Development Report

Introduction:

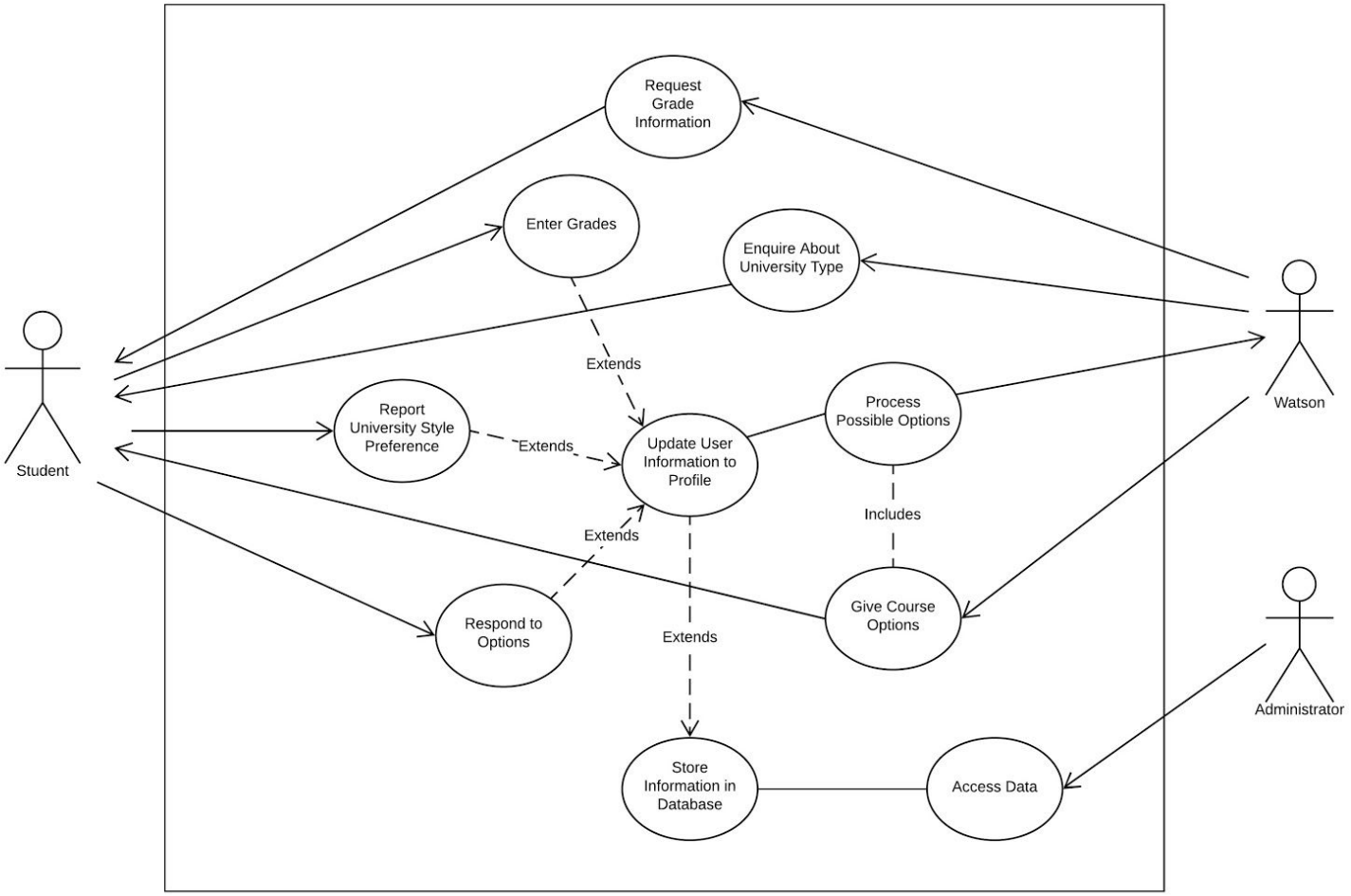
The purpose of the system is to give guidance to secondary school students deciding on their third level education path, based on information provided by the students (where they want to study, their desired discipline, their favourite subjects, leaving cert result predictions etc). It will be a web application, and give guidance using the Watson conversation tool on IBM cloud to make a guidance counsellor chat bot. We are aiming to provide relevant information about colleges around Ireland and information about specific courses within those colleges.

