Oregon State University
College of Engineering

CS Capstone Design Document

December 3, 2017

Kora

Prepared for

Autodesk

Patti Vrobel

Prepared by

Group8

James Stallkamp
Jeremy Fischer
Austin Row

Abstract

This Document describes the design views and components of Kora. This document begins by introducing the purpose, scope, and glossary, followed by a summary of the overall project. Then follows a list of stakeholders and their concerns. The rest of this document is divided into sections where each provides a different design viewpoint.

CONTENTS

1 Document Details

1.1 Date of Issue and Status

This document was issued December 1, 2017 and is the first complete iteration of the design document.

1.2 Authorship

Jeremy Fischer, Austin Row, and James Stallkamp are the authors of this document and the developers of Kora.

1.3 Change History

TABLE 1: Change History

Date	Change Description
November 30, 2017	First design document draft

2 Introduction

2.1 Purpose

The purpose of this design document is to outline how Kora's functionality will be achieved. More generally, this document describes how the client's requirements will be met. Kora's developers will use this document as a roadmap during implementation.

2.2 Scope

This document focuses on the relationships between Kora's components and their individual processes, and how they work together to satisfy the project's requirements.

2.3 Summary

Kora will be a speech-based virtual assistant for Fusion that lets users perform any one of a subset of tasks within the product, such as saving a document or opening a menu, by verbally instructing it to perform the task. Workflows in Fusion that are not suited for handling by a voice interface will not be supported by Kora. As a stretch goal, Kora will be capable of questioning the user and using responses to predict and automatically assist with future user behavior. This functionality will be implemented as a plugin that is bundled with Fusion and will be part of the product's standard download.

Kora will offer users a tool that decreases the time required to achieve their goals within Fusion by offering an interface that runs in parallel with and complements the keyboard and mouse. If the stretch goal is achieved, Kora will further increase productivity by learning to predict and automate specific workflows within the product.

3 GLOSSARY

TABLE 2: Glossary

Term	Definition
Kora	The virtual assistant that is the focus of this project
NLP	Natural Language Processing
API	Application Programing Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
UI	User Interface
Fusion	An Autodesk Cloud-based 3D CAD/CAM tool/product
Task	In the context of Fusion, a function or operation that can be performed in Fusion
Plugin	Software that adds specific new functionality to another piece of software
User	A person that interacts with Kora or Fusion depending on the context
Workflow	A sequence of related tasks

4 DESIGN STAKEHOLDERS AND CONCERNS

4.1 System Decomposition

Stakeholder(s): Developers

Concern: How is the software system decomposed into components?

4.2 Component Functional Responsibilities

Stakeholder(s): Developers

Concern: What functionality is each software component responsible for?

4.3 User-Application Interface

Stakeholder(s): Client, Developers

Concern: What interfaces will exist for users to interact with the application and for the application to communicate with users?

4.4 Internal Interfaces

Stakeholder(s): Developers

Concern: What internal interfaces will exist between software components?

4.5 Component Interactions

Stakeholder(s): Developers

Concern: Which components of the software system will interact?

4.6 Component Interaction Responsibilities

Stakeholder(s): Developers

Concern: What are the responsibilities of each component of the software system in the context of interactions?

4.7 Architecture

Stakeholder(s): Developers

Concern: What architecture defines the software system?

4.8 System States

Stakeholder(s): Developers

Concern: What are the possible states the software system can be in?

4.9 State Activities

Stakeholder(s): Developers

Concern: What defines each of the states of the system?

4.10 State Transitions

Stakeholder(s): Developers

Concern: When does the system transition from one state to another?

4.11 Log File Location

Stakeholder(s): Developers

Concern: Where will data regarding user interactions be logged?

4.12 Log File Content

Stakeholder(s): Developers

Concern: What will be stored in log files?

4.13 Log File Access

Stakeholder(s): Developers

Concern: How will stored log files be accessed?

5 Design Viewpoint: Composition

5.1 Addressed Design Concerns

- System Decomposition (defined in section 4.1)
- Component Functional Responsibilities (defined in section 4.2)

5.2 Design Elements

5.2.1 Master Module

Type: Module

Purpose: Facilitates communication between modules and manages application state.

