Software Design Description

Chorus Project

Yavuz Selim YEŞİLYURT 2259166
Ahmet Dara VEFA 2237899

GROUP 3

16/04/2017

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose

Chorus Project is designed as a crowd-powered conversational assistant that allows a user to receive assistance on any online task through a two-way natural language conversation. This project is created with the aim of providing answers to requesters in real-time with enabling a conversation.

Specifically, goal of this report is to explain complete details of the project according to IEEE standards with the proposed software architecture for the Chorus System. Mostly design-wise concepts (i.e. the ways how Chorus system is structured in order to satisfy the mandatory requirements specified in SRS Report) will be mentioned. Even though we may not be able to modify specific components in a detailed way, the changes that are thought to be relevant and required will be considered through report considering a lot of sides of the project is open to be developed further and enhanced.

1.2. Scope

This report will enlighten architectural patterns and features of the project according to IEEE Standards. Detailed design specifications and descriptions (i.e. implementation details of some individual components) will not be covered. Solutions to some specific design constraints and adaptations of them to general system architecture will be mentioned through the report. UML diagrams will be used to support the context as much as possible since Unified Modeling Language materials are one of the best kinds of illustration techniques of Software Design Documents.

Scope of the system is international since anybody in the world can use it. System is simplified as much as possible for the end users for utilization purposes. Since web interface and account related context are handled through Google (and Google Hangouts), any end user with a general knowledge of how to use general service-oriented applications, will be able to use Chorus system as how they want to use. Besides, since the system is designed as convenient as possible for conducting researches, researchers will be able to use the system easily for their scientific purposes as well. Workers which are recruited from MTurk will be able give service for the end users through the system properly, they will be recruited with proper interviews handled by the IT staff. Whole technical and maintenance stuff is done behind the scenes appropriately by the Administrators (IT Staff).

1.3. Stakeholders and Their Concerns

Stakeholders in this project can be categorised into 4 groups as End Users, Workers, Researchers and Administrators.

End User: Since Chorus system is designed as a service-oriented architecture, end users have a fundamental role in working mechanism of the system, they are the main service requesters. They ask questions and request assistance from the system about any topic through the interface provided to them by Google.

Worker: End users interactions with the system directly interest workers. They are responsible for providing real-time answers and assistance for the people who needed

assistance (interacted with the system) behind the scene. They are provided with special environment and interface for handling their job and they will get paid in accordance with their performance

Researcher: Researchers will be interested with the scientific aspects of the system. They will conduct researches, gather data, evaluate them and view their results for their scientific purposes. They will also contribute to enhancement of the system as much as they can.

Administrator: These people are concerned with the technical issues of the system. Their main responsibility is to provide maintenance through handling possible conflicts that arises while system is operating. They will make sure that Chorus is working well in general with controlling system servers, databases and other system users. Besides they will keep the system up-to-date with making necessary modifications related with bugs or improvements of the system whenever system needs so.

2. References

IEEE Computer Society. (2009). *IEEE Standard for Information Technology—Systems Design—Software Design Descriptions.* IEEE. New York: IEEE. IEEE Std 1016

Other sources:

Singh, M. P. (2005). The practical handbook of Internet computing . Boca Raton: Chapman & Hall/CRC.

Sommerville, I. (2016). Software engineering. Boston: Pearson Education Limited.

3. Glossary

- AWS: Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.
- PHP: It is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. PHP means PHP: Hypertext Preprocessor.
- SimpleDB: Amazon Simple Database Service (SimpleDB) is a highly available and

flexible non-relational database that allows developers to request and store data, with minimal database management and administrative responsibility.

- HTTPS: It is a protocol for secure communication over a computer network which is widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security, or its predecessor, Secure Sockets Layer. The main motivation for HTTPS is authentication of the visited website and protection of the privacy and integrity of the exchanged data.
- SHA512: SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the National Security Agency (NSA). Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity.
- JQuery: It is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT license. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.
- Laravel: It is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern.

4. Architectural Views

4.1. Context View

This viewpoint contains all stakeholders' use cases and their associated use case tables with detailed descriptions of normal flow, pre and postconditions and exceptions. These tables describe the flow of the system on different scenarios and they provide how system should behave when an exception related with some specific topic takes place. The following figure is the context diagram of the system which shows the relations between Chorus, actors and the others systems.

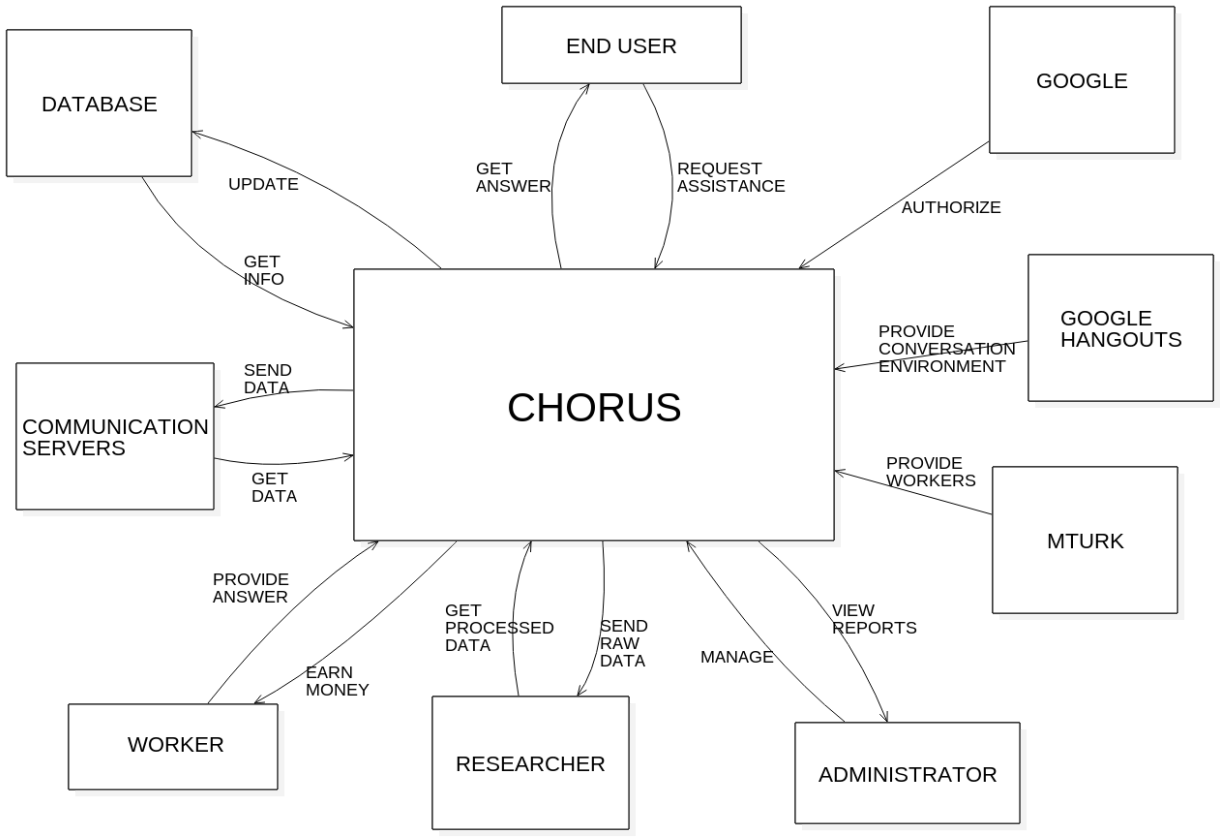


Figure 1: Context Diagram

4.1.1. Use Cases

4.1.1.1. End Users

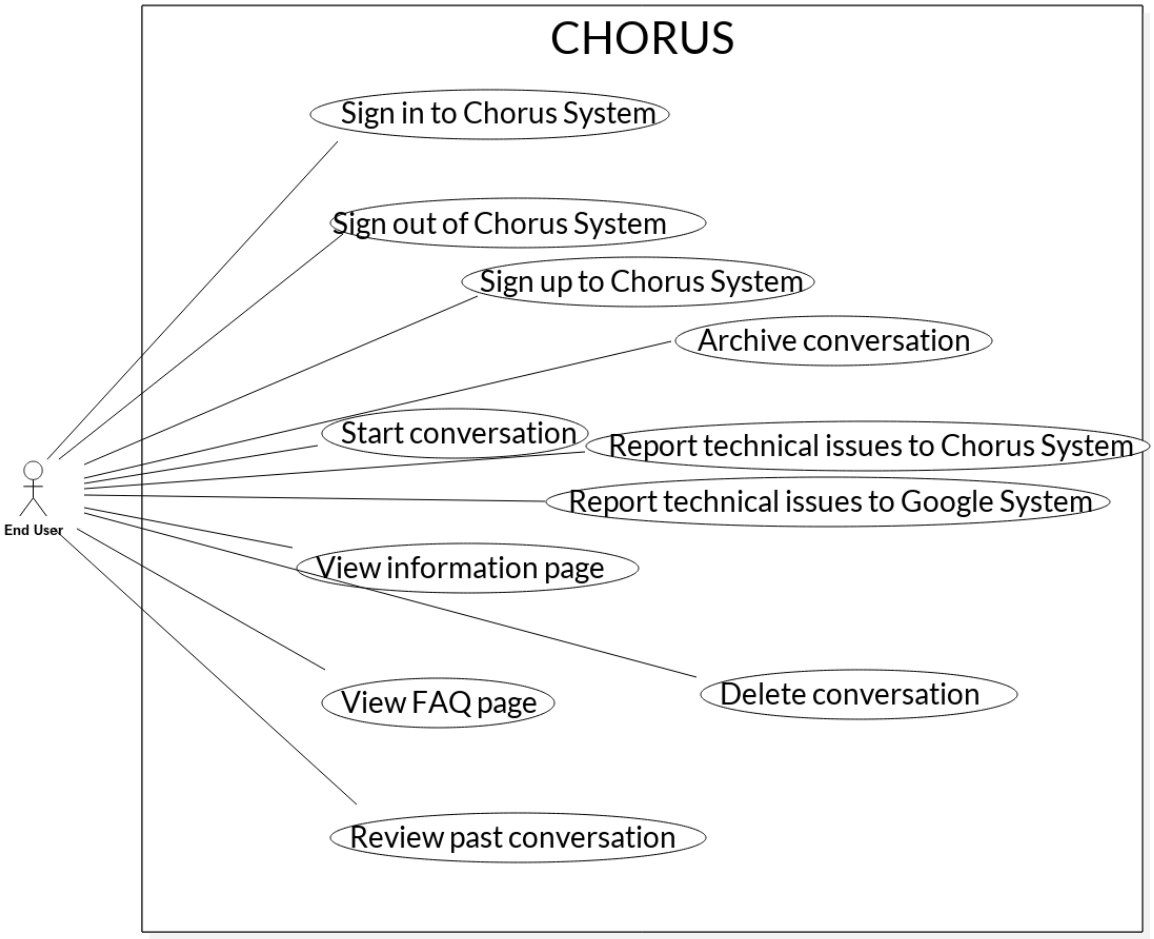


Figure 2: End User Use Case Model

Use Case ID	1
Use Case Name	Sign in
Actors	Chorus End User
Description	End User signs in to Chorus system in order to create conversation with workers.
Preconditions	End User exists in the system with a valid username and password.
Postconditions	End User is successfully signed in to Google

	system.
Normal Flow	1. End User requests to enter Google system. 2. System prompts end user to sign in. 3. End User enters his/her own gmail and password. 4. System verifies entered information. 5. Google validates the End User. 6. System signs End User in to Google system.
Exceptions	4. a. If the gmail and/or password entered by the End User is/are not valid: I. Google system displays “Invalid mail address or password” error message. II. Use case returns on step 2 of normal flow.

Table 1: Sign In For End User

Use Case ID	2
Use Case Name	Sign up
Actors	Chorus End User
Description	End User signs up to Chorus system in order to be able to access chat functionalities of the system.
Preconditions	End User that does not exist in the system fills in the blanks with valid first and last name, gmail, username and password.
Postconditions	End User is successfully signed up to Chorus system.
Normal Flow	1. End User opens sign up page located at http://talkingtothecrowd.org/signup. 2. End User enters their first and last name, gmail, username and password. End User also ticks the “i am 18 years old or older” and “i accept the End-User Agreement” boxes 3. System verifies entered information.

	4. Chorus validates the End User. 5. System successfully signs up the End User into Chorus system.
Exceptions	3. a. If the first or last name, email, username or password entered by the observer is/are not valid: I. Chorus system displays " first or last name, email, username or password is not valid" error message. End User needs to enter valid format information. For example, for email, End User needs to enter valid email type. II. Use case returns on step 2 of normal flow. 3. b. If the email entered by the user exist in the system: I. Chorus system displays "The email address you entered already exists in the system. Please enter another email address or click "forgot my password" button" error message. II. Use case returns on step 2 of normal flow 3. c. If the username entered by the End User exist in the system: I. Chorus system displays "The username you entered already exists in the system. Please enter another username" error message. II. Use case returns on step 2 of normal flow

Table 2: Sign Up For End User

Use Case ID	3
Use Case Name	Sign Out
Actors	Chorus End User
Description	End User signs out of Google system in a successfully way.

Preconditions	End User must be signed in to Google system
Postconditions	End User is successfully signed out of Google system.
Normal Flow	1. End User requests to sign out Google system. 2. System checks session information. 3. System signs out the End User from the Google system. 4. System returns End User to sign in page.
Exceptions	2. a. If the session is not created by the system: I. Google system automatically goes to step 4 of the use case. 2. b. If the time of the session is exceeded: I. Google system automatically goes to step 4 of the use case.

