

HOMEWORK#2

HW TITLE: HW2

TEAM TITLE: TEAM02

NAMES OF ALL MEMBERS: 4

Name	Email address	Signatures
<i>Xiaosong Lu</i>	xiaosong.lu@csu.fullerton.edu	xiaosong
<i>Deepak Prasanna Mayanattanmy Yagneswaran</i>	dee7@csu.fullerton.edu	MY Deepu//
<i>Guangyi Shang</i>	guangyi.shang@csu.fullerton.edu	Guangyi Shang
<i>Chongbei Wang</i>	chongbeiwang@csu.fullerton.edu	chongbeiwang

SUBMISSION DATE: 05/03/2018

COURSE TITLE: SOFTWARE DESIGN AND ARCHITECTURE

INSTRUCTOR NAME: CHANG-HYUN JO

REVISION HISTORY

<i>Version</i>	<i>Time</i>	<i>Author</i>	<i>Revision Summary</i>
1	3/14/2018	Guangyi Shang	Quality attribute scenario
2	3/18/2018	chongbeiwang	Use case description
3	3/24/2018	Xiaosong Lu	Merging the document
4	3/24/2018	Guangyi Shang	Utility ADRT PCM iteration1
5	3/25/2018	guangyishan g	Utility ADRT PCM iteration2
6	3/26/2018	chongbeiwang	Utility ADRT PCM iteration3
7	5/01/2018	chongbeiwang g	Table of contents
7	5/01/2018	chongbeiwang g	Table of contents
8	5/01/2018	Deepak	Overall documentation works
9	4/01/2018	Deepak	QAW
10	4/29/2018	Deepak	ADD v2

Table of Contents

<i>HOMEWORK#2</i>	1
<i>REVISION HISTORY</i>	2
<i>EXERCISE1</i>	4
VISION.....	4
SYSTEM CONTEXT DIAGRAM	5
TECHNICAL ENVIRONMENT	5
DOMAIN KNOWLEDGE:	7
TOP FEATURES OF FCHAT:	12
USE CASE	13
USE CASE DIAGRAM:	14
QUALITY ATTRIBUTES:.....	15
<i>EXERCISE 2</i>	21
QAW PRESENTATIONS AND INTRODUCTION:.....	21
<i>EXERCISE 3</i>	34
CONTEXT:	34
INPUTS OF ADD:.....	34
ADD STEPS:.....	35
DOCUMENTING BEYOND THE VIEWS:	76
<u> <i>lesson learned</i></u>	81
<i>TEAM CHARTER</i>	85
<i>Team Evaluation</i>	89

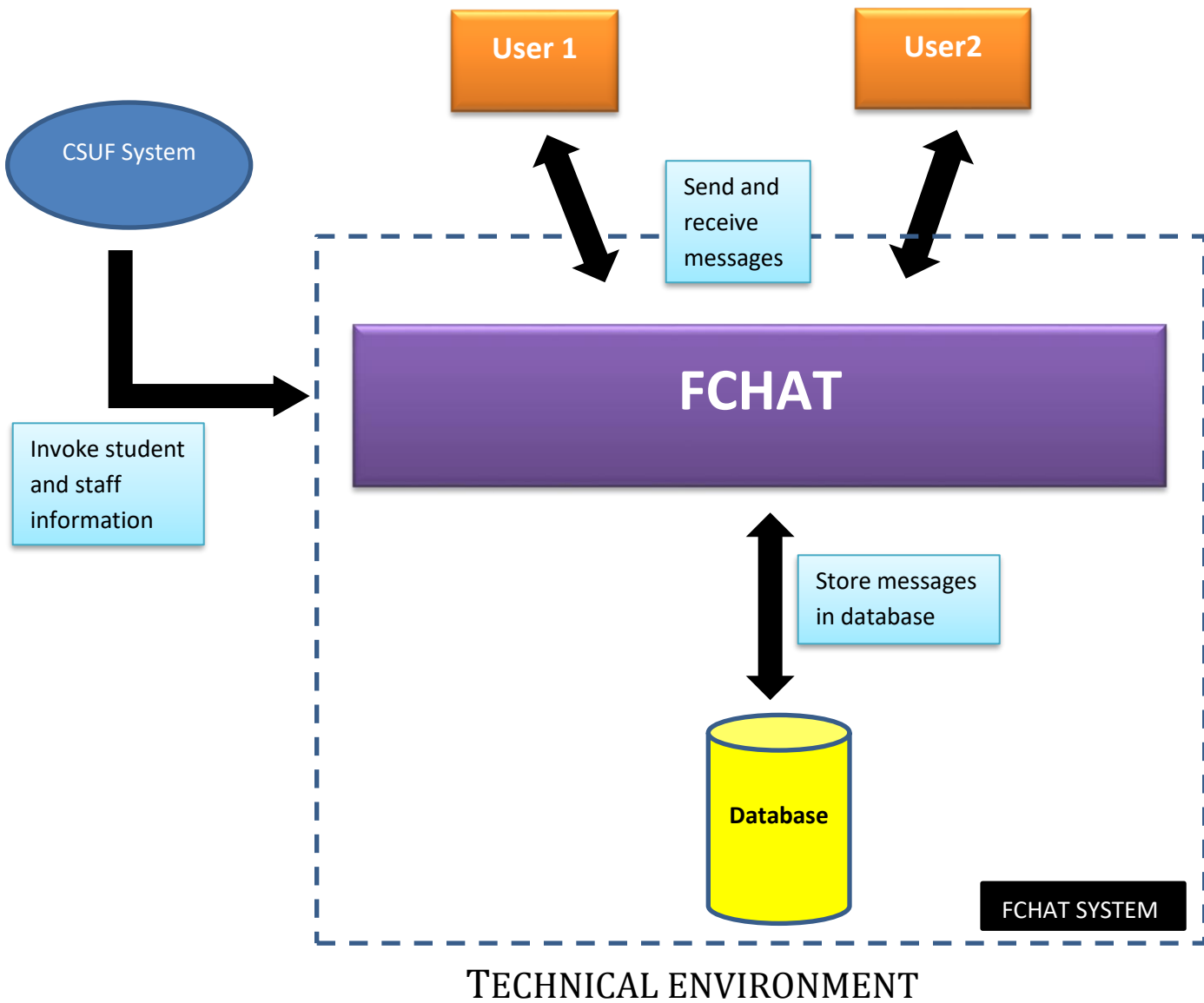
EXERCISE1

VISION

For the college students of California state university Fullerton, who is in need of an inter-communication smart network, the 'fchat (fullerton_chat)' is a chat suit that enhances the communication between various students and faculties of our university in a much effective manner. Unlike WhatsApp and Facebook messenger which are popularly known chat web applications, our product is exclusively for Fullerton students, Professors and department staffs who can communicate with each other in a safer and secure manner. It is believed that, a chat web application between various peoples of our university will enhance the relationship between students and faculties.

SYSTEM CONTEXT DIAGRAM

Figure 1.1 – System Context Diagram



TECHNICAL ENVIRONMENT

PROTOCOL: XMPP

XMPP is abbreviated as ‘eXtensible Messaging and Presence Protocol’ mainly used for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data.

Openfire is a real-time collaboration (RTC) server licensed under the Open Source Apache License. It uses the only widely adopted open protocol for instant messaging, XMPP. Openfire is incredibly easy to setup and administer but offers rock-solid security and performance.

Our project is based on XMPP protocol to delivery messages and files between users.

DATABASE: MySQL

MySQL is the world’s most popular open source database. Besides, MySQL is a relational database management system (RDBMS) for database management using the most commonly used database management language, Structured Query Language (SQL). In addition, MySQL supports ACID (Atomicity, Consistency, Isolation, Durability), which can maintain the integrity of the database. We can use MySQL to store text messages.

FILE SYSTEM: HDFS

HDFS is abbreviated as ‘Hadoop Distributed File System’. It is designed to reliably store files across machines in a large cluster. User’s pictures, videos, audio files can be stored in HDFS.

LOAD BALANCER: NGINX

Usually, we have more than one servers. So, we need someone to distribute incoming network traffic across a group of backend servers. Nginx is a good choice. It is not only a powerful web server but also a reverse proxy server. If one of our server goes down, nginx can redirect traffic to the remaining online servers.

DOMAIN KNOWLEDGE:

'Fullerton_Chat' (fchat) falls under Instant Messaging domain. Instant messaging is a set of communication technologies used for text-based communication between two or more participants over the Internet or other types of [networks](#).

IM–chat happens in real-time. Of importance is that online chat and instant messaging differ from other technologies such as email due to the perceived quasi-[synchrony](#) of the communications by the users. Some systems permit messages to be sent to users not then 'logged on' (*offline messages*), thus removing some differences between IM and email (often done by sending the message to the associated email account).

IM allows effective and efficient communication, allowing immediate receipt of acknowledgment or reply. However, IM is basically not necessarily supported by [transaction control](#). In many cases, instant messaging includes added features which can make it even more popular. For example, users may see each other via [webcams](#), or talk directly for free over the Internet using a [microphone](#) and [headphones](#) or loudspeakers. Many applications allow file transfers, although they are usually limited in the permissible file-size.

It is usually possible to save a text conversation for later reference. Instant messages are often logged in a local message history, making it similar to the persistent nature of emails. With the history of the communication pointing to a huge challenge to humanity, progressive development yielded to social networking that comes with instant messaging as the most crucial component. Both social media networks and special messaging programs utilize differing messaging protocols to affect the instant communication between users.

These protocols are the core of instant messaging by providing the key features. Two of the main protocols used for instant messaging in the market today are Web Socket and XMPP.

Based on my research, we have planned to use XMPP (Extensible Message Present Protocol)

XMPP is abbreviated as Extensible Messaging and Presence Protocol:

XMPP is originally developed as messaging platform, and the primary communication of this protocol is short messages between server and client or between client and client. XMPP is reactive to user's presence and status.

XMPP ABBREVIATION:

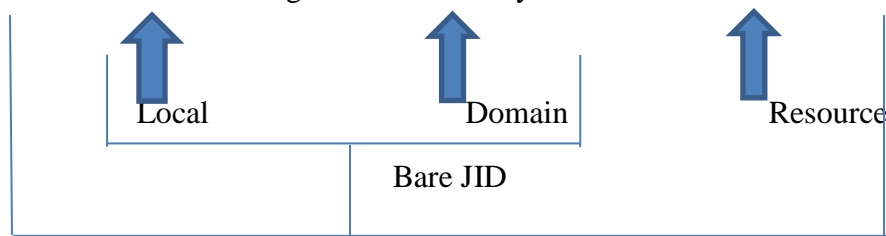
eXtensible	Protocol has customizing feature so that it can be customized to individual needs.
Messaging	the primary communication of this protocol is short messages between server and client or between client and client
Presence	Reactive to user's presence and status
Protocol	Based on standardization from XML Standards Federation. It is open to any platform.

XMPP DATA STREAM:

XMPP takes place between single data stream, between the user and the server.

