**Chorus**

Software Design Description

METU CENG350 2017-2018 Spring

|  |  |
| --- | --- |
| **Group 33** | |
| **Name** | **Student ID** |
| Furkan Doğan | 2098937 |
| Anxhela Ngjeliu | 2174753 |

**TABLE OF CONTENTS**

1. [Introduction 5](#_TOC_250016)
   1. [Purpose of the system 5](#_TOC_250015)
   2. [Scope 5](#_TOC_250014)
   3. [Stakeholders and their concerns 5](#_TOC_250013)
2. [References 6](#_TOC_250012)
3. [Glossary 7](#_TOC_250011)
4. [Architectural Views 7](#_TOC_250010)
   1. [Context view 7](#_TOC_250009)
   2. [Composition view 13](#_TOC_250008)
   3. [Information view 15](#_TOC_250007)
      1. [Service Interfaces 16](#_TOC_250006)
      2. [CRUD Operations 19](#_TOC_250005)
   4. [Interface view 22](#_TOC_250004)
      1. [Internal Interfaces 22](#_TOC_250003)
      2. [External Interfaces 23](#_TOC_250002)
         1. [User Interfaces 23](#_TOC_250001)
         2. [System Interfaces 25](#_TOC_250000)

**FIGURES**

Figure 1: Context Model 8

Figure 2: Use-case Diagram 9

Figure 3: Component Diagram 13

Figure 4: Deployment Diagram 14

Figure 5: Service Interfaces Class Diagram 16

Figure 6: Database Class Diagram 19

Figure 7: Vote Sequence Diagram 22

Figure 8: Chorus Bot Interface 23

Figure 9: Chorus Worker Interface 25

Figure 10: Send Message Sequence Diagram 26

Figure 11: Select Best Response Sequence Diagram 28

# TABLES

Table 1: Glossary 7

Table 2: Send Message Use-Case Table 10

Table 3: Select best response Use Case Table 11

Table 4: Use cases descriptions 12

Table 5: Service interfaces operation descriptions 17

Table 6: Service interfaces operation design 18

Table 7: CRUD Operations 21

# Introduction

# Purpose

# The purpose of this document is to describe the design of project Chorus by giving information about the content and the organization of it. So, it provides stakeholders information about the components of the project and how they are connected to make the system work.

# Scope

# On this document the system is described at an architectural level, explaining the components of the project and their relationships between each other which make them work together as a system. To be more specific, Context view, Composition view, Information view and Interface view are explained, and more details are given by specific diagrams such as use case, sequence and component diagrams.

# Stakeholders and their concerns

**End users** are ordinary people asking for assistance on different tasks. They have to be English speakers since they can only communicate in English with the crowd workers. End user’s concern is to take information about different tasks that they might be interested by chatting with the crowd workers.

**Crowd workers** are semi-anonymous people recruited for short periods of time from Amazon’s Mechanical Turk. They work online so they have to have some basic computer skills such that they will be able to search the web for tasks that they will be asked to give assistance for. Crowd workers’ concern is to take monetary rewards according to their contribution by giving proper answers or voting the answer that seems more right to be given to the end user.

**Administrators** can be divided into IT staff and researchers. Their concern is working to keep the maintenance of the system, keep the database up-to-date and also work on errors that the system can give.

# References

# This document is written with respect to the specifications of the document

# below:

# IEEE standard for information technology--systems design--software design

# descriptions. (2009). New York, NY: Institute of Electrical and Electronics Engineers.

# Other sources:

**Chorus: A Crowd-Powered Conversational Assistant**

W.S. Lasecki, R. Wesley, J. Nichols, A. Kulkarni, J.F. Allen, J.P. Bigham.

In Proceedings of the ACM Symposium on User Interface Software and Technology (**UIST 2013**). St Andrews, UK. P151-162.

