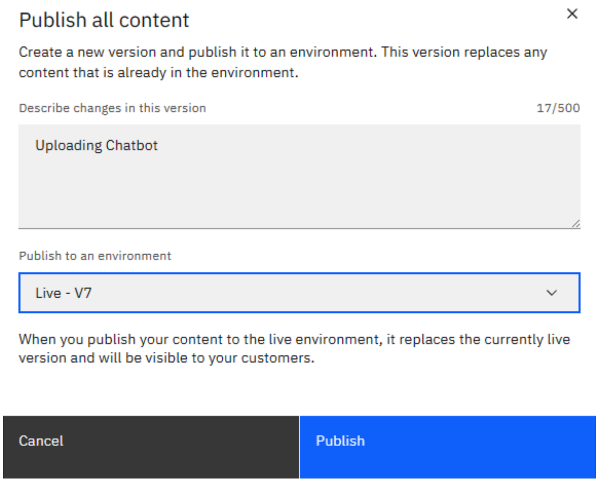


Image of the Publish Page with a Highlighted 'Publish' Button.

You will be prompted to describe the changes made to the chatbot, for this case it would be suitable to enter 'Initialising Chatbot'. Following this you will need to select a version to publish. You need to make sure it is on the Live Branch. If you are following along with this example it will likely say "**Live - V1**'.



Publish all content

Create a new version and publish it to an environment. This version replaces any content that is already in the environment.

Describe changes in this version 17/500

Uploading Chatbot

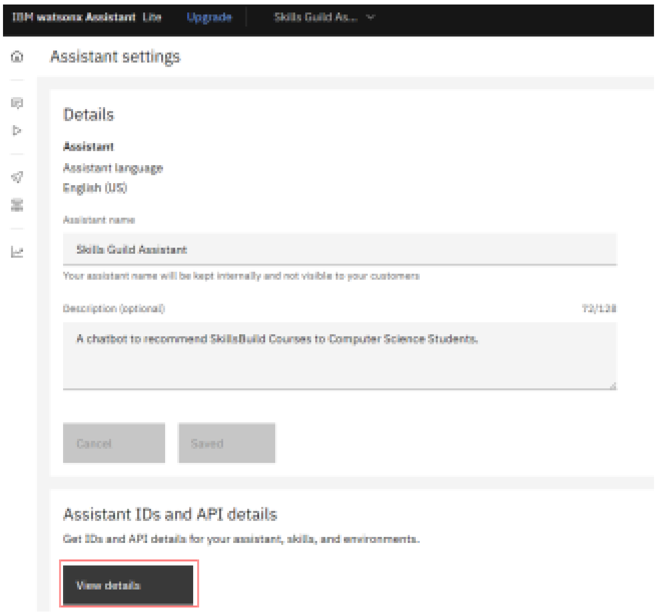
Publish to an environment

Live - V7

When you publish your content to the live environment, it replaces the currently live version and will be visible to your customers.

Image of the Content Publication Form.

Now you have to make some changes to the code to be able to run the newly hosted assistant. This will require some data from this assistant. Navigate back to the left hand menu and select '**Assistant Settings**' then select '**View Details**' in the Assistant IDs and API Details section.



Assistant settings

Details

Assistant

Assistant language

English (US)

Assistant name

Skills Guild Assistant

Your assistant name will be kept internally and not visible to your customers

Description (optional) 72/128

A chatbot to recommend SkillsBuild Courses to Computer Science Students.

Assistant IDs and API details

Get IDs and API details for your assistant, skills, and environments.

Image of Assistant Settings with a highlight around 'View Details' Button.

This will provide IDs and APIs required to link our application to this newly hosted chatbot. You will now be required to open webapp.py and make some changes to the credentials at the top. The API Key you copied earlier needs to be assigned to the '**api_key**' variable. Then you will need to head back to the Assistant IDs and API Details page to assign the '**Service Instance URL**' to the assistant_url variable and the '**Live Environment ID**' to the assistant_id variable.

