Chorus

A Crowd-Powered Conversational Assistant

Software Design Description

Name	Student Id
Mert Tunç	2099414
Sina Şehlaver	2099729

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Purpose of the System

Chorus is a crowd-powered conversational personal assistance service that aims to be more human like and precise compared to other chat bots or automated personal assistance services by combining human intelligence with computer supervision. System has two sides, one is the end-user who uses Chorus as a personal assistant and the other side of the system which is crowd-workers, responding the requests of the users.

1.2 Scope

- Chorus system will contain crowd-workers that are recruited from Amazon Mechanical Turk and MobileWorks Inc. Those crowd-workers will be responsible for proposing responses to the users.
- Chorus system will contain a graphical interface to allow crowd-workers to browse the chat history, propose responses to the users and vote for the responses.
- Chorus system will contain a sign-up form for new end-users to keep the track of the user statistics.
- Chorus system will contain a database for two different purposes,
    - Database will keep user information for statistical purposes.
    - Database will keep activity track of the crowd-workers. From this data, payments of the Crowd Workers, restrictions and statistical data will be composed.
- Chorus system will have an admin for keeping the integrity and the functionality of the system.
- Chorus system will not contain a graphical user chat interface. Currently, HangoutsBot on Google Hangouts is being used for this purpose.
- Chorus system will not have any interface for crowd-workers to apply directly. Amazon Mechanical Turk or MobileWorks will be handling those applications.

1.3 Stakeholders and their Concerns

The Chorus Project, by its nature is a project that its main concern and purpose is to study and gather data on a crowd powered project. Since it is a research project, classical stakeholder pattern can not be found in Chorus Project, for example Chorus does not have any investors.

Researchers : Researchers are the people whose concerns are to gather information and analyze the information about the Chorus Project. They are mainly interested in the outcomes and the real world performance of the Chorus Project. Studying and analyzing the data gathered from the system might lead to new developments and products.

Users : Users are the people that utilize the offerings of the Chorus Project. Their concerns are to get assisted and informed about any situation that they encounter by chatting with the Chorus System. The main two concerns of users are the response time of the Chorus Bot and the accuracy of the given information. The response of the Chorus System shall be in real time and still be precise and accurate to satisfy their need.

System Developers : These are the team members who are responsible of developing Chorus system. Their concern is having specific and predetermined workflow to work on since without a well defined workflow, the development process would not be easy.

2. References

IEEE standard for information technology--systems design--software design descriptions. (2009). New York, NY: Institute of Electrical and Electronics Engineers.

A Conversational Agent Powered by Crowdsourcing. (n.d.). Retrieved March 18, 2018, from http://talkingtothecrowd.org/

Huang, T., Lasecki, W. S., Azaria, A., & Bigham, J. P. (2016). “Is there anything else I can help you with?”: Challenges in Deploying an On-Demand Crowd-Powered Conversational Agent.

Lasecki, W. S., Wesley, R., Nichols, J., Kulkarni, A., Allen, J. F. & Bigham, J. P. (2013). Chorus: A Crowd-Powered Conversational Assistant.

3. Glossary

Term	**Definition**
Unregistered User	User who has not signed up to the Chorus system.
Registered User	User who has successfully signed up to the Chorus system.
Admin	Person who observes and maintains the integrity of the Chorus system.
Crowd Worker	Volunteers that make money by doing little tasks that are part of a bigger job.
Amazon EC2	A hosting service by Amazon Web Services which provides an instance that Chorus system runs on and a database for the Chorus system.
Amazon Mechanical Turk	A marketplace for Crowd Workers by Amazon.
MobileWorks	A marketplace for Crowd Workers by MobileWorks Inc.
LAMP	It is a common design choice that uses PHP and MySQL with Apache Server running on a Linux distribution.
SATA	Serial Advanced Technology Attachment. An interface for transferring data between a computer and a storage device.
SAS	Serial Attached Small Computer System Interface. An interface for transferring data between a computer and a storage device.
Cookie	Small data which are stored on user’s computer.
CRUD	Abbreviation for Create Read Update Delete, used for database operations.
SSL	Abbreviation for Secure Socket Layer.
TCP	Abbreviation for Transmission Control Protocol.

Table 1: Glossary

4. Architectural Views

4.1 Context View

In this viewpoint, context of the system with all actors are defined in general and detailed viewpoints. In the context diagram, actors and their interaction with the Chorus System will be explained in general terms. Use case diagrams and the detailed explanations of every possible use case of the system will be specified below the context diagram.

