Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: CSION

High Level Design Report

İsmail Yavuzselim Taşçı, Ayça Begüm Taşçıoğlu, Mehmet Sanisoğlu, Nursena Kurubaş, Mehmet Selim Özcan

Supervisor: Prof. Dr. Uğur Güdükbay
Jury Members: Prof. Dr. Özgür Ulusoy and Prof. Dr. İbrahim Körpeoğlu

TABLE OF CONTENTS

1. Introduction

Here, we will introduce the Csion application by mentioning the motives behind building such a system, the expected qualities, explanations of some words used in the report, and an overview of the application.

1.1 Purpose of the system

Csion aims to help people who have doubts in their minds by taking their questions regarding these doubts, and coming up with the optimal suggestion decided by considering the user's personality features that is obtained from a personality test. Its purpose is making users' lives easier by being a trustworthy consultant who knows its customers better than they know themselves.

Csion is intended to offer the user two ways to demonstrate their doubts, one is by choosing the topic that relates by the doubt/question that user has in mind, and the other will exist to be able to help the users who cannot find any related topic at categories section. This second recommendation system will be using a chatbot to give users the freedom of writing about their concerns without any restrictions.

1.2 Design goals

In this section, some qualities that are needed to be prioritized in order to achieve the best user experience are explained.

1.2.1 Usability

- Application should be understandable for every user. User interface will provide every information that user can access.
- Menus and sub-menus should be usable for every portion of the application. Also, menus should be easy to tap and easy to read since users can have difficulties in reading.
- Application should provide information such as tips and manuals for users when they first login to the system.

1.2.2 Efficiency

- Number of questions should not exceed 10 for categorized problems.
- Questions must cover at least 1 keyword of the personality type in order to decrease the number of questions.
- IBM Watson API customization must cover every keyword for every personality and result of the analyzed text must return at least 90% of the related keywords.
- Analized output should be direct and should give only necessary percentages.

1.2.3 Reliability

- Application should provide exact results of personality tests without any errors.
- Result of analyzed text that comes from IBM Watson should return correct emotion, correct keywords, correct overall sentiment, correct concept and correct syntax.
- Application should label each answer of questions correctly and their value as a variable for algorithm. Algorithms must work coherently for the same inputs.

1.2.4 Security

- The answers of questions should be accessible only by the user should they decide not to give feedback.
- Outside of the application, the result of personality tests should be only accessible by Crystal Knows since Crystal Knows has the right to keep the results of tests.
- Any given personal information such as passwords and social platform information should be encrypted and protected that only user and application developers can access.
- Personal data of the user should not stored if the user deletes it’s account

1.2.5 Maintainability

- Application should be fixable for less than an hour in case of any wrong output
- Application should be updatable for future implementations for less than an hour.
- Servers should be checked for any undesired behavior for every day

1.2.6 Scalability

- Application should process more than 100 thousand personal data.
- Application should answer more than 2500 users simultaneously.
- Database should be scalable upto 100 thousand users for the first stage.

1.2.7 Recoverability

- In case of a system shutdown or unexpected failure, application should return a functioning state and remember the questions that user has already answered.
- Database should keep the state of the last safe situation and in case of any problem database should return this state.

1.3 Definitions, acronyms, and abbreviations

Personality Test: The 16 Personalities - also known as Myers-Briggs Personalities - Test which will be integrated with Crystal Know API [1].

Personality Type: Myers-Briggs Personality types which are provided by Crystal Know API. Those personality types include: The Inspector - ISTJ, The Counselor - INFJ, The Mastermind - INTJ, The Giver - ENFJ, The Craftsman - ISTP, The Provider - ESFJ, The Idealist - INFP, The Performer - ESFP, The Champion - ENFP, The Doer - ESTP, The Supervisor - ESTJ, The Commander - ENTJ, The Thinker - INTP, The Nurturer - ISFJ, The Visionary - ENTP, The Composer - ISFP [2].