The function of the of the application was broken down into 3 parts. The front end which we wanted to make modular, scalable and simple, allowing it to fit into other websites or be later built up as a standalone service. For this we used Angular, a package that allows for the creation of modular non refreshing single page applications. The AI intelligence was done through the Watson Assistant API. This work was done in the Watson Assistant Workspace, training the conversational aspect of Watson. And lastly the backend, which created a UDP connection to a node server. The main service this provided was passing packets from the frontend to Watson and vice versa. The front end and Watson training took place separately and then was brought together with the implementation of the backend.

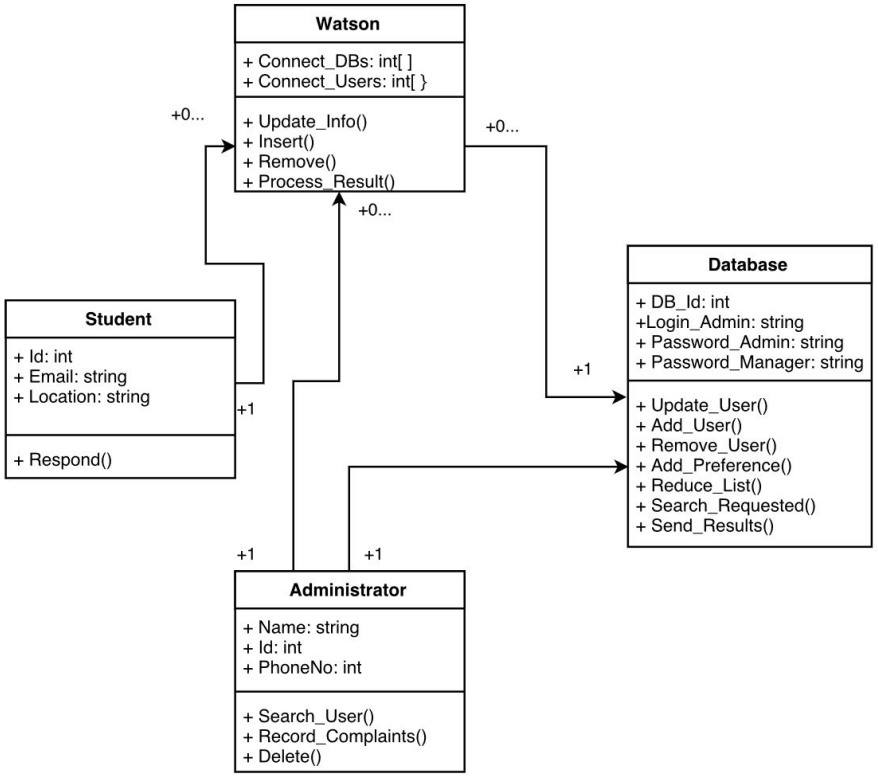
Requirements:

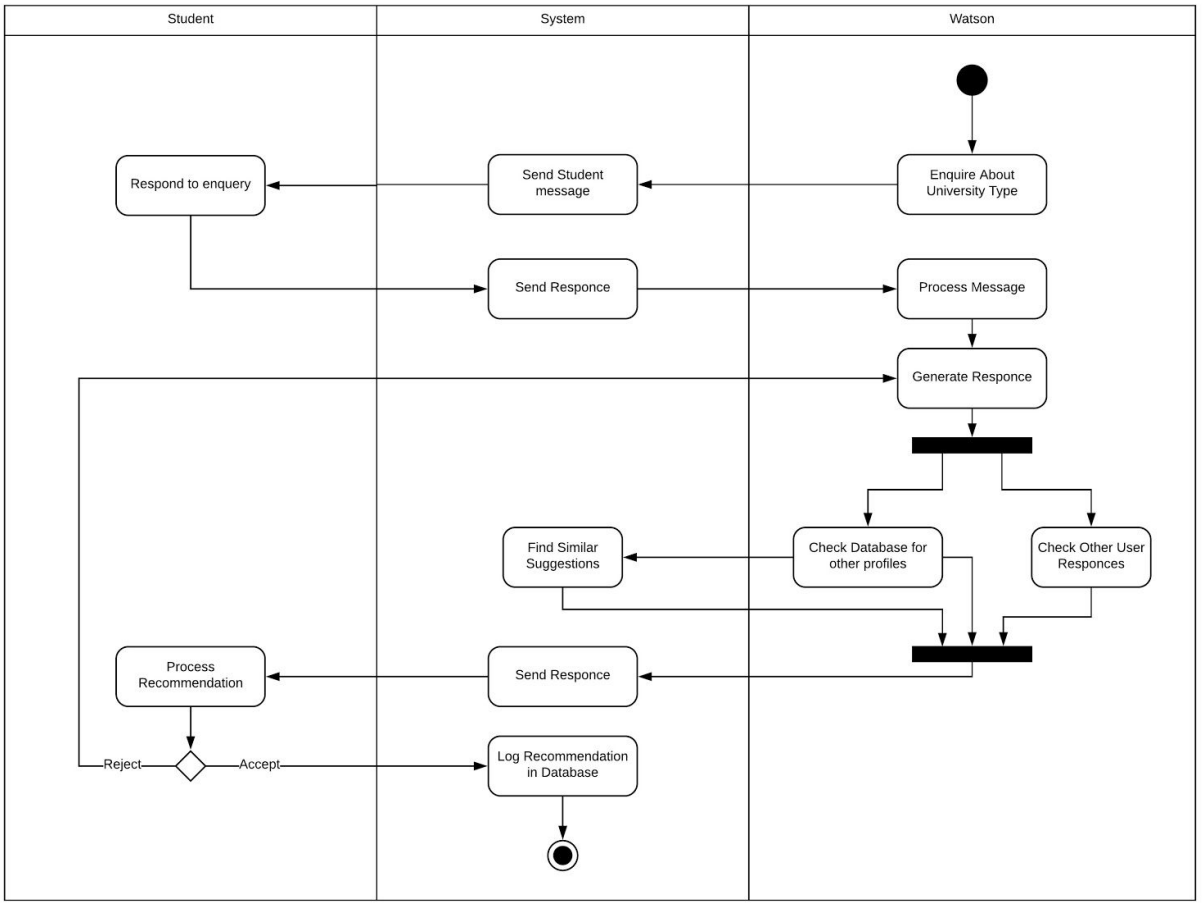
Functional Requirements	Non-Functional Requirements
● Allows user to query information about specific colleges ● Allows user to query information about specific courses. ● Allows user to narrow down course choices based on area of study, points, subject requirements, college location and more. ● Links user to websites to find more information on colleges and courses. ● Web app with chatbot.	● Allows user to have a natural conversation with the chatbot. ● Text to Speech and Speech to Text implementation. ● Slack, Facebook, Messenger chatbot. ● Further development for employment opportunities for third level students.

The main purpose of the system is the passing of information between Watson and the student in order for Watson to recommend a course and university that is tailored to the student's academic strengths and interests. The most common flow is Watson asking the student several questions and gathering responses. The responses will be used with data on current programmes at universities to narrow down courses and universities suited for the student. Each response is stored in anonymous profile that is both accessible to Waton and the administrations. The the few possible error scenarios were not added as they did little to help describe the system. There is no specific flow or order besides Watson making a recommendation and responding to the users indication of interest. This will more then likely happen at the end of a series of questions, once Watson has gathered enough information.