**"Is there anything else I can help you with?": Challenges in Deploying an On-Demand Crowd-Powered Conversational Agent**

Ting-Hao K. Huang, Walter S. Lasecki, Amos Azaria, Jeffrey P. Bigham.

In Proceedings of Conference on Human Computation & Crowdsourcing (**HCOMP 2016**), 2016, Austin, TX, USA.

# Glossary

|  |  |
| --- | --- |
| **End user** | Ordinary people that ask Chorus for assistance. |
| **Crowd worker** | Paid users of Chorus who propose answers for the end user. |
| **Crowd-powered** | Systems that combine computation with human intelligence, coordinated from large group of people who work online. |
| **Crowdsourcing** | The process getting work usually online from a crowd of people. |
| **Retainer pool** | Finding crowd workers who are paid to do some task in real time. |
| **Database** | A structured set of data. |
| **Multi-tiered** | Any application developed and distributed among more than one layer. |
| **Mturk** | Amazon Mechanical Turk |
| **Chatbot** | A computer program that attempts to simulate the conversation or "chatter" of a human being via text or voice interactions. |
| **API** | Application Programming Interface |
| **COTS** | Commercial Off-The-Shelf |

Table 1: Glossary

# Architectural Views

# Context view

# In this view all users and systems interactions are explained by Context and Use Case diagrams. The Context diagram gives a general view of all users and external systems that interact with Chorus system, while Use Case diagram explains also a basic flow of how all the actors and systems collaborate together and what kind of relationship they have between each other.

# 

Figure 1: Context model

# 

Figure 2: Use Case Diagram

|  |  |
| --- | --- |
| **Use case name** | Send message |
| **Actors** | User, Google Hangouts |
| **Description** | Users write his/her message and send it to the Chorus with Google Hangouts |
| **Data** | Chat history |
| **Pre-conditions** | User should sign up Chorus and sign in Google Hangouts to send message. |
| **Stimulus** | User write his/her message on Google Hangouts and click send button. |
| **Basic Flow** | Step 1 – User sign up Chorus.  Step 2 – User sign in via Google Hangouts.  Step 3 – User send his/her message to Chorus. |
| **Alternative Flow** | - |
| **Exception Flow** | If there is no internet connection user can not send message to Chorus. |
| **Post-conditions** | User waits for the response. |

Table 2: Send message Use Case Table

|  |  |
| --- | --- |
| **Use case name** | Select best response |
| **Actors** | Crowd Workers, Collaborative Reasoning System |
| **Description** | Users ask for information and crowd workers try to  provide the best information. Crowd workers choose best  information with the help of collaborative reasoning  system. They vote each other’s responses and system choose best one. |
| **Data** | Chat history |
| **Pre-conditions** | Workers should sign in with Amazon Mechanical  Turk |
| **Stimulus** | User open Google Hangout and after sign in they just click Chorus Bot and they write their message and click send button. |
| **Basic Flow** | Step 1 – Crowd workers propose reasonable answers.  Step 2 – Workers vote for the best answer.  Step 3 – Collaborative Reasoning System select best response |
| **Alternative Flow** | - |
| **Exception Flow** | If number of votes of two responses is same, then system choose one of them randomly. |

Table 3: Select best response Use Case Table

|  |  |
| --- | --- |
| **Use case name** | **Description** |
| Score the conversation | User score the conversation via Google Hangout when conversation is ended. |
| Sign up | User sign up Chorus on Chorus’s website after filling sign up form. |
| Send automatic message | When user starts chat with Chorus and when the conversation ends system send automatic welcome and good-bye messages. |
| User sign in | User sign in to Chorus with adding Chorus on Google Hangouts. |
| Get response | After user send message, workers write their responses and system choose best one and send it to the user. |
| Receive message | After user send message, crowd workers receive that message on Amazon Mechanical Turk |
| Respond | Crowd workers write their responses for voting. |
| Worker sign in | Workers sign in to the Chorus on Amazon Mechanical Turk with their username and passwords. |
| Add summaries of key facts | Crowd workers add summaries of key facts and important statement with the help of Curated Memory System to keep crowd workers up-to-date with details of current or past conversation that may be relevant for providing future responses. |
| Keep workers up to date | Curated Memory System keep workers up-to-date with the help of summaries of key facts. |
| Reward workers | Dynamic Scoring System reward workers for interactions they made with the user. |
| Check system errors | IT staff can see the error logs happened in the Chorus system. |
| Access data | IT staff can access all the data in the Chorus system to improve the system. |

