AVA An Automated Voice Activated Advisement System

(Design Document)

By: The Software Gurus

Team Members:

Omar Obidat

Jesse Parron

Allen Asencio

Dan Ferdetta

Sumit Mistry

Software Engineering 515

Fall 2022

Professor Johnson

October 31, 2022

**Table of Contents 2**

**Requirement and Specifications 3**

**Use Case Diagram 7**

**Activity Diagram 10**

**Top Level Design 11**

Language Recognition for recommendations and feedback: 11

Connecting to the Schools database: 12

GPA calculator: 13

Unofficial Transcript generation: 15

Course catalog viewer: 16

**Detailed Design 18**

**Pseudocode 19**

**Requirement and Specifications**

Product Scope:

College campuses are known for having a surplus of majors/minors and hundreds to thousands of potential options for classes a student may need to take. With that being said, it may be hard to figure out which courses one should take when pursuing a degree. Students may accidentally take classes they don’t need, or take too many classes and overload themselves. In order to advise students in an efficient manner that doesn’t take too many labor hours, a voice activated advisement system has been developed in order to provide assistance in all areas of student advising. The system will allow a user to verbally communicate with an automated adviser, who will guide them through the process of picking the correct course(s).

Product Features:

* Recognize user voices.
* System should allow administrative views for editing and updating software.
* Be up to date and flexible with changing university graduation requirements and courses no longer being taught.
* Allow students to log in and interact with the advising system. Such functionalities are:
  + View their course catalog.
  + View their transcript.
  + View available courses that can be taken.
  + Talk with an automated system that will provide feedback on any questions they have.
  + System should be user friendly and easy to use.
  + Student can filter the advisor system to specific:
    - Credit amounts in a given semester.
    - Full time/part time course loads, if an undergraduate student is partaking in 12 or more credits they are considered full-time, anything less is part-time.
    - Time period of classes in the day.
    - Preferable concentrations of what courses they want.
  + Ask the system to calculate the current GPA of students, and the GPA they would have after the completion of following semesters courses. If GPA is low, the advising system will notify students about possible academic probation.
  + System should be able to generate multiple courses the user can pick from and the amount of courses they would need in order for the student to graduate on time.
  + System can give different instructor choices for the classes that are offered during the semester of the students choice. It will display the names of each professor that will be teaching the course, so if a student has taken/likes that professor then they can choose that course.

Users Characteristics:

* Students
  + The typical undergraduate student is between 18-23 years of age, although there are cases where some will be outside of said range.
  + Education level is undergraduate.
  + All undergraduates using the system are majored and/or minored in the Computer Science and Information Technology (CSIT) Department.
* School Administration
  + Presidents, Deans, Chairmen, Professors.
  + Has the privileges to add/update any courses or students that need to be modified in the system.
  + Can view any information that the system advises to a student.
  + Bypass course size limits for students who require an already full class for graduation.

Constraints:

* Development tools: system will be built using Python, HTML, CSS, Bootstrap, JavaScript, Sqlite, and Flask.
* The developers are limited in terms of the level of security of the system; The developers can implement hashing for the passwords and disconnect user sessions when trying to go back after logging out.
* The developers will follow school regulatory procedures for the system.

Assumptions and Dependencies:

* The developers have access to the university's database.
* The development team all have access to the system’s code.
* The system has a reliable server or system to be installed on.
* Changes in the original requirements may occur.