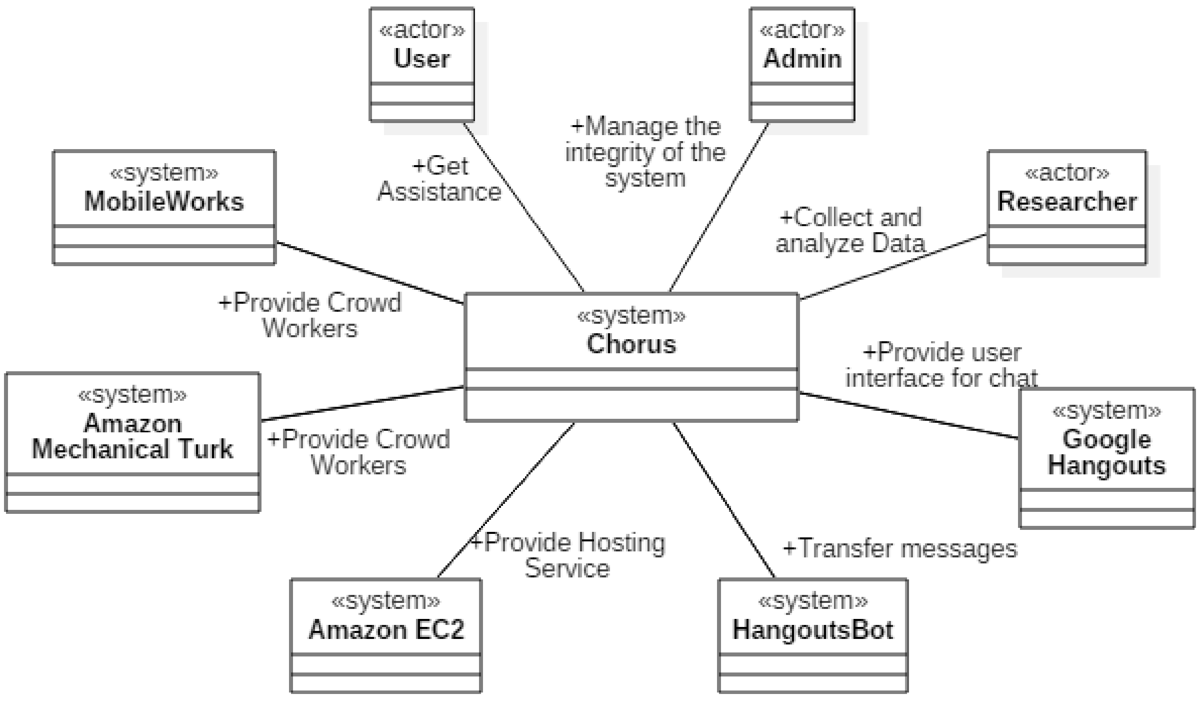


Figure 1: Context Diagram

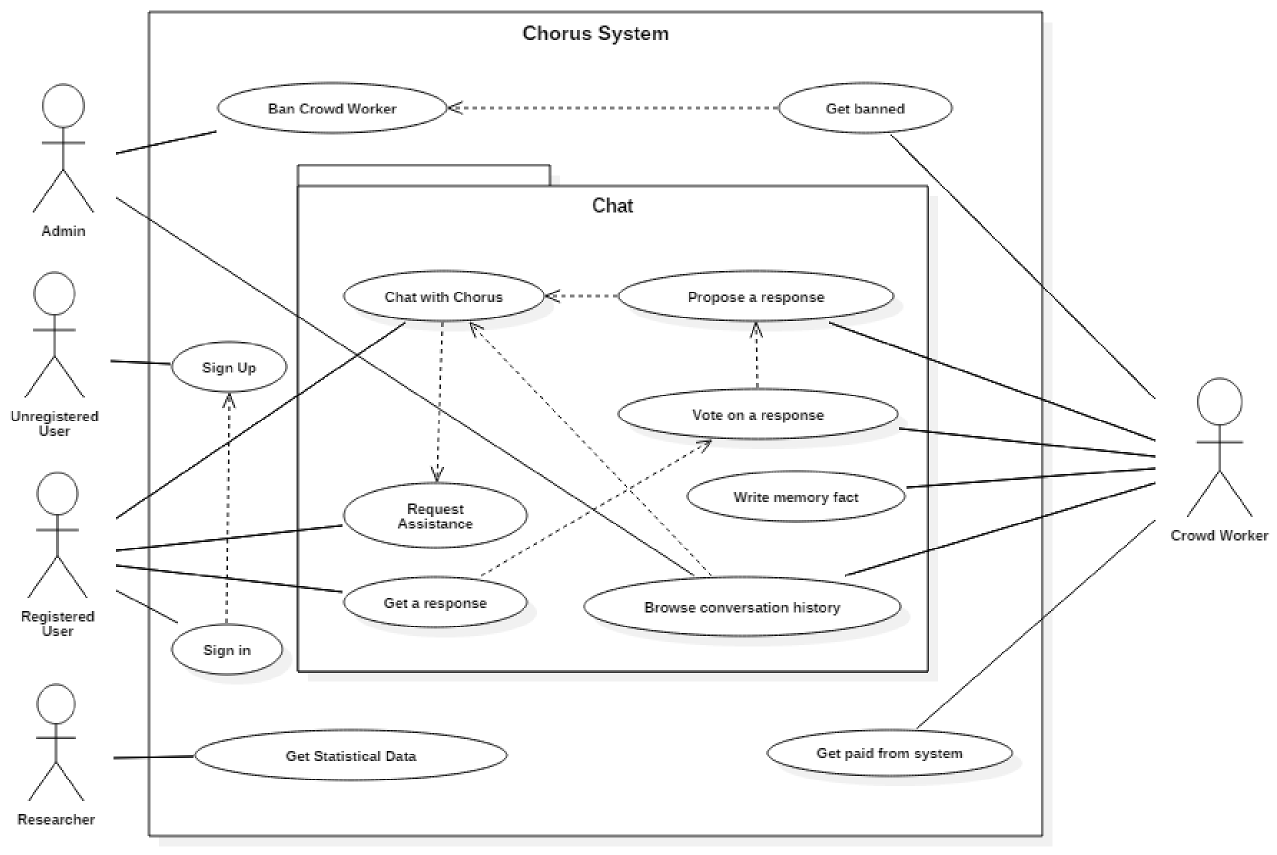


Figure 2: Use Case Diagram

Use case name	Request assistance
Actors	Registered User
Description	Registered User starts a chat with Chorus Bot, requesting assistance
Data	-
Preconditions	Registered User must be logged in to the Chorus system
Stimulus	Registered User clicks the send button

Basic Flow	Step 1 – Registered User enters the text that is to be sent to the Chorus System to the chat textbox Step 2 – Registered User clicks the send button Step 3 – A human intelligence task is created on the worker supplier system
Alternative Flow	-
Exception Flow	-
Postconditions	Crowd Workers are stimulated for replying

Table 2: Use case, Request assistance

Use case name	Chat with Chorus
Actors	Registered User
Description	Registered User chats with Chorus Bot
Data	-
Preconditions	Registered User must be logged in to the Chorus system, There must be an ongoing conversation
Stimulus	Registered User clicks the send button
Basic Flow	Step 1 – Registered User enters the text that is to be sent to the Chorus System to the chat textbox Step 2 – Registered User clicks the send button
Alternative Flow	Step 1 – Registered User selects a picture to be sent to the Chorus System Step 2 – Registered User clicks the send button

Exception Flow	-
Postconditions	Crowd Workers are stimulated for replying

Table 3: Use case, Chat with Chorus

Use case name	Propose a response
Actors	Crowd Worker
Description	Crowd Workers propose an answer for the chat message sent by the registered user.
Data	Memory of the Chorus about the particular registered user
Preconditions	Registered User must be logged in to the Chorus system, there must be a message sent by the registered user.
Stimulus	The Crowd Worker presses the enter button
Basic Flow	Step 1 – The Crowd Worker types the response to be proposed to the chat textbox Step 2 – The Crowd Worker presses the enter button
Alternative Flow	-
Exception Flow	-
Postconditions	-

Table 4: Use case, Propose a response

Use case name	Vote on a response
Actors	Crowd Worker
Description	Crowd Workers vote for proposed suitable responses to the request made by the registered user
Data	Memory of the Chorus about the particular registered user
Preconditions	Registered User must be logged in to the Chorus system, there must be a message sent by the user, there must be at least one response to the message
Stimulus	Crowd Worker presses the upvote button
Basic Flow	Step 1 – The Crowd Worker selects any of the proposed responses Step 2 - The Crowd Worker presses the upvote button
Alternative Flow	-
Exception Flow	-
Postconditions	Chorus system selects the highest voted response provided that it got %40 of the votes, then the id of the message is sent to the HangoutsBot

Table 5: Use case, Vote on a response

Use case name	Write Memory Fact
Actors	Crowd Worker
Description	Crowd Workers take notes on important information about current conversation
Data	-
Preconditions	Registered User must be logged in to the Chorus system, there must be a message sent by the user, there must be at least one response to the message
Stimulus	The Crowd Worker presses the enter button
Basic Flow	Step 1 – The Crowd Worker selects or types a fact about the current conversation in the memory textbox Step 2 – The Crowd Worker presses the enter button
Alternative Flow	-
Exception Flow	-
Postconditions	Chorus saves the fact

Table 6: Use case, Write Memory Fact

Use case name	Browse conversation history
Actors	Crowd Worker, Admin
Description	Crowd Workers can browse the history of the current conversation
Data	-
Preconditions	Registered User must be logged in to the Chorus system, there must be a question asked by the registered user
Stimulus	Actor scrolls the scroll bar
Basic Flow	Step 1 – Crowd Worker browse chat history by scrolling
Alternative Flow	-
Exception Flow	-
Postconditions	-

Table 7: Use case, Browse conversation history

Use case name	Get a response
Actors	Registered User
Description	After the selection of the highest voted proposed response, the selected response is sent to the registered user
Data	-
Preconditions	Registered User must be logged in to the Chorus system, there must be a question asked by the registered user, there must be at least one response to the requested question.

Stimulus	A new response is sent to the HangoutsBot
Basic Flow	Step 1 – HangoutsBot is sent the message id of the highest voted response Step 2 – HangoutsBot sends the corresponding message to the registered user
Alternative Flow	Step 1 – If there are less than 3 Crowd Workers currently active on the conversation, HangoutsBot is sent the message id of the response without any filtering Step 2 – HangoutsBot sends the corresponding message to the registered user
Exception Flow	-
Postconditions	Registered User gets the selected response

Table 8: Use case, Get a response

Use case name	Ban Crowd Worker
Actors	Admin
Description	If any of the Crowd Workers are acting against the integrity of the system, Admin can ban him/her from the Chorus system
Data	Conversation history of the current chat with the registered user
Preconditions	Any of the Crowd Workers should have behaved against the integrity rules of the Chorus System
Stimulus	Admin presses ban button

Basic Flow	Step 1 – System admin encounters any kind of misbehavior that is done by a Crowd Worker Step 2 – The admin selects the Crowd Worker from administration panel and ban it from the Chorus system permanently
Alternative Flow	-
Exception Flow	-
Postconditions	The Crowd Worker gets banned

Table 9: Use case, Ban Crowd Worker

Use case name	Get banned
Actors	Crowd Worker
Description	Misbehaved Crowd Worker is banned from Chorus Environment
Data	-
Preconditions	-
Stimulus	Corresponding Crowd Worker is banned by the admin
Basic Flow	Step 1 – The Crowd Worker gets banned from the Chorus system permanently
Alternative Flow	-
Exception Flow	-
Postconditions	-

Table 10: Use case, Get banned

Use case name	Sign Up
Actors	Unregistered User
Description	Unregistered User signs up to the Chorus system
Data	-
Preconditions	-
Stimulus	User presses sign up button
Basic Flow	Step 1 – Unregistered User opens the Chorus website Step 2 – Unregistered User opens the sign up form and fills the form. Step 3 – Unregistered User submits the form Step 4 – Unregistered User is given rights to login to the Chorus system by an activation mail
Alternative Flow	-
Exception Flow	-
Postconditions	-

Table 11: Use case, Sign Up

Use case name	Sign In
Actors	Registered User
Description	In order to use the Chorus system, users should sign in.
Data	-