Suggestion: After evaluating a user's personality results and his/her problem, provides an answer to the specified problem with the use of natural language processing.

Feedback: User's advice, criticism or information related to the provided answer by the Csion, and its future results.

Client: The mobile application interacts with user

Server: The system communicates with client, provides data for the client

NLP: Natural Language Processing

API: Application Program Interface

REST: Representational State Transfer

UI: User Interface

Chatbot: A computer program designed to simulate conversation with human agents.

AWS: Amazon Web Services

1.4 Overview

Csion aims to optimize people's decisions about any subject based on their characteristic behaviours within a short amount of time. We are planning to use Myers-Briggs Personality types **[2]** to have an initial categorization of the users' characters **[1]**. This test is highly standardized and is esteemed in the field of psychology. There are 16 unique personalities and each person can find their personality type by solving simple and quick test questions. These types have certain borderline character traits that generally conform to the way people behave. We will use the "Crystal Knows" API to implement this test and get the results to use inside our application. Crystal Knows will also provide us data of over 6 million individuals who have done this test.

Although it is a deciding factor, personality alone is not the only variable when a person makes a decision. A detailed analysis of the subject that the user will decide, also needed to provide reliable suggestions. Because of that, in our project Csion, we will implement a Chatbot which uses Natural Language Processing techniques to give detailed analysis of a sentence or paragraph. We will ask users to write the problem itself, pros and cons. Then by combining that analysis and the person's general personality, we will give detailed suggestions to our users to help them decide.

Another way of using Csion will be through our question-answer section. In that section, users will only be able to get suggestions from predetermined categories such as changing company or changing department etc. After selecting the problem, we will ask predetermined questions to the user to get important information that affect the decision for that problem. After getting answers from users, we will use scientific methods to combine the personality of the user and the answers that we get to determine what users should decide. After this processing phase, we

will again show a detailed suggestion to the user. To increase usability of this section, questions will only include only yes or no questions and users will answer the questions by using a very convenient way. For yes, users will swipe to the left and vice versa. In case of any wrong question, another answer will be irrelevant and to answer as irrelevant users will swipe to top.

After giving suggestions, we will ask the user to enter a deadline for the decision. After the deadline day, we will ask the user to give feedback to our suggestion whether he chose what we suggested or not and his satisfaction rate about his decision. With using this feedback data and deep learning implementation, we will make our application learn and improve itself to make better and better suggestions for future questions.

2. Proposed software architecture

In this section, we explained our project's proposed architecture. This section includes an overview of the architecture, subsystem decomposition of our project, hardware/software mapping, persistent data management, access control and security, global software control, boundary conditions.

2.1 Overview

Csion is based on server-client architecture. We divided the implementation of client side as Android and IOS implementation with native languages. Our client side will only consist of and manage the view of the application. All computations and business logic will reside on the server side. Our server side provides REST API to get necessary data from the client, and it will also handle the backend processes of the application such as database management, natural language processing, security control etc.