Author: James Stallkamp

5.2.2 UI Module

Type: Module

Purpose: Provides user a way to interact with Kora and expresses the application state to the user.

Author: James Stallkamp

5.2.3 Speech-to-Intent Module

Type: Module

Purpose: Anaylzes speech input and produces a intent object in JSON format.

Author: James Stallkamp

5.2.4 Logger Module

Type: Module

Purpose: Stores runtime and contextual information to be used for training Kora.

Author: James Stallkamp

5.2.5 Fusion Module

Type: Module

Purpose: Translates intent into Fusion API commands and executes them.

Author: James Stallkamp

5.2.6 Text-to-Speech Module

Type: Module

Purpose: Synthesizes an audio output from a given text input.

Author: James Stallkamp

5.2.7 Action Prediction Module

Type: Module

Purpose: Trains Kora to become a more powerful assistant.

Author: James Stallkamp

5.2.8 Global Data Store

Type: Data Store

Purpose: Stores data that for that can be accessed by any module.

Author: James Stallkamp

5.3 Design View: Modules

Kora is composed of seven primary modules.

5.3.1 Master Module

The Master module is responsible for coordinating all other modules.

5.3.2 UI Module

The UI module is responsible for all interaction with the user. This module will collect input and communicate it to the Master module as well output regular feedback to the user.

5.3.3 Speech-to-Intent Module

The Speech-to-Intent module will take in audio input and output a JSON object containing information on the spoken input. This intent module will construct the JSON object and return it to master.

5.3.4 Logger Module

The Logger Module will create a persist-able data object that will be loaded with run time information from Kora. These data objects will be analyzed and used to help train Kora for future development.

5.3.5 Fusion Module

The next module is the Fusion module, this module will handle executing Fusion commands. The Fusion module will parse information from the JSON intent object to construct and execute a Fusion command.

5.3.6 Text-to-Speech Module

In order for Kora to speak to the user it will need a voice synthesizer module. This module will receive text input and produce an audio output that can be played to the user.

5.3.7 Action Prediction Module

The Action Prediction module is responsible for training Kora to recognize patterns and improve functionality.

5.3.8 Global Data Store

The Global Data Store is responsible for storing data that will be needed in multiple parts of the application. The data that is stored there only persists until the application is closed or the data is explicitly deleted.

6 Design Viewpoint: Information

6.1 Addressed Design Concerns

- Log File Location (defined in section 4.11)
- Log File Content (defined in section 4.12)
- Log File Access (defined in section 4.13)

6.2 Design Elements

6.2.1 MongoDB

Type: Database
Purpose: Database to hold the log files
Author: Jeremy Fischer

6.2.2 JSON object

Type: Storage structure
Purpose: The structure the information will be stored in
Author: Jeremy Fischer

6.2.3 HTTP calls

Type: Access mechanism

Purpose: How the data will be posted and accessed

Author: Jeremy Fischer

6.3 Design View: Contents

The contents of the data will be:

- the intent generated by the Speech-to-Intent module
- the context generated by the Speech-to-Intent module such as quantities
- whether the user's request was successfully processed
- the posting date
- the posting time
- the user identification

6.4 Design View: Access

The data will be posted and accessed via HTTP calls to the database.

6.5 Design View: Structure

The data will be stored in a MongoDB database. Each entry in the database will resemble the JSON object that is returned from the Speech-to-Intent module driver method.

7 Design Viewpoint: Patterns

7.1 Addressed Design Concerns

- Architecture (defined in section 4.7)

7.2 Design Elements

7.2.1 Mediator Design Framework

Type: System Architecture

Purpose: Kora has a simple mediator that coordinates all interactions between all other components.

Author: James Stallkamp

7.3 Design View: Software Architecture

Kora is structured according to a mediator design pattern. Kora will have a Master module that coordinates interactions between itself and all other modules. The Master module controls Kora's flow and ensures that the correct data gets to the correct module.