<stream>	
	<stream>
<presence> <show/> <presence>	
<message to='user'> <body/>	
</message>	
<iq to='user' type='get'> <query/> </iq>	
	<iq from='serv' type='result'> <query/>
	</iq>
...	
<unavailable/>	...
</stream>	
	</stream>

The basic unit of message identification system in XMPP is the Jabber ID(JID)



STANZAS:

Communication in XMPP server is broken down into Stanzas. Stanzas are short contained XML messages sent through the server. Stanzas are divided into three types,

1. Presence

2. IQ
3. Message.

Basic Structure of Stanza is,

```
<[Stanza type] [from=""] [to=""] [type=""] [id=""]>
```

```
<child element>
```

```
<child element>
```

```
<sub-child element/>
```

```
</child element>
```

```
</[Stanza type]>
```

TYPES OF STANZAS:

Presence Stanza

Presence Stanzas is used to indicate user status. It can be used as a trigger for events inside the server. Presence is shared with all users on Roaster. For example, presence stanzas are used to indicate the status of the user like whether the user is online or offline. They may be sent by components, servers or clients. There are 5 different show tags for indicating the status of a user,

- Chat
- Away
- eXtended away
- Do not Disturb
- Unavailable

The presence stanza has child elements to elaborate or to provide more information about the status of a user. They are,

- i. <show/> - away, chat, child, extended away, unavailable
- ii. <status/> - any text string
- iii. <priority/> - Integer between -128 and -127
- iv. <c/> - Extended content with provided XMLNS

Example:

```
<presence from=scott@xmpp.example.com/LENOVO-Z_585 to=user@xmpp.example.com><priority>5</priority></presence>
```

Info-query stanzas (iq stanzas):

IQ Stanzas generally contain the text or media files entered by the user. It works on request and response mechanism. There are four types of IQ generally used, they are

- Set
- Get
- Result
- Error

Service Discovery or Disco:

Service Discovery or disco are used to find out details about users, servers, features etc. Disco is further divided into two categories,

- i. Disco#items:
Disco items requests available items like 'Multi User Chat (MUC) rooms', inputs from the peer user end and other information services.
- ii. Disco#info:
Disco info primarily requests information about a special identity.

Example:

```
<iq from='yourjid@example.com' id='22' to='example.com' type='get'>
```

```
<query xmlns='http://jabber.org/protocol/disco#items' /> </iq>
```

The 'to JID' in the above case refers may refer to a server, components or users.

'from JID' is not always necessary in IQ Stanzas.

Message Stanzas:

Message Stanzas are designed to carry person to person text, they can be extended to deliver rich text or different formatting. There are five types of message types,

- i. Chat – one on one message between two users.
- ii. Group chat – messages being sent to and from Multi User Chat rooms.
- iii. Headline – alert messages that do not expect a reply.
- iv. Normal – standalone message with no history, but reply is expected.
- v. Error – messages reserved for error delivery.

Messages have specific child elements,

<body/> - normally included, contains human readable xml data.

<subject/> - optional element specifying the topic of the message.

<thread/> - non-human readable using a unique ID to collect messages.

Example:

```
<message type="chat" to=user2@example.com id="aac9a">
```

```
<body>Hello, are you free to meet at 12:00 today?</body>
```

```
</message>
```

COMPONENTS:

Components are the items of code that can extend or provide services to the server. They typically come with a server install, and allows to turn on or off. Components have a JID assigned with them. The examples of components are,

- Pub-sub
- Multi User Chat environment
- HTTP
- AMP
- Jingle

TOP FEATURES OF FCHAT:

- The 'Fullerton_Chat' allows a user to share private text/voice messages, audio/video files, pictures between a single and multiple user groups with the help of internet.
- Fchat primarily works on user's phone number and campus wide ID as the input from user for registering with the application.
- The system allows the user to store his/her private data in a cloud server, so that the user is able to retrieve the data from the internet in any device.
- The system is flexible to run on different platforms like Android, MacOS, Windows and web browser.
- User can send and receive messages from different users, irrespective of the platform.
- The system provides data hiding from third party hackers through encryption and decryption message services.
- The availability of Fchat system is more than 99%, the system takes very less downtime in a year for maintenance.
- Even if we miss our notifications or turn off our phone, Fchat will save our recent messages until the next time we use the application
- Fchat makes use of phone address book to connect quickly and easily with our contacts who have already enrolled with 'Fullerton_chat' application.
- The users in fchat will be always logged in, so that they don't miss messages. No more confusion about whether a user is logged in or logged out.
- Fchat provides tag for all types of users. Students will be tagged with 'ST' symbol and professors will have a 'Pf' symbol. This feature allows the users to easily identify other students and faculties.



Display picture with Student Tag



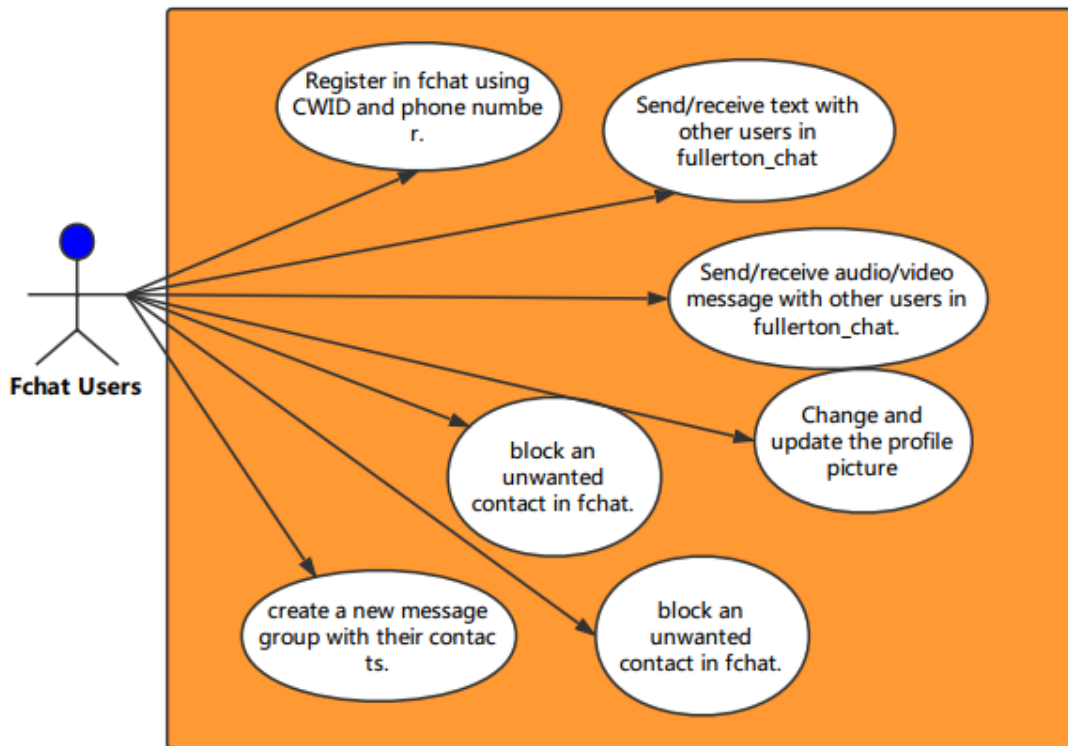
Display picture with Professor Tag

USE CASE

FUNCTIONAL REQUIREMENTS:

- FR1* User should be able to send/receive text with other users in fullerton_chat.
- FR2* User should be able to send/receive audio/video message with other users in fullerton_chat.
- FR3* User should be able to send message even though another user is not active in fchat.
- FR4* Fchat must be able to run between different devices with different operating systems.
- FR5* Fchat should update student's information and staff's information and shall display the same in profile section.
- FR6* User should register in fchat using CWID and phone number.
- FR7* Sharing files/text message should be done securely with some sorts of encryption and decryption.
- FR8* The fchat must be responding fast enough when exchanging texts and sharing audio and video.
- FR9* User interface should be easy to use.
- FR10* Users should be able to change and update the profile picture
- FR11* Users should have the ability to block an unwanted contact in fchat.
- FR12* Users should be able to create a new message group with their contacts.
- FR13* Fchat should display 'ST' tag for students and 'Pf.' Tag for faculties.

USE CASE DIAGRAM:



USER CASE 1: REGISTER IN FCHAT USING CWID AND PHONE NUMBER.

User can create their account by providing their name, phone number, CWID. Our system will verify their CWID and phone number.

USER CASE 2: SEND/RECEIVE TEXT WITH OTHER USERS IN FULLERTON_CHAT

User can send and receive text messages with other users in Fullerton_chat. If the user is online, he or she can receive the messages or files immediately. If the user is offline, he will receive the messages or files when he or she login.

USER CASE 3: SEND/RECEIVE AUDIO/VIDEO MESSAGE WITH OTHER USERS IN FULLERTON_CHAT.

User can send and receive audio or video message with other users in Fullerton_chat. If the user is online, he or she can receive the messages or files immediately. If the user is offline, he will receive the messages or files when he or she login.

USER CASE 4: CHANGE AND UPDATE THE PROFILE PICTURE

User logs Fullerton_chat and then select his account setting. he can update his profile picture in his account setting.

USER CASE 5: BLOCK AN UNWANTED CONTACT IN FCHAT.

User logs Fullerton_chat and open his contacts, and then he can block an unwanted contact from his contacts.

USER CASE 6: CREATE A NEW MESSAGE GROUP WITH THEIR CONTACTS.

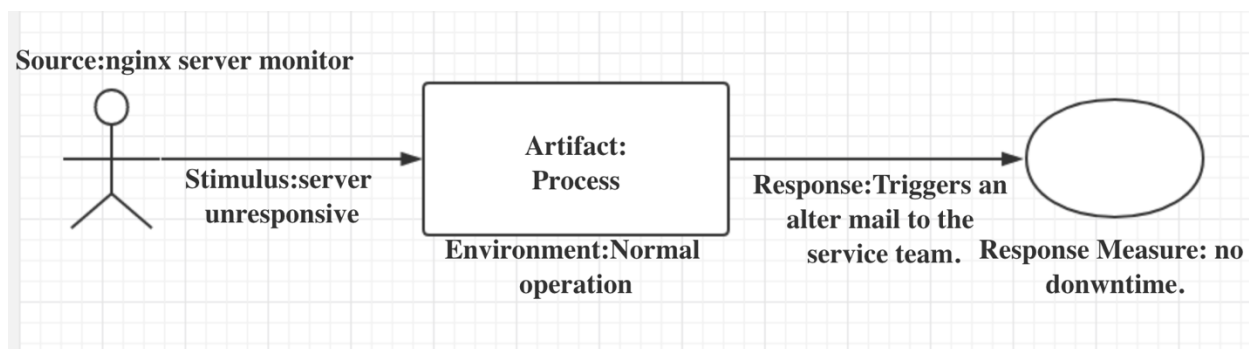
User logs in Fullerton_chat and open his contacts. he selects friends from his contacts and create a new message group with the selected friends.