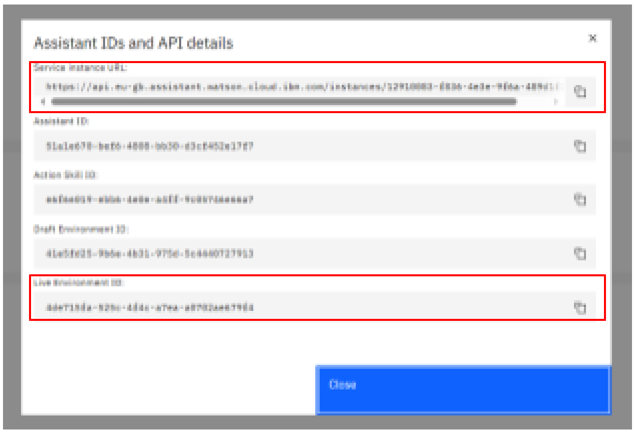
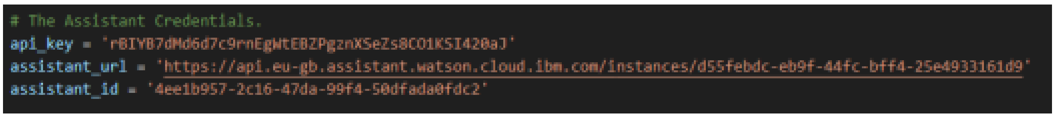


Image of Assistant IDs and API Details, 'The Service Instance URL' and 'Live Environment ID' are Highlighted.



```python
# The Assistant Credentials.
api_key = 'rBIYB7dMd6d7c9rnEgWtEBZPgznXSeZs8CO1KSI420aJ'
assistant_url = 'https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/d55febdc-eb9f-44fc-bff4-25e4933161d9'
assistant_id = '4ee1b957-2c16-47da-99f4-50dfada0fdc2'
```

Image of webapp.py Assistant Credentials.

Once these 3 variables are assigned the file needs to be saved and then if you relaunch the web application using the command:

```
python web_app.py
```

You can sign in and navigate to the 'Find Courses' page which will automatically create a session with the newly linked chatbot and begin a chat.

The IBM Cloud is how you would access the chatbot if you wanted to implement changes to the chatbot itself. This can include providing more content or even adding additional IBM SkillsBuild Courses as and when they are implemented.

3.9 Future Developments

Provide useful and usable information on how the system's possible future developments can be implemented (10%).

Using the MuShCo framework we were able to effectively prioritise the essential and high-priority aspects of the project, ensuring they were completed to a high standard before the deadline. However, this meant that some parts of our initial design were not able to be implemented like we had hoped.

3.9.1 Unimplemented Features

The main unimplemented feature is using Google BigQuery to store our database and handle our queries. This would have allowed for faster data retrieval, more secure storage, as well as a much greater capacity. Unfortunately, due to unforeseen serious adverse circumstances, this was replaced by comma-separated-value files stored in a directory titled 'database'. While on a small scale - where our product is currently, and may remain - this is no real difference, but if the product's usage was to expand further then this could cause to be an issue, due to lack of record locking or timestamp ordering protecting data from being incorrectly overwritten.

Furthermore, we had hoped to further gamify SkillsGuild, by adding in both an institution and global leaderboard to add a hint of competition to SkillsGuild. We believed this would further motivate people to complete more SkillsBuild courses, but unfortunately time constraints rendered this unachievable, at least for now.

The web app also doesn't currently fully function on macOS devices as the chatbot returns an error if a query is sent to it, however the error code that is returns is the same error code for session expiry so this is possibly a fault with the backend of IBM Watson and not our own implementation.

Additionally, the idea of implementing our application as a Progressive Web App (PWA) has been unprioritized. This is due to the absence of a team member. This feature has still been considered in development and there is room to implement the caching for offline functionality. There are performance improvements that can be made when developing this and now the application is done there will be less caching issues during the development of the PWA. Furthermore we would significantly be able to improve the accessibility of the application now that everything is in place prior to this being implemented. This would have been a nice addition for solving the problem provided by our client. However as the main functionality is dependent on an internet connection this was not essential, but would have been a nice addition. Even without this implementation, the performance and loading times are still efficient.