2.2 Subsystem decomposition

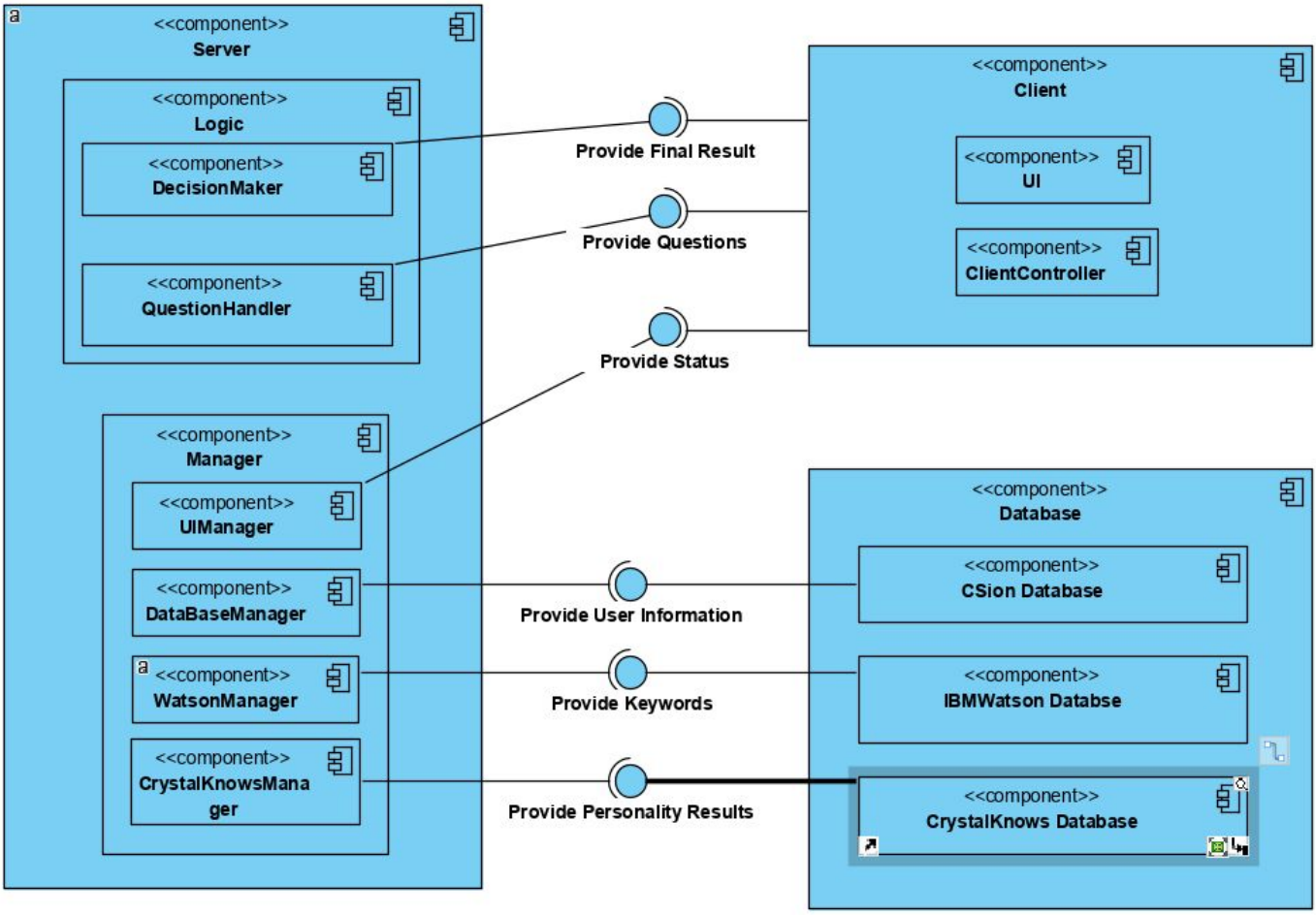

Figure 1. Component Diagram for Subsystem Decomposition

We have chosen a client-server architecture for implementing a decentralized structure to meet the design goals we have addressed. The subsystems are composed of Database, Server and Client. The Database component consists of several different Database components, each of which provides information for different managers in server-side. We have used two foreign databases: IBMWatson and CrystalKnows, due to using their APIs inside our own implementation. The server component has two container components for Logic and Manager. Logic handles providing the final processed results and selected questions to the client-side. Manager includes all of the components that receive the required relevant data from the database and maintains the application according to the input data. The last component, Client

is a visual representation tool to create a display according to the input data from the server component.

2.3 Hardware/software mapping

In Csion we have two main separation according to software that lives in different machines. One of them is our server-side software which is responsible for all the application logic, database management and calculations. The other one is our client-side software which is responsible for presentation and user interaction purely. While our server-side application will live inside a cloud machine like AWS or Azure, our client-side application will run in the users' smartphones. We plan to support both Android and iOS for our client-side application. In the following diagram you can see the visualization of our hardware/software mapping.

