Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: CSION

Low-Level Design Report

Group Members: 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

In this report we explained the low level, implementation details of our project such as design trade-offs, guidelines that we followed, engineering standards, definitions, packages, class interfaces etc.

1.1 Object design trade-offs

In this section, we explained the trade-offs that we have faced while designing our senior project, Csion.

1.1.1 Usability vs Functionality

In our mind there were many ideas and many features to add to this system but we have to give up from some of them because our system aims to provide fast suggestions to people who cannot decide about a topic. While doing that, users should not encounter complex interfaces or problems while navigating through different parts of the application because this would have a considerable effect on the user experience; no person would like to use such a system while trying to get a fast answer about their hesitant thoughts. Therefore, the system should be sufficiently usable for effective usage by the users. Hence, to be able to increase the usability of the system we need to diminish the number of functionalities for creating a better user experience and user satisfaction.

1.1.2 Quality vs Cost

Since we are working in the psychology field as well, we need to take help from psychological personality tests. Some of those tests are not free to use and cost too much money but are very accurate and effective. So, we have to decide about the quality of our application vs cost that it requires to be developed. In this decision we choose to go with the cost effective solution and drop some of the proprietary tests like Myers Briggs test and we decided to go with DISC and Big-Five tests which can be used freely. While those tests are also based on scientific research, they might not be as accurate as Myers Briggs and they might not give detailed answers as Myers Briggs can. However, we agreed on if the application shows some success, we will add many more tests to increase the quality of our application and these tests will include the proprietary tests like Myers Briggs.

1.1.3 Rapid Development vs Accuracy

One last trade-off that we faced is deciding between rapid development vs accuracy. Since Natural Language processing is a very advanced topic and we did not know deeply about it, rather than designing our own Natural Language Processor, we decided to go with the premade but configureable one, IBM Watson. By making this decision we really cut off the development time that we need for this project. We will configure Watson to fit our needs, and for minimum viable product that will be enough. However, as in the previous trade-off, if the application shows some success, accuracy of the Natural language Processor will be the first topic to improve.

1.2 Interface documentation guidelines

Table 1. Interface Documentation Table Example

Class (or Component)	
Class Name	
Class Description	
Variables (if any)	
Variable Type	Variable Identifier
Variable Description	
Methods (if any)	
Method Type (for Request Methods) Return Type (for normal methods)	Method Identifier
Method Description	

In this report, we will represent the class interfaces in the format of Table 1. Realize that the variable and method return types are defined in the following parts in a way that they do not necessarily need to match with the implementation itself since some languages does not have type declaration. For the sake of making the explanations more easy to understand, we chose to write types that we believe makes more sense.

The “+, -, #” symbols will be added as access modifiers whenever necessary. ‘+’ denoting public, ‘-’ denoting private, and ‘#’ denoting protected variables or methods.

1.3 Engineering standards (e.g., UML and IEEE)

The diagrams in this report follow the Unified Modeling Language (UML) standards. To cite the references, on the other hand, we have referred to The Institute of Electrical and Electronics Engineers (IEEE) guides.

1.4 Definitions, acronyms, and abbreviations

Here you will find the definitions, acronyms, and abbreviations that we used throughout the report.

- **Personality Test:** DISC and Big-Five personality tests.
- **Personality Type:** Short name for DISC and Big-Five Personality types. DISC personality types include: The Captain - D, The Driver - Di, The Initiator - DI, The Influencer - Id, The Motivator - I, The Encourager - Is, The Harmonizer - IS, The Counselor - Si, The Supporter - S, The Planner - Sc, The Stabilizer - SC, The Editor - Cs, The Analyst - C, The Skeptic - Cd, The Questioner - CD, The Architect -Dc [1].
- **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
- **HTML**: Hyper-Text Transfer Language

- **CSS**: Cascading Style Sheets
- **JSON**: JavaScript Object Notation

2. Packages

In this section, we explained the inner workings of our software, such as packages, managers, variables, methods etc.