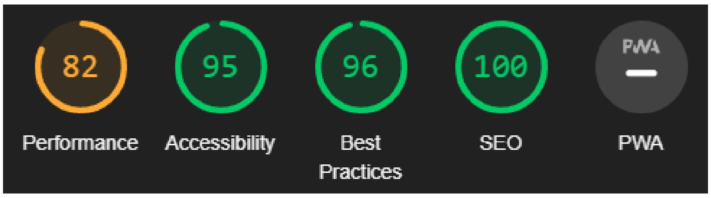


Image of a Chrome Lighthouse Report Prior to implementing the PWA.

The image above displays the current lighthouse report at the submission of our application. only one of the four categories requires improvement prior to the PWA being implemented, besides from the code for the PWA itself.

SkillsGuild was also meant to be usable on a mobile device, and while the application has been tested in a mobile environment on desktop (and functions as desired), due to financial restrictions we were unable to host the application online. Instead, the application must be run locally to be accessed and hence is not usable on a mobile device at this point.

Finally, while the user receives items for completing certain courses, we did not have enough time to create the art for each item. This would have meant that a user with, say, the Warhammer item equipped would have been able to navigate to their avatar and see them wielding the weapon. This also would have encouraged users to complete more courses in SkillsBuild.

3.9.2 Suggestions for Future Developments

The first features that should be developed in the future are the ones mentioned in 3.4.1 for the reasons discussed above. Other potential features include a 'friends list'. Similar to the proposed institution leaderboard, there could be a leaderboard ranking you and all of your friends, and you could become friends with people in your institution or in another.

Another potential feature is the addition of limited-time-events (LTEs). These can include thematic new courses, items or webpage designs. For example, in the Halloween period there could be a pumpkin item added for the helmet slot and the webpage can be themed appropriately for the holiday. There could also be LTEs that give exp boosts.

The most ambitious feature, but arguably the most valuable, would be linking SkillsGuild accounts to SkillsBuild. This would mean that when a user completes a course in SkillsBuild, the relevant data is sent to their SkillsGuild account, giving them their exp, new item and stat boosts. Currently SkillsGuild operates on an honesty system, where players say whether or not they have completed a course and the system takes that for a fact. While this should work in theory, as users are only on SkillsGuild to motivate and assist them with navigating SkillsBuild, some users may get over-competitive and lie about their completed courses so that they would rise higher on leaderboards, which would defeat the purpose of our product.

Unfortunately, as we were unable to access a Mac during our development phase, we were unable to make SkillsBuild Mac-compatible due to issues with the Watson chatbot. With more time and access to such devices, the application will be compatible with all devices, and hence it is a suggestion for the future.

The addition of security to the database and account system is also a potential development as the current version of SkillsGuild functions more as a proof-of-concept instead of a deployable product with fully developed security. This is as was advised by the client however the inclusion of this would make the web app more complete and deployable as a product.

There is also a warning regarding overriding the recommendations. If a user already has 3 courses recommended and enters conversation with Watson then a message pops up telling them they may be about to override a recommended course. This appears rather abruptly and early into the conversation, and so in the future we would improve the user experience, maybe altering the output and give the user the option to select from the recommended courses which they would like to work on. This way they are more in control of their progression and recommendations. Additionally, we could implement better methods for handling when the user already has courses recommended such as allowing them to select which course they would like to override.

The courses that are currently included in our application are just the technical courses however IBM SkillsBuild has developed some non-technical courses. These could be considered and implemented in future development phases to offer more functionality to the system.

Finally, as touched upon earlier, the idea of adding ‘streaks’ to SkillsBuild. If one were to complete courses in consecutive days they would receive either a cosmetic item or perhaps another type of reward. Depending on the route the product takes, users could use in-game currency (perhaps exp) to reclaim their streak if it is broken, with more of these purchasable. Of course, this is not the current plan for SkillsBuild, but shows how many routes are available.

4 - Requirements & Resources

4.1 Status of User Requirements

Provide a clear and comprehensive description for each of the user requirements, including the initial and current status, the extent to which it is fulfilled, and justification for the extent of fulfilment (10%).

4.1.1 Functional Requirements

These are split into 3 distinct categories: **Profile, Chatbot and Game Feature** functionality.

FR1 - Profile Functionality

The functionalities required for the creation and management of the user's profile.