8 Design Viewpoint: Interfaces

8.1 Addressed Design Concerns

- User-Application Interface (defined in section 4.3)
- Internal Interfaces (defined in section 4.4)

8.2 Design Elements

8.2.1 UI Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the UI module.

Author: Austin Row

8.2.2 Speech-to-Intent Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Speech-to-Intent module.

Author: Austin Row

8.2.3 Logging Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Logging module.

Author: Austin Row

8.2.4 Text-to-Speech Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Text-to-Speech module.

Author: Austin Row

8.2.5 Fusion Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Fusion module.

Author: Austin Row

8.2.6 Action Prediction Module Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Action Prediction module.

Author: Austin Row

8.2.7 Global Data Store Interface

Type: Internal Interface

Purpose: Defines rules governing interactions with the Global Data Store.

Author: Austin Row

8.2.8 User Interface

Type: External Interface

Purpose: Defines rules governing how the human user can interact with Kora.

Author: Austin Row

8.3 Design View: Module Interfaces

The interfaces to each module are driven by a single method that takes a JSON object as an argument and returns a JSON object back to the caller. Both the JSON object passed to the function and that which is returned contain the same two keys with values that are specific to each module and to whether it is an argument to the driver method or is a return value. One of the required keys is ”dataIDs” which maps to an object containing key-value pairs where the key is a name specific to the target module and the value is an ID used by the global data store to identify some stored data entity. The second required key is ”values” which maps to an object containing key-value pairs where each key is a name specific to the target module and the value is some piece of data that is to be used in the target module.

For JSON that is passed as an argument to the driver method, the data IDs map to data that is stored in the global data store that is needed by the module to perform its specific functionality. Also for this JSON, the ”values” key-value pairs are parameters that the target module uses in to accomplish its specific responsibilities. For returned JSON, the data IDs map to data that was stored in the global data store by the target module during the execution of its responsibilities. Also for returned JSON, the ”values” key-value pairs are return values that are used by modules later in the application logic flow. The following interface definitions describe which key-value pairs are defined for each module.

The INPUT section defines the expected format of the JSON that is passed to the module’s driver method. The OUTPUT section defines the expected format of the JSON that is returned from the module’s driver method.

8.3.1 UI Module Interface

INPUT: {
    dataIDs: {
        voiceResponseAudioFile: <dataStoreID or empty>
    }
    values: {
        UIActionCode: <number encoding what should be communicated to user through the UI>
    }
}
OUTPUT: {
    dataIDs: {
        userSpeechAudioFile: <dataStoreID or empty>
    }
    values: {
        errorOccurred: <0 if no errors else 1>
    }
}

8.3.2 Speech-to-Intent Module Interface

INPUT: {
dataIDs: {
userSpeechAudioFile: <dataStoreID>
}
values: { }
}
OUTPUT: {
dataIDs: {
}
values: {
errorOccurred: <0 if no errors else 1>
userCommandIntent: <intent JSON from Wit.ai API>
}
}

8.3.3 Logging Module Interface

INPUT: {
dataIDs: { }
values: {
logMessage: <message>
messageType: <error or warning or info>
}
}
OUTPUT: {
dataIDs: { }
values: {
errorOccurred: <0 if no errors else 1>
}
}

8.3.4 Text-to-Speech Module Interface

INPUT: {
dataIDs: { }
values: {
text: <text to be transcribed to speech>
}
}
OUTPUT: {
dataIDs: {
speechAudioFile: <dataStoreID>
}
values: {
errorOccurred: <0 if no errors else 1>
}
}

8.3.5 Fusion Module Interface

INPUT: {
dataIDs: { }
values: {
userCommandIntent: <intent JSON from Wit.ai API>
}
}
OUTPUT: {
dataIDs: {
}
values: {
errorOccurred: <0 if no errors else 1>
}
}

8.3.6 Action Prediction Module Interface

INPUT: {
dataIDs: {
userSpeechAudioFile: <dataStoreID>
}
values: {
get: <0 if not trying to get prediction for next user command else 1>
userCommandIntent: <intent JSON from Wit.ai API>
}
}
OUTPUT: {
dataIDs: { }
values: {
errorOccurred: <0 if no errors else 1>
prediction: <empty if get=0 in input else user prediction in same format as intent data from Wit.ai API>
}
}

8.4 Design View: Data Store Interface

The global data store has the following methods:

Name:	Get
Parameters:	dataID: The unique identifier for the data being retrieved.
Return Value:	The data associated with dataID.
Description:	Returns the data associated with the dataID parameter in the data store.