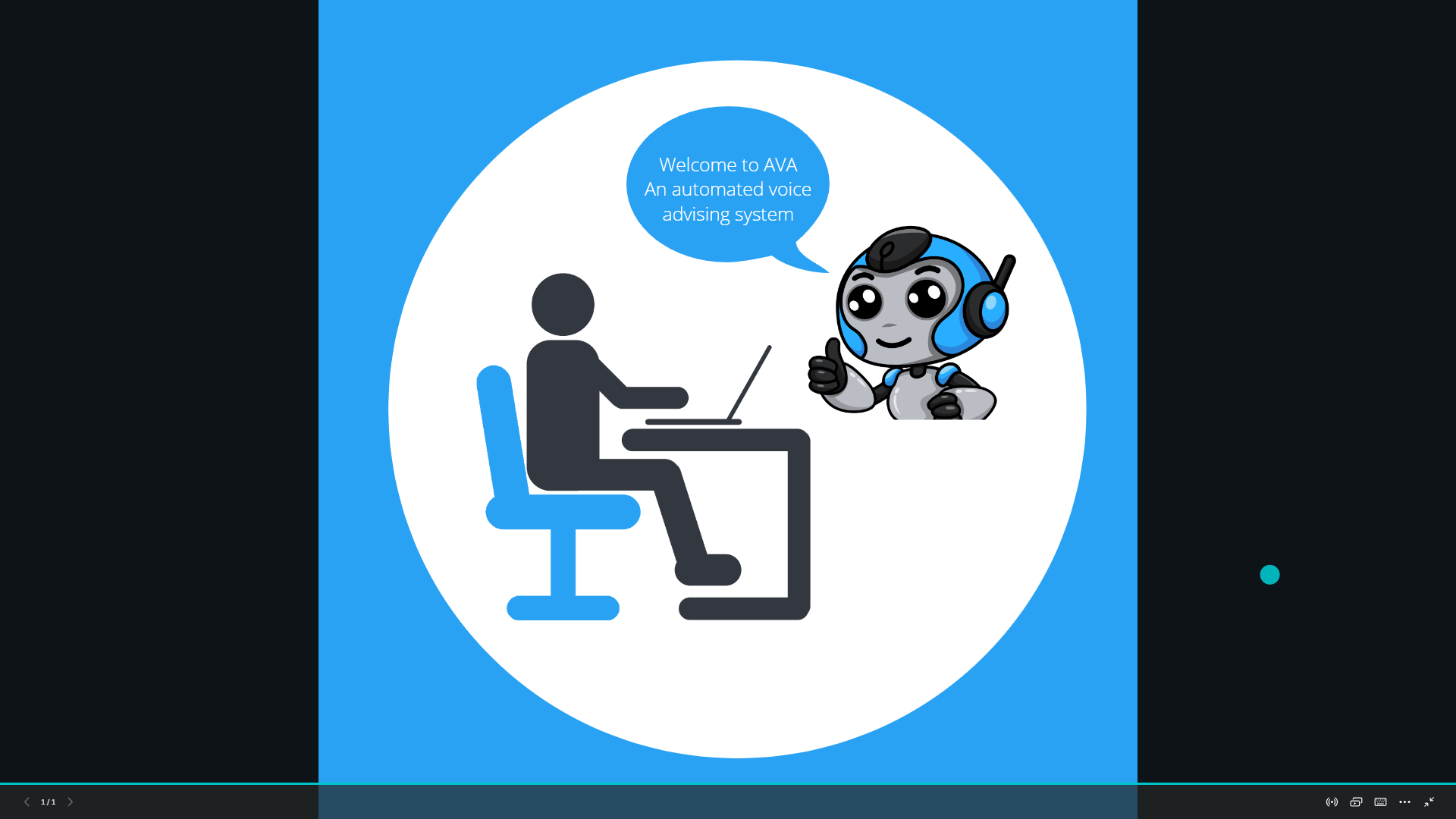


Figure 1

**Use Case Diagram**

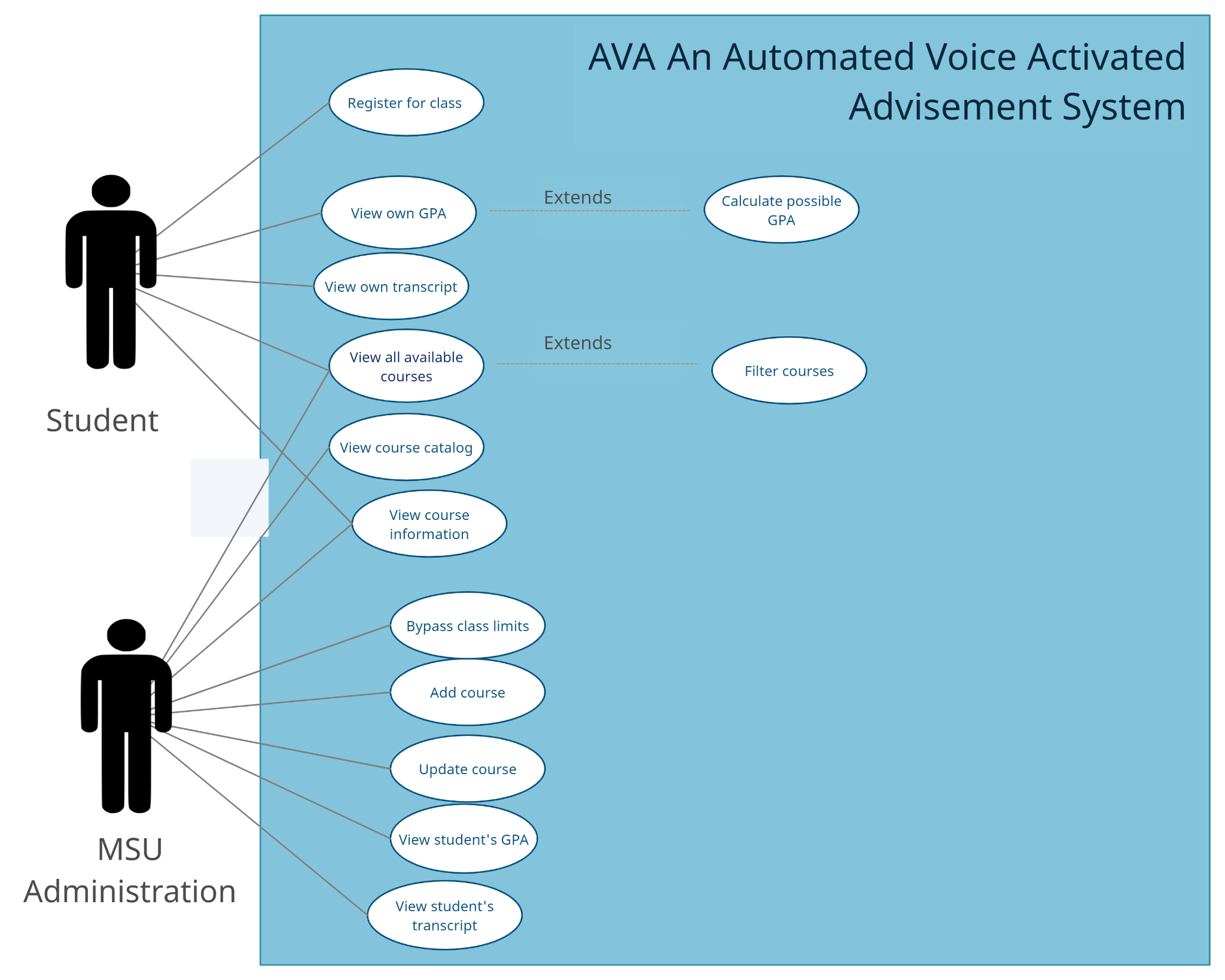
****

Figure 2

#### 

**Activity Diagram**

#### 

Figure 3

#### **Top Down Design**

#### **Connecting to School Database:**

* **Abstract**

The School Database stores the information about the student and Montclair State University’s various departments. For students to access the information, the system will need to establish access to the school’s database. This component will be an integral part of AVA system that will provide just the database authentication and a connection to the system. When a user asks for the information like; currently available course in semester or simply the last semester GPA, the system will authenticate the student first and upon successful authentication, the requested information will be provided. In the event of the above steps of communicating to the School database, the user will be able to see the series of events that will happen in the back end.

* **Implementation Document**

The authentication and a successful connection to the school’s database will need certain parameters to be passed by the users. The implementation document will allow developers to build the system that communicates to the school database and it will list all the possible parameters that are needed to pass during the application development. The document will contain the list of the school database server details, the database type, database name, and the authentication types to use.