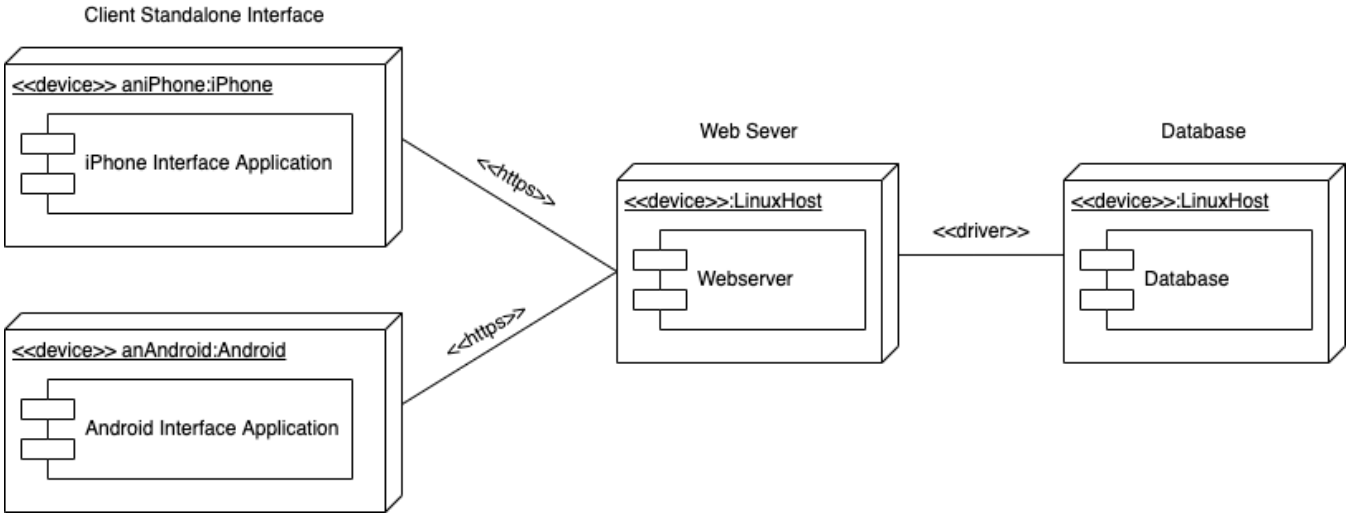

Figure 2. Hardware/Software Mapping

2.4 Persistent data management

The Csion will keep multiple users' data which are personality test results, previously asked questions, provided answers to these questions and users' feedbacks to these answers. Previously asked questions and the answers will be stored to be shown in users' personal profile pages. Personality test results will be stored to find educated answers to users' questions.

Multiple users are able to reach the system, at the same time. For this purpose, MongoDB will be used to store users' personal data. Nodejs will be used in database applications.

Two databases are planned to store information. All persistent data will be held in these two databases. One of these databases, which is the Client/User Database, will hold user data. Other, keeps the deals itself.

2.5 Access control and security

Every user in the system should be registered with a unique username and password, and take the first personality test to be given access to program's services. Since the users of the Csion shares personal data with the application, providing high as possible security is one of the main concerns of our program. Hence, the Csion will encrypt and store users' username and passwords.

2.6 Global software control

We have the client-server based architecture in our application where client is only responsible for showing output and taking input through user interface, and server is responsible for taking the input from the client and doing all the processing and computations to generate an output back to client.

Our server is designed to be event-driven in order to avoid race conditions because it should be able to serve multiple users at the same time. It will get data from many user requests simultaneously and since all of the functionality resides on the server, it should be capable of process and generating outputs for all client requests [3].