Name:	Store
Parameters:	data: any kind of data
Return Value:	The ID generated for and associated with data.
Description:	Generates a unique ID and stores data associated with the ID then returns the ID.

Name:	Remove
Parameters:	dataID: The unique identifier for the data being removed.
Return Value:	0 if data was not found, 1 if data was found and removed, 2 if data was found but not successfully removed.
Description:	Removes the data associated with the dataID parameter in the data store.

8.5 Design View: User Interface

The user interface can be divided into two interaction types: the user giving Kora a command and Kora replying to the user with the application status. There are two use patterns for a user when they give Kora a command. The first way

to give a command is to say the wake word followed by the command. When the user says the wake word, it signals to Kora that it should be actively listening for a command. The other method is to click the wake button that acts as an alternative to the wake word then say the command.

Through the user interface, the application responds back to the user with the status of the previous command. If the command succeeds, the application uses a voice synthesizer to alert the user that the previous command succeeded. If the command fails, the application uses the voice synthesizer to alert the user that the previous command failed.

9 Design Viewpoint: Interactions

9.1 Addressed Design Concerns

- Component Interactions (defined in section 4.5)
- Component Interaction Responsibilities (defined in section 4.6)

9.2 Design Elements

9.2.1 Master Module

See section 5.2.1 for element definition.

9.2.2 UI Module

See section 5.2.2 for element definition.

9.2.3 Speech-to-Intent Module

See section 5.2.3 for element definition.

9.2.4 Logger Module

See section 5.2.4 for element definition.

9.2.5 Fusion Module

See section 5.2.5 for element definition.

9.2.6 Text-to-Speech Module

See section 5.2.6 for element definition.

9.2.7 Action Prediction Module

See section 5.2.7 for element definition.

9.3 Design View: Module Interactions

The following diagram specifies the interactions that occur between modules within the application:

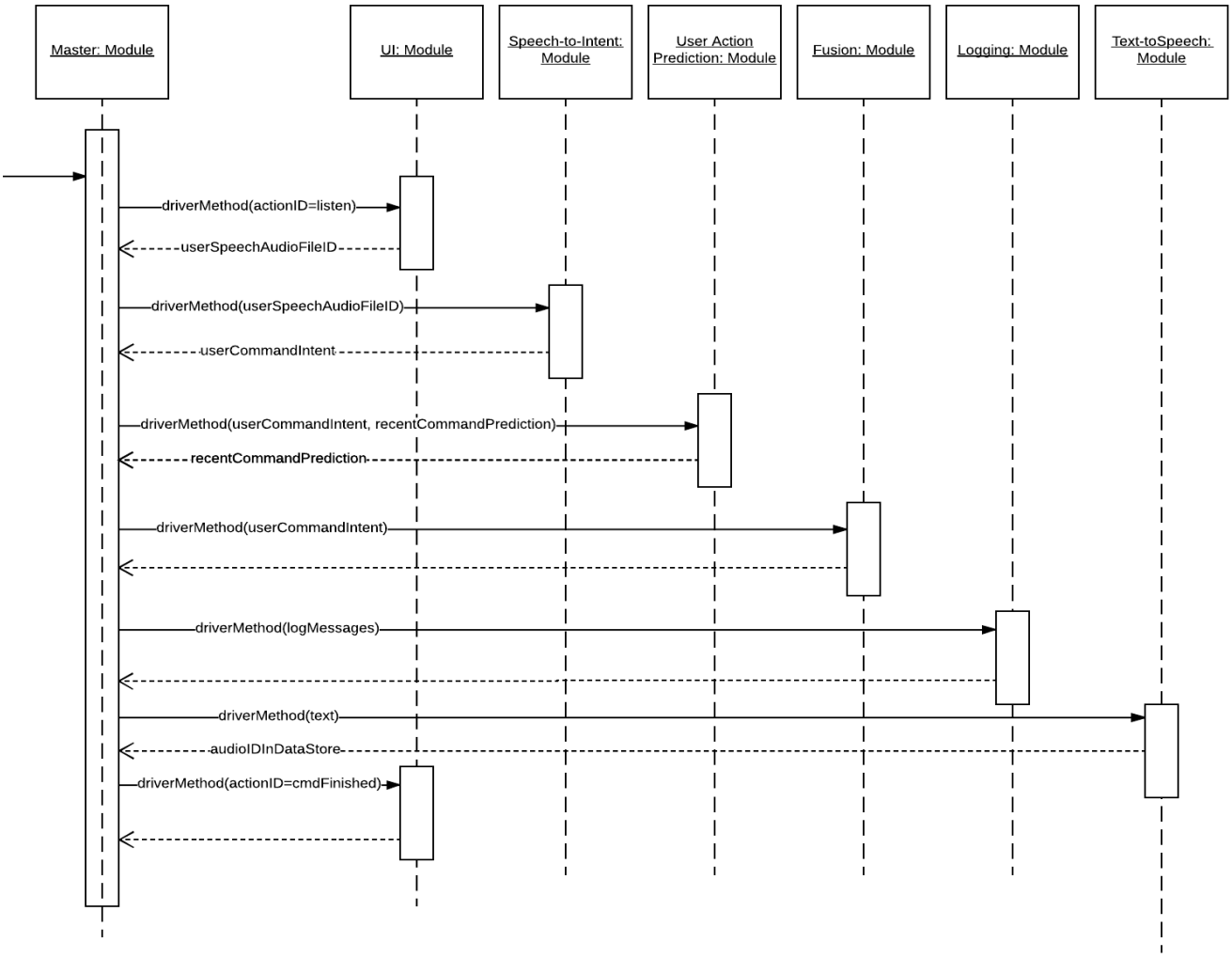

Fig. 1: The interactions that occur between the system's software modules.

10 Design Viewpoint: State Dynamics

10.1 Addressed Design Concerns

- System States (defined in section 4.8)
- State Activities (defined in section 4.9)
- State Transitions (defined in section 4.10)