* **Design**

The design section will cover the user-initiated database connection strategy. As illustrated in the figure-4, this component will need the authentication decision to begin for the user to interact with the system. Upon user’s passing in the default parameter, the authentication step will take care of the database access. Three consecutive unsuccessful tries will automatically disconnect the user for the further tries and the user will be asked to retry after some time. Upon a successful authentication, the user will be allowed to request the desired information of interest only as per the defined AVA system protocols. The user will be allowed to access a certain part of the database based on the role that was authenticated. Once the AVA system designs the query, it will retrieve the user requested information so the user can be informed of the resulting outcome retrieved from the database. Once the user ends the call session, the database connection will be dropped automatically.

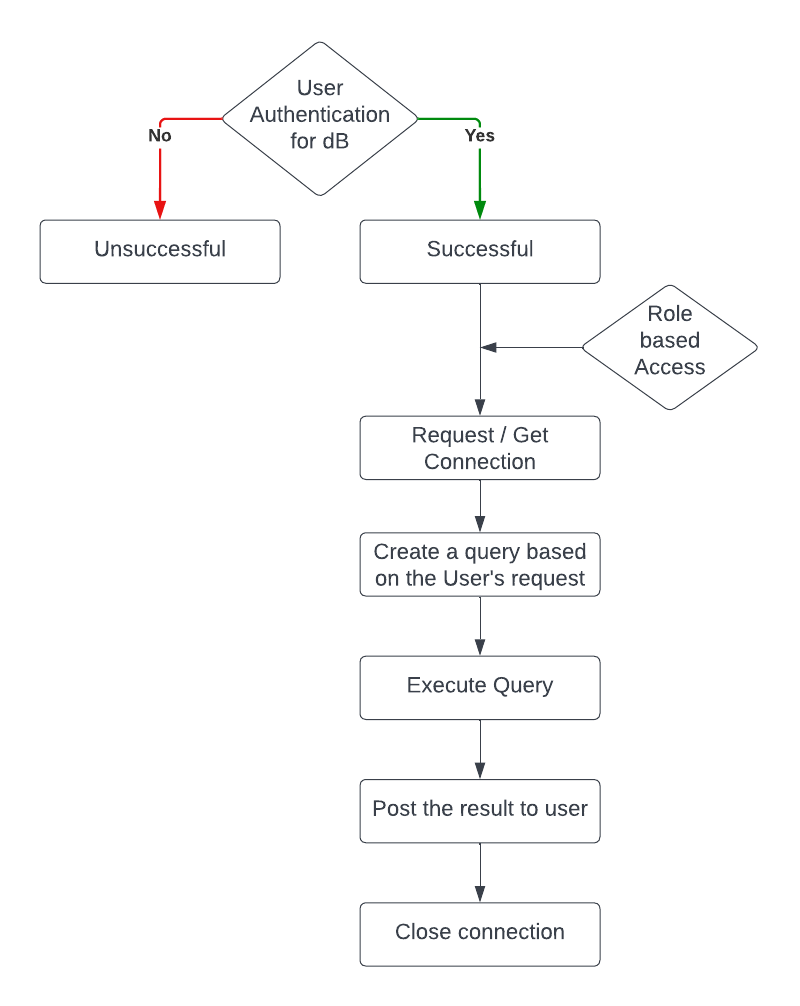


Figure 4: Database connection strategy

* **Exports**

Presence of Database and its connection make the entire system Live and so it can retrieve data from the database. This component exports information into Course Catalog Viewer that displays available courses from the catalog, Unofficial Transcript Generator which generates an unofficial transcript and GPA calculator that calculates the GPA. This hierarchy system is interdependent.

* **Imports**

When a user calls, this component imports the data from the Language recognition component to generate a successful database connection. The database imports data from the user's choice of course selection and class schedule for the semester.

* **Input/Output**

The Input for the component will be the user request transformed query. This query will help users to get the information from the school database.

The Output will be an action and or the information that the user requested. The result obtained by running the query will be the output of the component.

* **Pre and Post Conditions**

Preconditions for the successful database connection will be listed in the implementation document. The precondition for this component will be that the user is required to provide certain parameters to fully authenticate into the database and perform certain activities.

* **Error Handling**

After a user's successful authentication, if the data is found missing or has a glitch, the system will report the information using the logger, and such activities will be monitored to maintain the application integrity.