2.1 Client Package

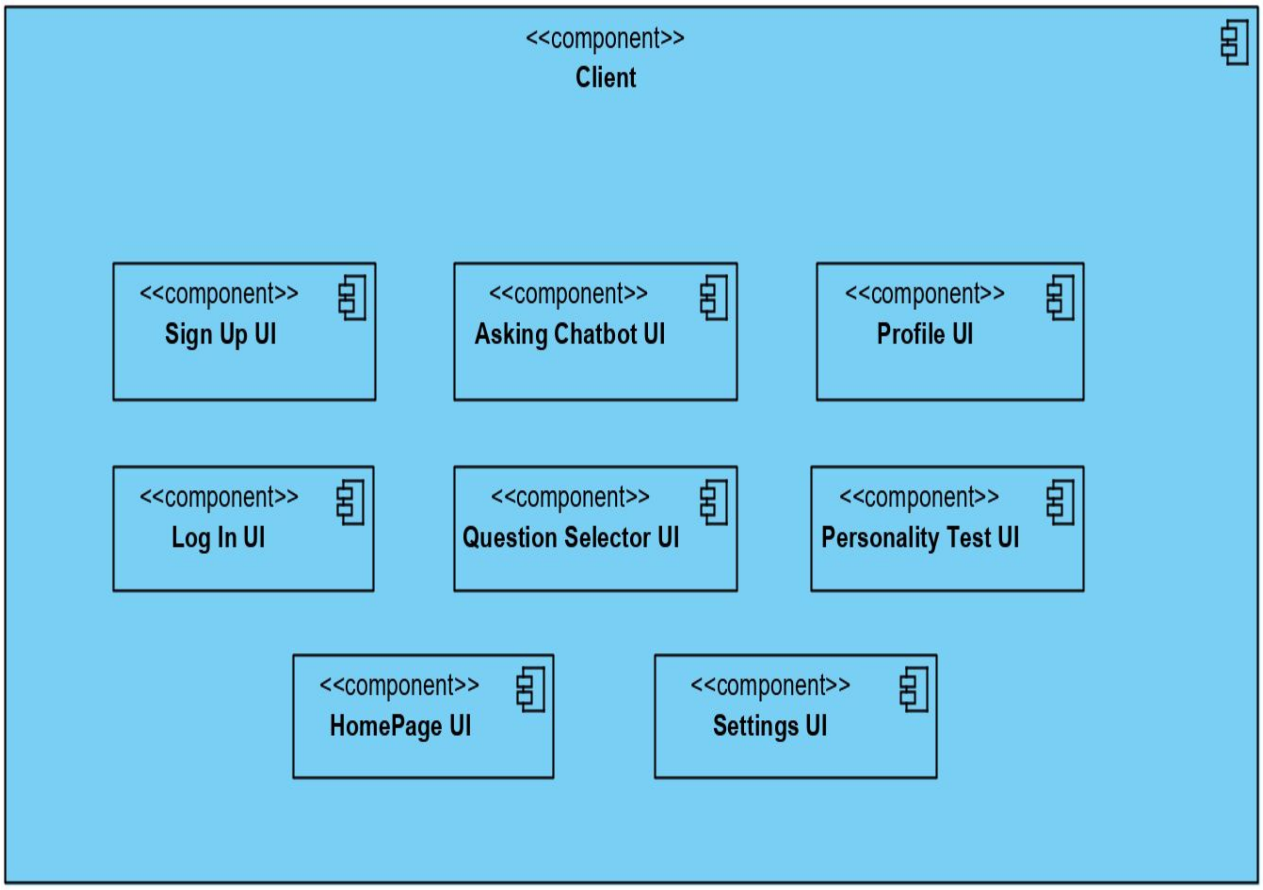

Figure 1. Client Package Representation

This package is composed of the user interface of different pages for clients to interact with the application. Client pages get input from users and send requests to the server-side if any information from the database is needed. Then, it gets the response from the server, processes the response if needed, and displays the required outputs to the user.

2.1.1 Sign Up UI

A registration form will be displayed to the user, and the user will fill the empty form with user information and click the *Sign Up* button. In that case, this component will send the user information to the server and process the response, if the written info is accepted, Personality Test UI will be displayed. If a user already has an account, by clicking *Log In*, s/he will be navigated to the Log In UI.

2.1.2 Log In UI

A login form will be displayed to the user, and the user will fill the empty form with necessary information and click the *Log In* button. In that case, this component will send the user information to the server to verify, if the user info is valid, HomePage UI will open. If a user does not have an account yet, by clicking *Sign Up*, s/he will be navigated to the Sign Up UI. User can click *Forgot your password?* link to request a new password via email through a pop-up.