QUALITY ATTRIBUTES:

Our Fchat system possesses 7 quality attributes. They are as follows:

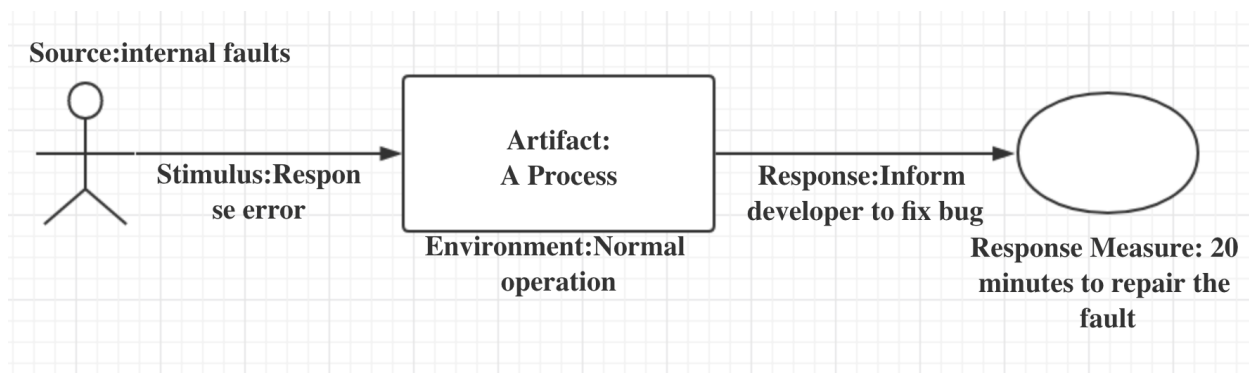
1. AVAILABILITY CONCRETE SCENARIO:

The Nginx server monitor finds that the apache Cassandra server is nonresponsive process during normal operations (send message). The monitor triggers an alert mail to the service team to repair the issue and continues to operate.



2. AVAILABILITY CONCRETE SCENARIO

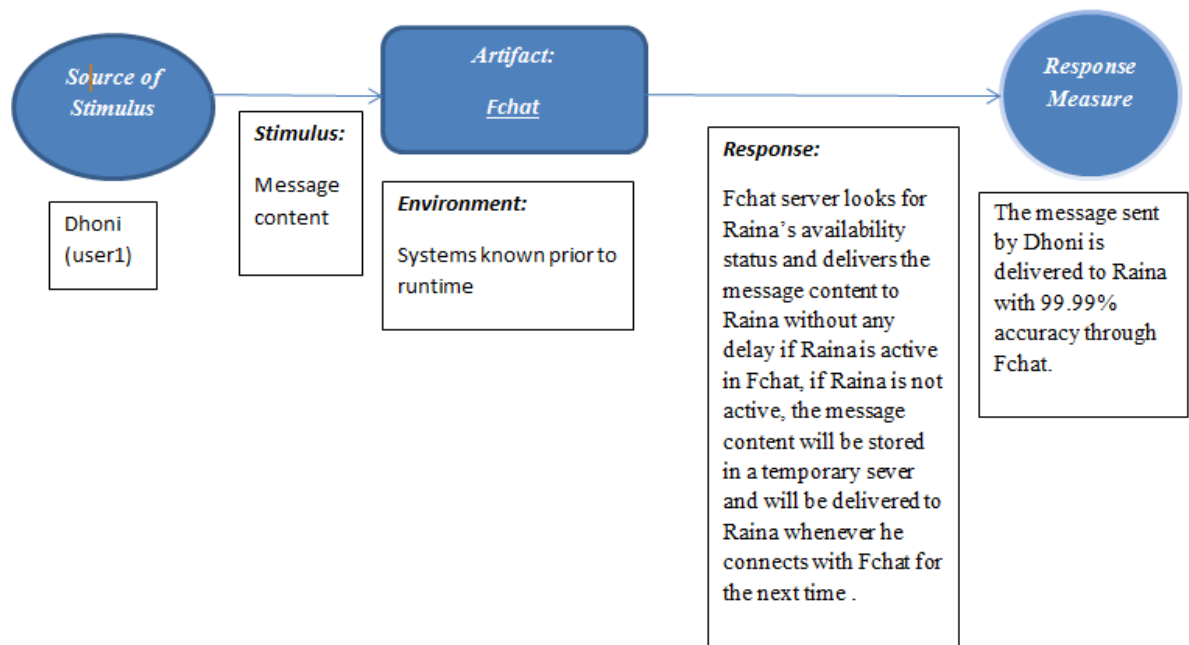
User cannot update their profiles, because one of backend server is unavailable, the developer fixes the bug and then restarts the backend server.



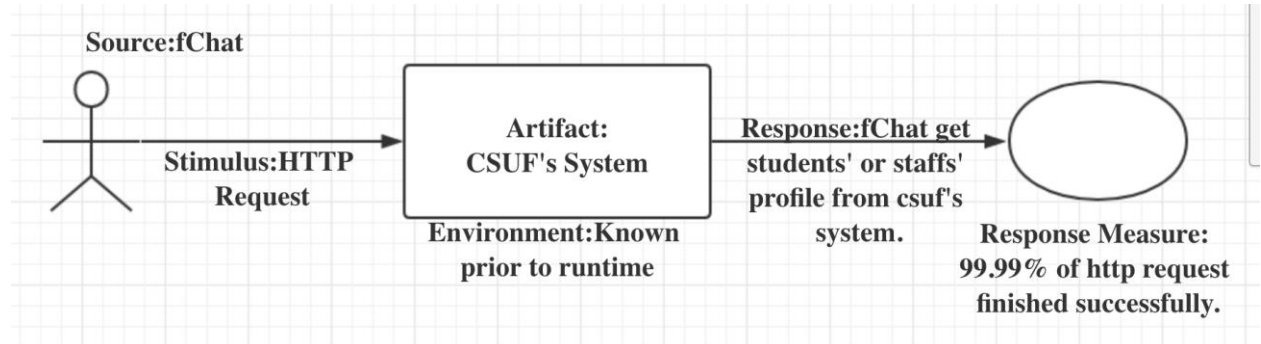
3. INTEROPERABILITY CONCRETE SCENARIO:

Dhoni communicates with Raina through Fchat. Dhoni's message content (We can assume a simple "Hello" in our case) will be first sent to our fchat server. Fchat server looks for Raina's availability status and delivers the message content to Raina without any delay if Raina is active in Fchat, if Raina is not active, the message content will be stored in a temporary sever and will

be delivered to Raina whenever he connects with Fchat for the next time . The message sent by Dhoni is delivered to Raina with 99.99% accuracy through Fchat.

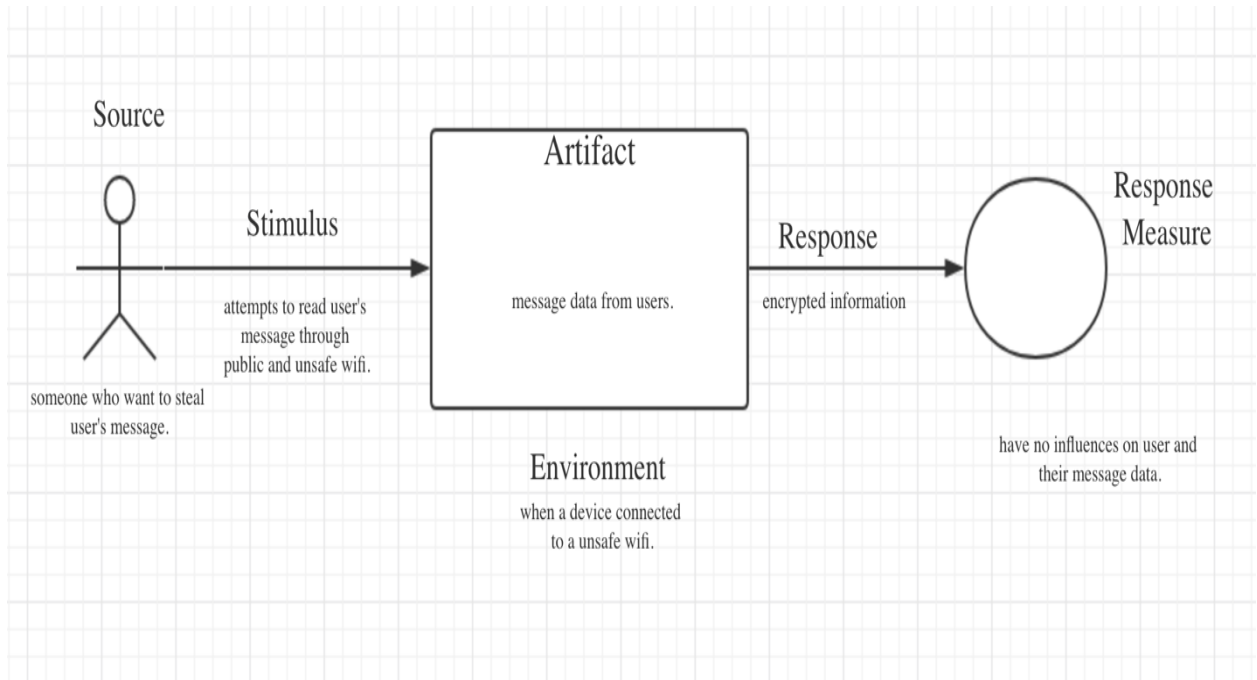


Fchat get and validate students or staffs' profile by using CSUF's system.

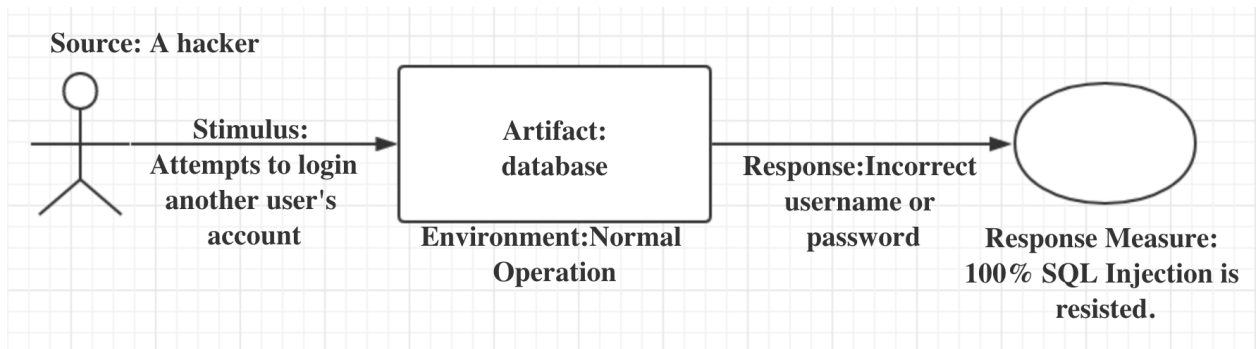


4. SECURITY CONCRETE SCENARIO:

A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.