Preconditions	User who is trying to sign in should have signed up to the Chorus before.
Stimulus	User presses sign in button
Basic Flow	Step 1 – User tries to authenticate by giving their credentials Step 2 – User successfully signs in, welcomed by a message
Alternative Flow	-
Exception Flow	Step 2 – Given credentials cannot be verified Step 3 – User is informed, sign in fails
Postconditions	-

Table 12: Use case, Sign In

Use case name	Get Paid From System
Actors	Crowd Worker
Description	For their effort, Crowd Workers are paid by Chorus System
Data	The points earned by the Crowd Worker
Preconditions	The worker must have earned enough points to get paid
Stimulus	A conversation ends
Basic Flow	Step 1 – Chorus system automatically calculates the amount of the payment from the points the worker earned Step 2 – Chorus system reports the calculated amount to the Crowd Worker's supplier firm
Alternative Flow	-

Exception Flow	-
Postconditions	-

Table 13: Use case, Get Paid From System

Use case name	Get Statistical Data
Actors	Researcher
Description	Researcher gathers statistical data about crowd sourcing and user base
Data	All applicable data in the database
Preconditions	-
Stimulus	Researcher presses the send query button
Basic Flow	Step 1 – Researcher connects to the Chorus system database Step 2 – Researcher writes a query text to the database management system Step 3 – Researcher gets the results of the query
Alternative Flow	-
Exception Flow	Step 1 – Researcher connects to the Chorus system database Step 2 – Researcher writes a query text to the database management system Step 3 – An error occurs while executing the query Step 4 – An error message is shown to the researcher
Postconditions	-

Table 14: Use case, Get Statistical Data

4.2 Composition View

This viewpoint will show the components of the system from a top-level point. Further information about the components is explained in their corresponding sections.

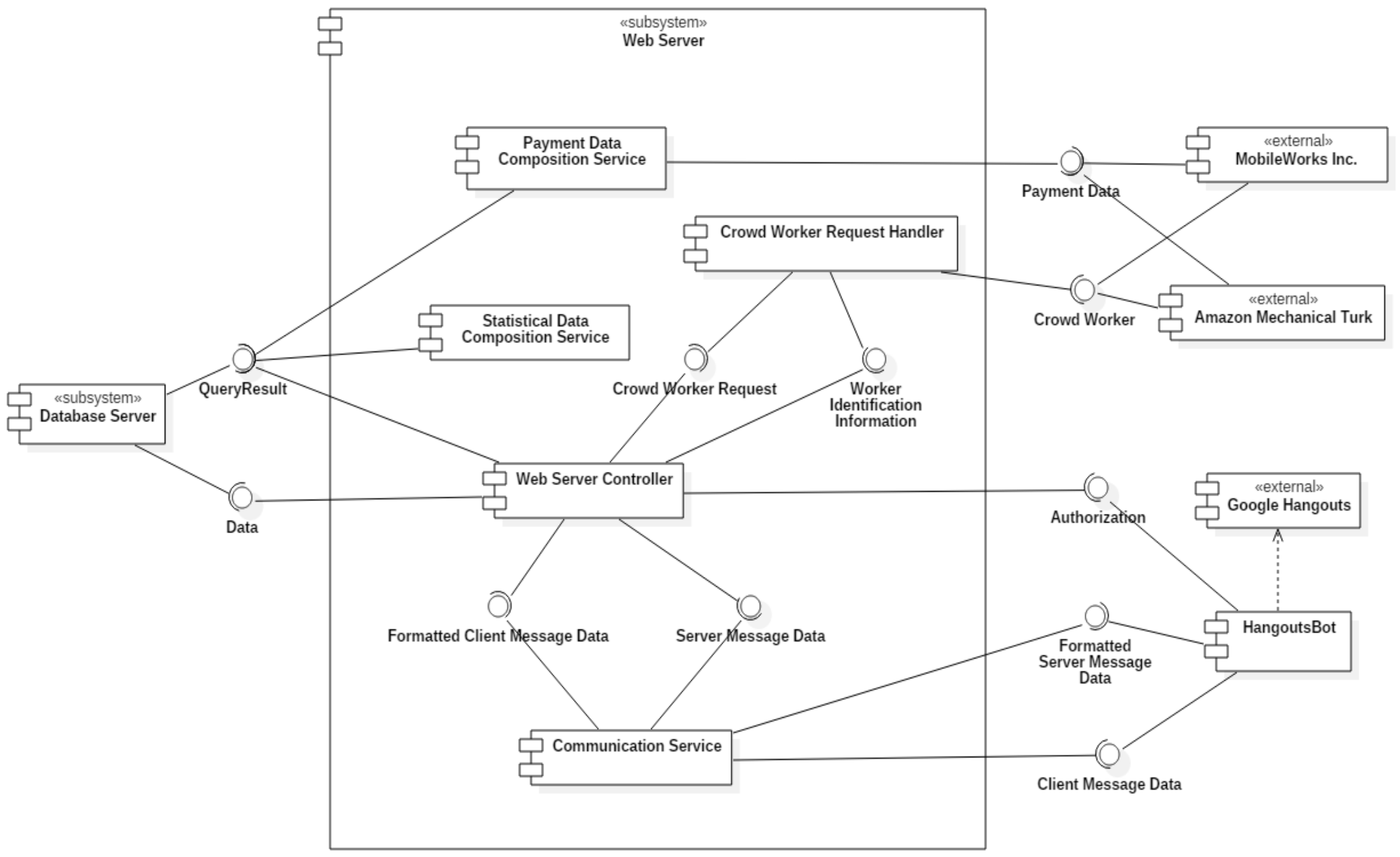


Figure 3: Component Diagram

Design Rationale:

- Web Server Controller Component will be in charge of handling user's interactions with the Web Server Component. It is responsible for sending and receiving chat messages with HangoutsBot through Communication Service Component; authorizing and identifying both users and crowd workers; storing user, worker and chat information in Database Server Component and finally requesting and acquiring crowd workers from crowd worker supplier firms via Crowd Worker Request Handler Component.

- Payment Data Composition Service Component is required in order to convert raw data coming from database (QueryResult) to Payment Data. Payment Data Composition Service Component will query the Database Server Component in order to deliver processed Payment Data to corresponding Crowd Worker supplier firm.
- Database Server Component will store the data coming from Web Server Controller Component. These data include; chat, user and worker information along with system logs.
- Communication Service Component is responsible for the communication between Web Server Component and the HangoutsBot Component. The Communication Service Component converts the message that is sent by the Web Server Controller Component to the communication protocol that HangoutsBot Component uses and sends the formatted data to the HangoutsBot Component and vice versa.
- Crowd Worker Request Handler Component will handle the requests for the acquisition of crowd workers. When received a request, the handler component will acquire, required number of crowd workers from the crowd worker supplier firms.
- Amazon Mechanical Turk and MobileWorks Inc. will provide crowd workers when requested by the Crowd Worker Request Handler Component. When an asynchronous payment and the information of the payment is received from the Payment Service Component, each of the crowd worker supplier will handle the received Payment.
- HangoutsBot is a subsystem of the Google Hangouts that is the medium between Users and the Web Server Component. It will send and receive the chat messages to and from both Users and Web Server Component. Google Hangouts is not a part of the Chorus system and there is no direct communication between Chorus and Google Hangouts.
- Statistical Data Composition Service Component will be used to analyze, visualize, filter, convert and export the raw data coming from the Database Server Component.
- Identification and authorization of the users will be handled in two different ways, first, every user that connect to the Google Hangouts should have to have a Google Account. With this account, they will authenticate to use the Google Hangouts to communicate with the Chorus. Second authorization is the one that Chorus does. At this layer of authorization, Chorus System will be given the identification of the user by the Google Hangouts via HangoutsBot to authorize the user.