2.1.3 HomePage UI

Users will be able to navigate through different UI's from here. There will be a footbar having navigations to Settings UI, Profile UI, Asking Chatbot UI, and Question Selector UI. The default UI displayed inside this UI will be Question Selector UI.

2.1.4 Settings UI

Users can change the notification and theme preferences through this screen and also change password by confirming old password, typing a new password and clicking the *Submit* button. User can suspend the account by clicking the related link, read further information about the application by clicking the Help button, read about the creators and sponsors of the application by clicking *About Us* button and finally read the Terms & Conditions again by clicking the related button.

2.1.5 Profile UI

At this screen, user can see and edit user information, see the personality test result by clicking *Personality* tab, specialties of former questions he asked by clicking *Former Q/A* tab and also give feedback to old questions he asked when the deadline is passed, and finally take additional tests and see their results by clicking *Additional Tests* tab. Clicking plus icon at *Former Q/A* tab navigates user to *Give Feedback* screen, and clicking *Take the test!* or *Repeat the Test* button at *Additional Tests* tab or *Take the test again!* at *Personality* tab will navigate user to *Personality Test Screens* having the test s/he chose.

2.1.6 Personality Test UI

At this screen, we will show users some questions about their personalities and except them to answer that questions by clicking responding answers. This page, then, send these answers to server for it t decide user's personality type. After receiving the personality type of user, the component will display the details of this personality type to user.

2.1.7 Question Selector UI

This part displays a list of categories and user can choose the category, and sub-category that is related with his/her question. After that, user should pick a deadline by using the datepicker below that represents until which date s/he should decide. By clicking the *Next* button, user navigates to a Q/A Screen. User will be able to give answers by swiping the question cards to left & right. This component will collect the answers and send them in a request to server to get a decision from system. Then, it displays the final decision the system gave for user.

2.1.8 Asking Chatbot UI

At this page, user will be given an empty form to fill out. After user fills the responding boxes, and submit by clicking related button, the component will send the form to server, and get a response. This response will include the decision made for the user, and it will be displayed on the screen.