ID, Type & Title	**FR1.1 - Profile - Sign Up**
Description	The user should be able to make an account using their username and institution name, the app will provide a list of basic questions that the user can answer and be added to the user's profile
MuShCo - Priority	**Must Have** - High
Dependencies	N/A
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR1.2 - Profile - Login**
Description	The user should be able to input a username and institution into the login page where the associated profile data should then successfully load and update the website's profile section to fit the data
MuShCo - Priority	**Must Have** - Medium/High
Dependencies	FR1.1
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR1.3 - Profile - Saving**
Description	Whenever a profile is updated locally, it should be saved and overwrite its previous instance
MuShCo - Priority	**Must Have** - Medium/High
Dependencies	FR1.1
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR1.4 - Course - Completion**
Description	The profile's currently recommended courses can be accessed at any time and marked as completed
MuShCo - Priority	**Must Have** - High

Dependencies	FR1.2
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

FR2 - Chatbot Functionality

The functionalities based around the chatbot and how the user should be able to interact with it as well as its purpose in the app.

ID, Type & Title	**FR2.1 - Chatbot - Interaction**
Description	The user is able to have a natural conversation with the chatbot where it asks questions regarding the user's academic pursuits
MuShCo - Priority	**Must Have** - High
Dependencies	N/A
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR2.2 - Chatbot - Recommendation**
Description	After the user has conversed with the chatbot, 3 SkillsBuild courses should be recommended based on the user's stats, completed courses and conversation with the chatbot
MuShCo - Priority	**Must Have** - High
Dependencies	FR1.2, FR2.1
Status	Slightly Changed
Extent of Fulfilment & Justification	After conversing with the chatbot, up to 3 courses are recommended based on the conversation with the chatbot and what has already been completed. The chatbot does not receive user data and so this is not taken into account when recommending courses. It can recommend interests which are similar to the questions asked when signing up a new user. However, this is not an automatic process at this time.

FR3 - Game Feature Functionality

The functionalities concerning the gamification features of the app and how they affect the user's experience with it.

ID, Type & Title	**FR3.1 - Profile - Stats & EXP**
Description	The user has stats in different computer science aspects that are updated whenever they complete a course in a specific field. The user also has a level and points towards the next level (EXP) should be gained after completing a course
MuShCo - Priority	**Should Have** - Low
Dependencies	FR1.2, FR1.4
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR3.2 - Equipment - Inventory**

Description	The user should have a list of unique equipment that is currently in their inventory which can be navigated where each piece of equipment is either a: helmet, chest piece, greaves, weapon or shoes
MuShCo - Priority	Should Have - Low/Medium
Dependencies	FR1.2
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR3.3 - Equipment - Acquisition**
Description	Each course has an assigned piece of equipment that the user can gain upon course completion where it will then be added to the user's inventory
MuShCo - Priority	Should Have - Low/Medium
Dependencies	FR1.4, FR3.2
Status	Unchanged
Extent of Fulfilment & Justification	Completely implemented

ID, Type & Title	**FR3.4 - Equipment - Avatar**
Description	The user has an avatar that can be equipped with the various equipment pieces in their inventory and the avatar will update to equip those pieces
MuShCo - Priority	Should Have - Low/Medium
Dependencies	FR3.2
Status	Slightly changed
Extent of Fulfilment & Justification	The user is still able to equip items to their avatar however this will not change the appearance of the avatar, the user has a static avatar that cannot be changed. This was done due to the time and resources manually creating item art assets would take, requiring significant work to create a unique sprite for each item. Despite this change, the items are still being equipped so the addition of visual changes should not be difficult if it were to be implemented in the future as the equipping logic is already there.

ID, Type & Title	**FR3.5 - Leaderboard - Display**
Description	The user can pull up a leaderboard of all the highest level profiles where their name, level and avatar are displayed
MuShCo - Priority	Should Have - Low
Dependencies	FR1.2, FR3.1, FR3.4
Status	Removed
Extent of Fulfilment & Justification	Due to the time it would take, the contribution to the product and the agreed priority of the leaderboard, it has been deemed infeasible to implement this feature to an acceptable standard within the development timeframe of the product.