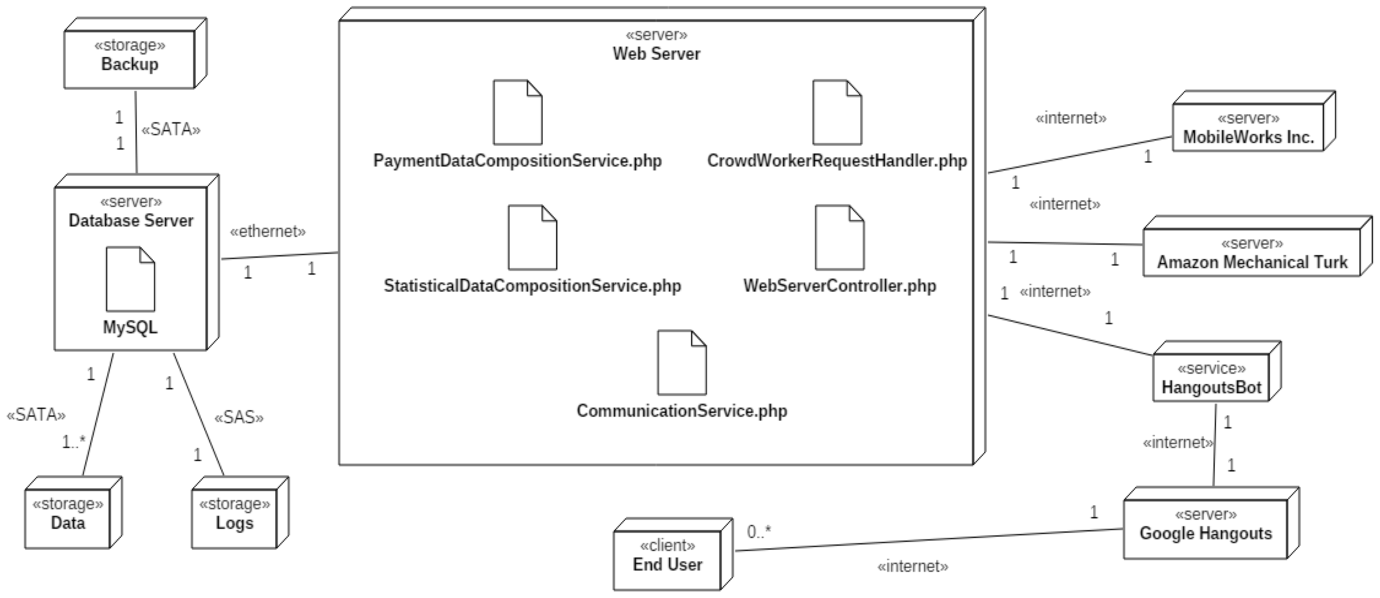


Figure 4: Deployment Diagram

Design Rationale:

- Since we use LAMP architecture, web server is developed and running with PHP and database is managed with MySQL.
- End Users are users who use Chorus to get assistance, they will communicate with the Web Server through Google Hangouts. Google Hangouts will send and receive the communication data to and from the Web Server via HangoutsBot.
- There are three separate database storages;
    - Data storage is for storing user, crowd worker and chat data
    - Logs storage is for storing system logs
    - Backup storage is for the backup of the Data storage
- We used SAS for storing logs since it provides faster connection than SATA.
- Only the external systems use internet connection because of the drawbacks in terms of data transfer rate, latency and other connection related issues and security.
- All internet based connections will use encrypted communication protocols for increased security (SSL,HTTPS).

4.3 Information View

In this view, the organization and the relations of the data that will be stored with the operations of the system that create, use, modify and delete the data will be specified. Also, the effects of the system operations on the data will be examined in terms of their effect type; create, read, update and delete.

4.3.1 Interfaces

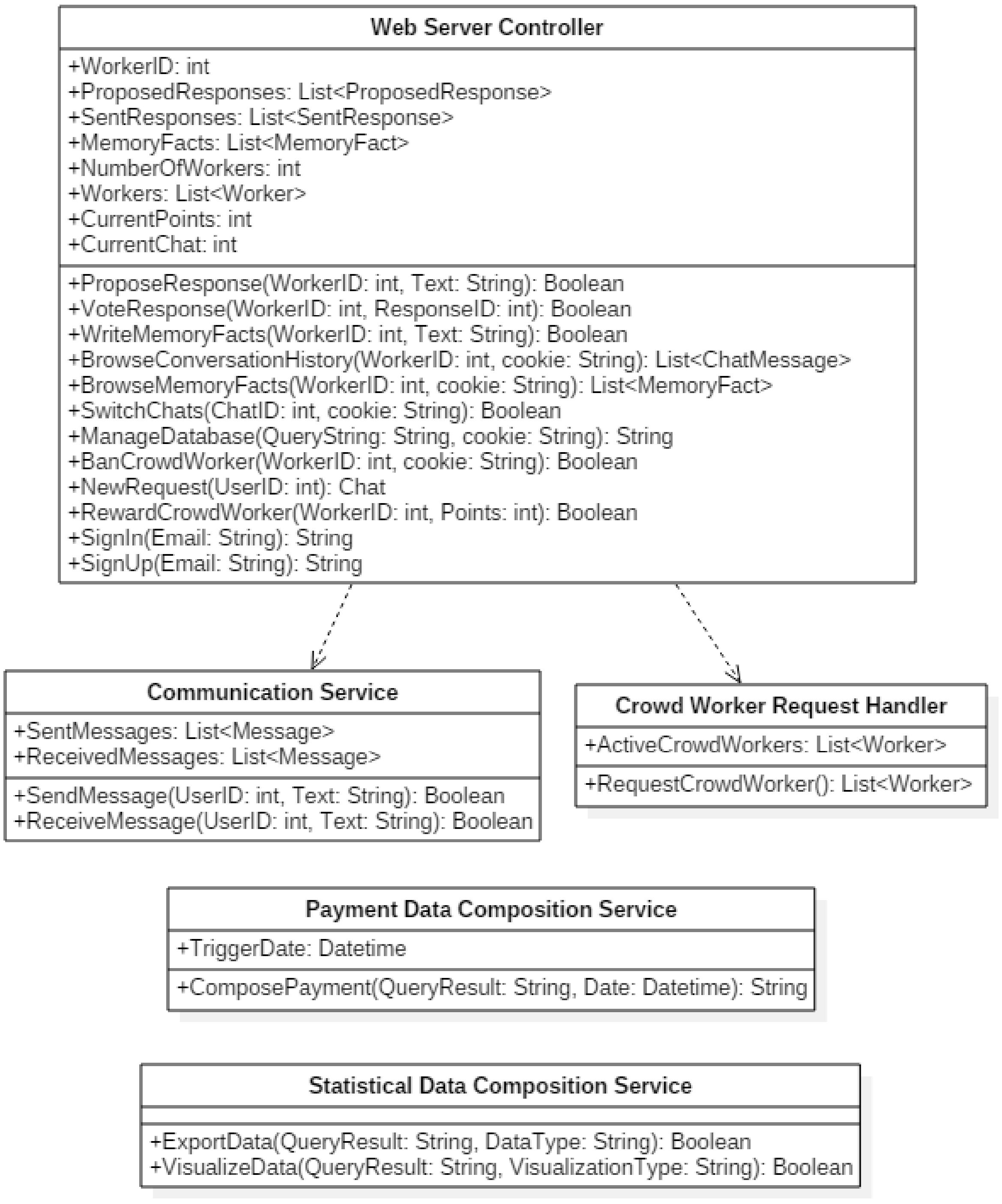

Web Server Controller
+WorkerID: int
+ProposedResponses: List<ProposedResponse>
+SentResponses: List<SentResponse>
+MemoryFacts: List<MemoryFact>
+NumberOfWorkers: int
+Workers: List<Worker>
+CurrentPoints: int
+CurrentChat: int
+ProposeResponse(WorkerID: int, Text: String): Boolean
+VoteResponse(WorkerID: int, ResponseID: int): Boolean
+WriteMemoryFacts(WorkerID: int, Text: String): Boolean
+BrowseConversationHistory(WorkerID: int, cookie: String): List<ChatMessage>
+BrowseMemoryFacts(WorkerID: int, cookie: String): List<MemoryFact>
+SwitchChats(ChatID: int, cookie: String): Boolean
+ManageDatabase(QueryString: String, cookie: String): String
+BanCrowdWorker(WorkerID: int, cookie: String): Boolean
+NewRequest(UserID: int): Chat
+RewardCrowdWorker(WorkerID: int, Points: int): Boolean
+SignIn(Email: String): String
+SignUp(Email: String): String
Communication Service
+SentMessages: List<Message>
+ReceivedMessages: List<Message>
+SendMessage(UserID: int, Text: String): Boolean
+ReceiveMessage(UserID: int, Text: String): Boolean
Crowd Worker Request Handler
+ActiveCrowdWorkers: List<Worker>
+RequestCrowdWorker(): List<Worker>
Payment Data Composition Service
+TriggerDate: Datetime
+ComposePayment(QueryResult: String, Date: Datetime): String
Statistical Data Composition Service
+ExportData(QueryResult: String, DataType: String): Boolean
+VisualizeData(QueryResult: String, VisualizationType: String): Boolean