* **Side Effects**

The Implementation document containing the school’s server address and database information will carry a risk for the school against unforeseen hacking and cyber-attacks.

#### **Language Recognition:**

* **Abstract**

The automated language recognition component of the system is the main component and the most important. This function allows the user to interact with all the other functions within the system. Once the user is logged on, they are then able to vocally ask the system to perform certain functions. These functions include: course advising, view a catalog of courses available, view an unofficial transcript, calculate the user's GPA, or simply ask questions on any other areas of concern that the system can handle. The system will handle these requests and respond accordingly. When a user asks for course advising, they may also filter recommendations, but a filter is not mandatory. This filter allows a student to specify courses they are interested in/prefer, academic status of part time or full time, and whether they want the classes to be in the morning or afternoon. If the user wants to utilize any of the other functions, they can just specify the name of it, and the system will take them there. The other functions that this component of the system can connect to, all have functions of their own, which will be discussed further. Lastly, a user may ask any questions regarding courses and academic information and the system will respond accordingly.

* **Implementation Document**

Users of the system should speak in a clear voice and ensure that questions are not vague. Users should also read the user guide prior to using the system to ensure complete understanding. Utilize functions listed as named, otherwise the system may not recognize what is being asked.

* **Design**

The language recognition advising component of the system is basically the overhead. This component interacts and connects with all the other functions across the platform. The user should be able to log in and directly interact with the advising system. The user can ask questions and utilize built in modules by stating the functions title. The first is the unofficial transcript, the user should be able to ask the system to use the “unofficial transcript” function. The overhead language recognition will pass the request on and the system will instantiate the function to produce an unofficial transcript for the user to view. If the user says to go back, or states another function name, the system should act accordingly. The second function connected to the language recognition overhead is the “course catalog”. The student may state that they want to “view the full course catalog”, this will be recognized, sent, and received by the course catalog function and will produce a full list of all available courses for the upcoming semester. The third function is the GPA calculator, a user can state that they want to “calculate their GPA”. This will produce the user's current GPA and a “what-if” GPA on the following semester. Lastly, the user can be advised on what classes to take, and also filter it to their preferences on day time, academic status as in full time or part time, and if they have class preferences they can also specify that too.

* **Exports**

This component of the system makes language recognition available to all other components of the system.

* **Imports**

The course catalog is used when advising students on what to take next semester. Since it is a list of all classes that are available, it will choose the classes from there. The students unofficial transcript is also utilized in order to track progression and see what courses were already taken and completed.

* **Input/Output**

1. Inputs would include: login credentials, vocal commands
2. Outputs: outputs of functions being utilized, class schedule, answered questions.

* **Subparts**

GPA calculator, course catalog viewer, unofficial transcript generator

* **Pre and Post Conditions**

A user must have a valid log in, a user must be able to speak clearly and concise for the system to understand.

* **Error Handling**

If the system does not recognize what was said, it will produce an error message stating that it did not understand and to ask again. Ways to test this would be by speaking at different volume levels, different languages, and different types of phrases.

#### **GPA Calculator:**

* **Abstract**

The GPA calculator will consist of simply two modules: calculateGPA and whatIfGPA. The calculateGPA module will calculate the GPA of all of a student’s completed courses with a valid letter grade listed. The whatIfGPA will calculate a student’s GPA with the addition of courses not yet completed that can be added in at the student’s discretion with a manually inputted letter grade.

* **Design**

The design of the Grade-Point Average calculator will grab all courses a student has taken and completed, then calculate a GPA based on those letter grades. As job applications often ask for a student’s GPA, this is an essential tool to include

* **Exports**

The GPA calculator will export a student’s calculated GPA based on all of their completed courses.

* **Imports**

The GPA calculator will import a list of a student’s completed courses with letter grades, to be converted into a standard 4-point GPA scale. There will additionally be optional user input if the student would like to test out not yet finished classes to determine potential GPA shifts.