2.7 Boundary conditions

In Csion, our web server should have automated methods such as prewritten scripts for startup and shutdown which will start and shutdown the server with the given parameters. Those parameters should be written in a configuration file which the script reads and starts the server accordingly. By doing that, we will automate most of the process. We will not have to remember startup or shutdown parameters, and if we want to make a change we can simply change the

configuration file. After those changes, since the script will automatically parse the configuration file startup and shutdown commands will stay the same.

2.7.1 Startup

Startup process of our web server will only be done by the authorized people. In the process of the startup, database should be the first component to start. After initialization of database component, web server should be started. Since our web server includes all the necessary managers, all of those managers will also be started with the start of our web server.

For our client-side application there is no authorization issue for startup process, so that everyone can start their own application whenever he or she wants. Startup process of client-side application is trivial since opening the smartphone application is enough.

2.7.2 Shutdown

Shutdown process of our web server will only be done by the authorized people. In the process of the shutdown, our web server should be the first component to shut down. Since our web server includes all the necessary managers, all of those managers will also shut down with the close of our web server. After web server is closed database component can be closed safely.

For our client-side application there is no authorization issue for shut down process, so that everyone can close their own application whenever he or she wants. Shut down process of client-side application is trivial since closing the smartphone application is enough.

2.7.3 Error

Our web server should catch all the errors and log them according their error type without crashing itself, so that the server would never go down without intentional shut down.

Our client-side application should also catch all the errors originating from itself and should notify the web server that the application gave an error so that the server can log those client-side errors too. This way we will be able to troubleshoot the client-side errors also by looking to the log files. After the web server notified application should notify the user as well

and close itself. Our client-side application should also be able to notify the user about the server shutdowns and server-side errors.

3. Subsystem services

Csion will be an application that mainly sends data back-and-forth between client-side, server and database subsystems.

3.1 Client

Client component is responsible for presenting a display to the user on their smartphones. It creates a relevant display according to the current status of the server-side and enables the user to interact with the application in a meaningful way.

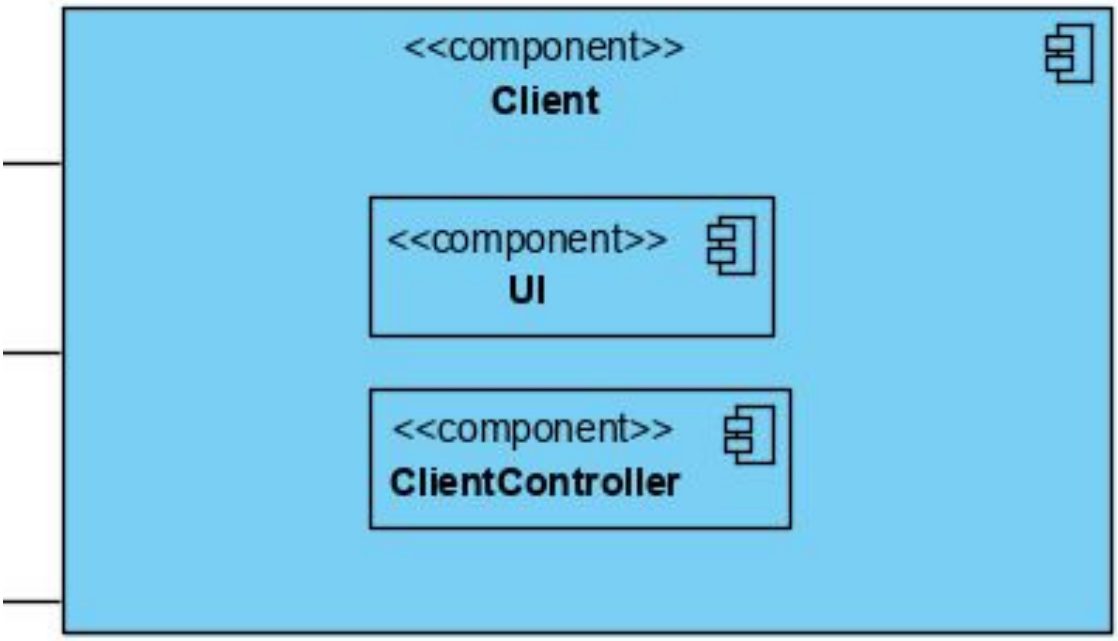

Figure 3. Client Service Components

3.1.1 User Interface

It enables the user to interact with the current visual representation and updates the view according to the input from the Client Controller. It has listeners to manage the current view of the application and current activity to display.