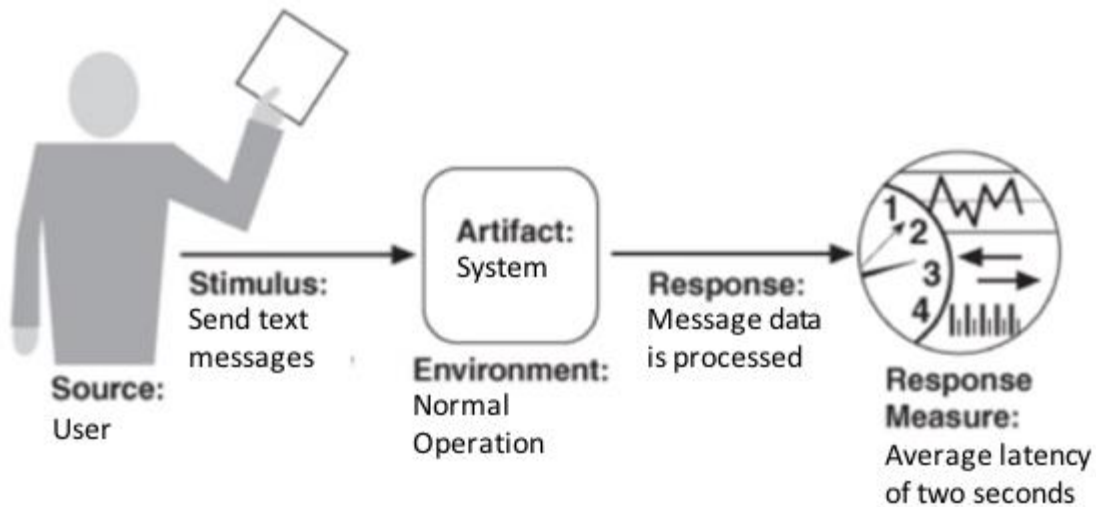


A hacker wants to login another user's account by SQL injection. However, fchat use SQL Parameters for Protection.

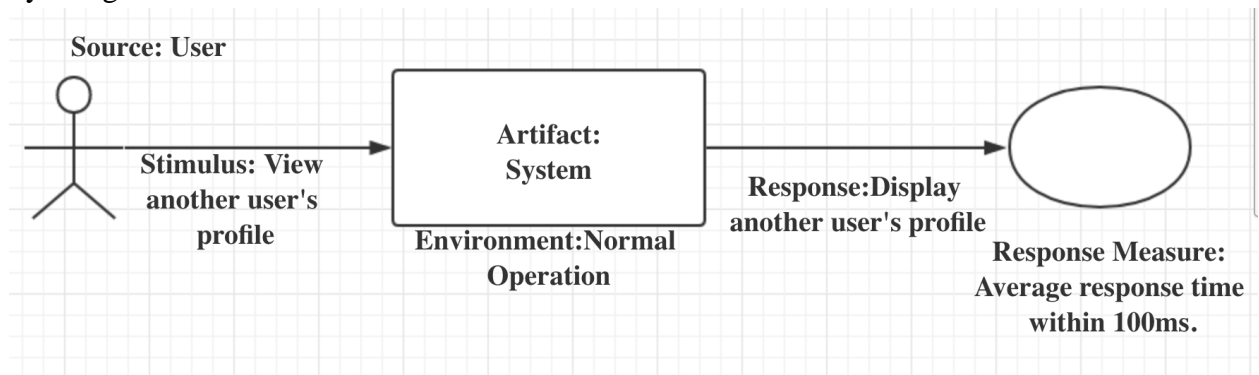


5. PERFORMANCE CONCRETE SCENARIO:

Fchat users send 1000 text messages per second under normal operations. Fchat system processes the message data with an average latency of two seconds.

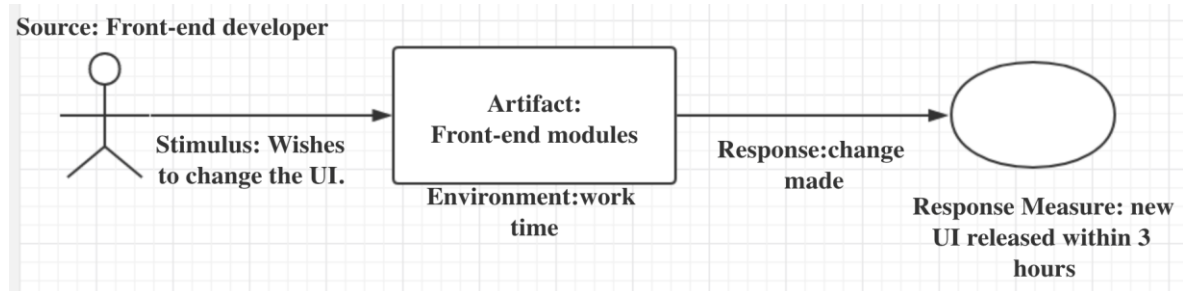


User can get another user's profile within 100ms because fchat copies user's profile into memory by using redis.



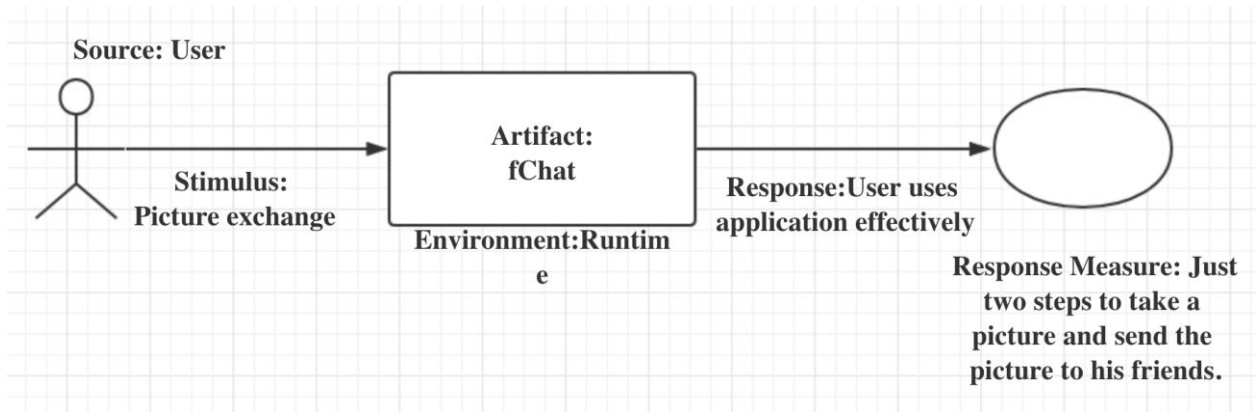
6. MODIFIABILITY CONCRETE SCENARIO:

The front-end developer can change user interface with 3 hours and no changes in backend.



7. USABILITY CONCRETE SCENARIO:

User can send pictures to his or her friends directly by using in-app camera.



OTHER QUALITY ATTRIBUTES: PORTABILITY

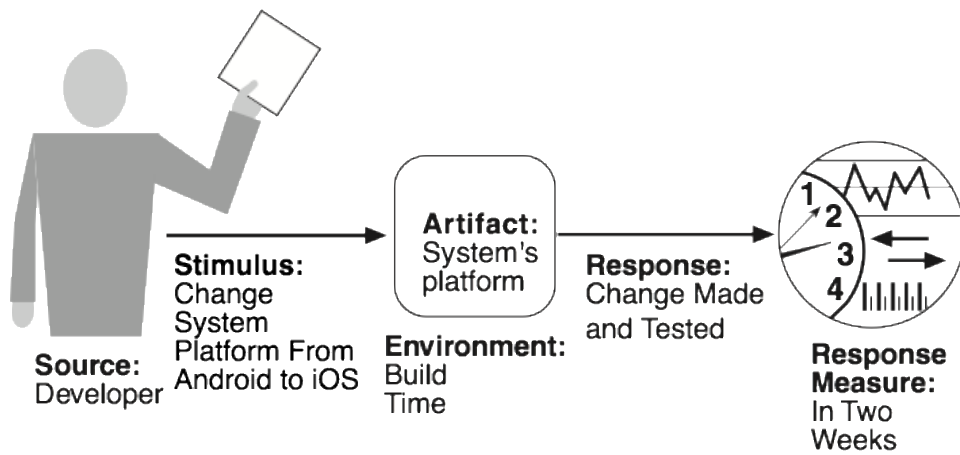
The developer is asked to change the system executing platform from Android to iOS at build time. The change is made and tested within 2 weeks.

PORTABILITY GENERAL SCENARIO:

- *Source of stimulus.* The developer.
- *Stimulus.* Change the system's executing platform from the one it has been developed for to a different software/hardware platform.
- *Artifact.* The system's platform.
- *Environment.* Design time, compile time, build time, initiation time, or runtime.
- *Response.* Make the change, test it, and deploy it.
- *Response measure.* All of the possible responses take time and cost money.

Following table enumerates the elements of the general scenario that characterize portability:

Portion of Scenario	Possible Values
Source	The developer
Stimulus	Change the system's executing platform
Artifacts	The system's platform
Environment	Design time, compile time, build time, initiation time, or runtime.
Response	One or more of the following: <ul style="list-style-type: none"> ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: <ul style="list-style-type: none"> ▪ Number, size, complexity of affected artifacts ▪ Effort ▪ Calendar time ▪ Money ▪ New defects introduced



EXERCISE 2

QAW PRESENTATIONS AND INTRODUCTION:

Quality attribute workshop (QAW) provides a method for evaluating the architecture of a software-intensive system during the acquisition phase of major programs. In our case, this Quality attribute Workshop is being held to figure the most prominent quality attributes for our Fullerton_Chat application.

Fullerton_chat is a chat application exclusively for the students and professors of California State University, Fullerton. Fchat is planned to develop, with an objective of enhancing the interaction between students and professors of CSUF.

The architecture is evaluated against a number of critical quality attributes, such as availability, performance, security, interoperability, and modifiability. The evaluation is based on test cases that capture questions and concerns elicited from various stakeholders associated with the system. The process of eliciting questions allows stakeholders to communicate directly, thereby exposing assumptions that may not have surfaced during requirements capture.

In our scenario, we divided our team members are various stakeholders of Fchat,

Stakeholders	Role
Deepak Mayanattanmy	Facilitator
Chongbei Wang	Developer
Guangyi Shang	End user
Song	Customer

Quality Assurance Workshop includes the following steps:

1. QAW Presentation and Introductions

QAW facilitators describe the motivation for the QAW and explain each step of the method.

2. Business/Programmatic Presentation

A stakeholder presents the business and/or programmatic drivers for the system.

3. Architectural Plan Presentation

A technical stakeholder presents the system architectural plans as they stand with respect to early documents, such as high-level system descriptions, context drawings, or other artifacts that describe the system's technical details.

4. Identification of Architectural Drivers

Architectural drivers often include high-level requirements, business/mission concerns, and various quality attributes. During this step, the facilitators and stakeholders reach a consensus about which drivers are keys to the system.

5. Scenario Brainstorming

Stakeholders generate real-world scenarios for the system. Scenarios comprise a related stimulus, an environmental condition, and a response. Facilitators ensure that at least one scenario addresses each of the architectural drivers identified.

6. Scenario Consolidation

Scenarios that are similar in content will be consolidated in this step.

7. Scenario Prioritization

Stakeholders prioritize the scenarios through a voting process.

8. Scenario Refinement

For the top four or five scenarios, the following are described:

- the business/programmatic goals that are affected by those scenarios
- the relevant quality attributes associated with those scenarios

.

2) BUSINESS/MISSION PRESENTATION:

BUSINESS OBJECTIVES OF FULLERTON_CHAT:

- Fullerton_chat is a chat application exclusively for the students and professors of California State University, Fullerton.
- Fchat is planned to develop, with an objective of enhancing the interaction between students and professors of CSUF.
- Fchat allows users to send and receive text messages and multimedia documents in a secured environment.