* **Input/Output**

Inputs will be taken from the system’s database while also providing functionality for the user to input potential “What-If” grades to see shifts in GPA. Outputs will be returned to the user in the form of a decimal number between the values of 0.00 and 4.00

* **Subparts**

The two subparts or modules of the GPA calculator include calculateGPA which will determine the student’s GPA based on completed courses with a letter grade between A and F. The other module, whatIfGPA must be manually called on and will allow a student to calculate a GPA with courses that are not yet completed that the student is enrolled in.

* **Pre and Post Conditions**

As a precondition, students must have a student ID and have completed at least one course. There are no postconditions in order to complete this task.

* **Error Handling**

Some courses may contain letter grades outside of the typical A to F range, for example if a student fails to complete a course, they may have the letter “I” instead. As a precaution, anything outside of the A to F range will not be included in GPA calculation.

* **Side Effects**

The basic GPA number will be available at all times.

* **Miscellaneous**

The calculateGPA module will always be immediately run upon the student opening the page. “What-If” GPAs must manually be selected and calculated.

#### 

#### **Course Catalog Viewer:**

* **Abstract**

For students to make an educated decision they require a clear understanding of the available and required courses they will need to take to graduate. The course catalog will include one module that will help students choose the most optimal route to graduate. The module will allow students to view all available courses in the Computer Science/ Information Technology department. It will also provide perimeters for the students to filter courses by time, name, and suitability.

* **Design**

The course catalog will use sql queries to filter the course list and display them appropriately. Using css and bootstrap the course catalog will have an easy to use interface.

* **Exports**

The module will export a list of courses that are available.

* **Imports**

The module will import the information of the student from the unofficial transcript generator module to be able to filter out the courses the student already took.

* **Input/Output**

The main input for the module will be the student asking the system to view the course catalog. The secondary input are the parameters the student defines to filter the courses. The output will be all available courses after filtering out the students previously taken courses and the parameters the student set in terms of time, date, or name.

* **Subparts**

The sub-modules will consist of the different perimeters for the students to filter courses by. Included in the perimeters are the name, time, and suitability.

* **Pre and Post Conditions**

As a precondition students must be logged in. This module has no postcondition.

#### **Unofficial Transcript Generator:**

* **Abstract**

The Unofficial Transcript Generator will be located on the students dashboard/side of the site and will be a place for students to access all the courses they have completed in list form. Along with all the courses that the student has taken in this list it will show when they took the course, and what grade they got on the course. Each student should have access to only their own unofficial transcript student.

* **Design**

The design for the unofficial transcript will be on the backend with each course that the student has taken attached to the student. On the front-end it will be built with css, html, and potentially bootstrap

* **Exports**

The generator will display a list of courses organized by semester taken along with grades associated with each course.

* **Imports**

The generator will take all the courses the student previously took along with when they took the course and the grade they achieved in the respective course.

* **Input/Output**

1. Input: students login credentials

2. Output: the students unofficial transcript, they had requested

* **Subparts**

There will be filters in the generator that will allow the student after they have the unofficial transcript on the screen to filter it by when they took the course, or on what grade the achieved in the course.

* **Pre and Post Conditions**

Precondition: The student must be logged in for them to access the unofficial transcript.

Postcondition: none

* **Error Handling**

If the student has not finished the course that they are currently enrolled in it will not appear on their transcript, since the course has not been finished and no grades have been given.

* **Side Effects**

The user should be able to access this whenever they want their unofficial transcript.

* **Miscellaneous**

Once the student hits the button to request for the unofficial transcript it should immediately display their transcript.

### 

### **Pseudocode**

### **Login**

User accesses the domain of the voice activated advising system

If username == username stored in database && password == password store in database:

Login and displays homepage

Else:

System displays error message: “user credentials are incorrect please try again”

**View course Catalog**

This function pulls up the course catalog through the usage of a voice command.