Table 4: Use cases descriptions

# Composition view

# The top-level decomposition into system-subsystem components is shown by the component and deployment diagrams given below.

Figure 3: Component Diagram

# Design Rationale:

# To be able to use Chorus users are given authorization by Account Management System via Chorus Login Authorization Handler which then also provides an authorization for Google Hangouts.

# Users are able to communicate with workers by Google Hangouts via Google Hangouts Handler. Messages are used by the Collaborative Reasoning Unit such that the best message will be chosen to be delivered to the user. The best message is chosen by workers through Amazon Mechanical Turk via Amazon Mechanical Turk handler.

# Old messages provided by Google Hangouts are used by Curated Memory System and after the most valuable messages are chosen, they are saved for future reference.

# Workers are rewarded by Amazon Mechanical Turk via Dynamic Scoring System, such that the reward is based on the contribution of each worker to choose or propose the best responses.

# Google Hangouts provides an automated respond for the users via Automatic Dialogue System.

# 

# 

Figure 4: Deployment Diagram

# Design Rationale:

# As a database server MySQL is used because it is open-source and free.

# Database server has 2 SDD disk to increase the reliability of the system.

# Database server is separate from the Web Server to make the system scalable.

# Information view

In this view, persistent information that is kept in the database and how these maps into classes with methods and attributes is shown in a class diagram, along with the relationships of these classes with each other. Create, read, update and delete functionality that is assigned to each database component or class is also specified.

# Service Interfaces

Figure 5: Service Interfaces Class Diagram

|  |  |
| --- | --- |
| **Operation** | **Description** |
| userGetMessage | After user send message, workers write their responses and system choose best one and send it to the user. |
| workerGetMessage | After user send message, crowd workers receive that message on Amazon Mechanical Turk |
| userPostMessage | Users write his/her message and send it to the Chorus with Google Hangouts |
| workerPostMessage | Crowd workers write their responses for voting. |
| vote | Users ask for information and crowd workers try to  provide the best information. Crowd workers choose best  information with the help of collaborative reasoning  system. They vote each other’s responses and system choose best one. |
| addNote | Crowd workers add summaries of key facts and important statement with the help of Curated Memory System to keep crowd workers up-to-date with details of current or past conversation that may be relevant for providing future responses. |
| score | Dynamic Scoring System reward workers for interactions they made with the user. After each interactions system give the worker his/her score. |
| showPage | When the worker requested a page with the operation pageRequest, this operation provides  the page and sends that to the user. |
| confirmEmail | Webserver requests to confirm email when it is requested by a member user. |
| endConversation | When the conversation ends system send automatic good-bye messages. |
| userLogin | User sign in to Chorus with adding Chorus on Google Hangouts. |
| workerLogin | Workers sign in to the Chorus on Amazon Mechanical Turk with their username and passwords. |
| manageAccount | The user requested this operation is only redirected to Google Hangouts API since Chorus system does not store any user information. |