Figure 5: Interface Class Diagram

Operation	**Description**
ProposeResponse	Given response text is added to the Responses table for the current Chat
VoteResponse	The VoteCount of the Response with the given ResponseID is increased
WriteMemoryFacts	After this operation, the written text is saved to the database as a MemoryFact.
BrowseConversationHistory	The Chat history of the requested Chat is shown to the Crowd Worker.
BrowseMemoryFacts	The Memory Facts of the current Chat is shown to the Crowd Worker.
BanCrowdWorker	If some of the Crowd Workers are acting against the integrity of the system, the Admin of the system can ban them from the Chorus system with this operation.
ManageDatabase	An Admin can access database by manually entering the queries to the DBMS directly using this operation.
SwitchChats	This operation allows the Admin of the system to switch current chat.
SendMessage	This operation sends the given message to the HangoutsBot using the proper communication protocol.
ReceiveMessage	This operation receives sent messages to the system from HangoutsBot and saves the retrieved information to the database.
NewRequest	This is the operation that initializes a new chat.
RewardCrowdWorker	This is a subroutine that is to be called after each demanding operation for rewarding the Crowd Workers for their participation in the assistance process.
RequestCrowdWorker	When a new conversation starts, this operation is called to collect the required number of Crowd Workers.
SignIn	This operation allows the system to authorize the connected user.
SignUp	This operation allows the users to register themselves to the system.

ComposePayment	This operation collects the raw payment data from the database and composes the data which is to be reported to the Crowd Worker supplier firms.
ExportData	The data that is queried by the Researcher or the Admin is exported in the requested data type.
VisualizeData	The data that is queried by the Researcher or the Admin is visualized in the requested visualization type.

Table 15: Operation descriptions

Operation	**Inputs**	**Outputs**	**Exceptions**
ProposeResponse	- WorkerID - ChatID - Text - Cookie	True if operation was successful, otherwise false	- Session id is not valid or expired - Crowd Worker with WorkerID is banned - Database connection error occurs
VoteResponse	- WorkerID - ResponseID - Cookie	True if operation was successful, otherwise false	- Session id is not valid or expired - Crowd Worker with WorkerID is banned - Database connection error occurs
WriteMemoryFacts	- WorkerID - Text - Cookie	True if operation was successful, otherwise false	- Session id is not valid or expired - Crowd Worker with WorkerID is banned - Database connection error occurs

BrowseConversationHistory	- ChatID - Cookie	Whole chat history with the given ChatID	- Session id is not valid or expired - Crowd Worker with WorkerID is banned - Database connection error occurs
BrowseMemoryFacts	- ChatID - Cookie	List of memory facts saved for the given ChatID	- Session id is not valid or expired - Crowd Worker with WorkerID is banned - Database connection error occurs
BanCrowdWorker	- WorkerID - Cookie	True if operation was successful, otherwise false	- Session id is not valid or expired - Database connection error occurs
ManageDatabase	- QueryString - Cookie	Result of the query	- Session id is not valid or expired - Database connection error occurs
SwitchChats	- ChatID - Cookie	True if operation was successful, otherwise false	- Session id is not valid or expired - Database connection error occurs
SendMessage	- UserID - ChatID - Text	True if operation was successful, otherwise false	- Connection error with the receiver occurs - Database connection error occurs
ReceiveMessage	- UserID - ChatID - Text	True if operation was successful, otherwise false	- Database connection error occurs
NewRequest	- UserID	Newly created Chat	- Database connection error occurs

RewardCrowdWorker	- WorkerID - Points	True if operation was successful, otherwise false	- Database connection error occurs
RequestCrowdWorker		List of acquired Crowd Workers	- Connection error with the Crowd Worker supplier system occurs
SignIn	- Email	Session id given by the system as a cookie	- Given Email is not recognized - Database connection error occurs
SignUp	- Email	Session id given by the system as a cookie	- Given Email is invalid - Database connection error occurs
ComposePayment		Payment Data formatted as string	- Database connection error occurs - Connection error with the Crowd Worker supplier system occurs
ExportData	- QueryResult - DataType	True if operation was successful, otherwise false	- Database connection error occurs
VisualizeData	- QueryResult - VisualizationType	True if operation was successful, otherwise false	- Database connection error occurs

Table 16: Operation design

Design Rationale:

- Web Server Controller is the main controller that is responsible for responding user interactions on the Web Server Interface, namely the Chorus Website and the User Interfaces provided to the Crowd Workers, Researchers and the Admin.
- Authentication for the operations that need it, will be established through the session id cookie input that each operation is given.
- ComposePayment operation will run on every TriggerDate date.
- ExportData operation will support the data types of ".xls, .xlsx, .csv, .m".
- Web Server Controller selects the highest voted response and checks whether the proposed response got %40 of the total votes for that chat. If so the response is sent to the HangoutsBot with the SendMessage operation.
- ReceiveMessage operation will be an asynchronous operation.
- When a message is received from a user that is not having an active chat, NewRequest operation will be called by the Web Server Controller in order to initialize a new chat.

4.3.2 Database Operations

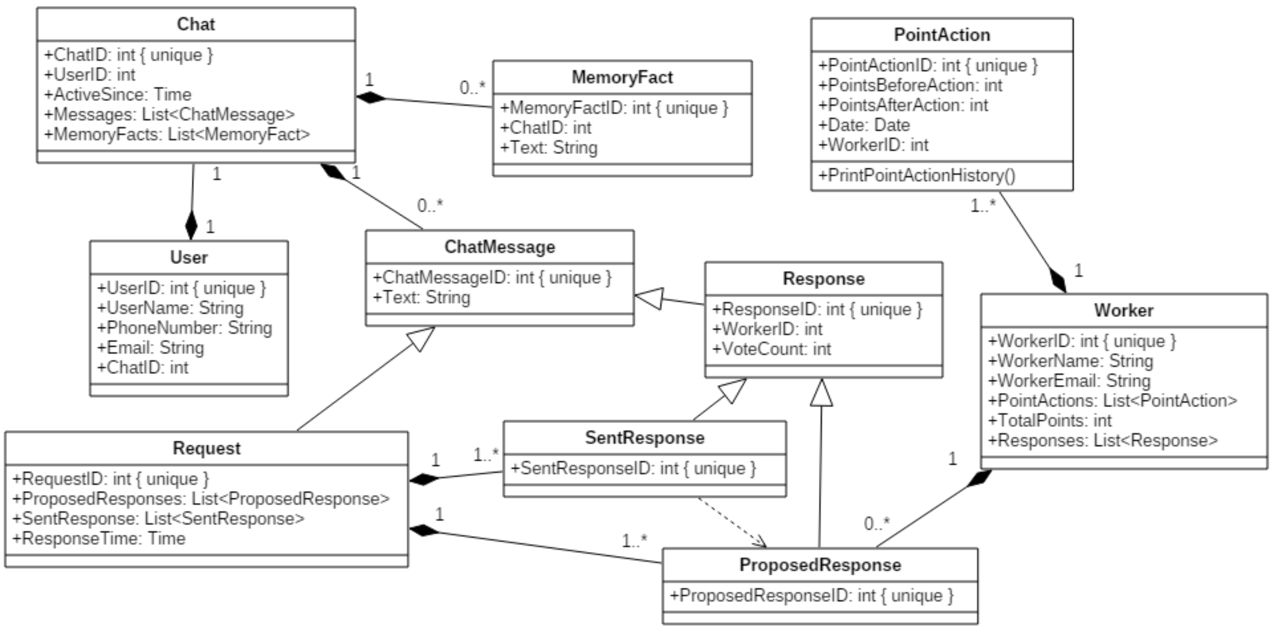


Figure 6: Database Class Diagram

Operation	**CRUD Operations**
ProposeResponse	Create : Response Read : Update : Request, Worker Delete :
VoteResponse	Create : Read : Response Update : Response Delete :
WriteMemoryFacts	Create : MemoryFact Read : Update : Delete :
BrowseConversationHistory	Create : Read : Chat, ChatMessage Update : Delete :
BrowseMemoryFacts	Create : Read : Chat, MemoryFact Update : Delete :
BanCrowdWorker	Create : Read : Worker Update : Worker Delete :
ManageDatabase	Create : Chat, User, ChatMessage, Response, Request, Worker, PointAction, MemoryFact Read : Chat, User, ChatMessage, Response, Request, Worker, PointAction, MemoryFact Update : Chat, User, ChatMessage, Response, Request, Worker, PointAction, MemoryFact Delete : Chat, User, ChatMessage, Response, Request, Worker, PointAction, MemoryFact
SwitchChats	Create : Read : Chat Update : Delete :
SendMessage	Create : Read : Response Update : Chat, Request Delete :
ReceiveMessage	Create : Request Read : Update : Chat Delete :