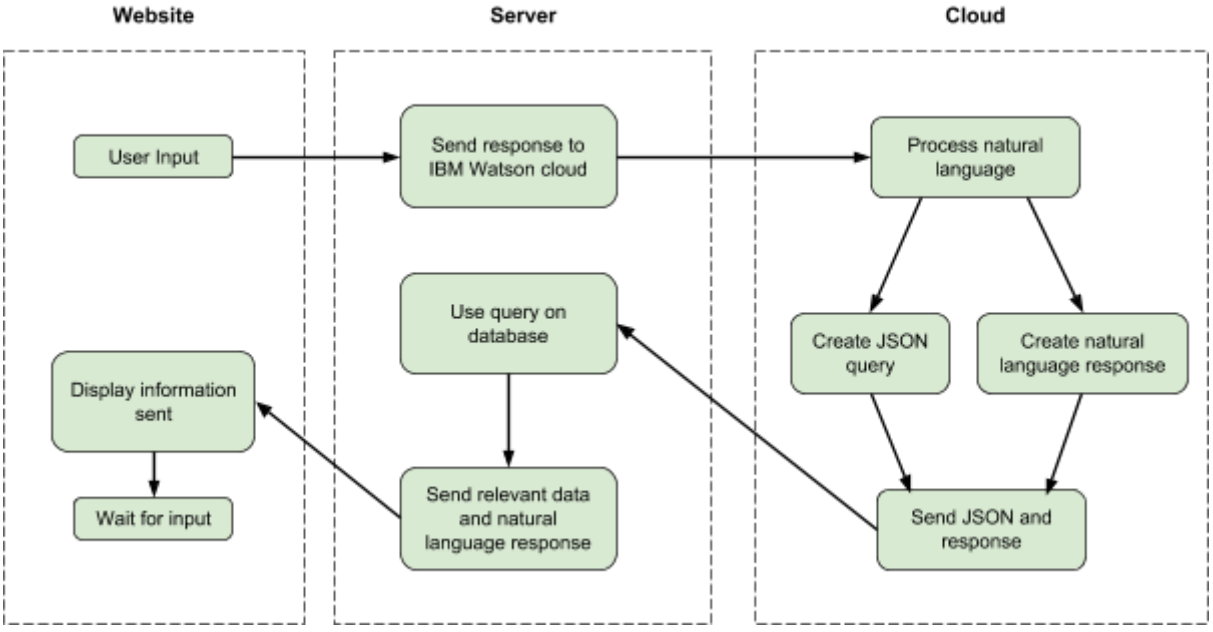

Request Grade Information
Enter Grades
Enquire About University Type
Extends
Report University Style Preference
Extends
Update User Information to Profile
Process Possible Options
Includes
Extends
Respond to Options
Extends
Give Course Options
Store Information in Database
Access Data
Student
Watson
Administrator

Design:


Watson
+ Connect_DBs: int[ ]
+ Connect_Users: int[}
+ Update_Info()
+ Insert()
+ Remove()
+ Process_Result()
+0...
+0...
+0...
Student
+ Id: int
+ Email: string
+ Location: string
+ Respond()
+1
Database
+ DB_Id: int
+Login_Admin: string
+ Password_Admin: string
+ Password_Manager: string
+ Update_User()
+ Add_User()
+ Remove_User()
+ Add_Preference()
+ Reduce_List()
+ Search_Requested()
+ Send_Results()
+1
+1
+1
Administrator
+ Name: string
+ Id: int
+ PhoneNo: int
+ Search_User()
+ Record_Complaints()
+ Delete()



Every scenario/use case within our system will all be pretty much the exact same as the student requests information from our database based off specific requirements that they wish to specify. If the database does not contain the information that has been requested, it learns what specifications were requested and adds them to the database to improve it's knowledge and increase the overall functionality and accuracy of the answers it provides.