2.2 Server Package

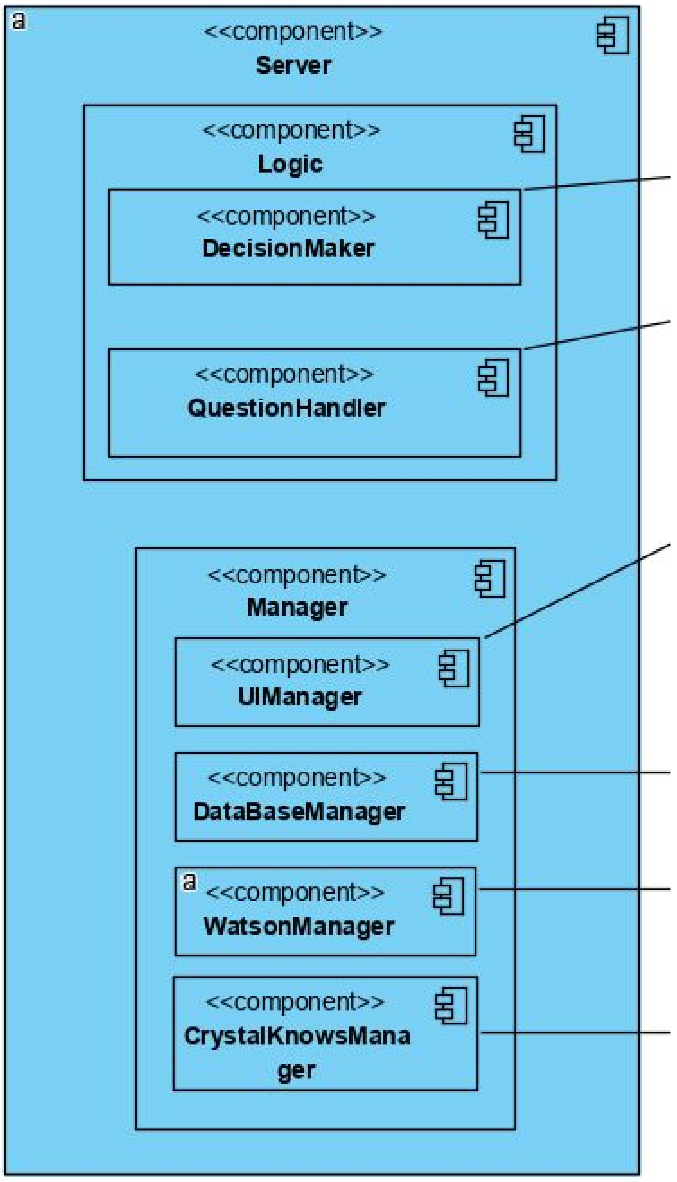

Figure 2. Server Package Representation

This package 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 from the client. It has managers to handle the data flow and maintain the application instance. It has 2 main parts: Logic and Manager.

2.2.1 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.

2.2.1.1 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.

2.2.1.2 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.

2.2.2 Managers

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

2.2.2.1 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.

2.2.2.2 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.

2.2.2.3 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.

2.2.2.4 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 DISC personality test and passes these to the other managers.

2.3 Database

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

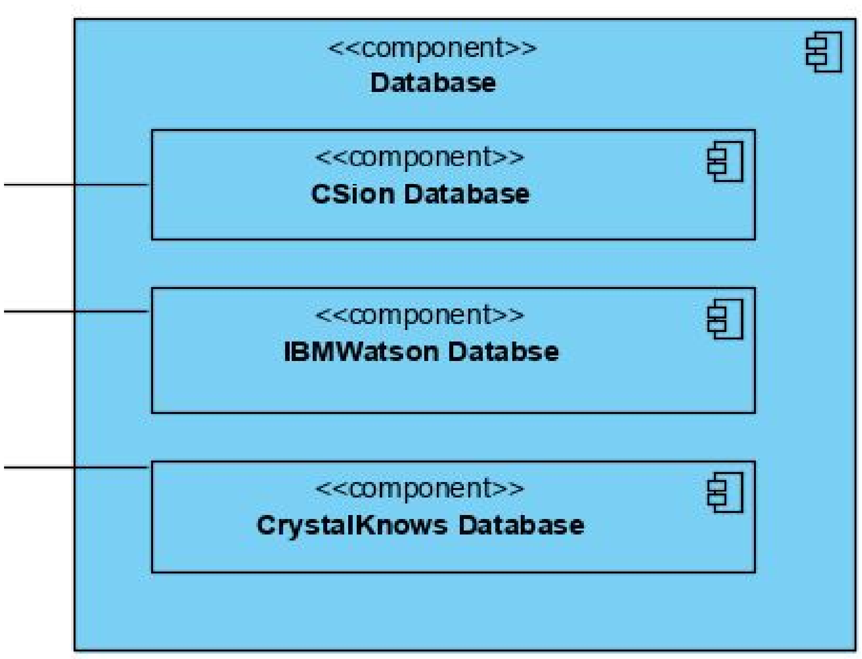

Figure 3. Database Service Components

2.3.1 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.

2.3.2 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.

2.3.3 CrystalKnows Database

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

3. Class Interfaces

The project consists mainly of two implementation tiers: Client Tier and Server Tier. The following are the class representations of the tiers and their component analyses.

3.1 Client Tier

Client side consists of the components that make up the user side's experience. These components form the basics of the control flow for the user.

Table 2. Sign Up UI

Component	
Sign Up	
This component will be used for registration of a new user.	
Variables	
String	e-mail
E-mail of users will be taken as an input from the form displayed on the screen.	
String	username
Username of user will be taken as an input from the form displayed on the screen.	
String	password
Password of the user will be taken as an input from the form displayed on the screen.	
Request Methods	
post	registerUser(username, password)
This method will send the variables taken from form to server. It will check the response coming from the server as well. If the response status is OK, then it will navigate to the Personality Test Page, if not, it will display the corresponding error and empty the form for the user to fill again.	