Table 3: Sign Out For End User

Use Case ID	4
Use Case Name	Start conversation with Chorus
Actors	Chorus End User
Description	End User starts talking with the Chorus.
Preconditions	End User must be signed in to Google System. End User must be signed up to Chorus System.
Postconditions	End User successfully conversed with Chorus.
Normal Flow	1. End User clicks the Chorus Bot's name on the left of the screen where conversations are. 2. A chat box pops up on the right. 3. User starts typing his/her message, emoji and/or sends a photo or a drawing. 4. Chorus responds.

	5. Use case returns on step 3 of normal flow.
Exceptions	4. a. If there is no workers to respond: I. A worker will be assigned to the user. II. Use case will return to step 4 of normal flow. 3.a. If user stops chatting: I. Workers will leave the channel.

Table 4: Start Conversation With Chorus

Use Case ID	5
Use Case Name	View information page
Actors	Chorus End User
Description	End User opens “information page” in order to get information about Chorus System.
Preconditions	End User must be in the Google Hangouts page.
Postconditions	End User is in “Information page”.
Normal Flow	1. End User presses the “About” button. 2. System redirects End User to the “Information page”.
Exceptions	-

Table 5: View Information Page

Use Case ID	6
Use Case Name	View past conversation
Actors	Chorus End User
Description	End User views past conversations in order to read them.
Preconditions	End User is signed in to Google System. End User is signed up to Chorus System. End User has conversed with Chorus before.
Postconditions	End User successfully reads past conversations.

Normal Flow	1. End User opens the chat box of the Chorus Bot. 2. End User scrolls through the chat until he/she reads anything he/she wants. 3. End User successfully read the past conversation.
Exceptions	2. a. If the past chat with Chorus bot is partially or fully deleted: I. End User can not read through those texts. II. System returns to step 1 of normal flow.

Table 6: View Past Conversation

Use Case ID	7
Use Case Name	Archive conversation
Actors	Chorus End User
Description	End User archives conversation to hide it from the inbox until the next time he/she receives a message from Chorus.
Preconditions	End User has to be signed in to Google. End User must have conversed with Chorus. The conversation must not be archived already.
Postconditions	The conversation is successfully archived.
Normal Flow	1. End User presses the 3 vertical dots next to the name of Chorus Bot in the contacts window. 2. End User presses archive button in the pop-up window. 3. Message is archived successfully.
Exceptions	-

Table 7: Archive Conversation

Use Case ID	8
Use Case Name	Delete conversation

Actors	Chorus End User
Description	End User deletes past conversation from the contact window
Preconditions	End User must be signed in to Google. End User must have had a conversation with Chorus that is not deleted.
Postconditions	End User successfully deletes past conversation.
Normal Flow	1. End User presses the 3 vertical dots next to the contacts name. 2. End User selects the delete option from the pop-up window. 3. System removes the contact from contact window.
Exceptions	3. a. If the conversation can not be found: I. Chorus system displays “Conversation could not be found” error message. II. Use case returns on step 1 of normal flow.

Table 8: Delete Conversation

Use Case ID	9
Use Case Name	Report technical issues to Google
Actors	Chorus End User
Description	End User reports technical issue to Google for the issue to be solved by Google.
Preconditions	End User must be signed in to Google.
Postconditions	The report is successfully sent to Google.
Normal Flow	1. End User presses the report button at the bottom of contacts window. 2. End User fills the necessary boxes of the pop-up window. 3. End User presses “send” button to send the report.

Exceptions	-

Table 9: Report Technical Issues To Google

Use Case ID	10
Use Case Name	Report technical issues to Chorus
Actors	Chorus End User
Description	End User reports technical issue to Chorus for the issue to be solved by Chorus.
Preconditions	End User is signed in to Google. End User is signed up to Chorus System. End User has conversed with Chorus.
Postconditions	The report is successfully sent to Chorus.
Normal Flow	1. End User presses the report button at the bottom of “X” button of the chat window. 2. End User fills the necessary boxes of the pop-up window. 3. End User presses “send” button to send the report.
Exceptions	-

Table 10: Report Technical Issues To Chorus

4.1.1.2. Workers

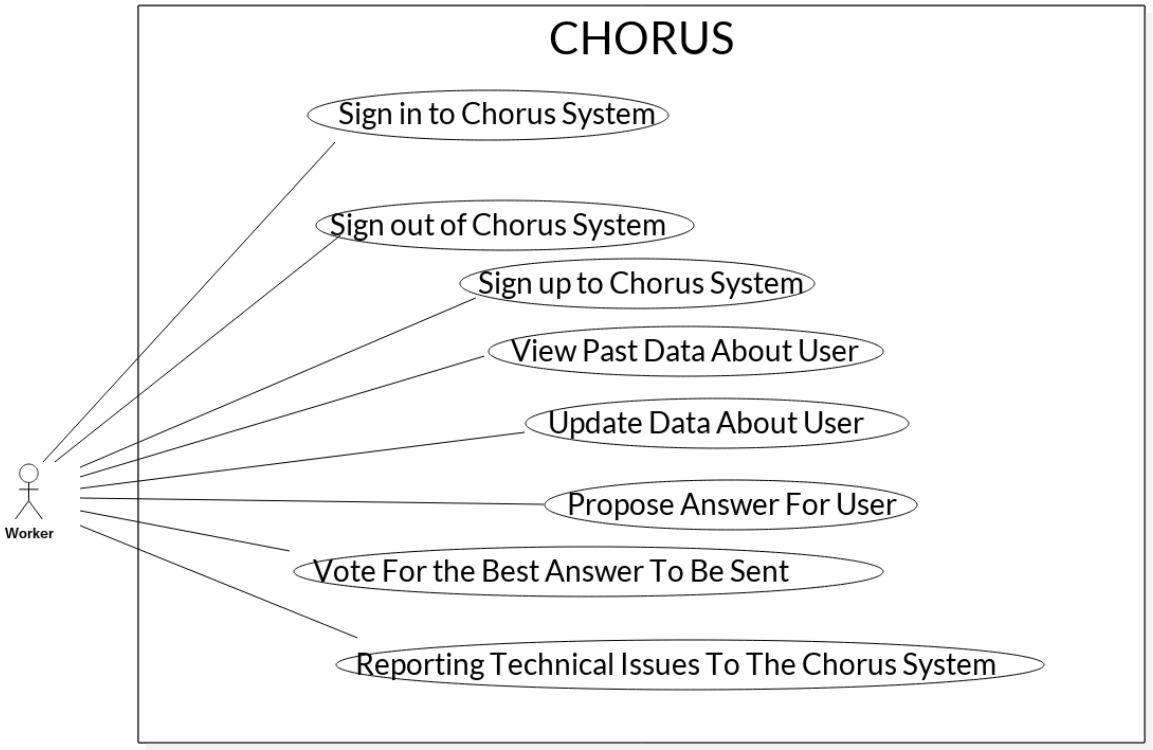


Figure 3: Worker Use Case Model

Use Case ID	1
Use Case Name	Sign in
Actors	Chorus Worker
Description	Worker signs in to the Chorus System successfully.
Preconditions	Worker is not already signed in to Chorus System. Worker is signed up to Chorus System. Worker is not banned from the Chorus System.
Postconditions	Worker is successfully signed in to Chorus System.
Normal Flow	1. Worker opens the worker sign in page.

	2. Worker enters his/her username and password. 3. Chorus System checks the information. 4. Chorus System validates the worker. 5. Worker is signed in to Chorus System.
Exceptions	3. a. If the username and/or password entered by the worker is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 2 of normal flow.

Table 11: Sign In For Worker

Use Case ID	2
Use Case Name	Sign Up
Actors	Chorus Worker
Description	Worker signs up to Chorus System in order to be able to send and receive messages.
Preconditions	Worker chooses a valid username. Worker chooses a valid password. Worker is not already signed up. Worker meets the minimum requirements specified on MTurk.
Postconditions	Worker is successfully signed up for Chorus.
Normal Flow	1. Worker presses the join button on MTurk. 2. MTurk redirects the worker to the worker signup page. 3. Worker types a username and a password. 4. Chorus System checks the information. 5. Chorus System validates the worker. 6. Chorus System adds the worker to the database. 7. User is given a link to the worker

	interface where he/she can do his/her job after signing in.
Exceptions	4. a. If the username and/or password entered by the worker is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 3 of normal flow.

Table 12: Sign Up For Worker

Use Case ID	3
Use Case Name	Sign Out
Actors	Chorus Worker
Description	Worker signs out of the chorus system in a successful way.
Preconditions	Worker must be signed in to chorus system.
Postconditions	Worker is successfully signed out of the Chorus System.
Normal Flow	1. Worker requests to sign out of the Chorus System. 2. System checks session information. 3. System signs out the Worker from the Chorus System. 4. System returns Worker to sign in page.
Exceptions	2. a. If the session not created by the system: I. Chorus system automatically goes to step 4 of normal flow. 2. b. If the time of the session is exceeded: I. Chorus system automatically goes to step 4 of normal flow.

Table 13: Sign Out For Worker

Use Case ID	4
Use Case Name	View past relevant data about end user
Actors	Chorus Worker
Description	Worker looks at the past data about the user in order to be more effective at answering End User's questions.
Preconditions	Worker must be signed in to Chorus System. End User must have past data in the Chorus System.
Postconditions	Worker successfully gets relevant data about End User.
Normal Flow	1. Worker joins the chat room. 2. Worker scrolls through the working memory to read the past data about the End User. 3. Worker successfully gets the data about the End User.
Exceptions	-

Table 14: View Past Relevant Data About End User

Use Case ID	5
Use Case Name	Update data about end user
Actors	Chorus Worker
Description	Worker updates the data about the user in order to contribute to system accuracy.
Preconditions	Worker must be signed in to Chorus System. The data must receive enough votes from the workers.
Postconditions	The data is successfully updated to the Chorust System.
Normal Flow	1. Worker types important data about the End User to the "Working Memory" section in his/her interface.

	2. Workers upvote the data. 3. Chorus System checks the minimum number of votes required and total number of votes the data has. 4. The data gets pushed to the Chorus System.
Exceptions	3. a. If the data does not get enough votes after 3 minutes or after 8 other entries have been added to the “Working Memory” section: I. Chorus System removes the data from “Working Memory” section. II. Use case returns on step 1 of normal flow. 3. b. If the data gets 2 downvotes: I. Chorus System removes the data from “Working Memory” section. II. Chorus System sends a report to the admins in order for them to review and penalize the worker. III. Use case returns on step 1 of normal flow. 3. c. If there is less than 4 workers in the chat room: I. The data will be removed with only 1 downvote. II. The data will be updated with minimum (# of workers-1) upvotes. III. Use case returns on step 1 of normal flow

Table 15: Update Data About End User

Use Case ID	6
Use Case Name	Propose answer for End User
Actors	Chorus Worker
Description	Worker types an answer in order to respond to End User.
Preconditions	Worker must be signed in to Chorus System. The answer must receive enough votes from the workers.

Postconditions	The answer is successfully given to End User.
Normal Flow	1. Worker types his/her response in the chat box under the “Chorus” sign. 2. Workers upvote the respond. 3. Chorus System checks the minimum number of votes required and total number of votes the response has. 4. The response is successfully given to End User.
Exceptions	3. a. If the answer does not get enough votes after 3 minutes or after 8 other entries have been added to the chat section: I. Chorus System removes the answer from chat section. II. Use case returns on step 1 of normal flow. 3. b. If the there is less than 4 workers in the chat: I. The answer will be given with minimum (# of workers-2) upvotes. II. Use case returns on step 1 of normal flow.

Table 16: Propose Answer For End User

Use Case ID	7
Use Case Name	Vote for answer to be sent to end user
Actors	Chorus Worker
Description	Worker votes for responses to send them to users.
Preconditions	Worker is signed in to Chorus System. A potential answer is posted on chat box.
Postconditions	The response is successfully upvoted.
Normal Flow	I. Worker presses the “click here if this answer the user” button on the right of the answer. II. Chorus System logs the worker who gave the answer, the worker who

	upvoted and the relevant chat history, and sends them to the admin in order to review and reward the worker who gave the answer. III. The answer is successfully upvoted.
Exceptions	1. a. If the answer is already upvoted by the worker: I. Worker will be unable to upvote the same answer again. II. Use case returns on step 1 of the normal flow. 1. b. If the answer has already gained enough upvotes: I. Worker will be unable to upvote the posted answer. II. Use case returns on step 1 of the normal flow.