ARCHITECTURAL PLAN PRESENTATION:

Fullerton_chat falls under instant messaging domain. Users are able to send and receive text messages and multimedia files only if the users are connected through internet.

TECHNICAL ENVIRONMENT:

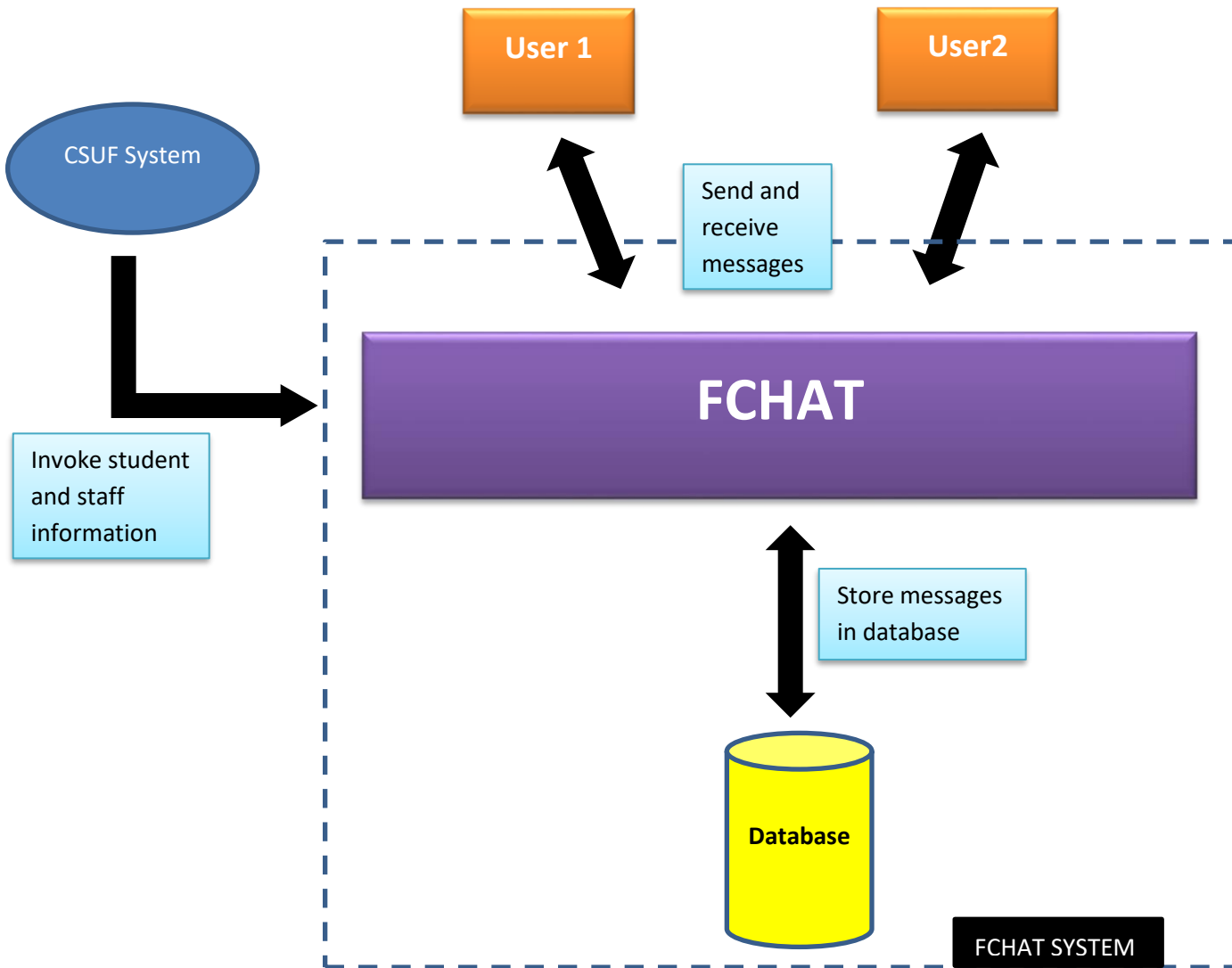
Fullerton_Chat is mainly configured with XMPP (eXtensible Messaging Presence Protocol). XMPP protocol is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language). It enables the near-real-time exchange of structured yet extensible data between any two or more network entities. XMPP is widely used in instant messaging domain.

Fullerton chat makes use of ‘Apache Cassandra’ server for storing messages temporarily and deletes the messages immediately after the message gets delivered to the peer end user.

Usually, we planned to implement two servers for load balancing. Nginx server distributes incoming network traffic across a group of backend servers. It is not only a powerful web server but also a reverse proxy server. If one of backend server goes down, Nginx redirects the traffic to the remaining online servers.

The system context diagram for fchat is shown in figure 1.1

FIGURE 2.1 SYSTEM CONTEXT DIAGRAM



4) IDENTIFICATION OF ARCHITECTURAL DRIVERS:

Functional requirements:

- FR1** User should be able to send/receive text with other users in fullerton_chat.
- FR2** User should be able to send/receive audio/video message with other users in fullerton_chat.
- FR3** User should be able to send message even though another user is not active in fchat.
- FR4** Fchat must be able to run between different devices with different operating systems.
- FR5** Fchat should update student's information and staff's information and shall display the same in profile section.
- FR6** User should register in fchat using CWID and phone number.

- FR7** Sharing files/text message should be done securely with some sorts of encryption and decryption.
- FR8** The fchat must be responding fast enough when exchanging texts and sharing audio and video.
- FR9** User interface should be easy to use.
- FR10** Users should be able to change and update the profile picture
- FR11** Users should have the ability to block an unwanted contact in fchat.
- FR12** Users should be able to create a new message group with their contacts.
- FR13** Fchat should display 'ST' tag for students and 'Pf.' Tag for faculties.

Non-functional requirements:

- NFR 1 Availability
- NFR 2 Interoperability
- NFR 3 Security
- NFR 4 Performance

Constraints:

The *design constraints* for Fchat are,

- Constraint 1: Fchat must be able to run on different platforms
- Constraint 2: Sharing messages should be highly secured
- Constraint 3: Time constraints – the deadline for Fchat project is 1st of December 2018.

5. SCENARIO BRAINSTORMING:

In this segment, the various scenarios proposed by the stakeholders will be addressed. The stakeholders propose scenarios in the way the system is expected to behave or the stakeholders question about target system's behavior under certain conditions. The various scenarios describes for Fullerton chat described by the stakeholders is as follows:

Scenario count	Scenarios
Scenario 1	Fault detection AVA-01: The Nginx server monitor finds that the apache Cassandra server is nonresponsive process during normal operations (send message). The monitor triggers an alert mail to the service team to repair the issue and continues to operate.
Scenario 2	Fault recovery AVA-02: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Scenario 3	Send/Receive messages between users INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Scenario 4	Data encryption SEC-01: A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Scenario 5	Inform actors SEC-02: When a hacker tries to access a user's data in a different device. Fchat informs the owner of the profile about the secondary login, requests for an acknowledgement from the user

	about the additional login. If the user denies the secondary login, the hacker won't be able to read the user's data.
Scenario 6	Introduce concurrency PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.
Scenario 7	User Interface changes should be completed within 3 man-hours of effort.
Scenario 8	Dhoni communicates with Raina through Fchat. Dhoni's message content (We can assume a simple "Hello" in our case) will be first sent to our fchat server. Fchat server looks for Raina's availability status and delivers the message content to Raina without any delay if Raina is active in Fchat, if Raina is not active, the message content will be stored in a temporary sever and will be delivered to Raina whenever he connects with Fchat for the next time . The message sent by Dhoni is delivered to Raina with 99.99% accuracy through Fchat.
Scenario 9	When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Scenario 10	When a hacker tries to access a user's data in a different device. Fchat informs the owner of the profile about the secondary login, requests for an acknowledgement from the user about the additional login. If the user denies the secondary login, the hacker won't be able to read the user's data.
Scenario 11	The developer is asked to change the system executing platform from Android to iOS at build time. The

change is made and tested within 2 weeks.

6. SCENARIO CONSOLIDATION:

From the table 2.2, which discusses about various scenarios proposed by Stakeholders, we find that, scenarios 4 and scenario 10 seems relevant to each other.

Scenario 4 and Scenario 10 mainly deals about ‘security’ quality attribute.

In addition to this, we could see that, scenario 8 and scenario 3 seems familiar to each other, as both scenarios discusses about interoperability attribute.

We have consolidated the repeated sounding scenarios from table 2.2. The following table 2.3 displays the updated version of table 2.2 with consolidated scenarios.

Scenario count	Scenario Heading	Scenarios
Scenario 1	Data encryption	A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Scenario 2	Send/Receive messages between users	When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Scenario 3	Portability	The developer is asked to change the system executing platform from Android to iOS at build time. The change is made and tested within 2 weeks.
Scenario 4	User Interface	User Interface changes should be completed within 3 man-hours of effort.
Scenario 5	Fault recovery	During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server

		and triggers an alert mail to the service team for immediate help.
Scenario 6	Introduce concurrency	During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.

SCENARIO PRIORITIZATION:

Among those consolidated scenarios, we find that, availability attribute deserves more weightage as the core component of Fchat will be the server, if the sever becomes unavailable, it will automatically bring down the entire Fchat system.

Hence, we prioritize availability quality attribute in our first priority.

Secondarily we figured out that, Fchat system interoperates with CSUF system to invoke student's and professor's information. Hence, we rate interoperability in the second priority.

Table 2.4 discusses the scenarios in prioritized order.

Scenario Priority	Scenario Heading	Scenarios
Scenario 1	Fault recovery	During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Scenario 2	Send/Receive messages between users	When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Scenario 3	Data encryption	A hacker attempts to read a user's message when the user connected to an unsafe and controlled

		Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Scenario 4	Introduce concurrency	During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.
Scenario 5	Portability	The developer is asked to change the system executing platform from Android to iOS at build time. The change is made and tested within 2 weeks.
Scenario 6	User Interface	User Interface changes should be completed within 3 man-hours of effort.

SCENARIO REFINEMENT:

As Fchat is an instant messaging application, interoperability, availability quality attributes play a major role for the normal functioning of the Fchat.

In addition to this, all the messages are valuable information from the user's perspective, and from the ownership front of Fchat, we find that all the message and file sharing should be secured in-order to preserve the user's information.

'An angry man becomes more angry with poor performance' – based on this factor, from architect's point of view, we feel to improvise the performance of Fchat, to enhance the user chatting experience by providing un-interrupted and super-fast communication.

From the following rationale, we refined the usability and portability scenarios from table 2.4.

Table 2.5 describes the updated refined scenarios with the quality attributes

Scenario Priority	Quality Attribute	Scenario Heading	Scenarios
-------------------	-------------------	------------------	-----------

Scenario 1	Availability	Fault recovery	During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Scenario 2	Interoperability	Send/Receive messages between users	When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Scenario 3	Security	Data encryption	A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Scenario 4	Performance	Introduce concurrency	During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.

END OF QUALITY ASSURANCE WORKSHOP:

- Initially, we discussed various business objects and the necessity of the requirement from the customer
- Post the discussion, we explained the basic architecture plan for building the Fchat architecture.
- Post repeated discussion on various quality attribute scenarios, we initially framed 11 scenarios pertaining to Fchat's operations.
- Among those we consolidated the scenarios in step 6 removing the similar sounding scenarios.
- We prioritized various scenarios and finally refined 4 scenarios from the prioritized scenarios.