Table 3. Log In UI

Component	
Log In	
This component will be used for registered users to log in to the system.	
Variables	
String	username
Username of user will be taken as an input from the form displayed on the screen.	
String	password
Password of user will be taken as an input from the form displayed on the screen.	
String	e-mail
E-mail of the user will be taken as an input from the form that will be displayed on the screen if a user clicks to “Forgot my password” link.	
Request Methods	
post	authCheck(username, password)
This method will be called when the user clicks the Login button. It will send the username and the password to server, and, check the response coming from the server as well. If the response status is OK, then it will navigate to HomePage, and if not, it will display the corresponding error and empty the form for user to try again.	
get	sendNewPassword(email)
This method will be called when user clicks forgot my password link, fill out the required data, and clicks to Send New Password button.	

Table 4. Personality Test UI

Component	
Personality Test	
This component will be used for the system to assign a personality type to newly registered users. It will also be used for giving additional tests to requested users. It will display a test consisting of many questions and collect the user input.	
Variables	
String[]	questions
Pre-typed questions that will be shown to users will be in this array.	
int[]	answers
The answers that users gave to each question (from 1 for very inaccurate to 5 for very accurate) will be held in this array.	
String	testName
This variable will be “big5” if the component is opened from sign up page, and, if it opened from additional tests, it will be the chosen test’s name.	
Request Methods	
post	getPersonalityType(answers)
This method will send the answer array to the server, and get a string response including the personality type of the user, and display it on the screen.	
get	getQuestions(testName)
This is a request for the server to respond by giving the questions of the required test.	

Table 5 HomePage UI

Component
Home Screen
This screen will have a footbar below and the first tab of the footbar will be selected as default. The interface of the tab selected will be displayed above the footbar in this component.

Table 6. Question Selector UI

Component	
Chatbot Decision Service	
This component is one of the components that serves for the main functionality of the system, that is, helping users to give decisions. Users should fill the given form and submit to see a decision advice on the screen.	
Variables	
String	question
The main question that the user has in mind is typed to displayed form.	
String[]	prosCons
The first element of the array will include the pros input, and the second element will include the cons input.	
Date	deadline
This variable responds to the deadline of the decision which will be made which is taken as an input from the user through a datepicker.	
Request Methods	
post	getDecisionChatbot(question, prosCons, deadline)
This method will send the parameter variables to the server and get a decision string as a response, and display it on the screen.	

Table 7. Asking Chatbot UI

Component	
Choosing Categories for Decision	
This component is the other component that serves for the main functionality of the system, that is, helping users to give decisions. Users should choose a main and sub-category, and answer several questions as yes or no to see the advised decision.	
Variables	
String[]	questions
The questions that will be displayed to users after they choose categories.	
String	category
The sub-category name that user chose on the screen by clicking the button with that name.	
boolean[]	answers
This answers that users gave to displayed questions. They will be held as true for yes, false for no, null for others.	
Date	deadline
The deadline that the user chooses after giving answers to questions.	
Request Methods	
post	getDecisionCategory(category, answers, deadline)
This method will send the parameter variable to the server and get a decision string as a response, and display it on the screen.	
post	getQuestions(category)
This method will send the chosen sub-category name to server and get and fill the questions array as the response.	

Table 8. Profile UI

Component	
User Profile	
This component displays all information -except the password- of the logged user, including the details of their personality type. Former questions that users asked to the application, and the answer they get will also be shown. Users will be able to give feedback for advised decisions. Users will also be able to take additional tests from this page.	
Variables	
String[]	userInfo
This array will hold the name, surname, username, email, age, profile picture link of the user.	
String[]	personalityDetails
This array will hold personality type name and details.	
FormerCategoryAdvice[] *	categoryAdvices
This array will hold the details of the former advice user has taken from the categorical decision advising system. The object of this array will consist of these parameters: enum status, enum category, String[] advice, boolean isAdviceFollowed, int satisfactionRate, boolean isFbGiven, Date deadline. isAdvicedFollowed and satisfactionRate will only be displayed when the feedback for the related advice is given, that is, when isFbGiven becomes true.	
FormerChatbotAdvice[] *	chatbotAdvices
This array will hold the details of the former advice user has taken from the chatbot decision advising system. The object of this array will consist of these parameters: enum status, String[] questionDetails, String[] advice, boolean isAdviceFollowed, int satisfactionRate, boolean isFbGiven, Date deadline. isAdvicedFollowed and satisfactionRate will only be displayed when the feedback for the related advice is given,	