Table 5: Service interfaces operation descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Inputs** | **Outputs** | **Exceptions** |
| userGetMessage | message | Message operation OK or failed | Authorization failed |
| userPostMessage | message | Message operation OK or failed | Authorization failed |
| workerGetMessage | message | Message operation OK or failed | Authorization failed |
| workerPostMessage | userID  message | Message operation OK or failed | Authorization failed |
| vote | - | - | Amazon Mechanical Turk not available. |
| addNote | Content Name Email Website  Authorization cookie | addNote operation OK or failed | Amazon Mechanical Turk not available. |
| score | userID | Scoring is OK or failed | Amazon Mechanical Turk not available. |
| showPage | pageID | Requested page | Authorization failed |
| confirmEmail | email | Confirmation is OK or failed | Authorization failed |
| endConversation |  |  | Automatic Dialogue System not available |
| userLogin | - | - | Authorization failed,  Google Hangouts not available. |
| workerLogin | - | - | Authorization failed, Amazon Mechanical Turk not available. |
| manageAccount | - | - | Authorization failed |

Table 6: Service interfaces operation design

# Design Rationale:

# There are handler classes to communicate with external classes.

# Users are able to communicate with workers by Google Hangouts via Google Hangouts Handler. Messages are used by the Collaborative Reasoning Unit such that the best message will be chosen to be delivered to the user. The best message is chosen by workers through Amazon Mechanical Turk via Amazon Mechanical Turk handler.

# Google Hangouts provides an automated respond for the users via Automatic Dialogue System.

# CRUD Operations

# 

Figure 6: Database Class Diagram

|  |  |
| --- | --- |
| **Operation** | **CRUD Operations** |
| userGetMessage | CREATE -  READ - Conversation  UPDATE –  DELETE - |
| workerGetMessage | CREATE -  READ - Conversation  UPDATE –  DELETE - |
| userPostMessage | CREATE - Conversation  READ -  UPDATE –  DELETE - |
| workerPostMessage | CREATE - Conversation  READ -  UPDATE –  DELETE - |
| vote | CREATE -  READ -  UPDATE - Conversation  DELETE - |
| addNote | CREATE – Working Memory  READ -  UPDATE –  DELETE - |
| score | CREATE -  READ – Statistical Data  UPDATE –  DELETE - |
| showPage | CREATE -  READ - Working Memory, StatisticalData, Conversation  UPDATE –  DELETE – |
| confirmEmail | CREATE -  READ – User,Staff Member  UPDATE –  DELETE - |
| endConversation | CREATE -  READ -  UPDATE –  DELETE -- Conversation |
| userLogin | CREATE -  READ - User  UPDATE –  DELETE - |
| workerLogin | CREATE -  READ – Staff Member  UPDATE –  DELETE - |
| manageAccount | CREATE -  READ - User  UPDATE –  DELETE - |
| informITStaff | CREATE – Error Log  READ - User  UPDATE –  DELETE - |

Table 7: CRUD Operations

# Design Rationale:

# Statistical data is a weak entity because Statistical data cannot exist on its own without a Staff Member.

# MySQL was chosen as a relational database management system to ensure all integrity constraints are satisfied

# Working memory is a weak entity because working memory cannot exist on its own without a conversation.

# Conversation needs User and Staff Member to start.

# Chorus servers keeps all information of users and staff members who are two parts of a conversation.

# Chorus servers keeps statistical data about staff members to score them.

# Interface view

This view provides information about internal and external interfaces of Chorus project that helps designers, programmers and testers to know how to better use the services provided by a design subject.

# Internal Interfaces

# The interface between Google Hangouts Handler and the Collaborative Reasoning system: The Collaborative Reasoning System uses the messages that come from the Google Hangouts Handler and after it chooses the best responses by the interface between Amazon Mechanical Turk Handler and Collaborative Reasoning System if forwards the best response to the user by means of the Google Hangouts Handler.

# Design Rationale: Since choosing the best response requires the votes of the workers then the help of Amazon Mechanical Turk Handler is required such that workers will be able to vote. Having many response proposals, messages must be filtered before the answer is given to the end user and since this can’t be handled by Google Hangouts, then Google Hangouts Handler is used.

Figure 7: Vote sequence diagram

# The interface between Amazon Mechanical Turk Handler and Collaborative Reasoning System: The voting system requires workers to vote on proposed responses such that the best response will be given to the end user, so the vote result is forwarded by the Amazon Mechanical Turk Handler to the Collaborative Reasoning System which will then produce the best message and deliver it to the Google Hangouts Handler.