Table 17: Vote For Answer To Be Sent To End User

Use Case ID	8
Use Case Name	Reporting technical issues to the Chorus System
Actors	Chorus Worker
Description	Worker reports technical issues to the Chorus System for the system to perform better.
Preconditions	Worker must be signed up to Chorus System.
Postconditions	The report is successfully sent.
Normal Flow	1. Worker finds a fault. 2. Worker clicks to the “Report” button in the worker interface. 3. Worker enters the necessary information. 4. The report is successfully sent.
Exceptions	-

Table 18: Reporting Technical Issues To The Chorus System

4.1.1.3. Researcher

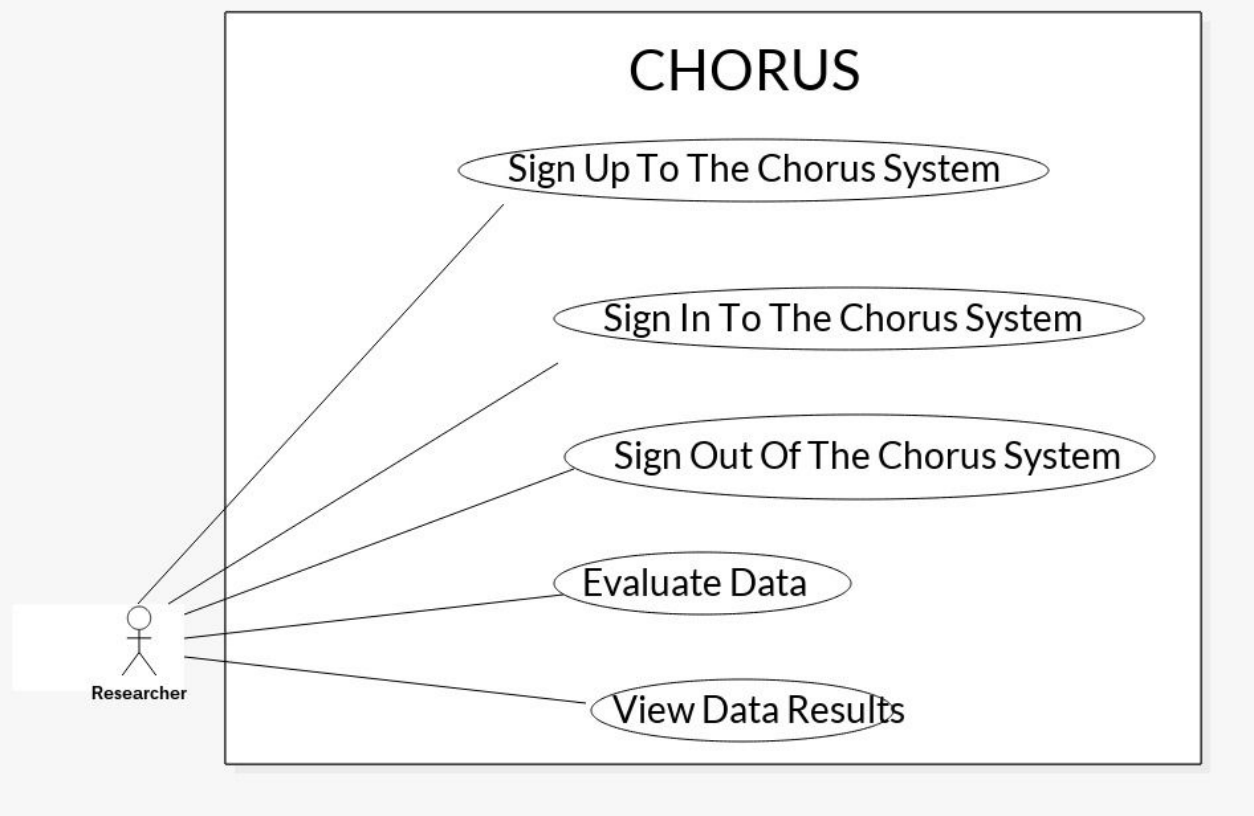


Figure 4: Researcher Use Case Model

Use Case ID	1
Use Case Name	Sign in
Actors	Chorus Researcher
Description	Researcher signs into Chorus system in order to analyze data that is collected from the conversations.
Preconditions	Researcher must be signed up to Chorus System. Researcher is not already signed in to Chorus System
Postconditions	Researcher is successfully signed in to Chorus system.
Normal Flow	1. Researcher opens the researcher sign

	in page. 2. Researcher enters his/her username and password. 3. Chorus System checks the information. 4. Chorus System validates the Researcher. 5. Researcher is signed in to Chorus System.
Exceptions	3. a. If the username and/or password entered by the researcher is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 2 of normal flow.

Table 19: Sign In For Researcher

Use Case ID	2
Use Case Name	Sign Up
Actors	Chorus Researcher
Description	Researcher signs up to Chorus System in order to be able to analyze the data collected from conversations.
Preconditions	Researcher chooses a valid username. Researcher chooses a valid password. Researcher is not already signed up. Researcher is chosen beforehand.
Postconditions	Researcher is successfully signed up for Chorus.
Normal Flow	1. Researcher receives the one time use sign up link. 2. Researcher types a username and a password.Researcher also ticks “I accept the Researcher Agreement” box. 3. Chorus System checks the

	information. 4. Chorus System validates the Researcher. 5. Chorus System adds the researcher to the database. 6. Researcher is given a link to the researcher interface where he/she can do his/her job after signing in.
Exceptions	3. a. If the username and/or password entered by the researcher is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 2 of normal flow.

Table 20: Sign Up For Researcher

Use Case ID	3
Use Case Name	Sign Out
Actors	Chorus Researcher
Description	Researcher signs out of the Chorus System in a successful way.
Preconditions	Researcher must be signed in to Chorus System.
Postconditions	Researcher is successfully signed out of the Chorus System.
Normal Flow	1. Researcher requests to sign out of the Chorus System. 2. System Checks session information. 3. System signs out the researcher from the Chorus System. 4. System returns researcher to sign in page.
Exceptions	2.a. If the session not created by the system: I. Chorus system automatically goes to step 4 of normal flow. 2. b. If the time of the session is exceeded:

	I. Chorus system automatically goes to step 4 of normal flow.

Table 21: Sign Out For Researcher

Use Case ID	4
Use Case Name	View data results
Actors	Chorus Researcher
Description	Researcher views the data from the conversations in order to analyze it.
Preconditions	Researcher must be signed in to Chorus System. There must be data from the conversations in the database.
Postconditions	Researcher successfully views the data.
Normal Flow	1. Researcher goes to the researcher database interface on Amazon SimpleDB. 2. Researcher presses the view data button. 3. Chorus System returns the data. 4. Researcher successfully views data.
Exceptions	3.a. If the result can not be found by the system: I. Chorus system automatically goes to step 1 of normal flow.

Table 22: View Data Results

Use Case ID	5
Use Case Name	Evaluate data
Actors	Chorus Researcher
Description	Researcher evaluates the viewed data in order to have a result.
Preconditions	Researcher must be signed in to Chorus

	System. There must be data from the conversations in the database. Researcher must have viewed new data.
Postconditions	Researcher successfully comes up with a result.
Normal Flow	1. Researcher evaluates data. 2. Researcher gets a result. 3. Researcher pushes the result into the database.
Exceptions	

Table 23: Evaluate Data

4.1.1.4. Administrator

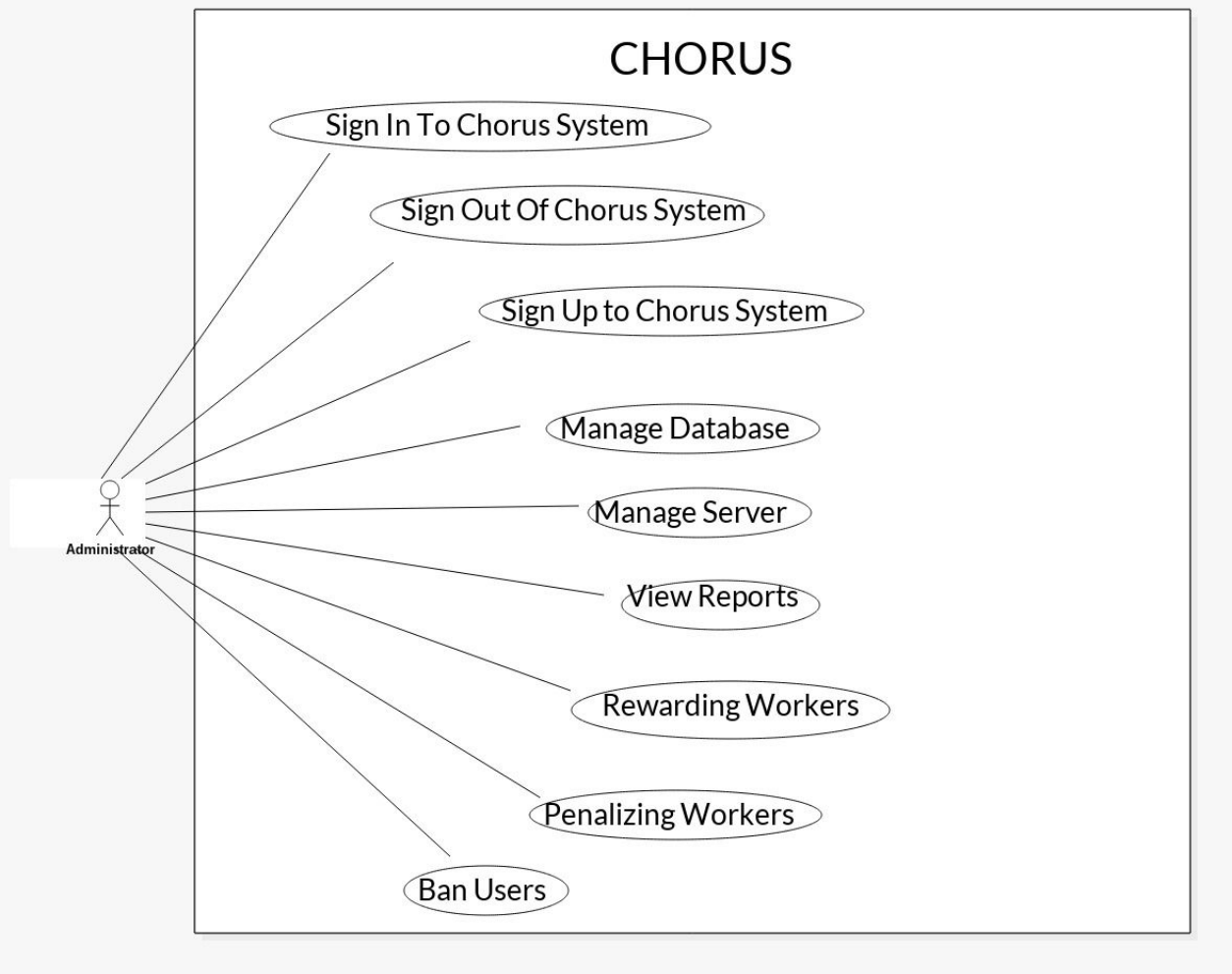


Figure 5: Administrator Use Case Model

Use Case ID	1
Use Case Name	Sign in
Actors	Chorus Administrator
Description	Administrator signs in to Chorus system in order to be able to do his/her job.

Preconditions	Administrator exists in the system with a valid username and password.
Postconditions	Administrator is successfully signed in to Chorus system.
Normal Flow	1. Administrator requests to enter Chorus system. 2. System prompts administrator to sign in. 3. Administrator enters his/her own username and password. 4. Chorus System checks the information. 5. Chorus System validates the administrator. 6. System logs administrator into Chorus system.
Exceptions	4. a. If the username and/or password entered by the administrator is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 3 of normal flow.

Table 24: Sign In For Administrator

Use Case ID	2
Use Case Name	Sign Up
Actors	Chorus Administrator
Description	Administrator signs up to Chorus system in order to be able to access the system management page of the system.
Preconditions	Administrator chooses a valid username. Administrator chooses a valid password.

	Administrator is not already signed up. Administrator will be chosen beforehand.
Postconditions	Administrator is successfully signed up to Chorus system.
Normal Flow	1. Administrator opens the single use sign up page. 2. Administrator enters username and password. Administrator also ticks the “I accept the Administrator Agreement” box. 3. Chorus System checks the information. 4. Chorus System validates the administrator. 5. Chorus system adds the administrator to the database. 6. Administrator is given a link which redirects administrator to admin interface when clicked.
Exceptions	3.a If the username and/or password entered by the administrator is/are not valid: I. Chorus system displays “Invalid username or password” error message. II. Use case returns on step 2 of normal flow.

Table 25: Sign Up For Administrator

Use Case ID	3
Use Case Name	Sign Out
Actors	Chorus Administrator
Description	Administrator signs out from Chorus system in a successfully way.
Preconditions	Administrator must be already signed in to Chorus system.
Postconditions	Administrator is successfully signed out

	Chorus system.
Normal Flow	1. Administrator requests to sign out Chorus system. 2. System checks session information. 3. System logs out administrator from the Chorus System. 4. System returns to the administrator sign in page.
Exceptions	2. a. If the session is not created by the system: I. Chorus system automatically goes to step 4 of the normal flow. 2. b. If the time of the session is exceeded: I. Chorus system automatically goes to step 4 of the normal flow.

Table 26: Sign Out For Administrator

Use Case ID	4
Use Case Name	Manage Database
Actors	Chorus Administrator
Description	Administrator manage databases. For example, s/he can easily add a new worker type in the “Manage Database” section.
Preconditions	Administrator must be signed in to Chorus system.
Postconditions	Administrator successfully manages the databases of Chorus system.
Normal Flow	1. Administrator requests to manage databases of the system by clicking ”Manage Databases” button in “Admin Dashboard” 2. System loads “Manage Databases” page. 3. Administrator selects which database s/he wants to manage. 4. Administrator does operation in the selected database and save it.

	5. System displays the result of the operation 6. System automatically returns “Admin Dashboard” page.
Exceptions	4. a. If the changes do not save in the database: I. Chorus system displays “The changes were not saved in the database” error message. Administrator needs to do it again. II. Chorus system automatically goes to step 3 of the normal flow.

Table 27: Manage Database

Use Case ID	5
Use Case Name	Manage Server
Actors	Chorus Administrator
Description	Administrator manages servers. For example, s/he can balance the conversations volume and flow via “Manage Server” section.
Preconditions	Administrator must be signed in to Chorus system.
Postconditions	Administrator successfully manages the servers of Chorus system.
Normal Flow	1. Administrator requests to manage servers of the system by clicking ”Manage Servers” button in “Admin Dashboard” 2. System loads “Manage Servers” page. 3. Administrator selects which operation s/he wants to do in server structure 4. Administrator does the operation and saves it. 5. System displays the result of the operation. 6. System automatically returns “Admin Dashboard” page.