3.1.2 Client Controller

This component is responsible for the management of the client-side. It keeps track of the previous, current and next status to control the activity flow for the User Interface. It receives input data from the UIManager inside the server component.

3.2 Server

This component is responsible for establishing a connection between the client-side and the database. Also, it includes the main logic of the application that enables us to create meaningful results according to the input form the client. It has managers to handle the data flow and maintain the application instance.

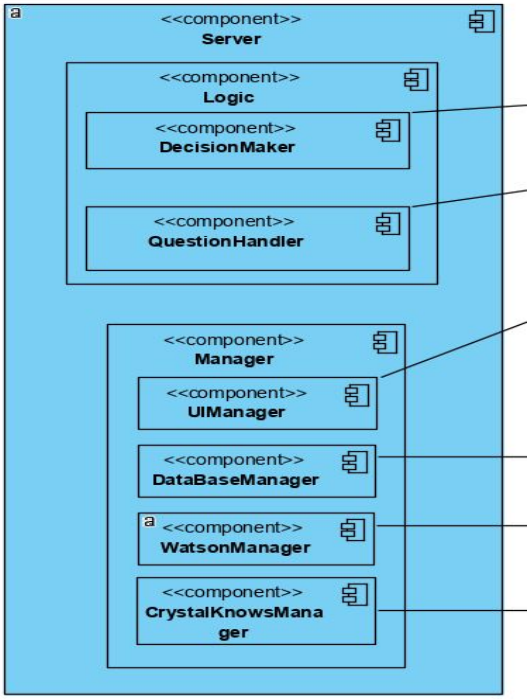

Figure 4. Server-side Components

3.2.1 Logic

This sub-component consists of algorithms to process the input data from the client and produce the optimum output for the client. The algorithms are implemented by considering our design goals.

3.2.1.1 Decision Maker

This sub-component is responsible for the production of the final output that is created to display to the user. It handles both the input from the answers from the questions and the keyword analysis of the text inputs.

3.2.1.2 Question Handler

This sub-component is responsible for the selection of the appropriate questions from the question pool according to the input from the user when they select the most relevant category for their problem. It provides a collection of selected questions to the client side to display.

3.2.2 Managers

There are four different managers to handle the data flow from the other components.

3.2.2.1 UI Manager

This sub-component is responsible for the interaction with the client controller in the client-side. It provides the current status to display to the user. It decides on the next status according to the input it receives from the other managers inside the server component. It serves as a link between the back-end and the front-end.

3.2.2.2 Database Manager

This sub-component is responsible for the control of the tables and entries within the Csion database. It provides the other managers with the necessary user data that comes from the Csion Database in the server.

3.2.2.3 Watson Manager

This sub-component is responsible for the connection between the foreign IBMWatson Database and the Csion server. It receives the relevant keywords that are generated from the IBMWatson API according to the user's text input and passes these keywords to the other managers.

3.2.2.4 CrystalKnows Manager

This sub-component is responsible for the connection between the foreign CrystalKnows Database and the Csion server. It receives the relevant personality type and the corresponding characteristic keywords from the CrystalKnows API according to user's answers to the Myers-Briggs personality test and passes these to the other managers.

3.3 Database

This component has one native and two foreign sub-components that handles the storage and manipulation of the used data.