NewRequest	Create : Chat, Request Read : Update : Delete :
RewardCrowdWorker	Create : PointAction Read : Worker Update : Worker Delete :
RequestCrowdWorker	Create : Worker Read : Update : Chat Delete :
SignIn	Create : Read : User Update : Delete :
SignUp	Create : User Read : Update : Delete :
ComposePayment	Create : Read : PointAction, Worker Update : Delete :
ExportData	Create : Read : Chat, MemoryFact, User, ChatMessage, Request, Response, Worker, PointAction Update : Delete :
VisualizeData	Create : Read : Chat, MemoryFact, User, ChatMessage, Request, Response, Worker, PointAction Update : Delete :

Table 17: CRUD Operations

Design Rationale:

- Every ChatMessage Entry is either a Request entry or a Response entry.
- Every entry that is inserted to the database will be kept for future investigation, i.e. research.
- MySQL will be used for database management.
- After each Crowd Worker operation that needs to be rewarded is completed, RewardCrowdWorker function will be called.

4.4 Interface View

In this view, the internal interfaces between the components of the system and the external interfaces between the Chorus System and the other systems will be specified in detail.

4.4.1 Internal Interfaces

The Interface between the Database Server and the Web Server Controller:

Web Server Controller will be responsible for storing the needed data in the Database Server for preserving the integrity of the system. Web Server Controller will query the database by sending a query string in SQL. When database server receives the query it will execute the query string in MySQL. In case of a MySQL error, database server component will report the error to the web server controller component.

Design Rationale:

- Database Server component will be storing past and present user, worker, chat and payment information together.
- No data will be deleted in order to be saved for research purposes, thus SQL queries will consist of combinations of insert, update and select operations.
- Web Server Controller will be responsible for composing the corresponding SQL queries for each operation.

The Interface between the Communication Service and the Web Service Controller:

Web Server Controller will use the Communication Service component in order to communicate with the HangoutsBot. When the highest voted response is selected to be sent to the user, Web Server Controller will provide the response as Server Message Data to the Communication Service so that it can be formatted and send to the HangoutsBot. Additionally, the messages received by the Communication Service is formatted and provided to the Web Server Controller so that the reverse communication is achieved.

Design Rationale:

- Communication Service component is necessary for the modularity of the service that is being used in order to chat with the users. In our case it is HangoutsBot, but in case of a change in the chat provider, the presence of the Communication Service component will ease the adaptation process.

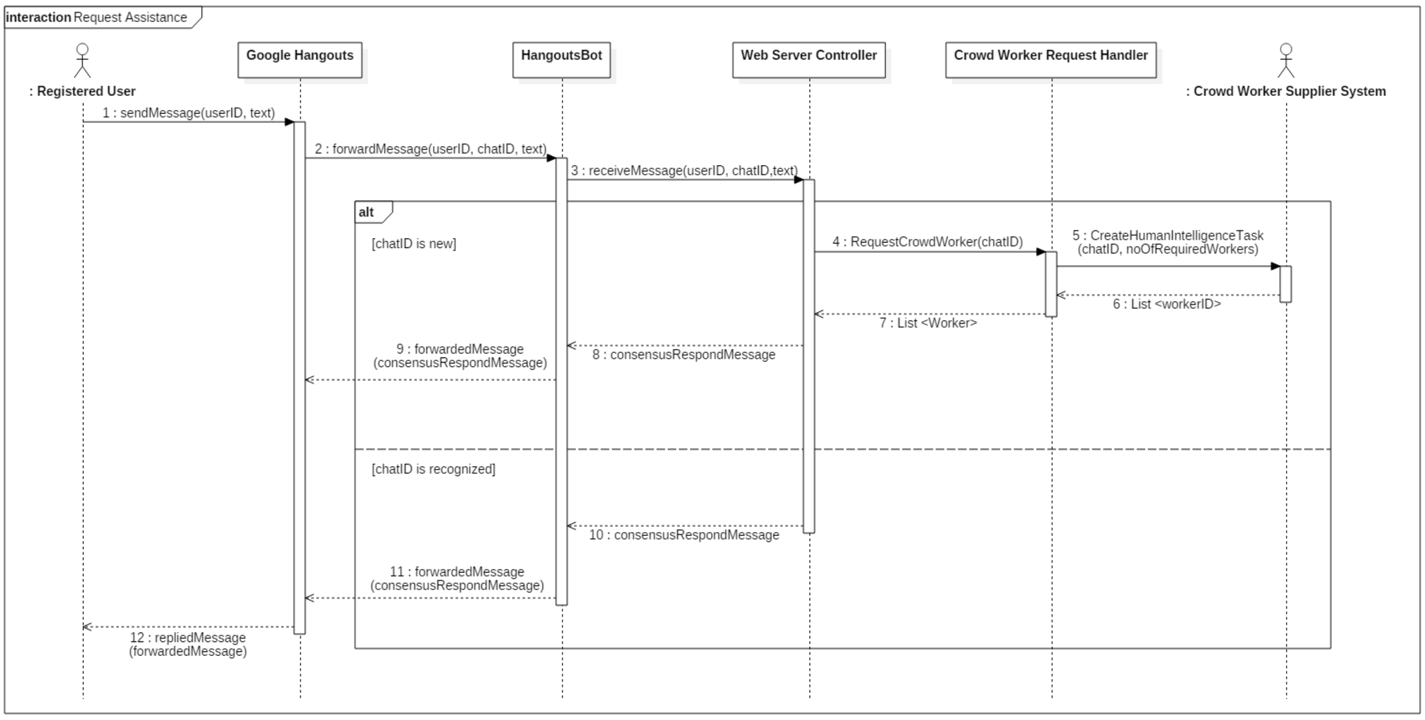

interaction Request Assistance
: Registered User
Google Hangouts
HangoutsBot
Web Server Controller
Crowd Worker Request Handler
: Crowd Worker Supplier System
1 : sendMessage(userID, text)
2 : forwardMessage(userID, chatID, text)
3 : receiveMessage(userID, chatID,text)
alt
[chatID is new]
4 : RequestCrowdWorker(chatID)
5 : CreateHumanIntelligenceTask (chatID, noOfRequiredWorkers)
6 : List <workerID>
7 : List <Worker>
8 : consensusRespondMessage
9 : forwardedMessage (consensusRespondMessage)
[chatID is recognized]
10 : consensusRespondMessage
11 : forwardedMessage (consensusRespondMessage)
12 : repliedMessage (forwardedMessage)

Figure 7: Request assistance sequence diagram showing the interface between Web Server Controller and Crowd Worker Request Handler components of the system

The Interface between the Crowd Worker Request Handler and the Web Server Controller:

When a new conversation request is present, Web Server Controller will request new crowd workers to attend to the request from Crowd Worker Request Handler. Crowd Worker Request Handler will collect the required number of workers and provide their information to the Web Server Controller, allowing Web Server Controller to associate those workers with the corresponding new conversation.

Design Rationale:

- Crowd Worker Request Handler will be the sole component which handles the crowd worker requests. Having a modular structure for the crowd worker requests, will ease up the adaptation of the changes in the supplier firms. In the case of another crowd worker supplier firm being added to the supplier firms that participate in providing crowd workers to the Chorus System, implementation of the conversion of crowd worker information format of the newly added firm to the Chorus format will suffice for adaptation.
- Collected crowd worker information will be sent as a list of crowd worker objects, so that it can be read by the Web Server Controller component.

The Interface between the Statistical Data Composition Service and the Database Server:

Statistical Data Composition Service is responsible for manipulating and searching data. Statistical Data Composition Service will compose the query that executes the operation needed for the retrieval of the requested data from the Researcher Interface and sends the query to the Database Server component. Result of the query will be sent to the Researcher Interface.