FURTHER STEPS:

Based on the refined scenarios we built Quality attribute utility tree using Attribute trade-off analysis methods, which gives an insightful knowledge about our target system.

Table 2.6 describes Quality Attribute Utility tree including the Architecturally significant requirements(ASR), business value, architectural impact, attribute refinement.

Quality Attribute	Attribute Refinement	(Business Value, Architecture Impact)	ASR
Availability	No downtime	(H, H)	Fault recovery During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Interoperability	Information exchange	(H,H)	Send/Receive messages between users When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Security	Data confidentiality	(H,H)	Data encryption A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Performance	Speed	(H,H)	Introduce concurrency During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.

We figured out various tactics and tentative patterns for building Fchat from the Quality attribute Utility tree.

We built Attribute Design Relational Table 2.7, using Utility tree.

ADRT table includes, architecture tactics and patterns and various ASR from Utility tree.

Table 2.7: Attribute Design Relational Table

Quality Attribute	ASR Scenario	Tactics		Architecture Patterns
		Category	Tactics	
Availability	During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.	Fault Recovery	Active Redundancy	Client/Server
Interoperability	INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.	Locate	Discover Services	Service Oriented Architecture
Security	When a hacker tries to access a user's data in a different device. Fchat informs the owner of the profile about the secondary login, requests for an acknowledgement from the user about the additional login. If the user denies the secondary login, the hacker won't be able to read the user's data.	Resist attacks	Encrypt data	Client/Server

Performance	PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.	Manage resources	Introduce Concurrency	Multi-tier
--------------------	--	------------------	-----------------------	------------

EXERCISE 3

CONTEXT:

In this exercise, we performed ADD for Fchat using the utility tree, ADRT, functional and non-functional requirements constructed from the previous exercises. In addition to this, we also documented the architecture for Fchat at the later part of this section.

INPUTS OF ADD:

The inputs of ADD are functional, non-functional requirements and design constraints,

Functional requirements:

- FR14* User should be able to send/receive text with other users in fullerton_chat.
- FR15* User should be able to send/receive audio/video message with other users in fullerton_chat.
- FR16* User should be able to send message even though another user is not active in fchat.
- FR17* Fchat must be able to run between different devices with different operating systems.
- FR18* Fchat should update student's information and staff's information and shall display the same in profile section.
- FR19* User should register in fchat using CWID and phone number.
- FR20* Sharing files/text message should be done securely with some sorts of encryption and decryption.
- FR21* The fchat must be responding fast enough when exchanging texts and sharing audio and video.
- FR22* User interface should be easy to use.
- FR23* Users should be able to change and update the profile picture
- FR24* Users should have the ability to block an unwanted contact in fchat.
- FR25* Users should be able to create a new message group with their contacts.
- FR26* Fchat should display 'ST' tag for students and 'Pf.' Tag for faculties.

Non-functional requirements:

- NFR 5 Availability
- NFR 6 Interoperability
- NFR 7 Security
- NFR 8 Performance

Constraints:

The *design constraints* for Fchat are,

Constraint 4: Fchat must be able to run on different platforms

Constraint 5: Sharing messages should be highly secured

Constraint 6: Time constraints – the deadline for Fchat project is 1st of December 2018.

ADD STEPS:

Step 0:

Fchat stakeholders have prioritized the requirements according to business and mission goals.

We performed Quality Assurance Workshop(QAW) to all our important stakeholders and fetched all the prioritized quality attributes.

STEP 1: CONFIRMING THE REQUIREMENTS INFORMATION:

- Quality Attribute utility tree is constructed from the inputs provided by our stakeholders during QAW.
- All the requirements are prioritized and elements are considered for instantiating in the architecture during design.
- Each quality attribute requirement is expressed in a “stimulus response” form.

Step 2: ITERATION – 1:

- ✓ As this is our first iteration, we are choosing the entire system to decompose.

STEP 3: - IDENTIFYING CANDIDATE ARCHITECTURAL DRIVERS

We constructed Quality Attribute Utility Tree - Attribute Trade off analysis method (ATAM) to predict the Architectural Significant Requirements (ASR) and candidate architectural drivers.

The four quality attribute requirements as the result of QAW workshop are availability, interoperability, performance and security.

Table 3.1 QUALITY ATTRIBUTE UTILITY TREE

Quality Attribute	Attribute Refinement	(Business Value, Architecture Impact)	ASR
----------------------	-------------------------	--	-----

Availability	No downtime	(H, H)	Fault recovery AVA-02: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Interoperability	Information exchange	(H,H)	Scenario: Send/Receive messages between users INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.
Security	Data confidentiality	(H,H)	Scenario 1: Data encryption SEC-01: A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.
Performance	Speed	(H,H)	Introduce concurrency PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.

Step 4: -CHOOSING A DESIGN CONCEPT:

Table 3.2 ARCHITECTURAL DESIGN RELATIONAL TABLE:

Quality Attribute	ASR Scenario	Tactics		Architecture Patterns
		Category	Tactics	
Availability	During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and	Fault Recovery	Active Redundancy	Client/Server

	triggers an alert mail to the service team for immediate help.			
Interoperability	INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.	Locate	Discover Services	Service Oriented Architecture
Security	When a hacker tries to access a user's data in a different device. Fchat informs the owner of the profile about the secondary login, requests for an acknowledgement from the user about the additional login. If the user denies the secondary login, the hacker won't be able to read the user's data.	Resist attacks	Encrypt data	Client/Server
Performance	PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.	Manage resources	Introduce Concurrency	Multi-tier

Table 3.3 Pros. & Cons. Matrix:

ASR Scenario	Client-Server		SOA		Multi-tier	
	Pros.	Cons	Pros.	Cons.	Pros.	Cons.
Availability Scenario AVA-01: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the	Distribute the roles and responsibilities to several standalone computers	NONE	Ability to scale operations to meet different demand levels	NONE	It adds reliability and more independence of the underlying servers or services.	NONE

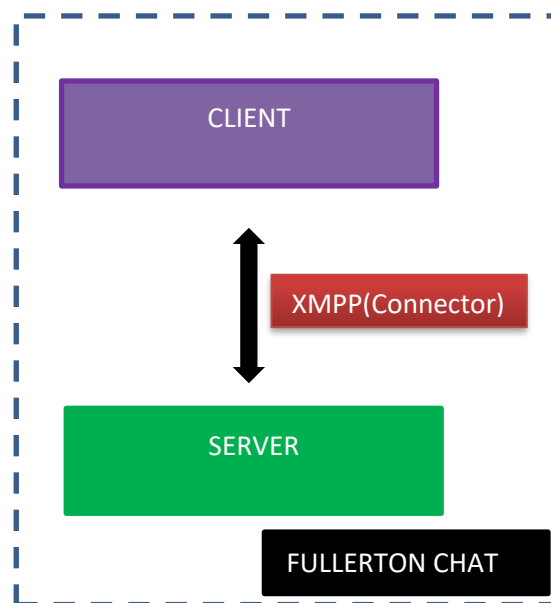
backend spare server and triggers an alert mail to the service team for immediate help.						
Interoperability Scenario INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.	Server side can build interface for data interaction with other system to benefit for interoperability.	NONE	Ability to adapt quickly to different external environments	NONE	NONE	It allows for different development teams to each work on their own areas of expertise.
Security Scenario SEC-01: A hacker attempts to read a user's message when the user connected to an unsafe and controlled Wi-Fi. The hacker fails to understand the messages because the messages are encrypted.	Add data encrypt on client side before data sending, hacker cannot read user's message.	NONE	Don't have much affection on Security	NONE	Security attack detection	NONE
Performance Scenario PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.	We can add load balance and more spare servers to improve the server performance.	NONE	NONE	There is a performance overhead associated with the SOA middleware.	Performance can be improved when separate different module separated to only perform related function.	NONE

Conclusive Statement from Pros. & Cons. Matrix:

Based on the process control matrix table, we find that *Client Server pattern fits best* for instant messaging applications. Client-Server pattern has the ability to distribute the roles and responsibilities to various systems.

Figure 3.1 describes the basic client-server pattern consideration as the result of Pros. & Cons. Matrix table.

Figure 3.1 – Basic Client- Server Pattern consideration for Fchat.



STEP 5: DESIGN STEP

In our Fchat system, we planned to instantiate load balancer server, proxy server, database, and CSUF interface system along-side with Client-Server pattern.

Rationale:

LOAD BALANCER:

- A load balancer acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that *maximizes speed and capacity utilization* and ensures that no one server is overworked, which could degrade performance.
- If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

DATABASE:

- Database acts as a data-repository for Fchat.
- Database stores the user profile information, the dashboard activities of all servers associated with Fchat system.
- In general, as Fchat is an instant messaging application, NoSQL database such as Apache Cassandra is preferred.

CSUF INTERFACE SYSTEM:

- The students or staff's information is seamlessly shared between CSUF system and Fchat.
- Fchat can only access students or staff's information from CSUF system via invoking CSUF system interfaces.

PROXY SERVER:

- A proxy server is a server that sits between a client application, such as a Web browser, and a real server.
- Proxy server intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.
- In Fchat, we preferred to instantiate *Client end proxy server* and *Server end proxy server*.
- In Fchat, due to high security considerations, *High Anonymity Proxy server* is preferred in both client and server.

Figure 3.2 describes the instantiation of Proxy server, load balancer server, CSUF interface and database alongside with Client- Server Pattern.

Figure 3.2 Instantiation Step.