Exceptions	4. a. If the changes do not save in the database: I. Chorus system displays “The changes were not saved in the database” error message. Administrator needs to do it again. II. Chorus system automatically goes to step 3 of the normal flow.

Table 28: Manage Server

Use Case ID	6
Use Case Name	View Reports
Actors	Chorus Administrator
Description	Administrator views the reports that is sent from the workers and end users.
Preconditions	Administrator must be signed in to Chorus system.
Postconditions	Administrator successfully views the reports that is sent from workers and end users.
Normal Flow	1. Administrator requests to view the reports by clicking ”Reports” button in “Admin Dashboard” 2. System loads “Reports” page. 3. Administrator selects report s/he wants to view. 4. Administrator views the selected report.
Exceptions	3. a. If there is no report to be seen: I. System goes to step 2 of the normal flow.

Table 29: View Reports

Use Case ID	7
Use Case Name	Banning End Users

Actors	Chorus Administrator
Description	Administrator bans the end users for a time period or indefinitely.
Preconditions	Administrator must be signed in to Chorus system.
Postconditions	Administrator successfully bans the end user.
Normal Flow	1. Administrator requests to ban end user by clicking ”Bans” button in “Admin Dashboard” 2. System loads “Bans” page. 3. Administrator types the username of the end user s/he wants to ban to the search bar. 4. Administrator adjusts the time limit that end user will be prohibited from the system. 5. Administrator bans the end user. 6. System displays the result of the operation. 7. System automatically returns “Admin Dashboard” page.
Exceptions	5. a. If the process is interrupted: I. Chorus system displays “The process is interrupted” error message. Administrator needs to do it again. II. Chorus system automatically goes to step 3 of the normal flow.

Table 30: Banning End Users

Use Case ID	8
Use Case Name	Penalize workers
Actors	Chorus Administrator
Description	Administrator penalizes the workers by lowering the payments of them or suspending them for a time period.
Preconditions	Administrator must be signed in to Chorus

	system.
Postconditions	Administrator successfully penalizes the worker.
Normal Flow	1. Administrator requests to penalize workers by clicking ”Bans” button in “Admin Dashboard” 2. System loads “Bans” page. 3. Administrator types the username of the worker s/he wants to penalize to the search bar. 4. Administrator selects the penalizing method from the list. 5. Administrator penalizes the worker. 6. System displays the result of the operation. 7. System automatically returns “Admin Dashboard” page.
Exceptions	5. a. If the process is interrupted: I. Chorus system displays “The process is interrupted” error message. Administrator needs to do it again. II. Chorus system automatically goes to step 4 of the normal flow.

Table 31: Penalize Workers

Use Case ID	9
Use Case Name	Rewarding Workers
Actors	Chorus Administrator
Description	Administrator rewards the workers in accordance with their performance.
Preconditions	Administrator must be signed in to Chorus system.
Postconditions	Administrator successfully rewards the worker.
Normal Flow	1. Administrator requests to reward worker by clicking ”View Staff” button

	in “Admin Dashboard” 2. System loads “View Staff” page. 3. Administrator types the name of worker s/he wants to reward to the search bar. 4. Administrator selects the rewarding method from the list. 5. Administrator rewards the worker. 6. System displays the result of the operation. 7. System automatically returns to the “Admin Dashboard Page”.
Exceptions	5. a. If the worker is currently banned: I. System responds with “Worker is banned, can not reward worker” message. II. System returns to step 2 of normal flow

Table 32: Rewarding Workers

4.2. Composition View

The composition viewpoint contains information about project components and the functionalities of these components (subsystems). It shows how subsystems and external systems related with each other and illustrates the functions from a top-level point of view. These relationships can be seen in the component diagram below:

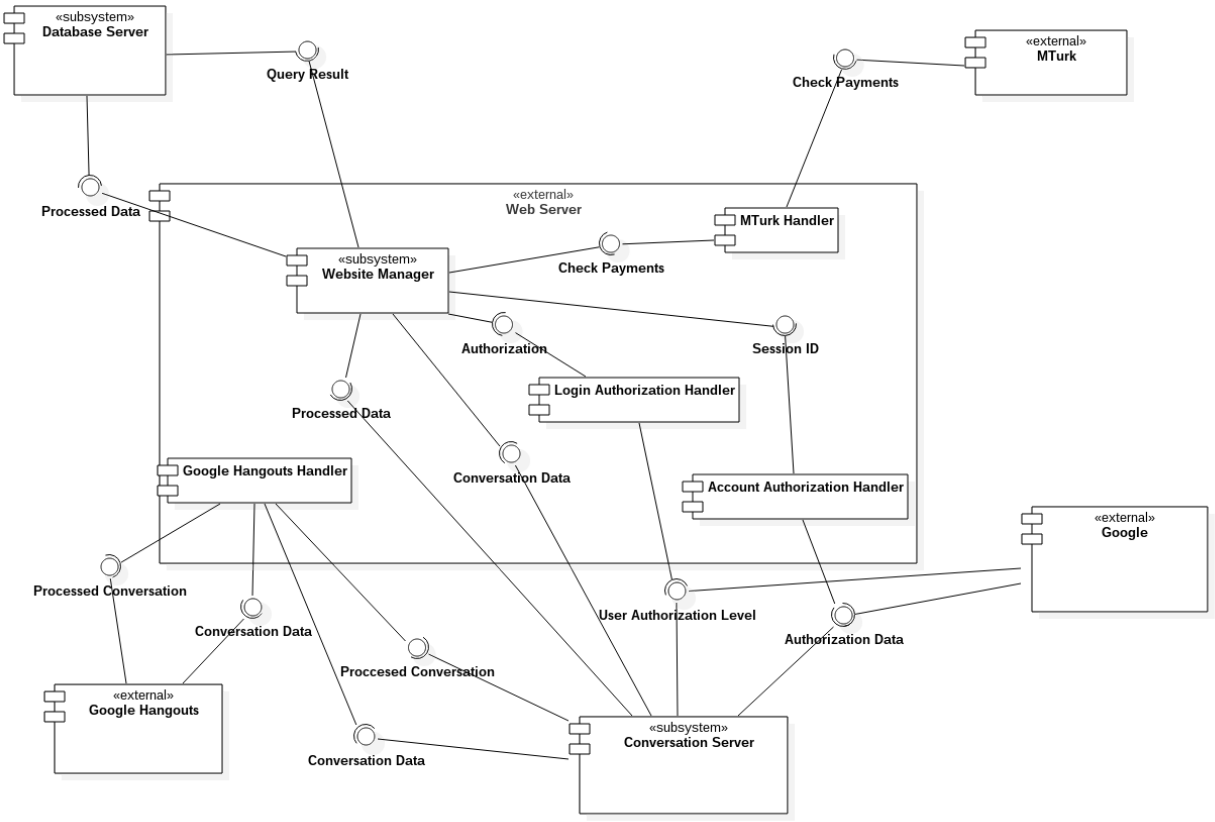


Figure 6: Component Diagram

Physical deployment of the information generated by the system (i.e. conversation data, authorization data, processed data, account data) on hardware components can be seen in the deployment diagram below:

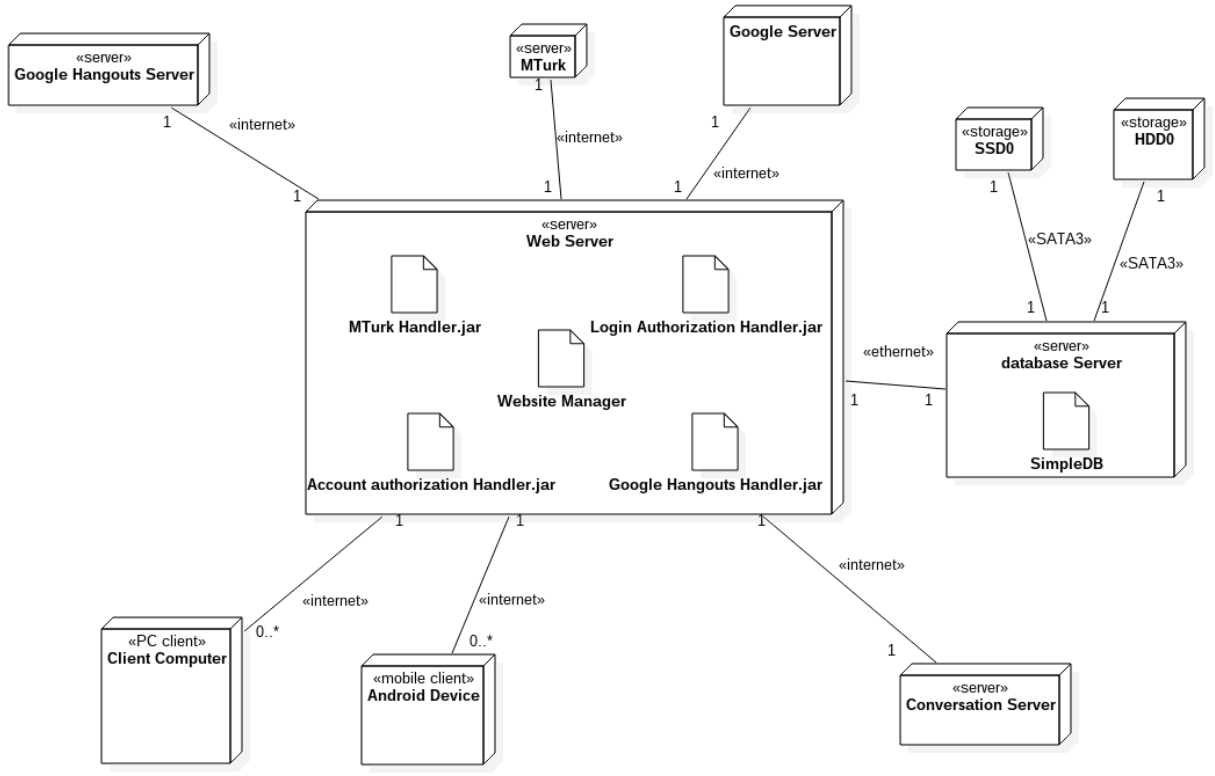


Figure 7: Deployment Diagram

4.2.1. Design Rationale

- Web Server subsystem will contain general internal elements of the Chorus that interacts with the external systems or other subsystems like Database Server. This will make Web Server component a core component of the system where all the relevant subsystems and handlers are wrapped in. This structure of Web Server makes it simpler to keep interactions between systems and other components more clear and organized. Besides containing the small components like Handlers for different operations in one main subsystem will decrease the complexity of the system structure.

- Since the system has different user interfaces for different users, we have divided the interfaces of the users into two system components, namely Website Manager and Google Hangouts. Website Manager will be responsible for providing interfaces to workers, administrators and researchers and Google Hangouts will be responsible for providing interface to end users. The data which flows between Website Manager and Google Hangouts will always arrive Conversation Server before arriving to the destination, since system needs to process all the ongoing data transactions. The purpose of this architecture is to make sure that data transactions between end users

and workers are safe and there are no data package losses due to problems that can occur within network.

- For security and scalability purposes Database Server subsystem will not be covered by the Web Server. Access to Databases will only be allowed from Website Manager component since only the workers, administrators and researchers can access to databases for their specific purposes.

- Authorization issues will be handled by Google for End Users and by System itself (Conversation Server) for the other users of the system. Their authorization data will be collected by Website Manager to give users Session IDs.

- Website Manager component will act like a core inside core (Web Server component), since all the processes will eventually arrive to that component. The aim of this structure is to keep all the processes and tasks organized and classified.