that is, when isFbGiven becomes true.	
AdditionalTest[] *	additionalTests
This array will hold the information of additional tests. The object of this array will consist of these parameters: String testName, String details, boolean isTaken, String results, boolean isNew.	
Feedback *	feedback
This variable will only be defined if the user clicks on a give feedback button. As the feedback form is filled and saved, this object will be defined and sent to the server through the newFeedback method. Feedback object will include 4 variables: String question, Date deadline, boolean isAdviceFollowed, int satisfactionRate.	
Request Methods	
get	getProfileDetails()
This method will be called at the initialization of the component and it will send a request to server and get a response that includes all necessary information for variables of the component to display the user on the screen.	
post	editDetails(userInfo)
When / If the user info is edited by the user, after the submission of the edited information, this function will be called. It will send the edited details to the server.	
post	newFeedback(feedback)
When the user gives feedback by filling out the feedback form, the feedback variable is set and when the user submits the form, it will be sent to the server by this function call.	

**These identifier types will not exist as distinct classes, they are JS objects that only the Profile component will have.*

Table 9. Settings UI

Component
Settings

This component is mainly for the arrangement of user preferences & giving information. User can change theme, toggle on/off the notification choice, change password, read help documentation, credits or terms & conditions, and suspend the account.	
Variables	
String	themeChoice
The hex color code of the theme color chosen by the user.	
boolean	allowNotifications
Notification preference of the user.	
Request Methods	
get	getPreferences()
This method will be called at the initialization of the component. It will request the preferences from the server and the returned response will include the info about themeChoice and allowNotifications variables.	
post	changeNotificationPref(allowNotifications)
This method will send the notification preference to the server, if changed.	
post	changeThemePref(themeChoice)
This method will send the new theme choice of the user to the server.	
post	changePassword(newPassword)
This method will send the input it get from the change password form to the server, if the form is filled and submitted. According to the status of the response, it will notify the user if the password changing is successful or not.	
get	suspendAccount()
This method will send the server a request to let it know that the user wants to suspend the account.	

3.2 Server Tier

Server side is responsible for all of the calculations, scoring and data retrieval operations for the application to operate.

Table 10. UIManager Class

Class	
UIManager	
This class acts as a mediator between the server side's managers and the client side	
Variables	
express	router
An instance for handling the requests	
Methods	
post	postRequest(req, res)
The listener function for post requests	
get	getRequest(req, res)
The listener function for get requests	
delete	deleteRequest(req, res)
The listener function for delete requests	

Table 11. ServerManager Class

Class
ServerManager
Handles the manager classes in the server sides.
Variables

CrystalKnowsManager	CKManager
An instance of CrystalKnowsManager for operating its method calls. When the user completes the DISC personality test, ServerManager gets results through CrystalKnowsManager.	
WatsonManager	WManager
An instance of WatsonManager for operating its method calls. Handles the text analysis for emotions and sentiment values for the texts to be analyzed.	
QuestionManager	QManager
An instance of QuestionManager for operating its method calls. Controls the process of ordering and evaluation of consecutive questions in the decision tree implementation.	
DecisionMaker	DMaker
An instance of DecisionMaker for operating its method calls. According to the results from the WatsonManager and QuestionManager, it provides a final output by evaluating the scores.	
DatabaseManager	DBManager
An instance of DatabaseManager for operating its method calls. This manager is responsible for the interaction between server and the database.	

Table 12. CrystalKnowsManager Class

Class	
CrystalKnowsManager	
Handles the analytical analysis part of the personality test of CrystalKnowsAPI	
Variables	
String[]	testAnswers
The results from the personality test.	

String[]	keywords
The characteristic tags for the personality type.	
String[]	resultsOutput
The overall output of the personality test result.	
Methods	
none	sendAnswers (testAnswers)
Send test answers to UIManager	
none	setResults (resultOutput)
Get the results from the CrystalKnowsManager and set the results output accordingly.	
none	setKeywords (keywords)
Get the keywords from the CrystalKnowsManager and set the keywords from output accordingly.	
none	sendKeywordsToClient (keywords)
Sends the keywords to UIManager.	
none	sendOutputToClient (resultOutput)
Sends the output to UIManager.	