Implementation:

As described above we broke the project down into three parts. The frontend, backend, and Watson api. These each required different tools and libraries to make all things work with one another.

The frontend was developed in Angular using typescript. Typescript allowed for the team to code javascript using the same conventions as java, such as objects and type declarations. This allowed for earlier comfort with a new platform. Angular allowed us to code the site in modular components. This helped in two regards. One the development of the application could be broken down into separate tasks that could be easily developed concurrently. The second use for modular development is that just the chat page could be implemented as a component in another website that was developed modularly. The style and look of the project was developed using bootstrap. Bootstrap uses a grid system that allowed us to develop the application with scalability in mind. This was important because the application could be implemented as a small chat window on another website or could act as a standalone application.

The backend was connect to the front end using socket.io which handles the reception and creation of http requests. Socket.io also allowed us to handle multiple instances of conversations as each socket has a seperate connection to the watson api and its own independent conversation context. The backend was coded using node. We chose node for several reasons. One being that Watson API has a node package which can be downloaded and used via the npm command line tool. The second reason is node's compatibility with Angular, what the frontend was developed in. Lastly we were able to code a high functionality server using fewer lines due to the packages node provides and supports. The Watson node package allowed us to create a Watson Assistant instance for each unique UDP connection to the server. Each instance has its own conversation context which is stored as a local json file. This means that the server is able to provide personal information without the collection of user data.

The Watson api was connected to a workspace where we were able to create a flow for our conversations and train the AI that handles the natural language processing of the messages. This training and creation was done through the bluemix website provided by IBM. This dashboard gave visual tools to create conversation flow and test the watson assistant. Using the API key for our workspace we are able to create a instance of the watson assistant on the backend which then communicates the front end via the chat page.

The user interface is a simple two page design. The landing page and the chatpage. The landing page purpose is simply to provide information on the service and acts a buffer between the user and the main aspect of the application. The chat page is a simple scalable design. It is an environment where the user is able to send and receive messages. The chat page now only handles a simple text form of interaction, but this could be expanded to handle video and photos or other types of media. Both of these pages follow as simple clean design that follows the general styling of IBM, our clients.

Conclusion:

The main problem we faced was communication. Some of the team completed their aspect of the project earlier and was well prepared to begin working on nonfunctional requirements while the other part left a majority of their work to the end of the project timeline. This was largely due to a communication error where one part of the team was under the assumption that the remaining segment of the project needed to be finished for their work to begin. This created inefficiency in the workings of the group, postponing features of the project that would have otherwise been completed at the time

of turnover. This stagnated development resulted in a less intelligent and informed assistant, due to a smaller set of trained conversations and data. This could have been remedied with clearer communication between the subdivisions of the development team.

As a team we were able to reach all of the major key requirements and made an effort to reach the nonfunctional requirements. The core project could only be improved if the AI was trained better. This was mainly a result of the short time span for the project. With more time the assistant would be better trained and more of the nonfunctional requirements would be reached. Overall we believe we reached the majority of the requirements of the project to a high degree.

For the most part the group functioned well. The subdivision of the development team helped in achieving the majority of project objectives. The division of the labour between the subgroups could have been better allocated. One of the groups was left doing 80% of the project while the other team completed the other 20%. This was also due to varying degrees of familiarity with the tools used. Given more time, the different levels of ability to use the technologies would be equivalent among members, this would possibly avoid this poor division of labour and delayed execution of the project.

Some suggestions for enhancement to the project are to the intelligence of Watson and its communication skills. Other improvements could be the addition of information tools and the graphical representation of user resources to be used by third level institutions. Watson would be improved with a larger set of data as well as more time spent in conversational training. The creation of a central database containing information about user interaction and queries could added to help third parties get insight on the incoming students and their academic interests. The overall design and execution of the project provides a robust scalable proof of concept of the main problem. With more time it could truly be developed as a industry quality service.