# 

# Design Rationale: Since the workers can’t access the messages directly by Amazon Mechanical Turk, Amazon Mechanical Turk Handler is used. So, after the voting result is ready, it must access Google Hangouts Handler to deliver the response to the user.

# External Interfaces

# User Interface

Figure 8: Chorus Bot Interface

# Chorus bot: Users are able to communicate with the crowd agents through the Chorus bot in Google Hangouts. First the user makes a request which is forwarded to the crowd and after the crowd has chosen the best response, only that answer is provided to the user and not all the responses that were suggested by the crowd agents.

# Voting Interface: Crowd agents are able to propose responses and also select reasonable responses by voting on other workers responses to be able to provide the best answer for the user. The system shows all the suggested responses to the worker and givest he worker the opportunity to vote on the response the seam to be the best. Worker is also able to retract the vote or change it. Also, if a worker disconnects their votes are removed.

# Working Memory Interface: A memory space allows crowd agents to save important portions of conversations that helps in maintaining continuity even when a new worker has joined the conversation. Worker are able to vote on lines of information to make the most valuable lines appear first such that workers will be able to find important information easily.

# Scoring Interface: Workers are rewarded based on their contribution of suggesting responses and also voting on responses of others. The workers are paid a small amount for each interaction with the interface, a medium amount for voting on an answer that it will later be chosen by the crows to be forwarded to the end user and a large reward when the worker suggests an answer that the crowd will eventually chose. To prevent the rewards from being abused workers are limited to only 3 contributions per user message.

# Admin Panel Interface: This interface is used by the IT staff and researchers where they can keep maintenance of the system and also work on errors that the system may give.

# System Interfaces

Figure 9: Chorus’s Worker Interface

# The interface between Google Hangouts, Curated Memory System and Amazon Mechanical Turk: The conversations between users and workers are saved for future reference by the Curated Memory System. These saved messages are delivered to the Amazon Mechanical Turk such that they can be accessed by the workers and used by them to give better responses for the users.

# 

# Design Rationale: Since the Curated Memory System must save data it should use a database such that it will be able to distinguish messages from different users.