Table 13. DecisionMaker Class

Class
DecisionMaker
The class that is responsible for the evaluation of the given answers by the user and deciding on the key features to be used later in the finalAlgorithm.
Variables

String[]	keywords
The keywords from the answers of the questions in the decision tree.	
String[]	questions
Asked questions to the user.	
String[]	NLPText
The inputted to the NLP part.	
String[]	analyzedResult
The text evaluation of the comparison between the keywords from the questions and the personality type keywords.	
String[]	finalOutput
The output from the finalAlgorithm.	
double[]	scores
The scores from the comparison between the keywords from the questions and the personality type keywords.	
Methods	
none	getKeywordsFromQuestions (questions)
Get keywords from the asked questions.	
none	getKeywordsFromPersonality ()
Gets the keywords from the user's personality type.	
none	getResultsFromWatson (NLPText)
Get keywords and scores from the Watson according to the text that the user has written.	
none	createScores (questions)
Sets scores according to the comparison between the keywords.	

none	finalAlgorithm (keywords, scores)
The main algorithm to produce the result report according to the keywords and scores of the current inquiry.	

Table 14. QuestionManager Class

Class	
QuestionManager	
The class that is responsible for the implementation of the decision tree. Enables the program to select the most relevant questions to be asked to the user as new answers are provided..	
Variables	
String[][]	questions
A double dimension array which holds questions and their tags/keywords.	
String[]	keywords
A list of major tags which will help to classify questions.	
Methods	
none	selectQuestions():
A method to select the most relevant question to given context.	
none	sendQuestions
A method to send relevant questions to the user to detect the final decision answer.	
none	setKeywords(keywords)
A method to organize keywords/tags of the questions.	
none	sendKeywords(keywords)

A method to send keywords of the questions to the system. In that way, the user will be faced with relevant questions of the question pool for the further investigation of the user’s problem.

Table 15. WatsonManager Class

Class	
WatsonManager	
This class is responsible for communication between IBM Watson and Csion	
Variables	
String	text
the text message to call and use Watson Manager’s methods.	
String []	keywords
The important keywords in the content, organized by relevance.	
double[]	scores
Rates of major tags in given text input	
String []	analyzedOutput
The result which is evaluated according to keywords in the given text and major tags.	
String []	emotion
Emotion analysis results for the keyword, enabled with the emotion option.	
Methods	
none	sendQuestion(text):
Send user input to the assistant and receive a response.	
none	getKeywords()

A method to get keywords, which are important words in the content, organized by relevance.	
none	setKeywords(keywords)
A method to organize keywords/tags of the questions.	
none	setEmotion(analyzedOutput, emotion)
A method to organize keywords according to the analyzed output.	
none	getEmotion(analyzedOutput, emotion)
A method to reach emotion, which is emotional analysis of the input keywords.	
none	sendResults()
A method to send final results to the system. By these results, our application will be able to understand user input, and its analysis.	
none	sendAnalysis(analyzedOutput)
A method to send final analysis according to keywords in input and emotional analysis.	

Table 16. DatabaseManager Class

Class	
DatabaseManager	
A class that handles communication between Csion Database and Csion Server	
Variables	
mongoose.Schema	userSchema
The schema for users that store database information	
mongoose.Schema	questionsSchema
The schema for questions with keywords and tags for each of the questions.	
mongoose.Schema	personalityTypeSchema

The schema for user's personality types and characteristic keywords according to DISC assessment.	
Methods	
none	initDatabase ()
Initializes the database to be ready in the running state.	
none	dropTable (mongoose.Schema)
Drops the database table.	
none	updateTable (mongoose.Schema)
Updates the database table	
none	getEntity(mongoose.Schema)
Returns the required entity from the database	
none	getEmotion(analyzedOutput, emotion)
A method to reach emotion, which is emotional analysis of the input keywords.	
none	insertOperation (mongoose.Schema)
Insert an entity to the specified place in the database.	
none	deleteOperation(mongoose.Schema)
Deletes an entity to the specified place in the database.	

4. References

[1] “Free DISC Personality Test: Crystal Knows,” *Free DISC Personality Test | Crystal Knows*. [Online]. Available: https://www.crystalknows.com/disc-personality-test. [Accessed: 10-Feb-2020].