The user chooses to filter by a department and inputs a department code

for (every course matching that code)

print all applicable courses

The user chooses to filter within a course number range and inputs a minimum and maximum course number

for (every course within that range)

print all applicable courses

The user chooses to filter by a time of day and inputs that time

for (every course matching that time)

print all applicable courses

The user chooses to filter by a day and inputs a day

for (every course matching that code)

print all applicable courses

**Unofficial Transcript generator function**

This function creates an unofficial transcript for a student

The user requests an unofficial transcript through the usage of a voice command

for (every course applicable to the student)

print course information including letter grade, credit hours, course name, and course number

return the result to the advisement tool

**Advisement on classes and questions**

This function will take the recognizer, microphone, and transcript as its arguments

This will allow the advisement component to understand what the student is asking

Print: “do you want advisement on classes or do you have questions”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “question”

While (True)

Print “What type of questions do you have?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Process questions and return answer

Print “Do you have another question?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “Yes”

Loop until questions are answered

Else if Speech == “No”

Exit back to beginning of advisement

Else Display error message

Else if Speech == “classes”

Check classes already taken from transcript info

Print “Do you want to apply a filter?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “Yes”

Print “do you want to be considered full time or part time?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “full time”

Fulltime = True

Parttime = False

Print “Do you want morning, afternoon or evening classes?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “morning”

Morning = True

Afternoon = False

Evening = False

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “afternoon”

Morning = False

Afternoon = True

Evening = False

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “evening”

Morning = False

Afternoon = False

Evening = True

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “part time”

Parttime = True

Fulltime = False

Print “Do you want morning, afternoon or evening classes?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “morning”

Morning = True

Afternoon = False

Evening = False

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “afternoon”

Morning = False

Afternoon = True

Evening = False

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “evening”

Morning = False

Afternoon = False

Evening = True

Print “Do you have any class preferences?”

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

If Speech == “yes”

While (1)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

Store classes spoken in an array, if the word stop

occurs exit loop

Display courses that should be taken

Else if Speech == “No”

Display courses that should be taken

Else if Speech == “No”

Check transcript details of what classes were taken

Display classes that should be taken next semester

Else Display error message

Else display error message

**Language processing**

System utilizes python's “pyAudio” library for speech recognition.

Import speech\_recognition as sprec (sprec stands for speech recognition)

Create a recognizer instance: recognize = sprec.Recognizer

Create an instance for the microphone: microphone = sprec.Microphone(microphone on the hardware)

With microphone as source:

recognize.adjust\_for\_ambient\_noise(source)

Speech = recognize.listen(source)

rec\_Sentence = recognize.recognize\_google(Speech)

Print (rec\_Sentence)

This will output a string for what was interpreted from the microphone and recognized by the google recognizer API.

If rec\_Sentence == “array of words that a user may say for “view course catalog”

view\_Course\_Catalog(recognize, microphone)

Else if rec\_Sentence == “array of words that a user may say for “view unofficial transcript”

unofficial\_Transcript(recognize, microphone)

Else if rec\_Sentence == “array of words that a user may say for “generate gpa”

GPA(recognize, microphone)

Else if rec\_Sentence == “array of words that a user may say for “academic advisement”

academic\_advisement(recognize, microphone)

**GPA calculator**

This function calculates the user’s GPA through the usage of a voice command

for every course in student’s transcript

if letterGrade = (grade A - F)

increment coursecount

add (call LGN with letterGrade) to gpa

divide gpa by coursecount

return new gpa

**What-If GPA**

This function calculates the user’s what-if GPA through the usage of a voice command

for every course in student’s transcript

if letterGrade = (grade A - F)

increment coursecount

add (call LGN with letterGrade) to gpa

for every course the student is enrolled in currently

name a letter grade

convert letter grade to 4.0 scale

increment coursecount

add (call LGN with letterGrade) to gpa

divide gpa by coursecount

return new what-if gpa

**Letter Grade to Number (LGN)**

This is a helper function to convert letter grades to values

double LGN (char num)

switch :

A case for each distinction from A to F including pluses and minuses

return num

**Session**

This function keeps track of a user’s activity

User enters a page

for timer = 0, timer <= 15 minutes

if user does action

timer = 0

if user tells advisement tool to log out

log user out

log user out

### **Detailed Design**

Within this document is an encapsulation of the entire design of the automated voice advisement system also known as AVA. With a thorough understanding of how each module works in order to achieve the goal specified by the stakeholders. The system will have the ability to listen to a user's vocal commands, and produce the expected result the user is requesting. After a user signs in with official credentials, the system will have the capabilities to produce a course catalog, display an unofficial transcript for the user, calculate one's GPA for the current and following semester, and allow the user to ask questions and be advised on what courses to take for the semester. All undergraduate students apart of the CS/IT department will have access to use this system to their advantage in achieving academic success.