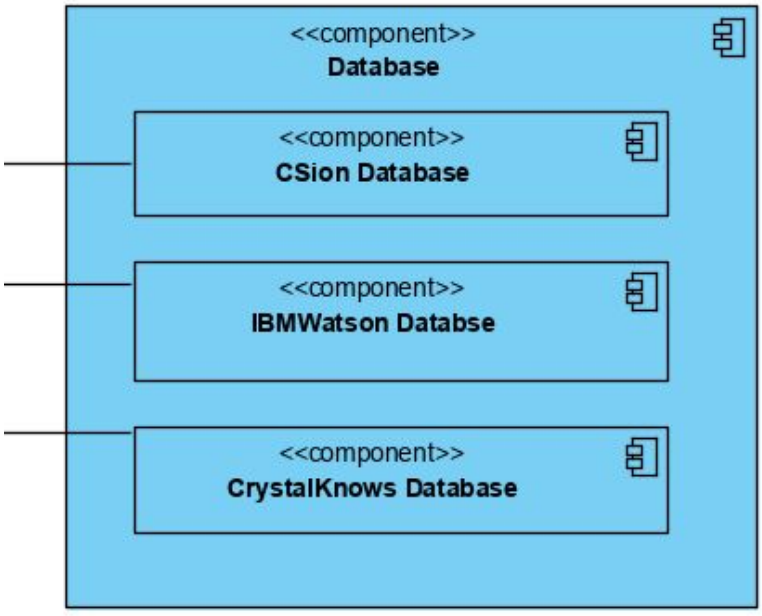

Figure 5. Database Service Components

3.3.1 Csion Database

This component is responsible for the storage of all user information for each user as well as the question pool that includes the questions to identify the problem more accurately. It provides the user information and the question sets to the Database Manager on the server-side.

3.3.2 IBMWatson Database

This component is responsible for the storage of all customized NLP keywords. This foreign database provides the matched keywords with emotions and sentiment values to Watson Manager on the server-side.

3.3.3 CrystalKnows Database

This component is responsible for the storage of all personality types and characteristic keywords according to the Myers-Briggs analysis. This foreign database provides the matched personality type and characteristic keywords to CrystalKnows Manager on the server-side.

4. New Knowledge Acquired and Learning Strategies Used

Although we have not started the implementation of the project yet, we have started to acquire new knowledge about many issues. For starters, we have done many online research about the personality assessment APIs, such as CrystalKnows API. These researches made us realize that these APIs can satisfy our needs to create a personality test, so we decided to use them in future.

We made some interviews with software engineers who have experience of building cross platform applications, and they recommended us to build our application using native languages because of the instabilities of cross-platform toolkits, and lack of sources for us to get help when we need. Since we need to make our implementations as fast and efficient as we can, we decided to proceed with native approach.

We spent some time on IBM Watson to understand its usage and manipulate it to fit our purposes. We made online research for this purpose and already made some progress on its usage. We also practiced on NodeJS by doing exercises and watching videos, and we built a working server. On top of it, we have made research on databases and decided to use MongoDB for our project. We connected the database to our server, but we still have more on it to learn, so we will continue to make research on MongoDB.

For future learning, we need to learn the IOS application development for our project to support IOS as well, and Android XML design for our UI on smartphones with Android operating system. We will watch crash-course videos and make use of online resources, like Coursera. We need to understand IBM Watson technology better and work more on it to make it fit for our application. We also did not make a significant process on chatbot APIs and natural language processing yet, so we need to make more research on them, and ask for advice from experts, if needed.

5. References

[1] “The purpose of the Myers-Briggs Type Indicator® (MBTI®) ,” The Myers & Briggs Foundation - MBTI® Basics. [Online]. Available: https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1. [Accessed: 13-Oct-2019].

[2] “Personality Types,” *16Personalities*. [Online]. Available: https://www.16personalities.com/personality-types. [Accessed: 30-Dec-2019].

[3] “Global software control,” *Assembla*. [Online]. Available: https://app.assembla.com/spaces/Melange/wiki/Global_software_control. [Accessed: 30-Dec-2019].