10.2 Design Elements

10.2.1 Idle Listen

Type: State

Purpose: Kora is waiting for a wake action to take place

Author: Jeremy Fischer

10.2.2 Active Listen

Type: State

Purpose: Kora begins listening to the user

Author: Jeremy Fischer

10.2.3 Process Speech

Type: State

Purpose: Kora is deriving intent from the user's verbal request

Author: Jeremy Fischer

10.2.4 Endpoint Mapping

Type: State

Purpose: Kora is discerning which API endpoint to call

Author: Jeremy Fischer

10.2.5 Fusion Action

Type: State

Purpose: Kora makes the Fusion API call

Author: Jeremy Fischer

10.2.6 UI Feedback

Type: State

Purpose: Kora is verbalizing the state of the latest request to the user

Author: Jeremy Fischer

10.2.7 Error

Type: State

Purpose: Kora is in a state of error

Author: Jeremy Fischer

10.3 Design View: States

10.3.1 State Diagram

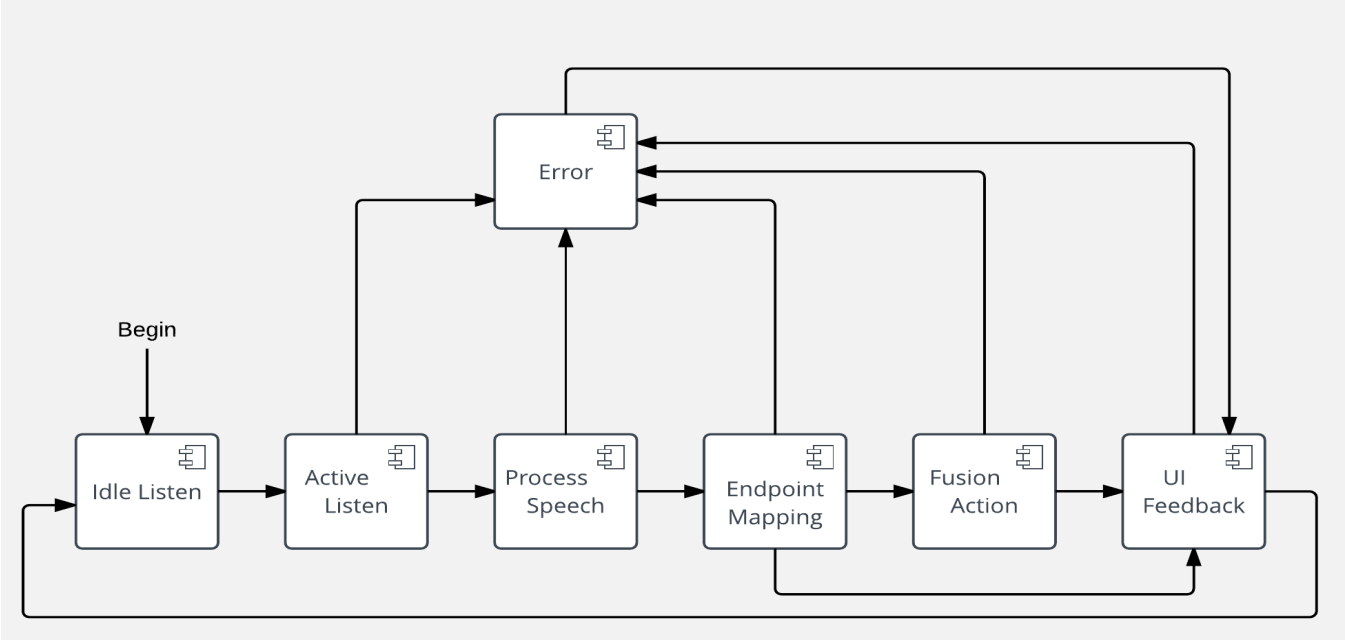

Fig. 2: The possible states Kora can be in, as well as the states the system can transfer to from within a given state.

10.4 Design View: State Activity

10.4.1 Idle Listen

The Speech-to-Intent module is being streamed audio, but does not do anything with it until it hears the wake word or is signaled to via a wake action such as a button press. In this state the system is simply awaiting for the user to signal it to begin listening.

10.4.2 Active Listen

The system tells the Speech-to-Intent module that it should treat the incoming audio as a request.

10.4.3 Process Speech

The Speech-to-Intent module is being streamed audio and is attempting to gather intent and context from it.

10.4.4 Endpoint Mapping

Kora is attempting to discern the correct Fusion API endpoint based off of the intent and context variables in the JSON received from the Speech-to-Intent module.

10.4.5 Fusion Action

Kora is making the call to the Fusion API and then waiting for the API to return an error code.

10.4.6 UI Feedback

Kora is signaling feedback regarding the latest transaction to the user.

10.4.7 Error

The system is in a state of failure and is resetting the application so Kora can attempt another request.

10.5 Design View: State Transitions

10.5.1 Idle Listen

- *Idle Listen → Active Listen*

  Kora moves from the Idle Listen state to the Active listen state when a wake action takes place.

10.5.2 Active Listen

- *Active Listen → Process Speech*

  Kora moves from the Active Listen state to the Process Speech state when the Speech-to-Intent module does not receive speech for a half second.

- *Active Listen → Error*

  Kora moves from the Active Listen state to the Error state when an internal failure occurs when Kora is in the Active Listen state.

10.5.3 Process Speech

- *Process Speech → Endpoint Mapping*

  Kora moves from the Process Speech state to the Endpoint Mapping state when the Speech-to-Intent module returns the JSON object holding the processed speech's intent and arguments.