Users that have access to the system would be all undergraduate students apart of the computer science and information technology department in the school. These students will need a valid ID and password that the school gave them in order to sign in. With valid credentials the user can then log in and will be brought to a home page where they can choose from different functions that the system supports. These functions are: academic advisement, view course catalog, view unofficial transcript, and calculate GPA. The user can then vocally select which function they would like to utilize. Once a student vocally announces which function they would like to use, the system interprets the language, and activates the function. If any problems were to occur with the language interpretation, the system will output an error message such as “Sorry speech not recognized, please try again”.

If the user vocalizes that they would like to partake in academic advisement with the system, the system will activate the academic advisement function and switch to a new interface where it will display what was vocalized by the user, and the output (answer) from the system after interpreting what was asked. Once the interface switches, the user will be asked by the system if they have questions, or would like academic assistance in course scheduling. The user can then reply, if “questions” is stated, then the system will activate a listener which will interpret any questions the user may ask. If the system is capable of answering the question, it will produce an output, and will ask if the user would like to continue asking questions. If the student says “yes” then the function will loop, and continue answering questions the student asks. If a question is unable to be answered, then it will display a message stating “Sorry that question could not be answered”. Once the user is satisfied with asking questions, then they can state that they are finished by saying “stop”. This will cause the function to exit the question interface and loop back to asking the user if they would like to ask a question or receive academic assistance in course scheduling. If the user states that they want help with “course scheduling” then the function again will switch to a text interface where it will then ask if the user would like to apply filters. If the student replied with “yes” then the filter application process will start, first asking if they want to be “full time or part time” for the semester. The user can then state their answer by vocalizing “full time” or “part time”. The system will then ask if they want “morning, afternoon, or evening classes”, again the student can vocalize their choice by saying “morning”, “afternoon”, or “evening”. Lastly, the module will ask if the student has any “class preferences”. Class preferences means the student has the ability to list the classes they want to prioritize first if they have any in mind. The system will then store these preferences into an array if there are any. And will use this information to try and prioritize the listed classes before other classes, as long as they benefit the students progress. If the student has no preference then they can state “no preference”. The function will then retrieve information from the unofficial transcript, the course catalog function, and use all filter information to then create a course schedule for the user to view and take into consideration of registering for the classes listed. If the user chooses not to use a filter, they can simply say “no filter” when asked, and then the function will retrieve information from the transcript module, course catalog module, and produce a course schedule that they again can take into consideration. Afterwards the module will then return to the homepage where the user can then choose another function or log out.

From the homepage, the second function is the ability to view a full course catalog. The user will be able to vocally state “view course catalog” the system will interpret the command and activate the course catalog function. This function will pull all data from the schools database for all classes that are available for computer science and information technology undergraduate students. It will then create a list of all classes that will be available and display all the courses for the user to look through. This function will also transfer all the data to the advisement module, so the advisement module has a list of all classes that are available to be taken. Afterwards the module will then return to the homepage where the user can then choose another function or log out.

If the user vocalizes “transcript” the system will instantiate the unofficial transcript function. This function will then pull all the information from the schools database regarding the logged in user. This includes all the classes taken in the past, all classes registered for currently, and all grades that were received from those classes that were taken. The function will then produce an unofficial transcript in pdf form for the user to view. Afterwards it will then direct the user back to the homepage.

If the user vocalizes that they want to calculate their GPA, they can state “calculate GPA”. The system will then switch interfaces to the text based interaction, and will calculate the GPA from all classes taken in the past and output the cumulative GPA the student has. The student can then ask for the what-if GPA, by stating “what-if”. The system will then calculate what the student needs for grades in the following semester in order to keep or raise their GPA. All instances will be displayed so the student can view all possibilities for whatever grades they may receive in the following semester.