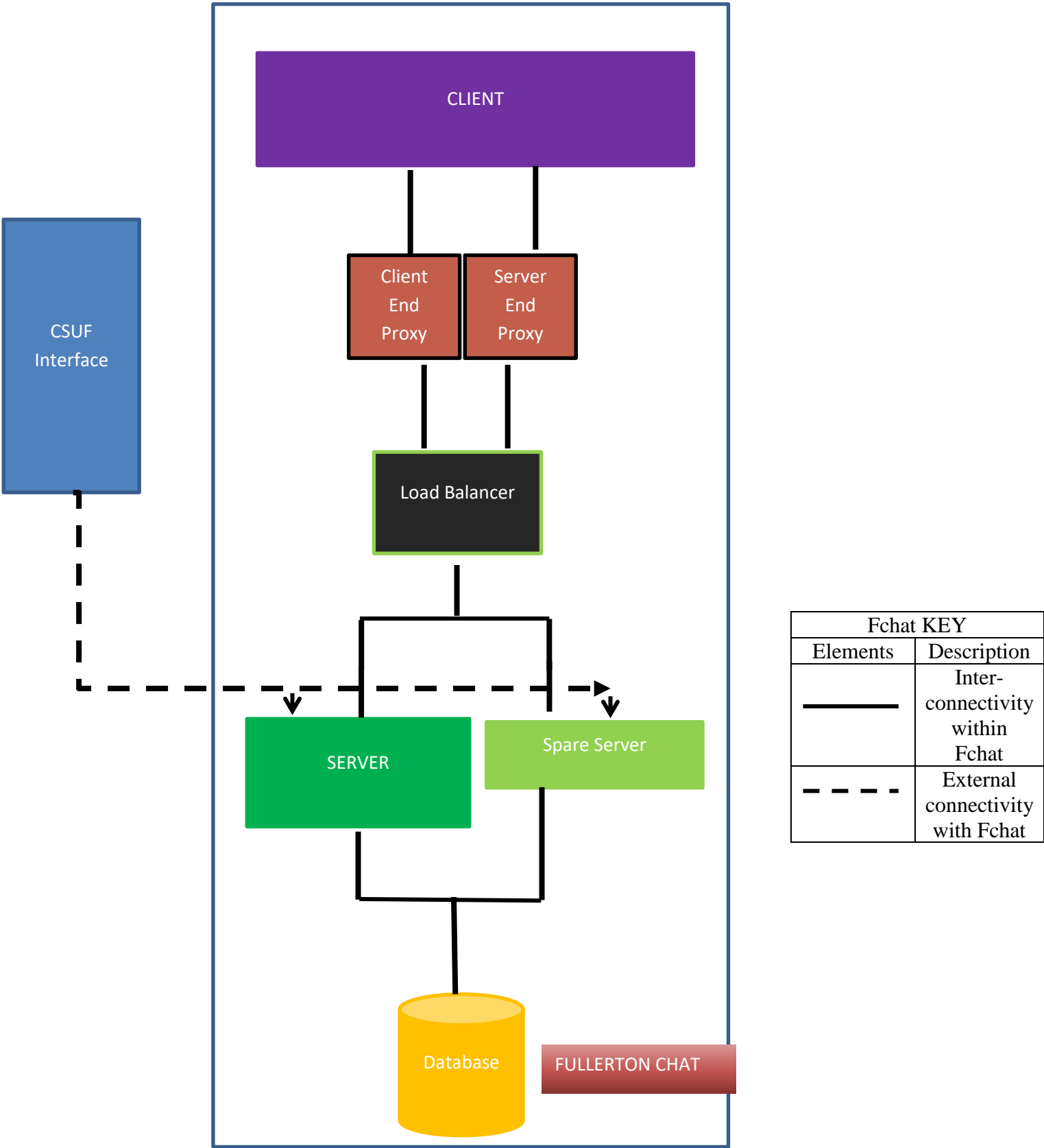


Table 3.4 Instantiation table

Element	Functional requirements	Non-functional requirements	Constraints
Client	FR1, FR2, FR5, FR6, FR7, FR9, FR10, FR11, FR12	NFR1, NFR2, NFR3, NFR4	Constraint No. 2
Server	FR1, FR2, FR3, FR4, FR5, FR6, FR8, FR10, FR11, FR12, FR13	NFR1, NFR2, NFR4,	Constraint No. 2
Proxy Server	FR7	NFR3	
Load Balancer	FR8	NFR4	
Database	FR3	NFR2	
CUSF Interface	FR6	NFR2	

STEP 6: DEFINING INTERFACE BETWEEN THE INSTANTIATED ELEMENTS:

As Fchat is an instant messaging application, the core communication interface between client and the server is XMPP.

Information flow in Fchat through XMPP between Client and Server:

- Flow 1: XMPP carries the message content from the user-1.
- Flow 2: Client end Proxy server checks the proxy information and blocks the information if the incoming request is from a malware.
- Flow 3: The load balancer distributes the load uniformly to all the web servers.
- Flow 4: The web server processes the information and sends it to the user-2 through server end proxy.

Information flow in Fchat through TCP between Web server and the Database:

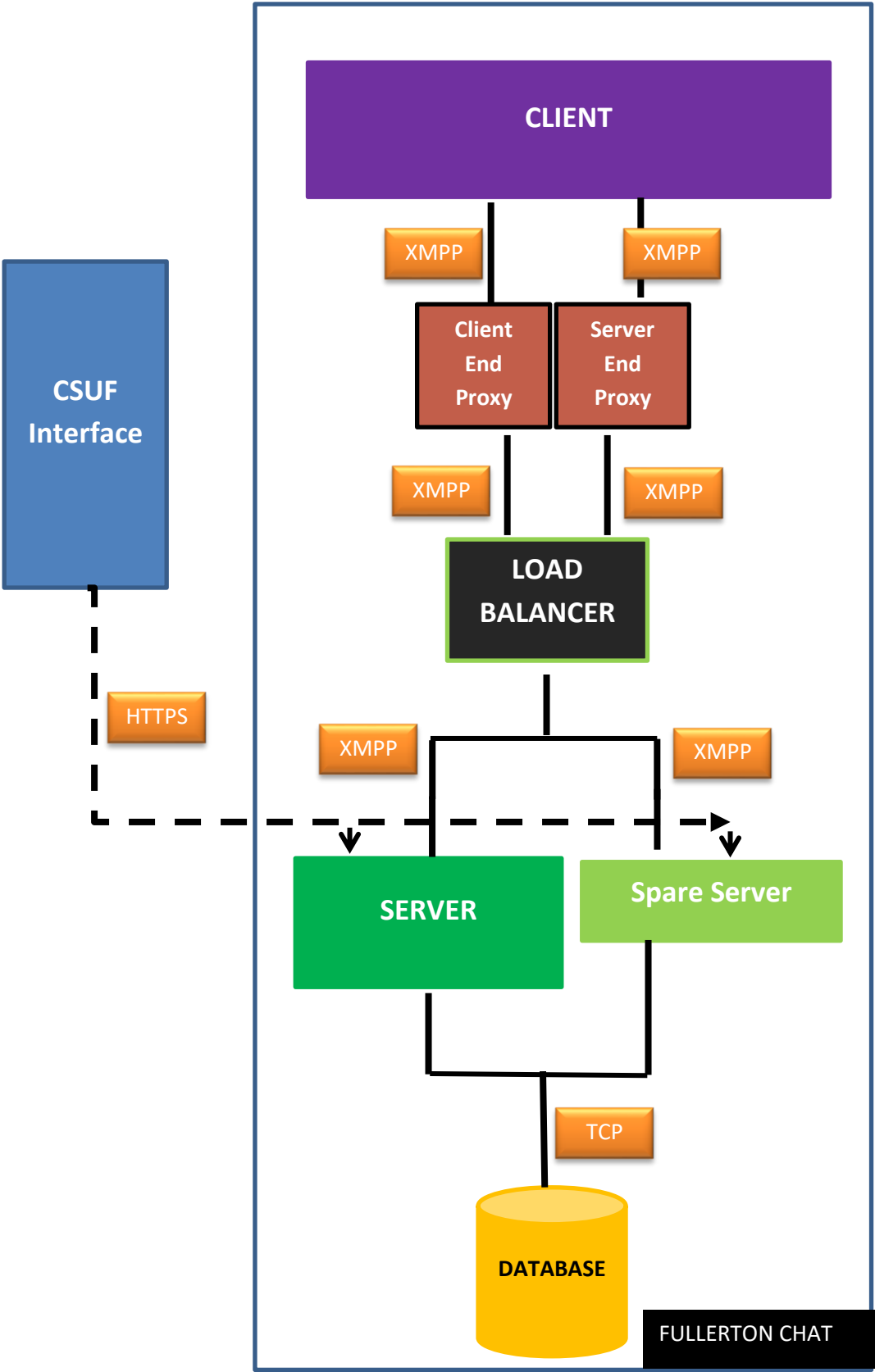
- Flow 1: Server element transmits the data to database through TCP
- Flow 2: TCP works in request and reply fashion.
- Flow 3: Before transmitting/remitting data from database, server sends an TCP request and Database responds back with reply TCP.

Table 3.5 explains in detail about the various interfaces in Fchat.

Interoperating elements		Interface
Element 1	Element 2	
Client	Proxy Server	XMPP
Proxy Server	Load Balancer	XMPP
Load Balancer	Web Server, Spare Web Server	XMPP
Web Server, Spare Web Server	Database	TCP
Web Server, Spare Web Server	CSUF Interface	HTTPS

Figure 3.3 describes the various interfaces instantiated in Fchat.

Figure 3.3 Allocating interfaces to various elements in Fchat.



Fchat KEY	
Elements	Description
—	Inter-connectivity within Fchat
- - - -	External connectivity with Fchat

STEP 7: VERIFYING AND REFINING REQUIREMENTS

SUMMARY OF ITERATION 1:

- ✓ In iteration 1, we decomposed the entire system using Client-Server pattern with connector as XMPP.
- ✓ We instantiated additional individual elements along-side with Client and Server components.
- ✓ Translated the responsibilities assigned to child elements into functional requirements for the individual elements.
- ✓ Refined the quality attribute requirements for individual child elements.
- ✓ Finally, assigned various interfaces elements between the elements to communicate with each other.

Step 8: CARRYING OUT FURTHER ITERATIONS

As we decomposed the entire Fchat system at initial phase of architecture, we will be decomposing the server and client element completely in next iterations.

Iteration 2

STEP 2: CHOOSING SERVER ELEMENT TO DECOMPOSE FIRST

Rationale for choosing Server element to decompose:

We have chosen to decompose Server element at the earliest for the following reasons,

Factor 1: Current knowledge on the architecture

Based on the instantiation table, we find that, Server element possess the maximum weightage of comprising majority of functional and non-functional requirements associated with it. Hence, we choose server element to decompose at the earliest.

Factor 2: Risk and difficulty

- ✓ We infer that, server element is interfacing with load balancer, database, and CSUF interface which are the sub-ordinate elements for Fchat.
- ✓ Additionally, we also find that, constructing the Server element as a whole, would cost much risk and complexity in design.

Factor 3: Business Criteria

- ✓ Server element holds the business logic for processing the incoming XMPP request from the client end.
- ✓ We also find that, Server element demands availability, performance, interoperability in most of the cases.
- ✓ Server element holds majority of the business weightage in Fchat system.

Factor 4: Organizational Criteria

- ✓ Architecting server element will cost up majority of the business value, as Server element holds the business logic for operating the Fchat.
- ✓ Constructing the Server element as a whole, would cost would be tedious, as it holds the business logic for interfacing with load balancer, database, and CSUF interface which are the sub-ordinate elements for Fchat.
- ✓ Server element has the drastic impact on the customer's business, so Server element should be framed with much care.
- ✓ For these reasons, we decide to decompose Server element at the earliest.

STEP 3: - IDENTIFYING CANDIDATE ARCHITECTURAL DRIVERS

The four quality attribute requirements as the result of QAW workshop are availability, interoperability, performance and security.

Table 3.6 QUALITY ATTRIBUTE UTILITY TREE

Quality Attribute	Attribute Refinement	(Business Value, Architecture Impact)	ASR
Availability	No downtime	(H, L)	Scenario 1: Fault detection AVA-01: The Nginx server monitor finds that the apache Cassandra server is nonresponsive process during normal operations (send message). The monitor triggers an alert mail to the service team to repair the issue and continues to operate.
			Scenario 2: Fault recovery AVA-02: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.
Interoperability	Information exchange	(H,H)	Scenario: Send/Receive messages between users INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The

			message has been delivered to user 2 with 99.9% accuracy.
Performance	Speed	(H,H)	Scenario: Introduce concurrency PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.