Design Rationale:

- In the case of a MySQL error due to the sent query, error will be redirected to the Researcher Interface.
- The results of the queries will be sent to the Researcher Interface in JSON format.

The Interface between the Payment Data Composition Service and the Database Server:

Payment Data Composition Service is responsible for composing payment data of the crowd workers. Payment Data Composition Service will send a query to the Database Server component to gather the point information of all crowd workers in the system. Gathered data will be converted to payment data to be sent to the corresponding crowd worker supplier firms.

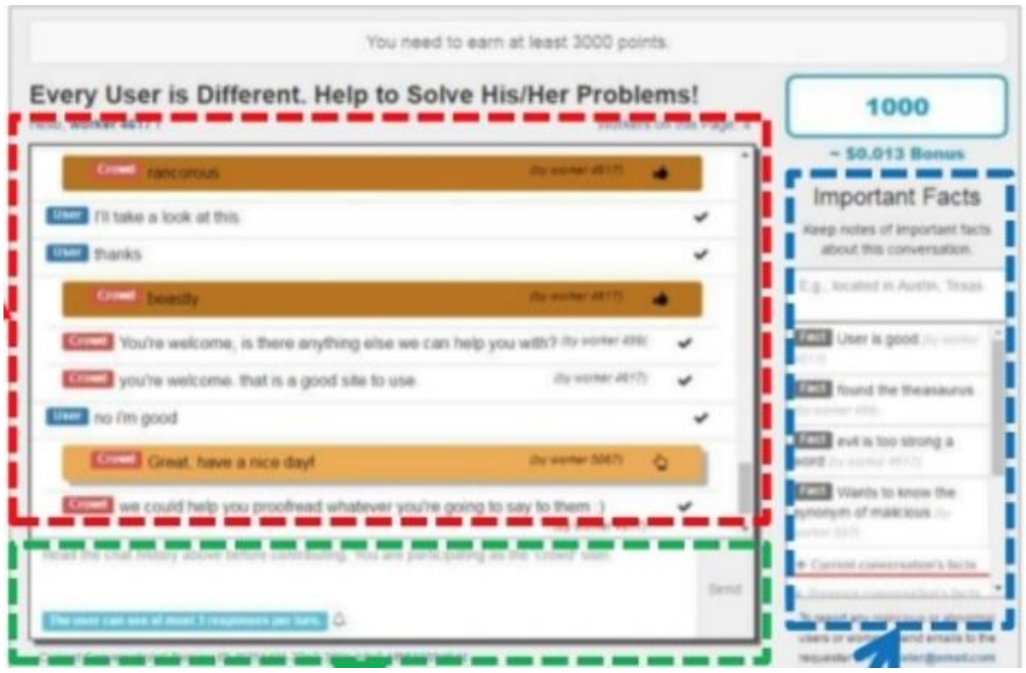
Design Rationale:

- In the case of a MySQL error due to the sent query, error will be logged to the database and a report will be sent to the crowd worker and the corresponding crowd worker supplier firm.

4.4.2 External Interfaces

4.4.2.1 User Interfaces

The interfaces that are in the Chorus system are the interface for the crowd workers, the interface for the researchers and the interface for the admins, so in other words the "users" of the Chorus system in this context are not the users who ask assistance from the Chorus system. All of the interfaces will be explained in detail further in this section.



You need to earn at least 3000 points

Every User is Different. Help to Solve His/Her Problems!

Crowd rancorous

User I'll take a look at this

User thanks

Crowd beastly

Crowd You're welcome, is there anything else we can help you with?

Crowd you're welcome. that is a good site to use

User no i'm good

Crowd Great, have a nice day!

Crowd we could help you proofread whatever you're going to say to them :)

Send

1000

~ $0.013 Bonus

Important Facts

Keep notes of important facts about this conversation.

E.g., located in Austin, Texas

Fact User is good

Fact found the theasaurus

Fact evil is too strong a word

Fact Wants to know the synonym of malicious

Figure 8: The Crowd Worker Interface

Crowd Worker Interface:

Crowd Worker Interface allows the crowd workers to interact with the Chorus system. Most of the functionality of the system is provided by this interface. The interface consists of the chat section and the memory section with information provided about the current chat.

Chat section shows the conversation between crowd workers and the end user, it also provides buttons for voting proposed responses. End user’s messages are shown in blue and the crowd worker responses are shown in red. If a proposed response is selected and sent as the highest voted response, the background of the message entry will become a mustard yellow colored, bumped, soft-cornered rectangle. In the bottom part of the chat section there is a text box to fill for the crowd workers to enter their proposed responses and a “Send” button that sends the message to the Chorus.

In the memory section there is a text box to fill for the proposed memory facts and below that there are the memory facts already proposed. The text box does not have a send button but hitting the “Enter” key triggers the creation of a memory fact. Above the memory section there is a soft-cornered rectangle that shows the points earned up until that moment.

Design Rationale:

- Since the design of this interface is simple, the probability of misusing the service is reduced.
- The single paged interface design provides, functionality and information to be accessible altogether in one place.
- The multi-colored and shaped design of the interface ease up the usage of the system.

Researcher Interface:

Researcher Interface allows the researchers to access and manipulate the data in the Chorus system without being interrupted by other system activity. Via this interface researchers will be query the database without need of the SQL knowledge. There are two tabs in the interface; “selection and representation” tab and “extraction” tab.

In the selection and representation tab, in the left part of the screen, researchers will be able to select the database table to be queried and the filters that will be applied to the data via dropdown boxes and text boxes. Also, there will be a textbox below of the visual selection tools that allows manual input for an SQL query that composes the required data, and below of that there is an “execute” button that triggers the execution of the provided query that results in the representation of the data. In addition, there is a “extract” button next to the “execute” button that redirects the researcher to the extraction tab. In the right part of the page, representation of the data that is the result of the query will be shown. There is both tabular representation and graphical representation, if applicable. Researcher is able to sort the data by an attribute by clicking on the corresponding column name in the table. If the result of the query contains numerical information, the graphical representation of the information will appear. The type of the graphical representation will be modifiable via a dropdown selection box appearing in the right top corner of the graphic.

The Extraction tab will consist of a dropdown selection box that allows the researcher to select the extracted data type and a “download” button that triggers the conversion and extraction of the data from the Chorus system to a file. The file that has been generated will be downloadable from the web browser.

Design Rationale:

- Since it will be in HTML5 all web browsers that support HTML5 will support this interface.
- Supported extraction types will be “.xls, .xlsx, .csv, .m”.
- Supported graphical representation types are: Pie charts, bar and Line graphs.
- The operations that change the database will not be allowed from this interface. In other words this interface will provide “Read-Only” database access.

Admin Interface:

Admin Interface will be provide access to all functionalities and data stored in the system for the admin. Thus the Admin Interface will contain and extend the Crowd Worker Interface and the Researcher Interface. The Admin Interface will consist of three tabs; extended version of Crowd Worker Interface tab, Researcher Interface tab and Database Access Console tab.

The extended Crowd Worker Interface tab will include the Crowd Worker Interface plus a list of the current active conversations. Each html item in the list will be a link that allows us to redirect to the corresponding conversation. The Researcher Interface tab will be the same as the original Researcher Interface. The Database Access Console tab will consist of a console that is directly connected to the database.

Design Rationale:

- The console in the Database Access Console tab is connected to the Web Server Controller that functions as a bridge between Database Server and the Database Access Console tab. The connection is authorized with the credentials of the Admin, so that unauthorized access to the database is restricted.
- Unlike Researcher Interface, the Admin Interface is capable and allowed to perform manipulative operations on the database. In other words the Admin Interface provides a "Read-Write" database access.