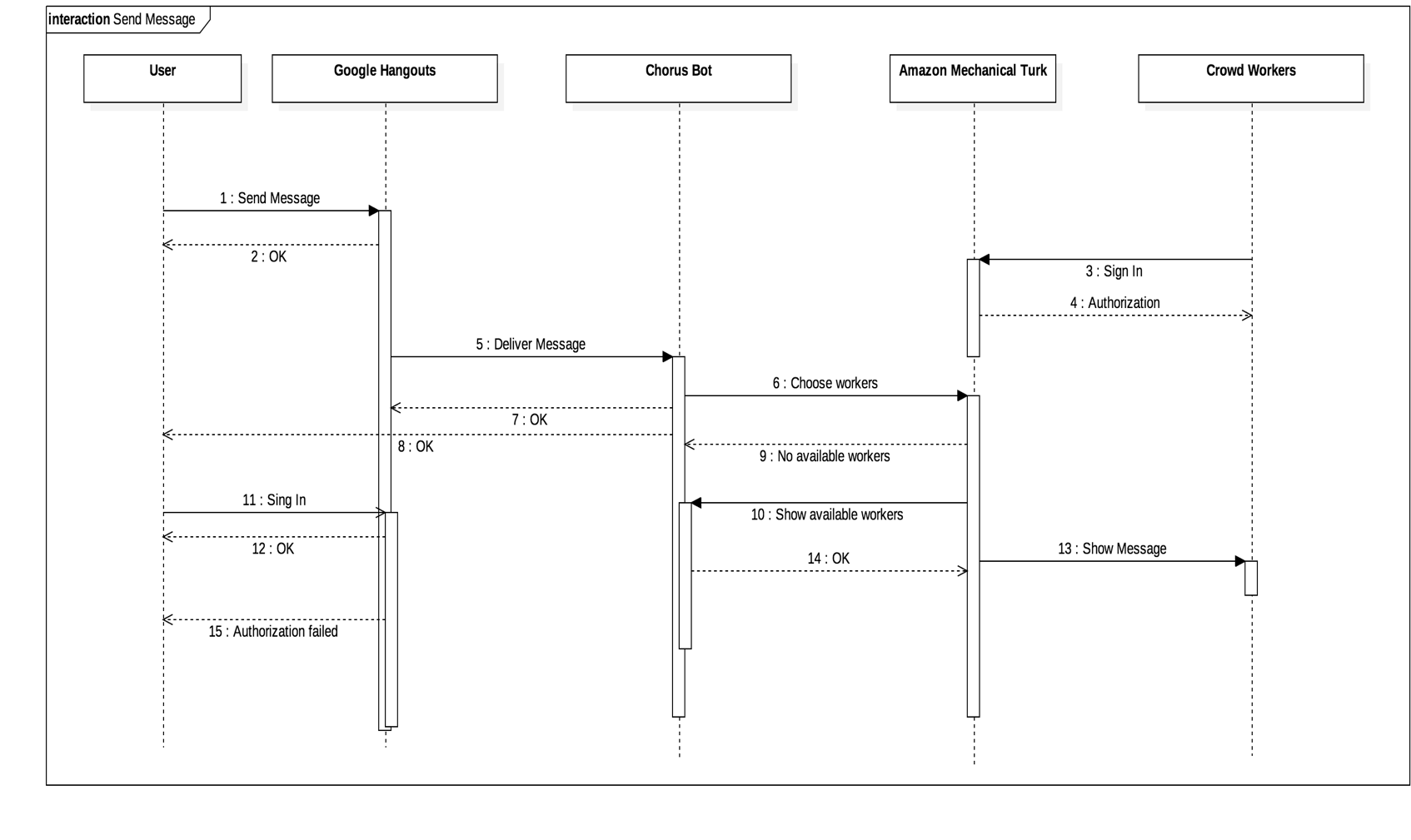


Figure 10: Send Message sequence diagram

# The interface between Google Hangouts, Chorus Login Authorization Handler and Account Management System:

# Since users have to get a permission before using the Chorus System, this authorization is given by the Account Management System, which delivers it to the Chorus Login Authorization Handler and which is then delivered by it to users through Google Hangouts.

# Design Rationale: The authorization for using Chorus it must be given by the Account Management System so for this authorization to be delivered to the user, Chorus Login Authorization Handler must be used.

# 

# The interface between Automatic Dialogue System and Google hangouts: This interface provides automated responses which are delivered to Google Hangouts such that they will be provided to users.

# Design Rationale: The Automated Dialogue System must be able to detect certain messages from users such that it will be able to give the right responses.

# The interface between Dynamic scoring system and Amazon Mechanical Turk: Since there is a rewarding system for users according to their contributions for giving or voting the best response, the Dynamic Scoring system decides about these rewards and forwarded them to the Amazon Mechanical Turk.

# Design Rationale: Since the Dynamic scoring system requires information about the previous contributions of workers, it must have access to the database to save all the data containing this information.