ID, Type & Title	**FR3.6 - Leaderboard - Types**
Description	The user can switch between 2 types of leaderboard, an overall leaderboard and an intra-institution leaderboard
MuShCo - Priority	Could Have - Low
Dependencies	FR3.5

Status	Removed
Extent of Fulfilment & Justification	Due to the time it would take, the contribution to the product and the agreed priority of the leaderboard, it has been deemed infeasible to implement this feature to an acceptable standard within the development timeframe of the product.

4.1.2 Non-functional Requirements

These are split into 5 distinct categories: **Performance, Reliability, Security, Compliance and Usability** requirements.

NFR1 - Performance Requirements

The requirements governing the performance of the app such as the speed and resource usage.

ID/Title	**NFR1.1 - Web app load times**
Description	The app should quickly load upon entering and switching pages to ensure it retains the user's attention and maintains a streamlined experience
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR1.2 - IBM Watson processing times**
Description	The inclusion of IBM Watson means Watson’s processes must operate at a sufficiently rapid pace to avoid interrupting the chatbot's conversational flow, ensuring a natural and seamless user experience, it will also constantly keep the 3 potential recommendations updated as new information is fed to it
Status	Unchanged
Extent of Fulfilment & Justification	The synchronous processing involved while the user engages with the chatbot is designed to be efficient. The

NFR2 - Reliability Requirements

The requirements governing the consistency of the app to ensure the experience remains reliable over several sessions.

ID/Title	**NFR2.1 - IBM Watson acceptable responses**
Description	To maintain a natural conversation with the chatbot, Watson should consistently provide relevant responses, encouraging the user to interact naturally and contribute more information to the recommendation algorithm
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR2.2 - Course recommendation correctness**
Description	The app's primary purpose is to recommend courses. It is crucial that it consistently suggests relevant courses to ensure reliability and fulfil its purpose as a valuable resource for users
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR2.3 - Internet disconnection**

Description	If the user disconnects from the internet, the app should gracefully cease operation and display an informative error message
Status	Unchanged
Extent of Fulfilment & Justification	The application will still function however the chatbot does not gracefully handle losing internet connection. It just remains blank. It will output the error to the console but this is not classified as gracefully in our books. Due to unforeseen circumstances,

NFR3 - Security Requirements

The requirements ensuring the app is secure and safe to use while staying in compliance with regulations.

ID/Title	**NFR3.1 - Remote app data**
Description	The profile data for users must be securely stored with IBM's Cloud servers, ensuring only necessary data is stored and utilised
Status	Changed
Extent of Fulfilment & Justification	For the purposes of development, considering the budgetary constraints and the effect this would have on product development and testing, the data the product utilises has been changed to using local csv files instead of a cloud server. This is due to the operation cost of a cloud server which is unnecessary during development as the structure of the database is simulated using the csv files. Using local csv files allows for rapid development and bypasses server costs. Due to the structure of the local files imitating the cloud server's, a future switch to using cloud servers is made easy with very little changes to the rest of the product allowing this to be a viable alternative for product feasibility purposes.

ID/Title	**NFR3.2 - Data transfer**
Description	The app will frequently send data to IBM servers to utilise Watson and the Cloud, this data transfer process must be secure and regulation compliant, it should also be consistent with little errors or retransmissions
Status	Changed
Extent of Fulfilment & Justification	Due to the change from using cloud servers to local csv files to store data, this requirement is completely different due to the data remaining local at all times, security when passing data is still ensured during operation however to accommodate for a potential change to larger scale cloud server usage. Data transfer to IBM Watson is unchanged however as this is a black box process we cannot influence meaningfully.

NFR4 - Compliance Requirements

Requirements dictating how the app will stay in compliance to regulations and standards for a polished, safe experience.

ID/Title	**NFR4.1 - GDPR compliance**
Description	The app should comply with the GDPR to safely handle user data according to established regulations
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR4.2 - Accessibility**
Description	The web app should be accessible to as many people as possible, including those with disabilities and thus must comply with WCAG guidelines
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR4.3 - Markup validation**
Description	The HTML and CSS for the web app must be valid for online publication and meet the World Wide Web Consortium standards to ensure quality
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR4.4 - Code consistency**
Description	Consistency in language and conventions across HTML, CSS, code, and other development aspects ensures long-term sustainability and promotes code comprehension among developers
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

NFR5 - Usability Requirements

The requirements governing how usable the app is and ensuring its ease of use and accessibility.