4.3. Information View

In Information viewpoint the relationship between the classes and the persistent information kept in database with their corresponding methods and attributes are described. The class diagram which shows these mappings is given below:

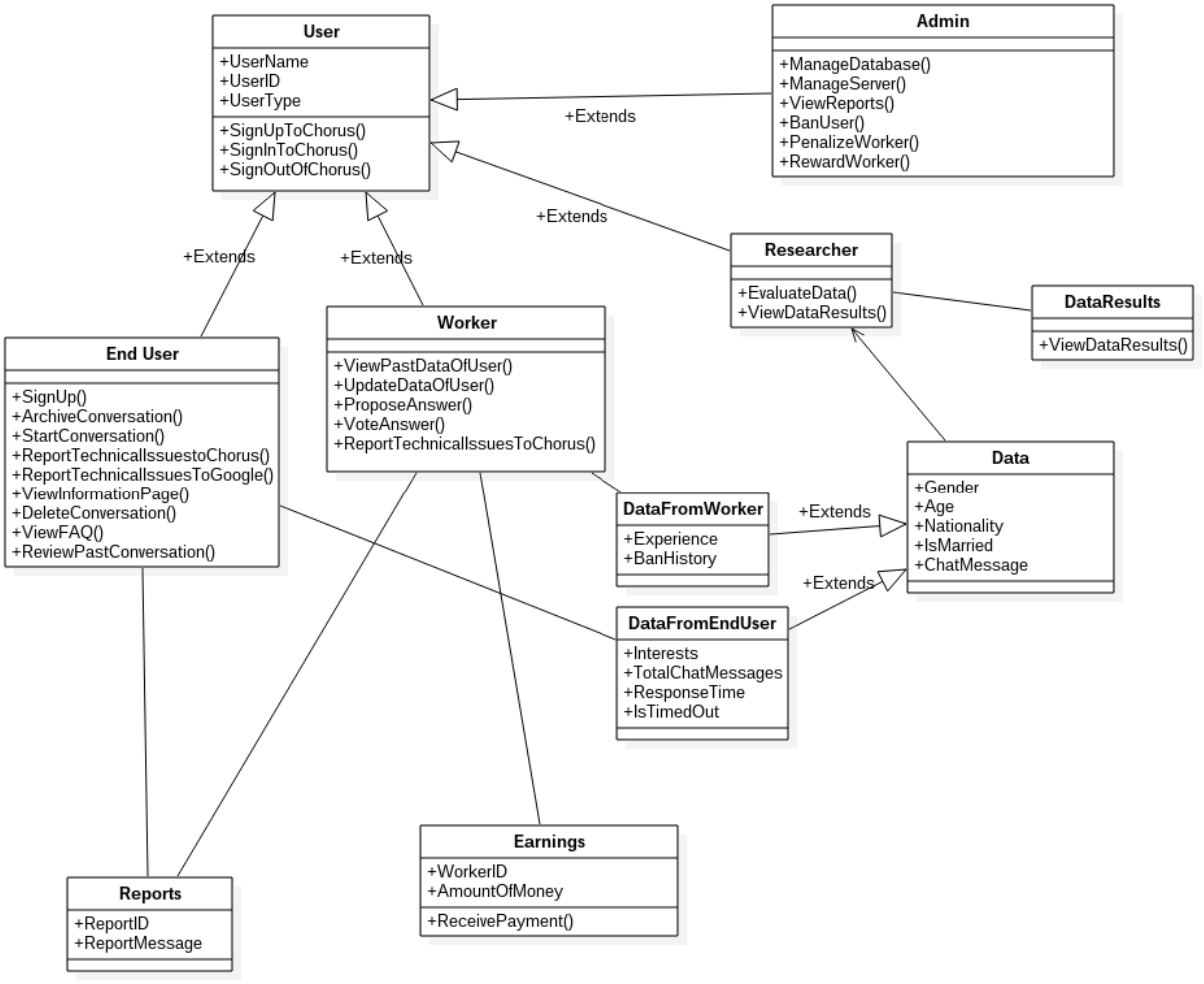


Figure 8: Class Diagram

Also to identify and describe the relationship between different entities of the system with each other, E/R diagram for the Chorus system is given below:

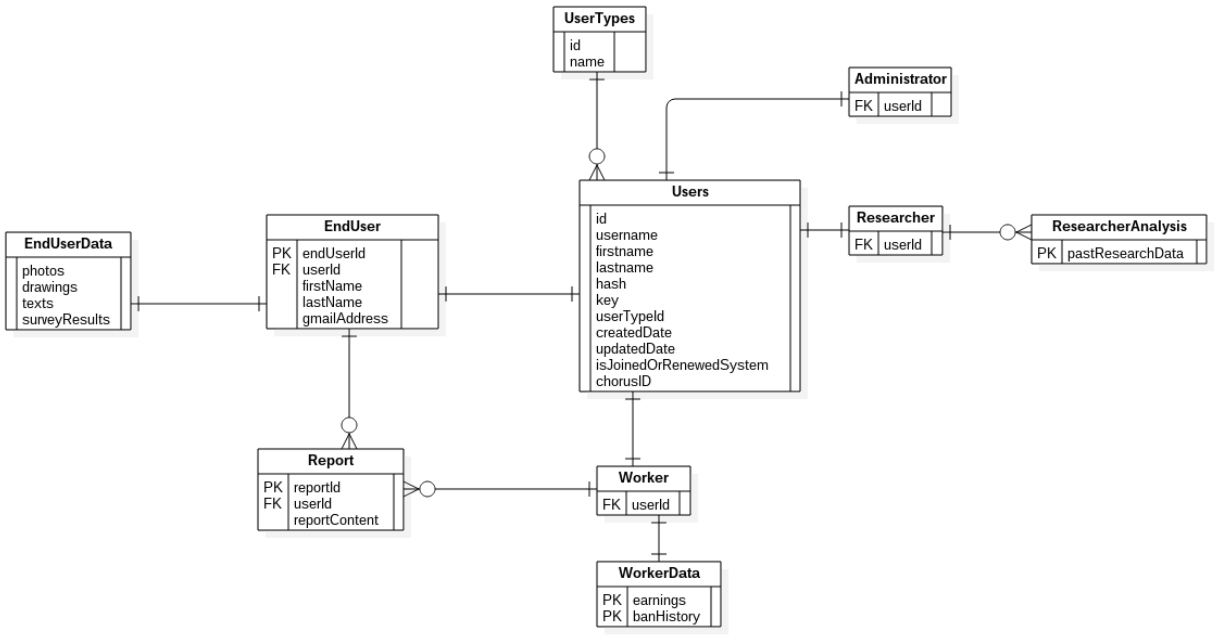


Figure 9: E/R Diagram

As you can see, there are 10 tables in the database. These are User, EndUser, Worker, Administrator, Researcher, EndUserData, Report, WorkerData, ResearchAnalysis andUserTypes. Now, we're inspecting their data attributes below:

- UserTypes: id, name
- Users: id, username, firstname, lastname, hash, key, userTypeId, createdDate, updatedDate, isJoinedOrRenewedSystem, chorusId
- EndUser: endUserId(PK), userId(FK), firstName, lastName, gmailAddress
- EndUserData: photos, drawings, texts, surveyResults
- Report: reportId(PK), userId(FK), reportContent
- Worker: userId(FK)
- WorkerData: earnings(PK), banHistory(PK)
- Administrator: userId(FK)
- Researcher: userId(FK)
- ResearcherAnalysis: pastResearchData(PK)

For entities in E/R Diagram above we used one-to-many and 1-1 relations. The relations are listed as the following:

- The relation between UserTypes and Users is one-to-many because one user type has more than one user.
- The relation between Users and Administrator, Researcher, Worker, EndUser is 1-1 because one user can only be one of Administrator, Researcher, Worker, EndUser.
- The relation between Researcher and ResearcherAnalysis is one-to-many because one researcher can have multiple analysis.
- The relation between Worker and WorkerData is 1-1 because one worker can have only one WorkerData.
- The relation between Report and Worker is one-to-many because one worker can have multiple reports.
- The relation between EndUser and Report is one-to-many because one end user can have multiple reports.
- The relation between EndUser and EndUserData is 1-1 because one end user can only have one EndUserData.

4.3.1 Service Interface and Design for CRUD Operations

The interactions between web pages and database management systems are provided by the options of the service interfaces. All users of the system except End Users can see and use those. Administrators have the authority to manage these interfaces (i.e. update) and manage the access of the other users who have the authority to use them. Researchers and Workers can use them for their responsibilities in the system and they can not manage the way they use these interfaces.

Administrators of the system are using CRUD (create, read, update and delete) approach for managing persistent storage in databases because of its performance. End User relational information, staff information, system logs and conversational data can be processed with CRUD operations.

4.3.2 Design Rationale

We considered certain non-functional requirements, which we specified in SRS Document, as more important and decided to give more value to them. For instance, portability and performance for data operations over databases and overall system architecture are important subjects in this view. Therefore we designed our database system and data access operations accordingly. We simplified the basic functions that can be used to modify, copy or transfer data from system database to another system component like IT environment. Besides, admins of the system use CRUD operations to satisfy this requirement. Also we designed the data forms portable and provided a controlling mechanism for that and their integrity while

transactions. The reason for this is, since our system has a service-oriented nature, among the 4 user types, End Users are the ones that interacts with the system and gets the most of the service from it. They ask questions, request assistance, report bugs and with this way contribute to general system architecture, which requires also that the system must be stable and run on every platform and device without any problem. Hence, in this viewpoint we tried to eliminate the problems regarding data forms and their integrity.

Another subject that we considered as important is maintainability of the system. This is important mostly due to the long-term performance of the project. To satisfy the parts of this requirement that is related with information architecture of the system, we decided to log every feedback (report) to our databases and design an automated notification mechanism for the administrators to inform them about these feedbacks that are logged into databases. Administrators will review these and make necessary adjustments on system architecture periodically.

4.4. Interface View

There are 2 kinds of interface classes. First one is internal interfaces which is mostly about subsystem routines and major components and concerns only the administrators and second one is external interfaces which shows the interaction between system service interface and user interface in a straightforward way and is open to all users.

Below describes the corresponding main page mock-ups and some UI Features of the system interfaces related with different types of users of the Chorus,

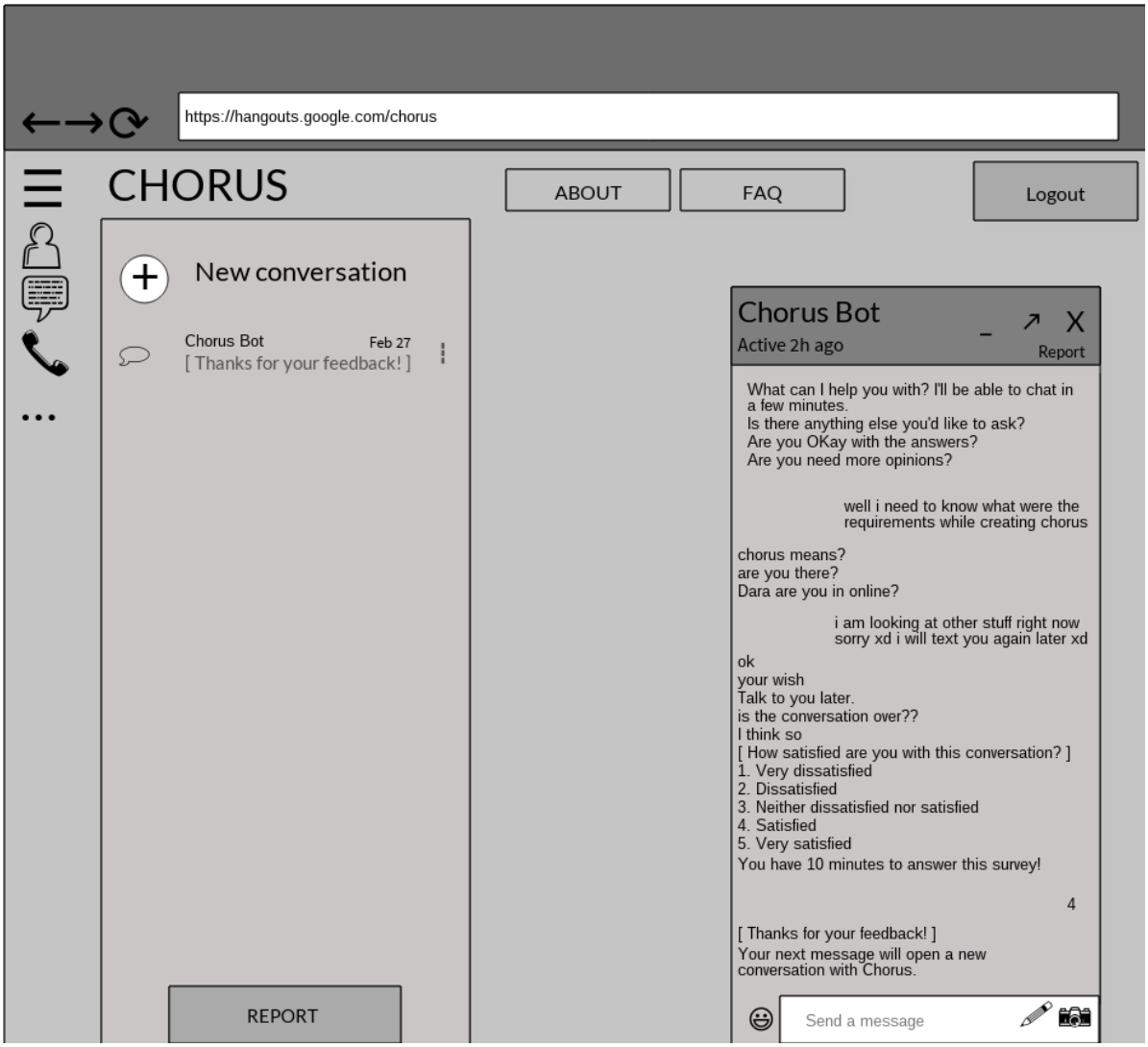


Figure 10: Main Screen Mock-up For End User

End users interface for both browser and smartphone parts are entirely on Google Hangouts. For the browser interface, on the main page there is a sub window on the left (contact window) which shows a list of users of Google Hangouts including Chorus Bot as small subsections. End Users can archive or delete the past conversations they had with Chorus Bot by clicking the three dots symbol on the rightmost section of the subsection. They can view the past conversations they had with Chorus by clicking to the subsection. Also that button enables end user to start a conversation with the Chorus system.

The end users will maintain their conversation with Chorus through a sub window on the right (chat window) which only pops out when user starts a conversation with Chorus. End Users can send photos to the Chorus via clicking the button at the bottom right corner of the chat

window, they can send emojis through bottom left corner of the chat window. End Users are also able to draw pictures and forward them through clicking pencil button near photo button.

End Users can see the status of the Chorus from the top left corner of the chat window and they can close the window by clicking the cross button at the top right corner of the chat window. There will be a active Chorus logo (status indicator) which will symbolize the “typing” information about Chorus when a response is being composed by the workers.

End Users can view information page about Chorus by clicking to “About” button at top right corner, near “Sign out” button and they can see the Frequently Asked Questions page by clicking to the “FAQ” button near “About” button. End Users can report technical issues about Google system by clicking to the “Report” button from bottom left corner of the main page and they can report technical issues about Chorus system by clicking to the “Report” button from the top right corner of the chat window. End Users can sign out through clicking to the “”Sign out” button near “About” button.