- *Process Speech → Error*

  Kora moves from the Process Speech state to the Error state when an internal failure occurs when Kora is in the Process Speech state.

10.5.4 Endpoint Mapping

- *Endpoint Mapping → Fusion Action*

  Kora moves from the Endpoint Mapping state to the Fusion Action state when the Mapping module successfully maps the intent to a Fusion API command.

- *Endpoint Mapping → UI Feedback*

  Kora moves from the Endpoint Mapping state to the UI Feedback state when the Mapping module fails to map the intent to a Fusion API command.

- *Endpoint Mapping → Error*

  Kora moves from the Endpoint Mapping state to the Error state when an internal failure occurs when Kora is in the Process Speech state.

10.5.5 Fusion Action

- *Fusion Action → UI Feedback*

  Kora moves from the Fusion Action state to the UI Feedback state when the Fusion API returns the error code indicating whether the API call was successful or not.

- *Fusion Action → Error*

  Kora moves from the Fusion Action state to the Error state when an internal failure occurs when Kora is in the Fusion Action state.

10.5.6 UI Feedback

- *UI Feedback → Idle Listen*

  Kora moves from the UI Feedback state to the Idle Listen state after it indicates to the user the outcome of processing the request.

- *UI Feedback → Error*

  Kora moves from the UI Feedback state to the Error state when an internal failure occurs when Kora is in the UI Feedback state.

10.5.7 Error

- *Error → Idle Listen*

  Kora moves from the Error state to the UI Feedback state after it resets all internal variables to their initial state.

11 GANTT CHART

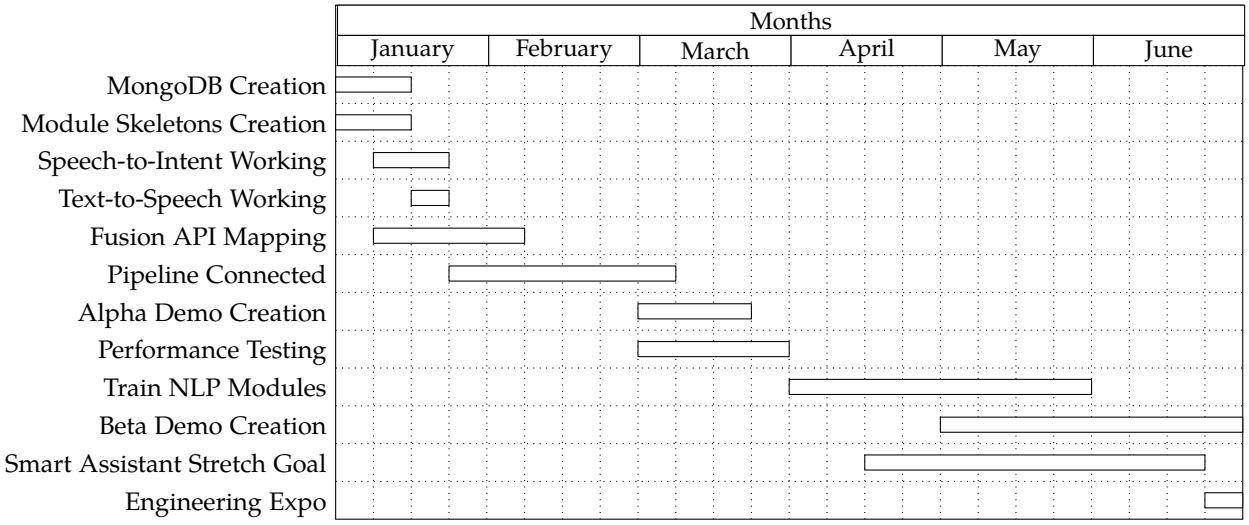

Fig. 3: Kora Development Schedule

12 CONCLUSION

This document has described Kora's development roadmap. This document explains how Kora's underlying system is decomposed into components and what each component is responsible for. Above is a Gantt chart that details the development schedule which the Kora developers will abide by. There is a change history table at the beginning of this document which will explain any modifications made to the design described in this document.