ID/Title	**NFR5.1 - Streamlined UX**
Description	Given its use with IBM systems, the web app must maintain the streamlined UX and minimalist appeal of IBM websites
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR5.2 - Compatibility**
Description	The app is designed for both mobile and PC use, ensuring compatibility, visual consistency, and uniform operability on both platforms
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

ID/Title	**NFR5.3 - Understandability**
Description	The game mechanics, responses given by Watson and general purpose of the app should be easy to understand and self-explanatory
Status	Unchanged
Extent of Fulfilment & Justification	Fulfilled

4.2 Resources & References

Describe clearly and understandably the source materials upon which the conceptualisation and development of the system is built (5%).

Group 20 - IBM SkillsGuild

In our project, we utilised various websites to enhance user experience and provide valuable information. Glassdoor's salary page offers insights into compensation data, while IBM's course catalogue provides detailed information on courses. Moreover, the integration of IBM Watson ensures dynamic chatbox responses, enhancing user engagement.

Glassdoor. (n.d.) *Salaries*. Available at: https://www.glassdoor.co.uk/Salaries/index.htm

IBM. (2024) *Course Catalog*. Available at: https://students.yourlearning.ibm.com/recommended/aoi/TECHNICAL_SKILLS

IBM. (2024) *IBM Watson Natural Language Understanding*. Available at: https://www.ibm.com/products/natural-language-understanding?lnk=flatitem

In our Python code, we used Flask, IBM-Watson and Pandas modules to create a robust and dynamic application. Flask provided a flexible and efficient web framework, while ibm-watson added powerful natural language understanding capabilities, enhancing the application's interactivity and responsiveness. Additionally, the use of Pandas facilitated data manipulation, ensuring that our project could efficiently handle and process information. Together, these modules played a crucial role in creating a user-friendly chat box application.

Flask. (2024) *Flask 3.0.2*. Available at: https://pypi.org/project/Flask/

Ibm-Watson. (2024) *ibm-watson 8.0.0*. Available at: https://pypi.org/project/ibm-watson/

Pandas. (2024) *Pandas 2.2.1*. Available at: https://pypi.org/project/pandas/

Python. (2023) *Beginner's Guide to Python*. Available at: https://wiki.python.org/moin/BeginnersGuide

Python. (2024) *Installing Packages.* Available at: https://packaging.python.org/en/latest/tutorials/installing-packages/

Python. (2024) *Install Python.* Available at: https://www.python.org/downloads/

During the development of the application, we drew inspiration from three notable sources. Myagi served as a reference for incorporating gamification elements into our project. Replika's AI Online Companion Chatbot and Ubisend's Conversational AI influenced the implementation of our chatbot AI, providing inspiration into creating engaging and responsive conversational interfaces.

Myagi (2014) *Get Myagi, Home Page.* Available at: https://www.getmyagi.com/

Replika (2017) *Replika AI Online Companion Chatbot.* Available at: https://replika.ai/

Ubisend (2016) *Ubisend Home Page, Conversational AI.* Available at: https://www.ubisend.com/features/embedded

We utilised IBM Cloud for efficient database functionality, ensuring secure storage of user information in our application. This choice reflects our commitment to reliability and data integrity, contributing to a seamless user experience.

IBM. (2024) *IBM Cloud*. Available at: https://www.ibm.com/cloud

Visual Studio Code served as our primary IDE, offering a user-friendly environment for code development. Its features streamlined the editing and execution of our web application's source code, contributing to the project's overall success.

Microsoft. (2024) *Visual Studio Code.*Available at: https://code.visualstudio.com/download

In the development of our application, we utilised IBM Cloud registration to gain access to the necessary resources and services. Leveraging the Watson API, specifically for natural language understanding, played a pivotal role in enhancing the frontend functionalities of our chatbox. By integrating IBM Watson's capabilities, we ensured dynamic and intelligent interactions within the chatbox, offering users a seamless and engaging experience.

Group 20 - IBM SkillsGuild

IBM. (2024) *IBM Cloud Registration*. Available at: https://cloud.ibm.com/registration

IBM. (2024) *IBM Watson Assistant V2 API*. Available at: https://cloud.ibm.com/apidocs/assistant-v2