Android end user interface is represented below:

Figure 11: Android Main Screen Mock-up For End User

Smartphone interface of the system is also handled by Google Hangouts application. End Users can interact with the system via main chat screen which has the same layout as the chat window in the browser. When the end user taps the message box, a default phone

keyboard appears on the screen where the user can type. This keyboard is small enough to let the end user see chat and the message he/she is typing when it is on the screen. The end user can also send emojis by clicking the smiley face, drawings by clicking the pencil, photos by clicking the camera buttons. The end user can take new photos when he/she clicks the camera button.

For workers a specialized interface for Chorus is designed. On the main page there is three sub windows which shows the chat screen, shared chat history about end users and earnings box respectively. Workers are able to both propose responses and vote on the responses of others via main chat screen. They can also make use of a collective shared chat history (working memory) via shared chat history sub window on the right which allows them to maintain continuity throughout the conversation, even as some leave and new workers join. Above shared memory sub window there is an earnings window which shows workers how much money they earned from the system in total.

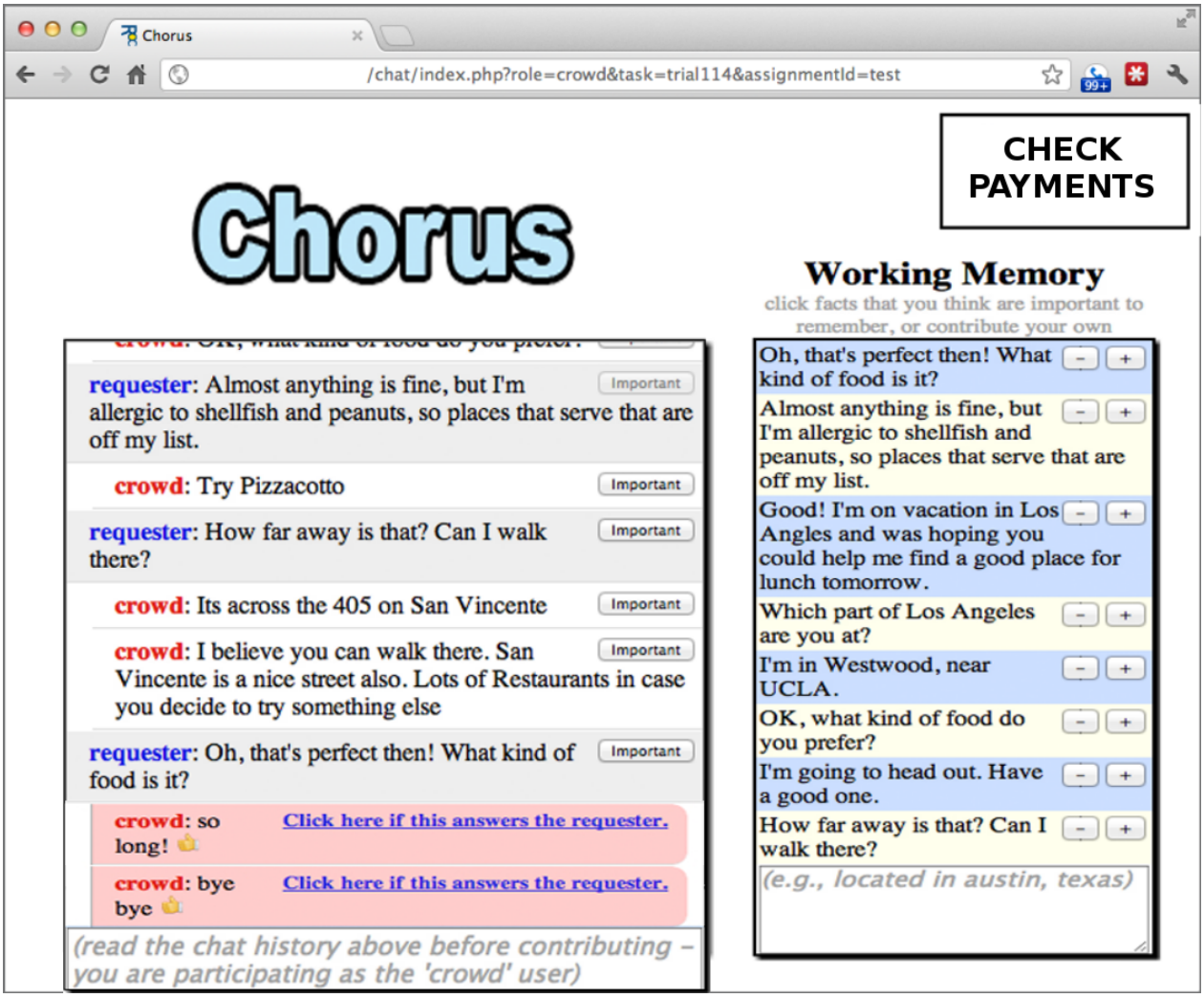


Figure 12: Main Screen Mock-up For Worker

Another specialized interface is designed for the admin interface. On the main page there is a navigation bar which contains all the functionality tools that admin can utilize. On “Dashboard” page admin can view all the end user and worker activity with time informations and can view all warnings, errors, logs that come from the system. Other buttons on the bar are also available for utilizing different features. For example the “Bans” page shows ban protests, ban list, ban submissions and provides an environment for the management of banning mechanism of the system.

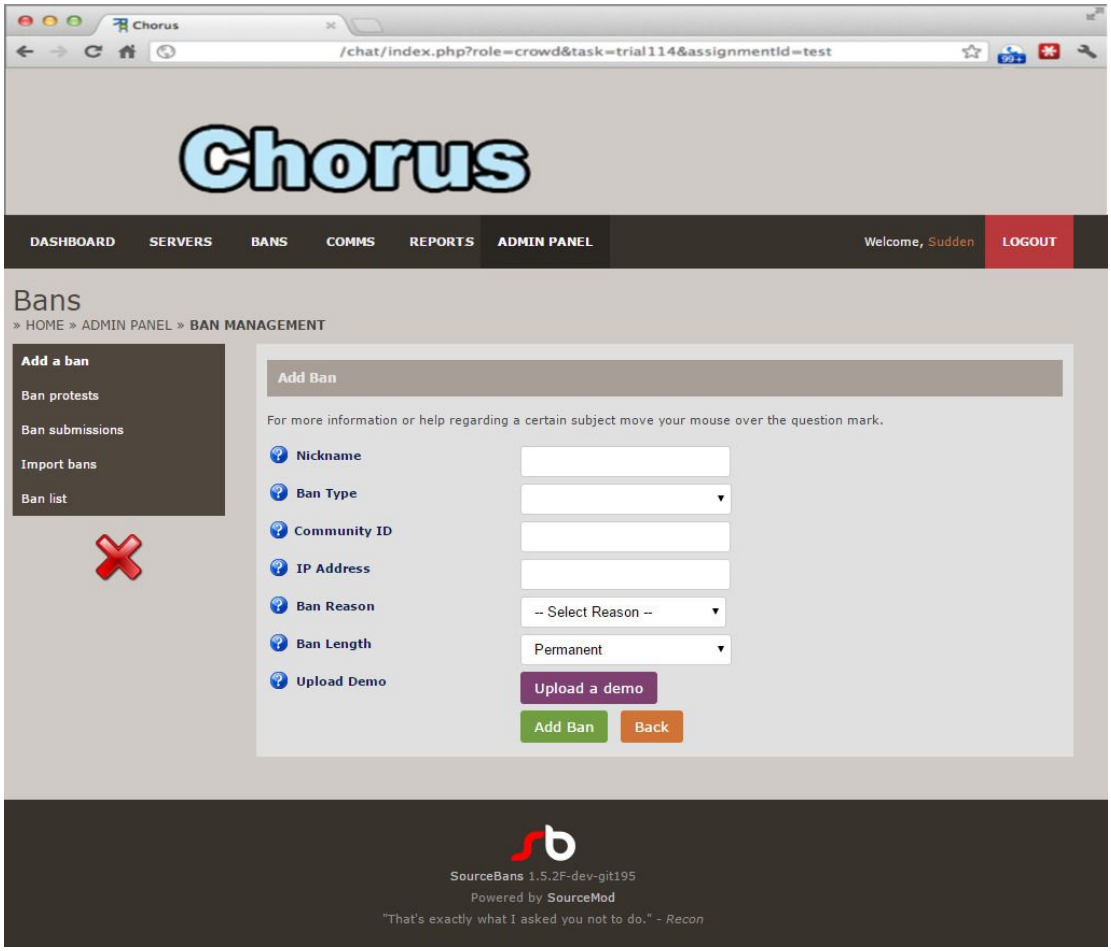
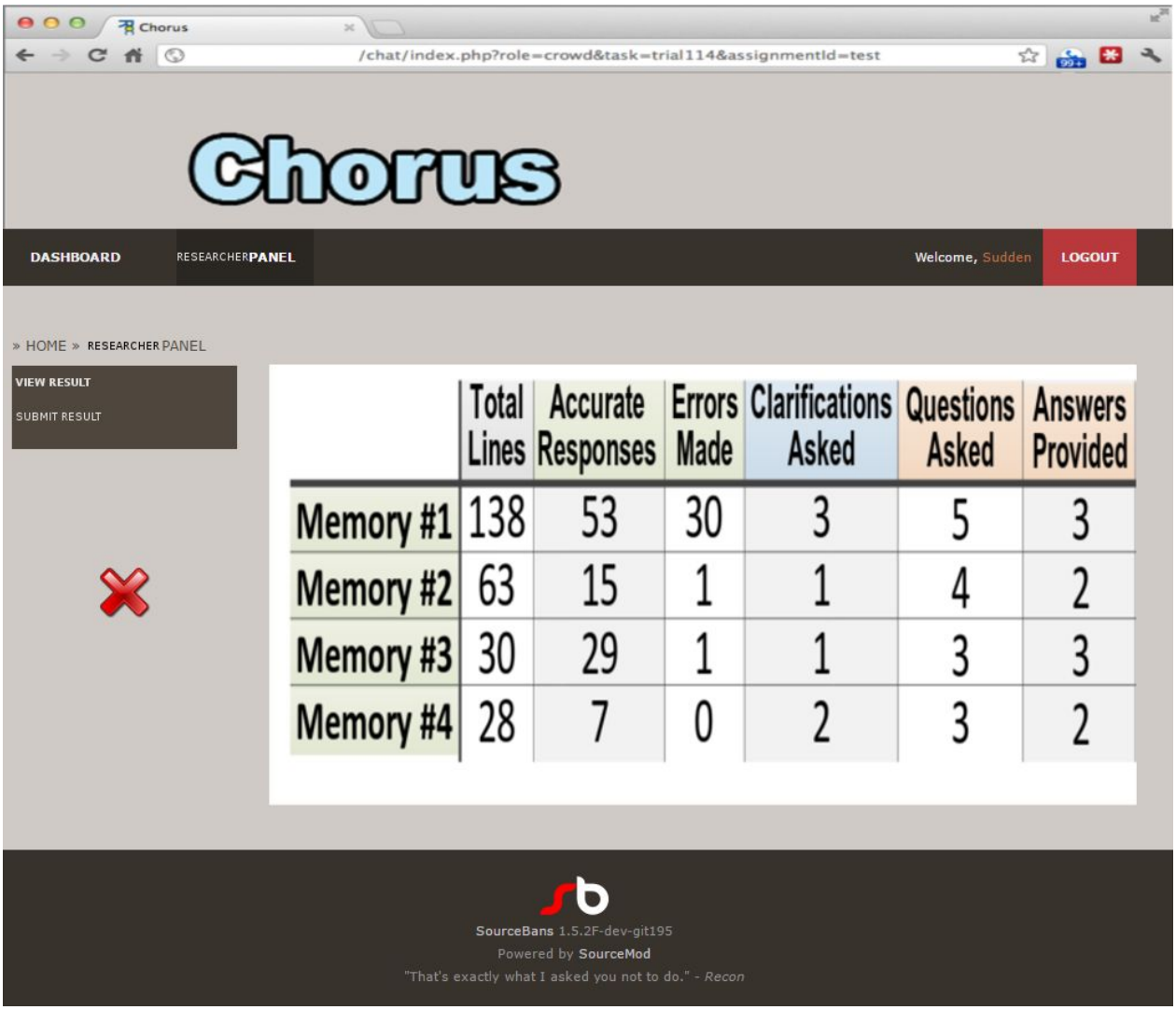


Figure 13: Main Screen Mock-up For Admin

In the researcher interface there are a couple of features that researcher can utilize for the research, testing and development purposes. There is a “Manage Researches” page which redirects researcher to another page which has a set of functionalities that provide suitable and convenient environment for the research purposes. For instance, researchers can access databases to carry out their researches and they can maintain their tests and analyze results

from this page. Also there is a “Results” page which shows the result history based on the past researches.



	Total Lines	Accurate Responses	Errors Made	Clarifications Asked	Questions Asked	Answers Provided
Memory #1	138	53	30	3	5	3
Memory #2	63	15	1	1	4	2
Memory #3	30	29	1	1	3	3
Memory #4	28	7	0	2	3	2

Figure 14:Main Page Mock-up For Researcher

Sign in system is different for each user type but they all have common aspects such as hashing technique for validating username and password, holding sessions etc. The figure below demonstrates sign in screen for end user which is done via Google (external interface).