4.4.2.2 System Interfaces

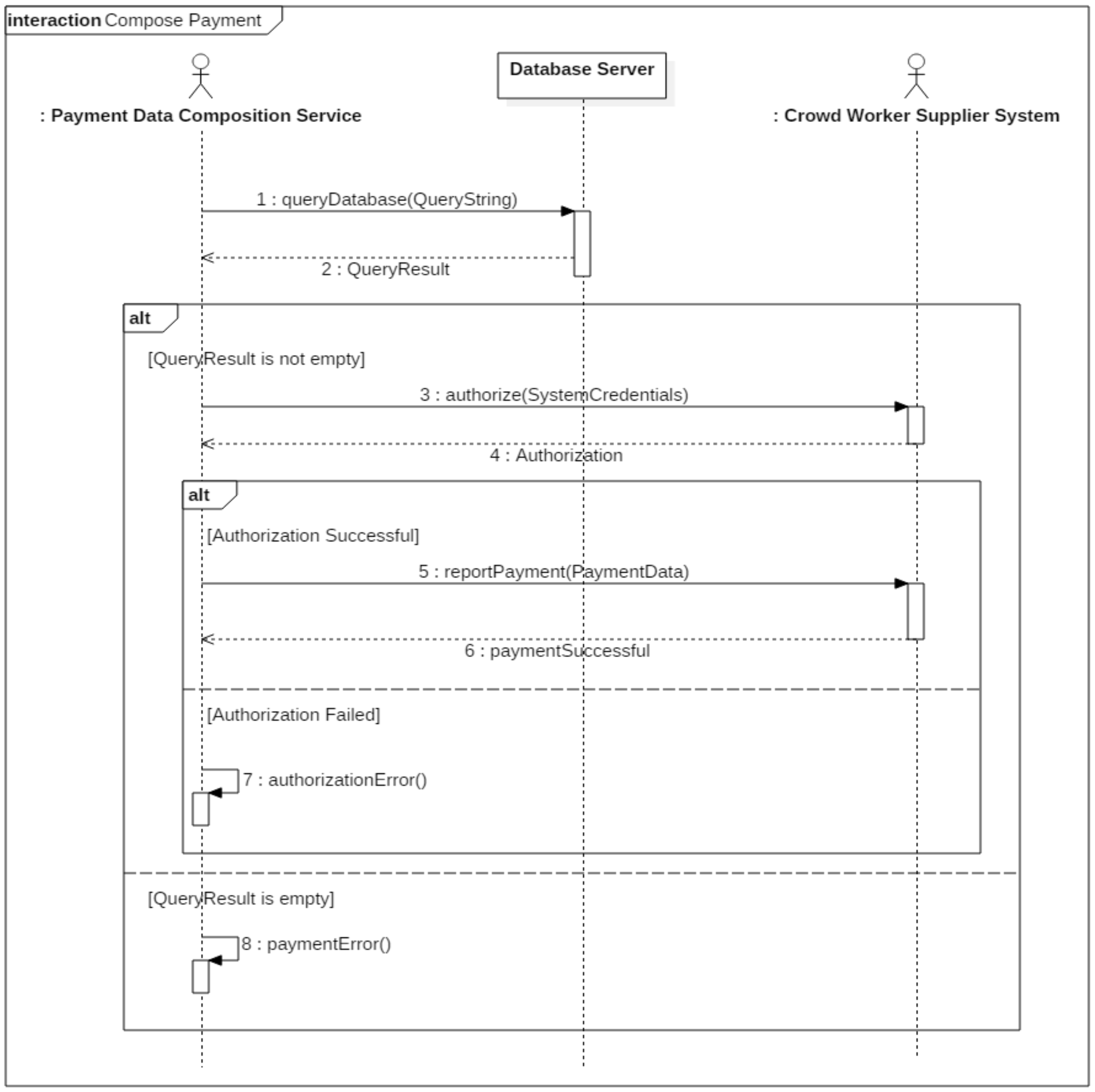

interaction Compose Payment
: Payment Data Composition Service
Database Server
: Crowd Worker Supplier System
1 : queryDatabase(QueryString)
2 : QueryResult
alt
[QueryResult is not empty]
3 : authorize(SystemCredentials)
4 : Authorization
alt
[Authorization Successful]
5 : reportPayment(PaymentData)
6 : paymentSuccessful
[Authorization Failed]
7 : authorizationError()
[QueryResult is empty]
8 : paymentError()

Figure 9: Compose Payment sequence diagram showing the interfaces between Payment Data Composition Service and external components; Database Server component, Crowd Worker Supplier System

The Interface between the Payment Data Composition Service and the Crowd Worker Supplier Systems:

Payment Data Composition Service will be responsible for composing and reporting the payment data to the corresponding Crowd Worker Supplier Systems. After the composition of the data, Payment Data Composition Service component will request authorization from Crowd Worker Supplier System with its credentials. If the authorization is successful, it will send the composed data to the Crowd Worker Supplier Systems as Payment Data.

Design Rationale:

- If an error occurs in any part of the processes, it will be reported back to the Payment Data Composition Service component.
- If the payment is successful, the Payment Data Composition Service will reset the points of each crowd worker.

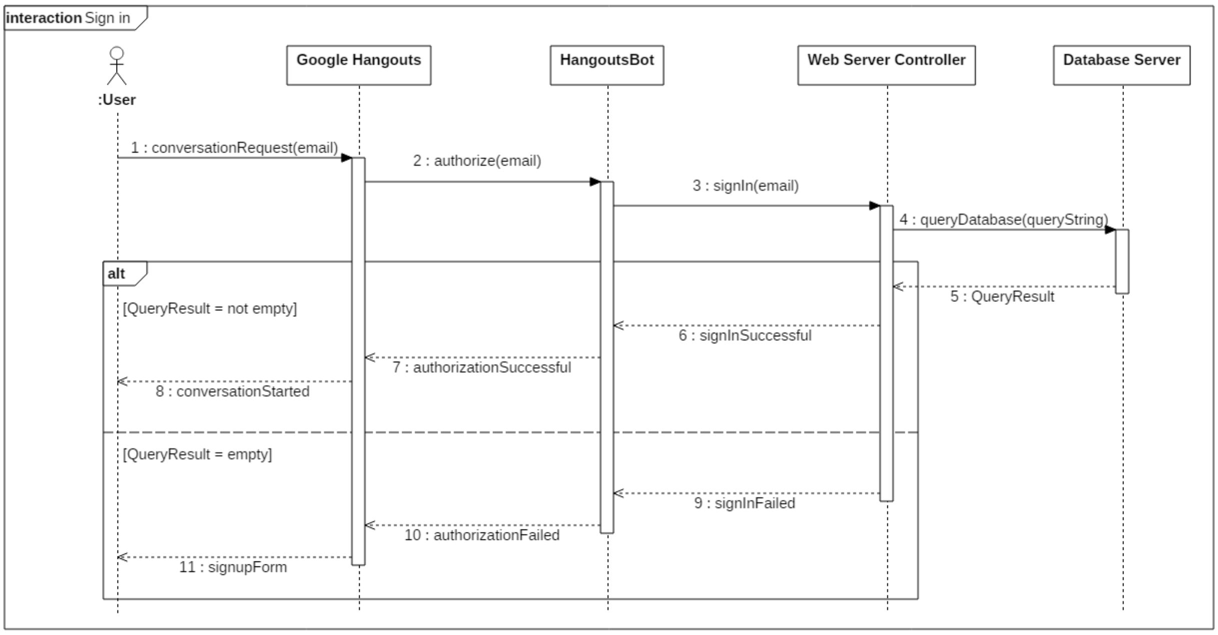


Figure 10: Sign-in sequence diagram showing the interface between Web Server Controller component and the external system HangoutsBot component

The Interface between the HangoutsBot and the Web Server Controller:

When a user starts a new conversation, HangoutsBot will request authentication of the user from the Web Server Controller. Web Server Controller will query the database in order to access the user's credentials. If the user is not registered, a sign up form will be sent to the user. If the user is registered, authorization will be successful and the conversation will begin.

Design Rationale:

- The communication between HangoutsBot and the Web Server Controller is not persistent, in other words, after the authorization request is responded, the need for the connection is over. Hence HTTP is used for the communication protocol.

The Interface between the HangoutsBot and the Communication Service:

Communication Service will be responsible of receiving, formatting and sending the messages sent by HangoutsBot to the Web Server Controller.

Design Rationale:

- Communication Service component will include all socket connections with the HangoutsBot. TCP with SSL will be used for secured connection.

The Interface between the Crowd Worker Request Handler and the Crowd Worker Supplier Systems:

When a crowd worker request is made by the Web Server Controller, Crowd Worker Request Handler reports the required number of crowd workers to the Crowd Worker Supplier Systems and waits for the acquisition of the crowd workers. When the process is completed the information of the supplied crowd workers are sent to the Crowd Worker Request Handler.

Design Rationale:

- The algorithm for requesting a crowd worker is as follows: First the request is sent to all of the supplier firms, when any of the supplier firms send a crowd worker back the system will check if it has reached the required number, if yes the requesting is over, if not it will ask for another worker from each supplier firm that send back a crowd worker and go back to checking the required number.