# 

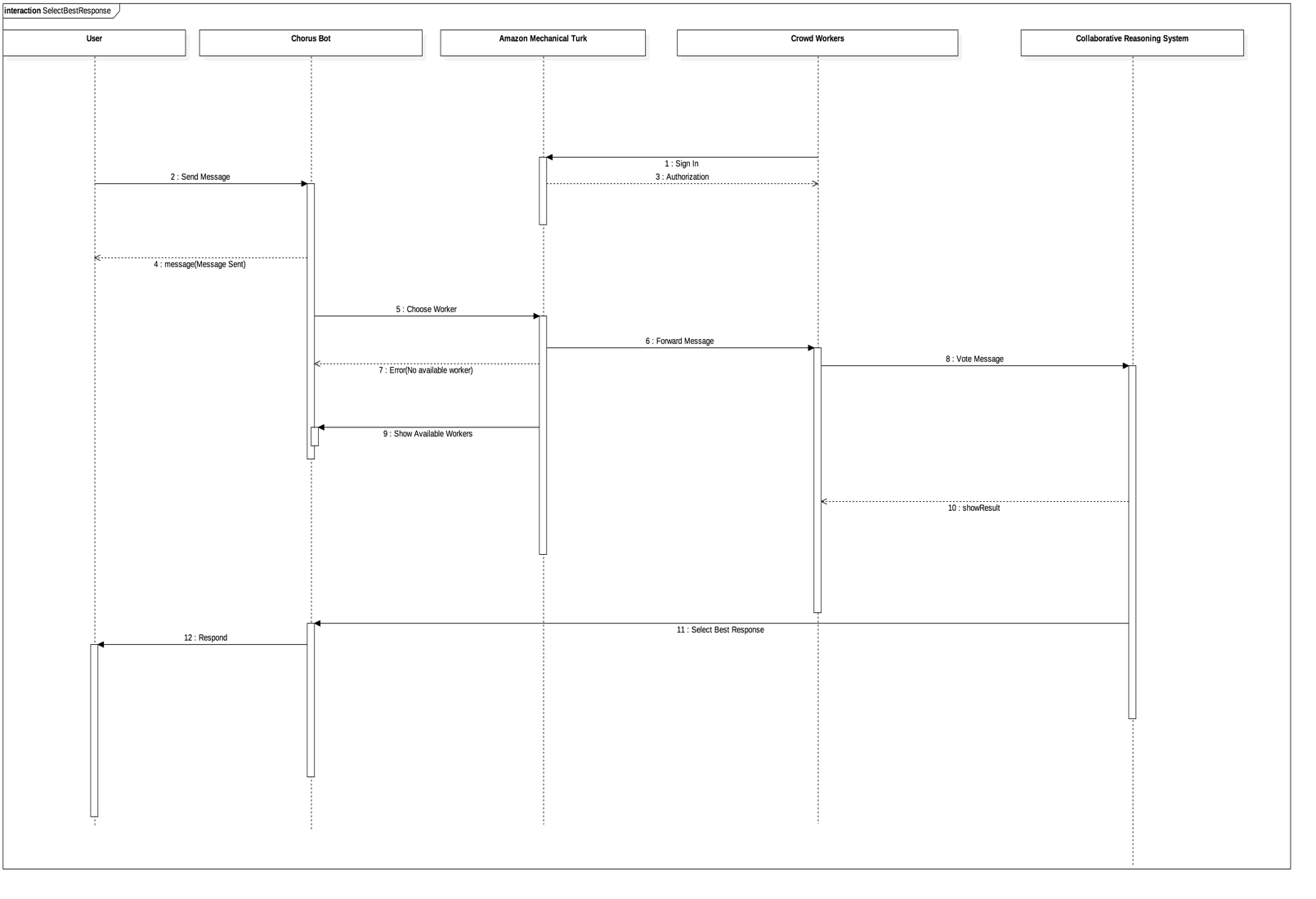


Figure 11: Select Best Response sequence diagram

# The interface between Amazon Mechanical Turk and Amazon Mechanical Turk Handler: This interface provides access of Amazon Mechanical Turk for the Chorus system.

# Design Rationale: Since workers must have access to messages and be able to provide answers, then Amazon Mechanical Turk Handler is used to provide it.

# The interface between Google Hangouts and Google Hangouts Handler: This interface provides access of Google Hangouts for Chorus system.

# Design Rationale: Since messages provided to users must be filtered and only one answer must be chosen, then Google Hangouts Handler is used to provide it.