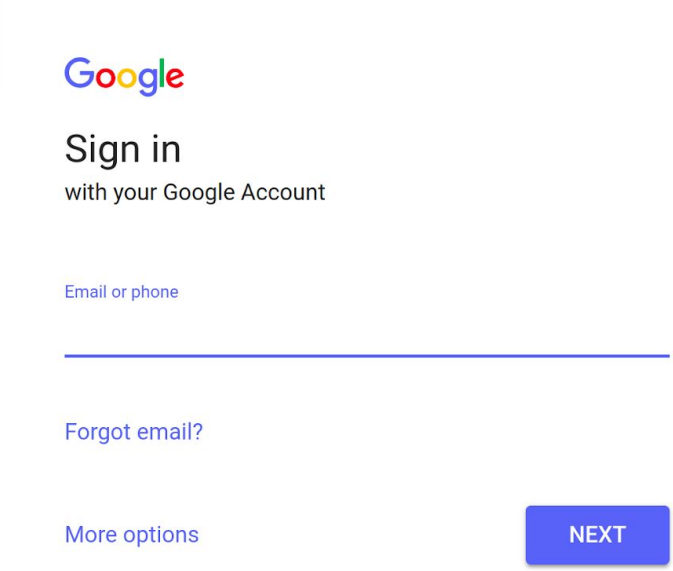


Figure 15:Sign In Screen Mock-up for End User

Sequence diagrams for sign in processes for different user types are given below:

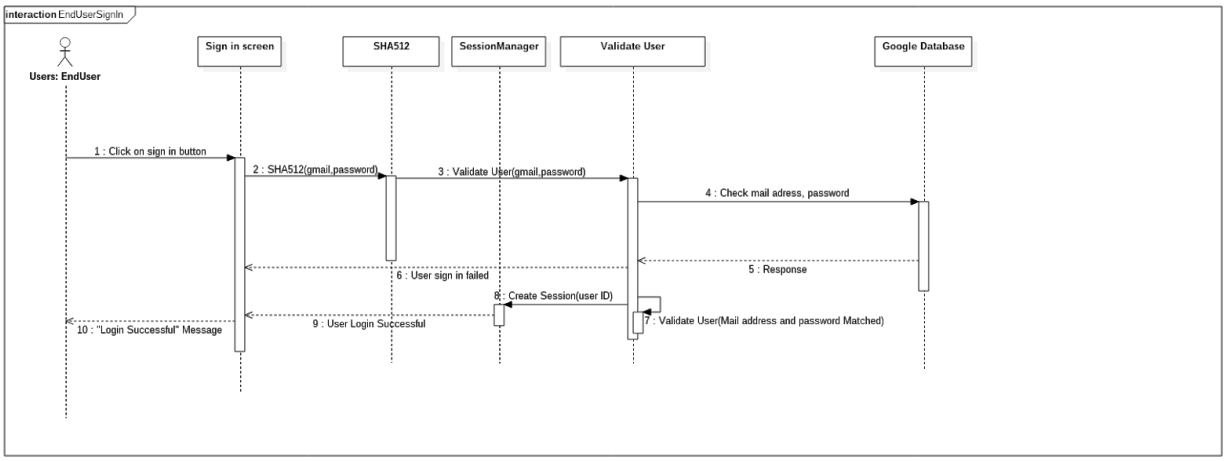


Figure 16:End User Sign In Sequence Diagram

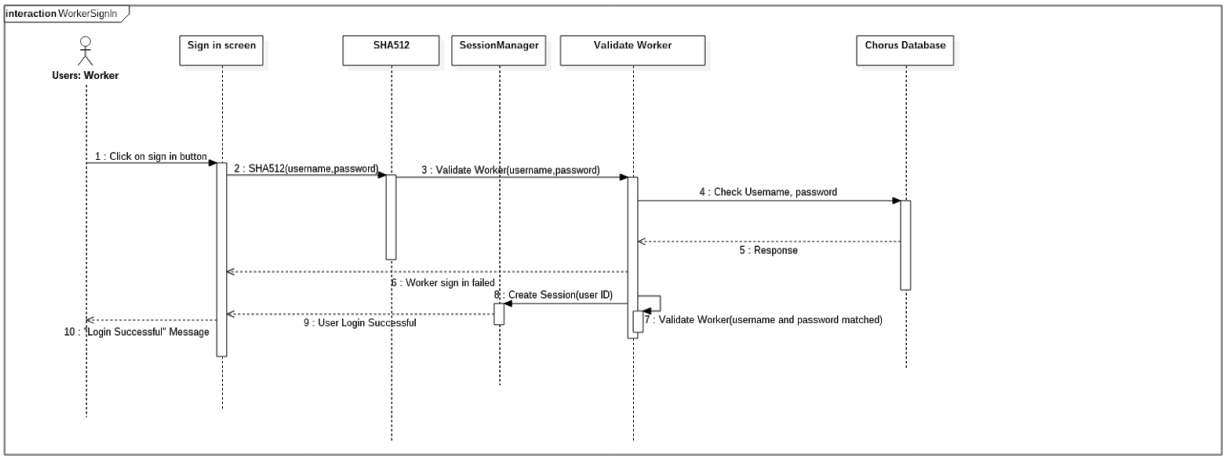


Figure 17:Worker Sign In Sequence Diagram

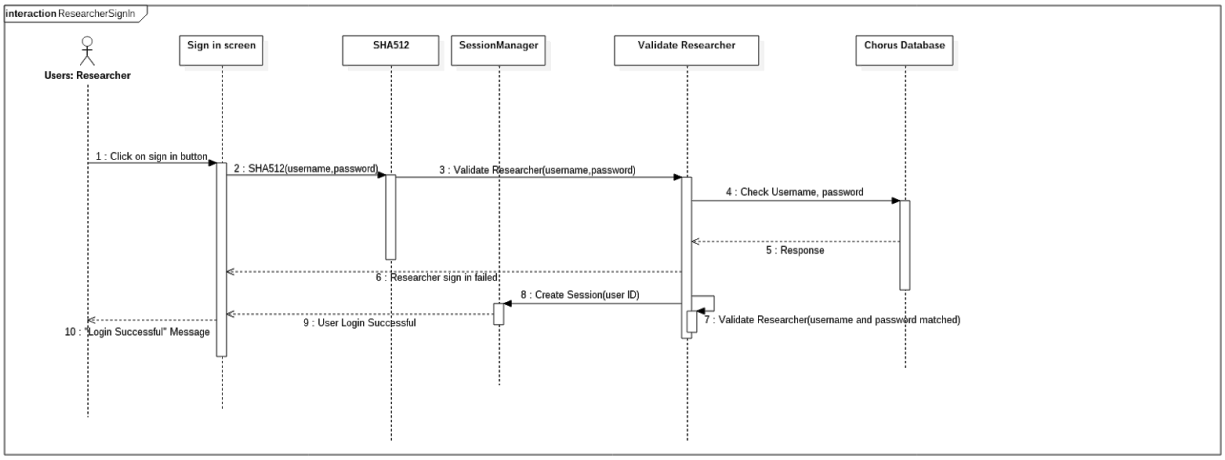


Figure 18:Researcher Sign In Sequence Diagram

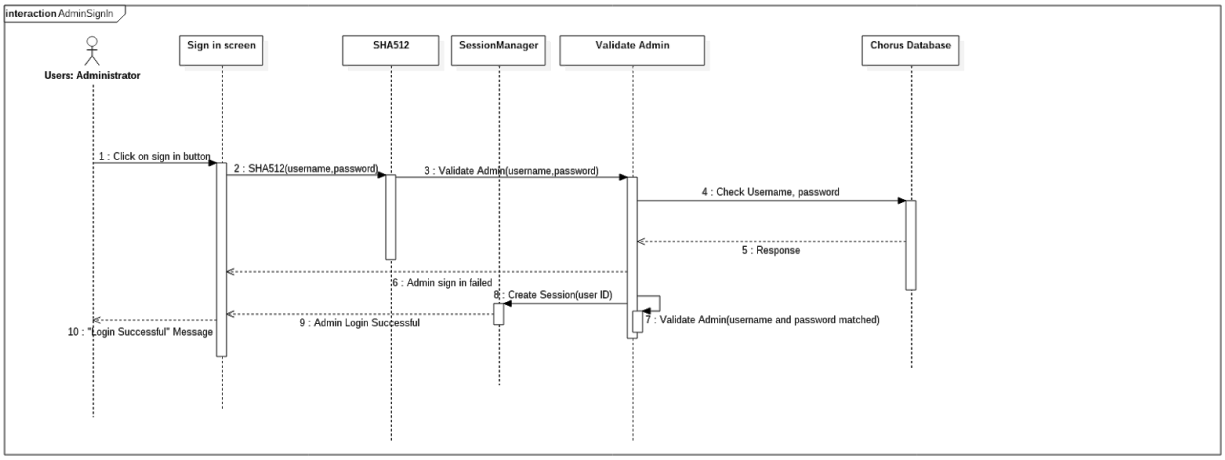


Figure 19:Administrator Sign In Sequence Diagram

Sign up screen for the all users of the system are again different but they all have common features. Below mock-up illustrates the sign up screen for end users, the other users of the system, namely, workers, researchers and administrators are registered to system through similar interfaces which have different options for different users while registering (e.g. education level for workers). Below mock-up for representing sign up screen for end users is followed by sequence diagrams for all users of the system which illustrates process more clearly.

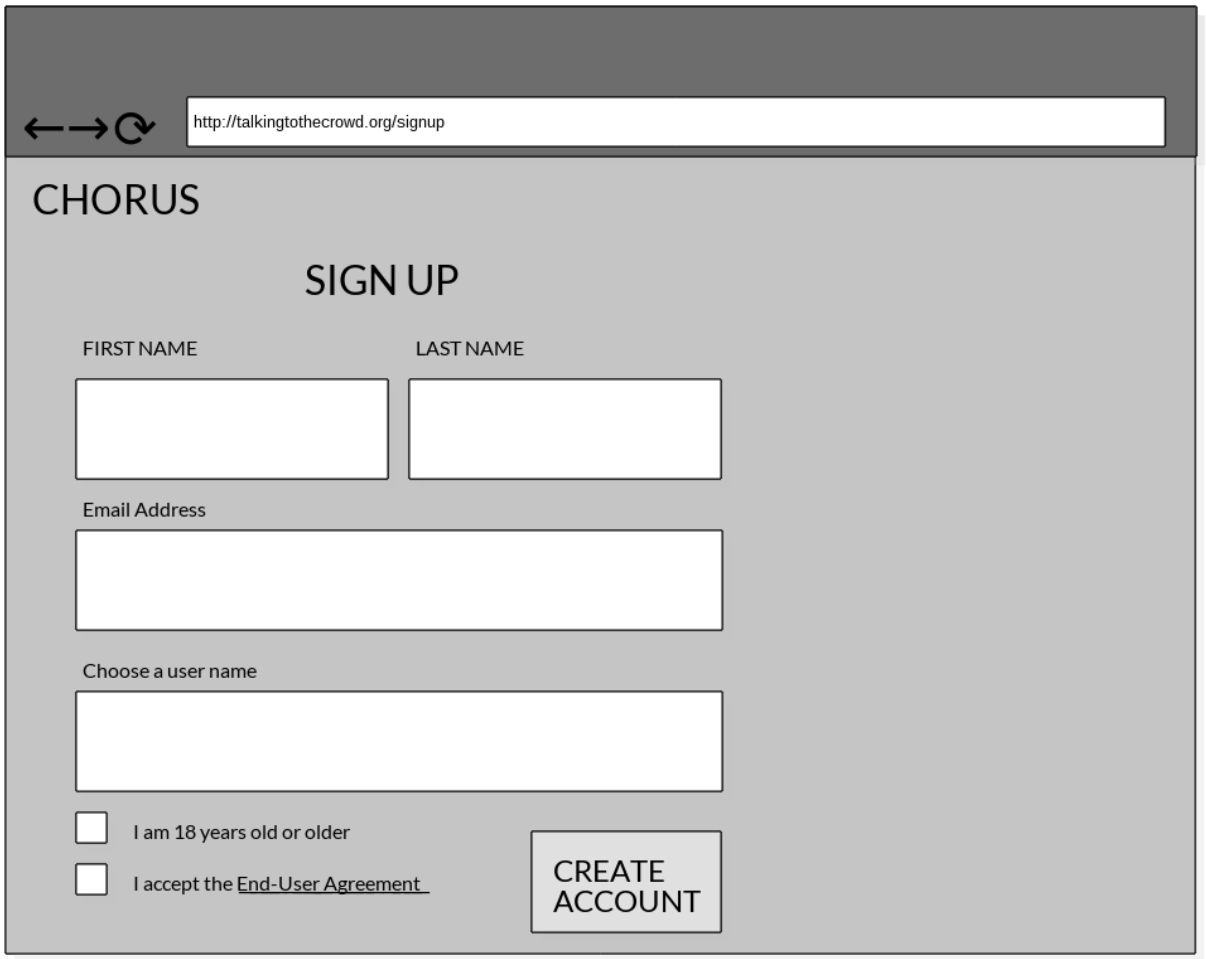


Figure 20:End User Sign Up Mock-up

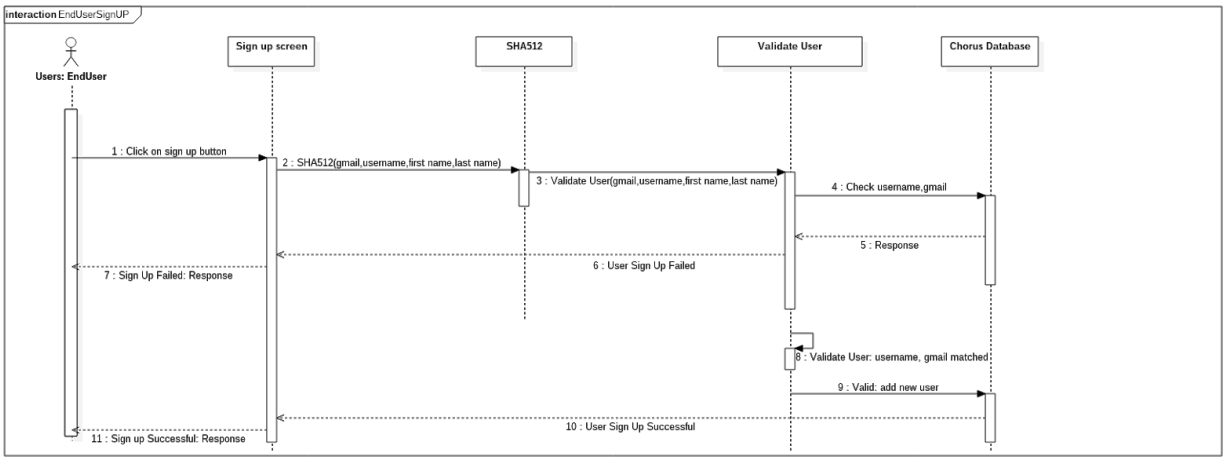


Figure 21:End User Sign Up Sequence Diagram

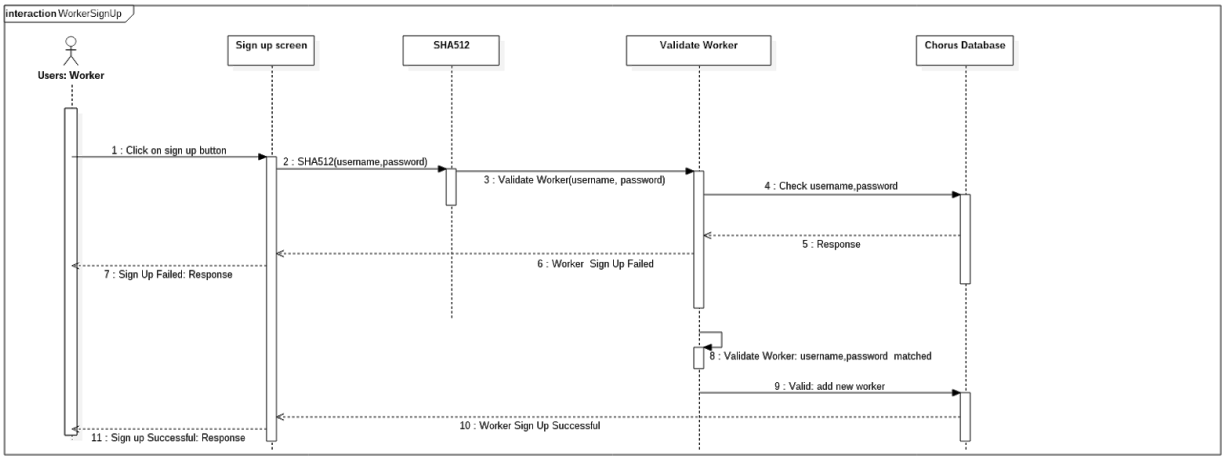


Figure 22:Worker Sign Up Sequence Diagram

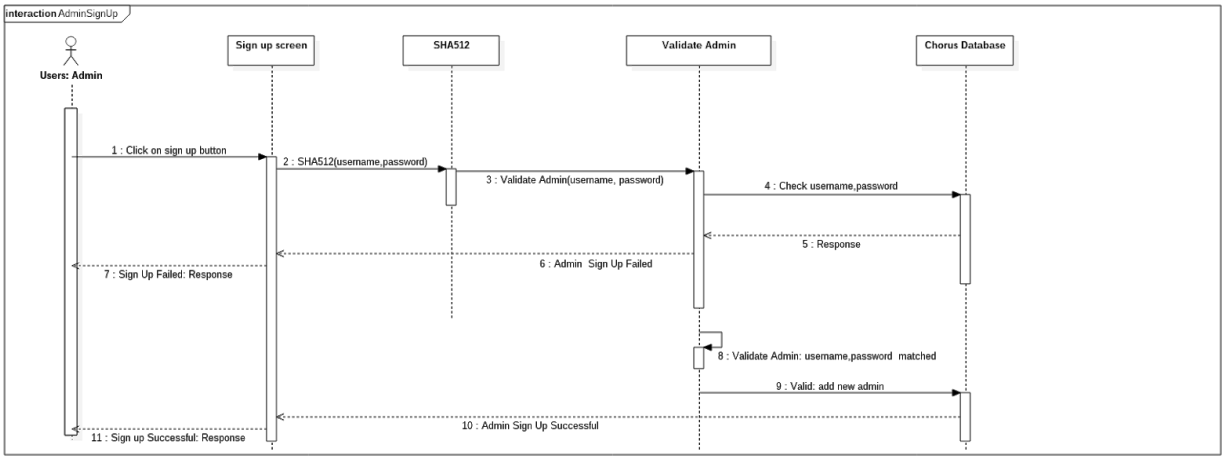


Figure 23:Administrator Sign Up Sequence Diagram

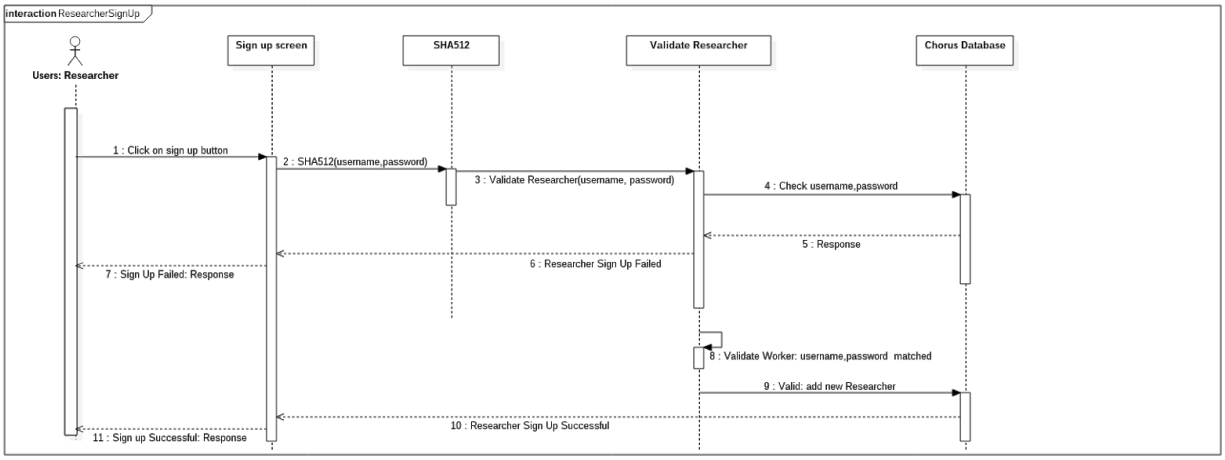


Figure 24:Researcher Sign Up Sequence Diagram

As we can see in the worker interface mock-up screen below, when a worker clicks the “Check Payments” button on the top right of the corner, server requests the amount of money the worker earned from the MTurk, and posts in a pop up screen.

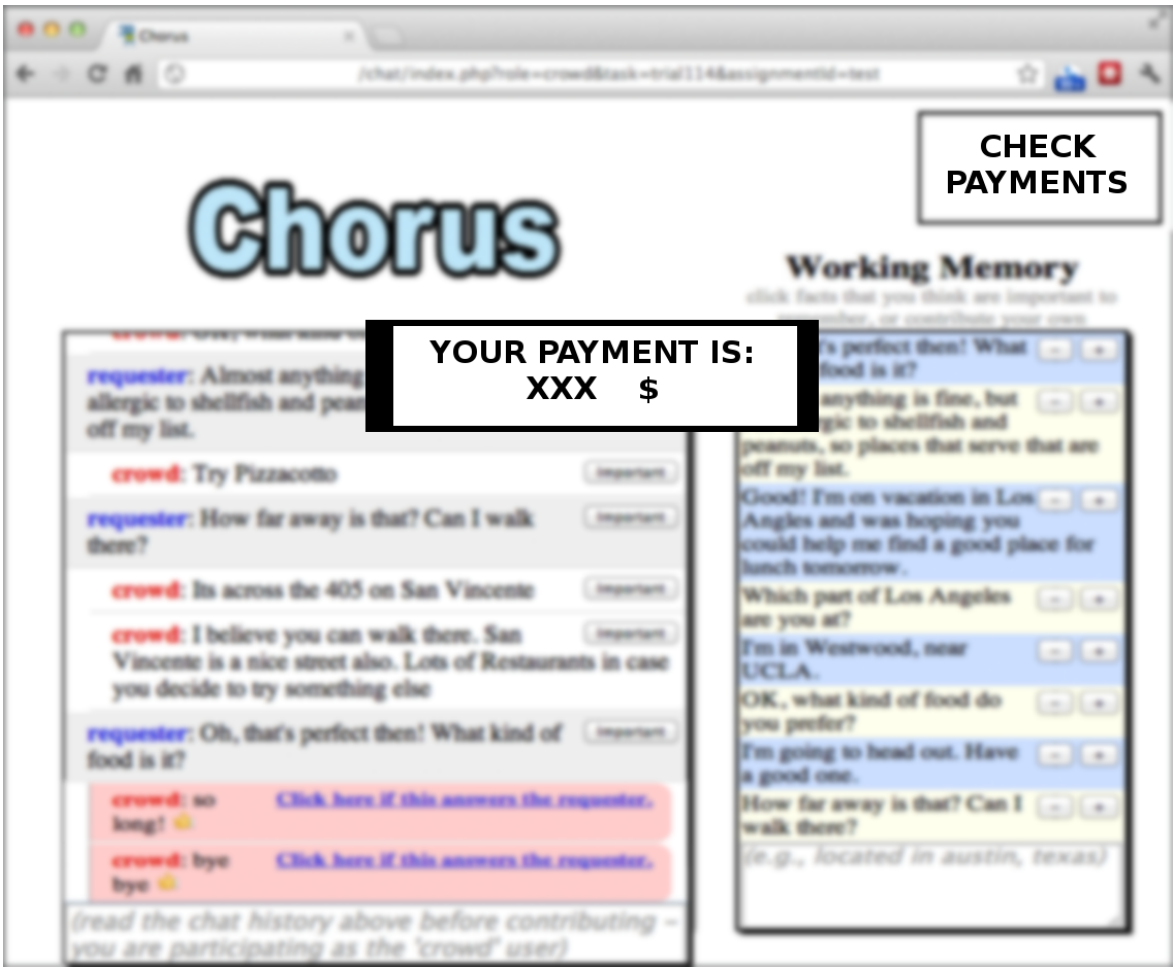


Figure 25:Worker Check Payment Mock-Up Screen

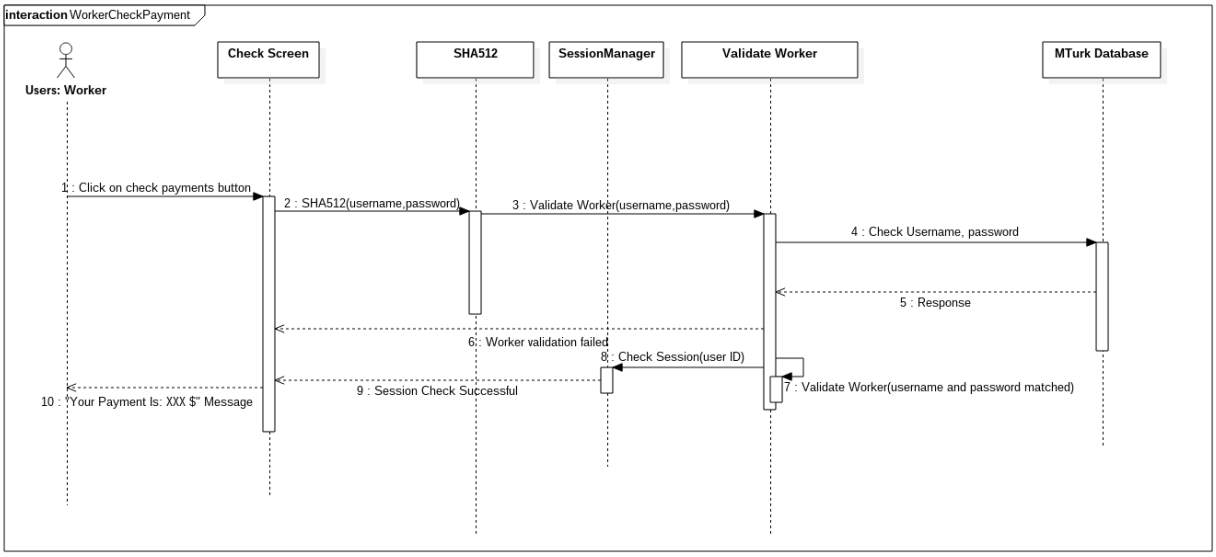


Figure 26:Worker Check Payment Sequence Diagram

4.4.1. Design Rationale

Due to their special concerns and responsibilities in system, users of the system are not using the same interfaces with each other; each user has its own type of interface which allows them to fulfill their own responsibilities. Merging all interfaces in one for all users would be a crucial mistake since the conflicts within system interfaces that is originated by different users would be inevitable. System GUI for end users is handled by external interfaces (Google Hangouts) because system GUI for end users needs simplicity since we consider profile of the end users of system as general public profile and therefore system GUI needs to be as clear and simple as possible. The GUIs for other users of the system are handled by internal interfaces. They are also simple yet adequate. Among the other users, administrators will have the most complex interface and will have a authority to manage and update all system interfaces.