Step 4: -CHOOSING A DESIGN CONCEPT:

Table 3.2 ARCHITECTURAL DESIGN RELATIONAL TABLE:

Quality Attribute	ASR Scenario	Tactics		Architecture Patterns
		Category	Tactics	
Availability	AVA-01: The Nginx server monitor finds that the apache Cassandra server is nonresponsive process during normal operations (send message). The monitor triggers an alert mail to the service team to repair the issue and continues to operate.	Fault Detection	1.Monitor 2.Ping/Echo	Layer
	AVA-02: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.	Fault Recovery	Active Redundancy	
Interoperability	INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.	Locate	Discover Services	Broker
Performance	PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.	Manage resources	Introduce Concurrency	Multi-tier

Table 3.3 Pros. & Cons. Matrix:

ASR Scenario	Layer	Broker	Multi-tier
--------------	-------	--------	------------

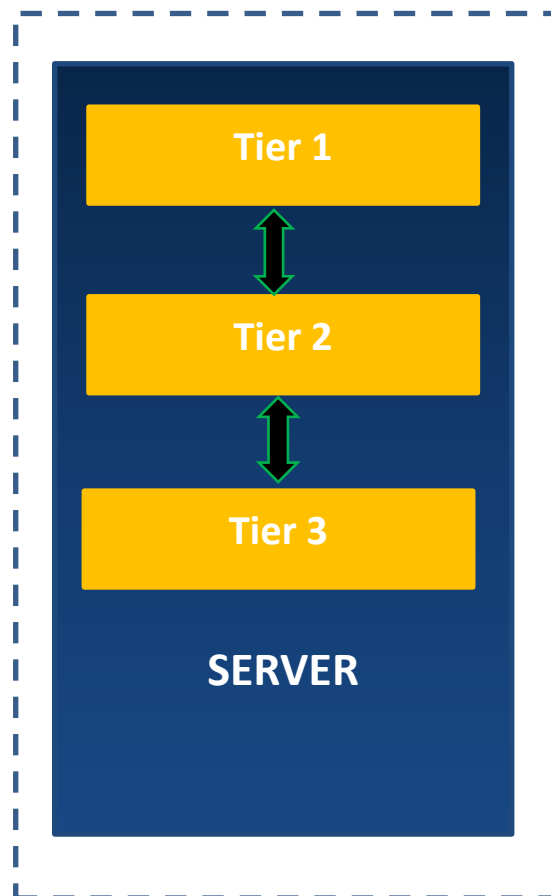
	Pros.	Cons	Pros.	Cons.	Pros.	Cons.
Availability Scenario AVA-02: During peak operating conditions, the load balancer monitor finds that one of the backend servers is crashed. The load balancer automatically distributes the load to the backend spare server and triggers an alert mail to the service team for immediate help.	Separated layer for different functions, make availability easy to control and be enhanced.	NONE.	NONE	Broker can be a single failure point of the server, which is a risk for availability.	It adds reliability and more independence of the underlying servers or services, it's good for availability.	NONE
Interoperability Scenario INT-01: When User 1 sends a text message to user 2 through fchat during normal operating conditions. The fchat system delivers the message by locating user 2. The message has been delivered to user 2 with 99.9% accuracy.	Server side can build separated layer for data interaction with other system to benefit for interoperability.	NONE	Ability to adapt quickly to different external system interaction. Good for interoperability.	NONE	Different tier can focus of its own function, the tier for interaction can be considered and built specially for interoperability, so it's benefit.	NONE
Performance Scenario PER-01: During normal/peak operations, Fchat process the incoming message requests using multiple backend servers operating concurrently. The load will be distributed equally on all the backend servers ensuring an average latency of two seconds.	NONE	Add complexity to the server by using layer, not good for performance.	NONE	There is a performance bottleneck in broker to cause risk in performance.	Performance can be improved when separate different module separated to only perform related function. Good for performance.	NONE

Conclusive Statement from Pros. & Cons. Matrix:

Based on the Pros.& Cons. matrix table, we find that *Multi-tier pattern fits best* for decomposing into server element. Server element holds the principal business logic for processing the incoming XMPP request. In addition to this, Server element also interfaces with database for storing the messages, communicates with load balancer for intaking the XMPP requests, and interacts with CSUF interface for invoking the student and staff information. Server element carries out multiple functionalities as a whole, and hence we choose Multi-tier pattern to decompose with Server element.

Figure 3.1 describes the basic Multi-tier pattern consideration as the result of Pros. & Cons. Matrix table.

Figure 3.4 – Basic Multi-Tier patter consideration for decomposing server element in Fchat.



STEP 5: DESIGN STEP

While decomposing the server element, we plan to add three sub-elements as Presentation Tier, Business Tier and Data-Access Tier in our Multi-tier architecture.

Rationale:

PRESENTATION TIER:

- The presentation tier consists of various interface sub-elements through which the server communicates with other elements.
- Among those, the server mainly communicates with Load Balancer for fetching the XMPP requests from client end. In addition to this, the server also communicates with CSUF interface for invoking the Student and professor profile information.
- For these reasons we planned to instantiate *load balancer interface* and *CSUF system interface*

BUSINESS TIER:

- Fchat system's server element is comprised of various functionalities. The business tier in fchat is responsible for carrying out all the business logic that runs the Fchat.
- In business tier, we have two modules, one module for core-business logic and the another one for handling the user requests.
- *Core business logic* module includes the main business program such processing the incoming XMPP requests, and sending it to the destination client.
- Whereas, there is another module named '*Managing user profile*', which handles the requests from the user, such as changing the display picture and storing it in the database, creating a group, blocking another user, integrating with CSUF interface and updating the same in Fchat and tracking user's data.
- For example, if a user is currently enrolled in his final spring semester and the user is getting graduated, the 'User Profile' module take care of those features and updates the user profile information in Fchat.

DATABASE ACCESS TIER:

- Database acts as a data-repository for Fchat.
- Database stores the user profile information, the dashboard activities of all servers associated with Fchat system.
- Database access tier acts as an interface for accessing the Fchat database
- We included *Database communication interface*, through which the server element communicates with Fchat database.

Figure 3.5 describes the instantiation of Presentation tier, Business Tier and Data-access tier with Multi-tier patter together in Server element.

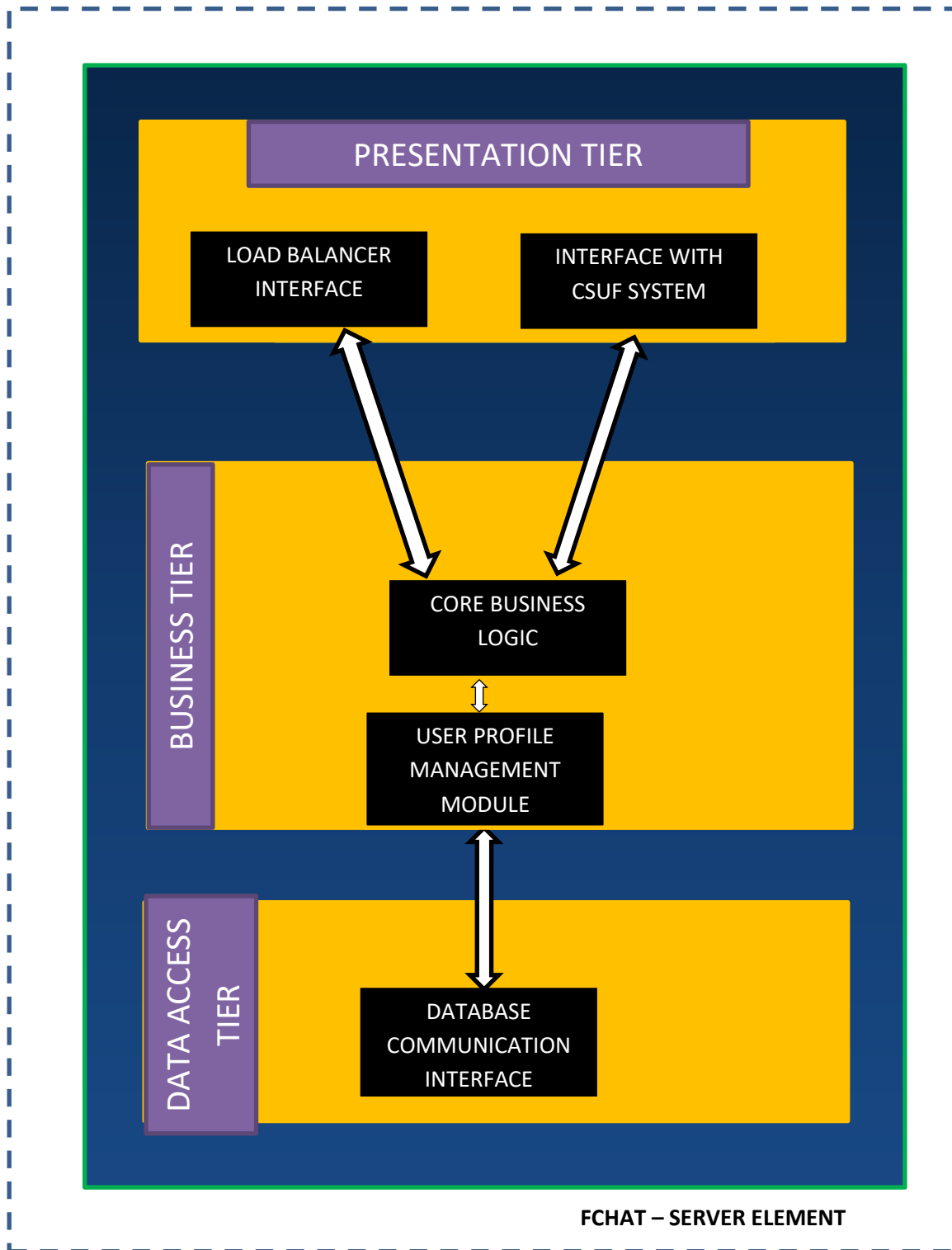


Table 3.9 describes the instantiation of various functional and non-functional requirements associated with server element.

Element	Functional requirements	Non-functional requirements	Constraints
Load Balancer Interface	FR1, FR2, FR3, FR4, FR6, FR8, FR12.	NFR1, NFR4	
Interface with CSUF system	FR1, FR2, FR3, FR5, FR6, FR12	NFR1	
Core Business Logic Module	FR1, FR2, FR4, FR7	NFR4, NFR1	Constraint No. 2
User profile Management module	FR3, FR5, FR7, FR10, FR11, FR12	NFR4	
Database Access interface	FR3, FR6, FR10, FR11, FR12	NFR1	

STEP 6: DEFINING INTERFACE BETWEEN THE INSTANTIATED ELEMENTS:

As Fchat is an instant messaging application, the core communication interface between client and the server is XMPP. However, we also use TCP for communicating with Fchat Database.

Information flow in Fchat Server through XMPP between load balancer interface and Business logic module:

- Flow 1: XMPP carries the message content from the user-1.
- Flow 2: The load balancer distributes the load uniformly to all the web servers.
- Flow 3: The load balancer interface receives the XMPP request distributed by load balancer in the server and transfers the load to the business logic module.
- Flow 4: The load The business logic module processes the XMPP request and reverts it back to the load balancer interface.
- Flow 5: The load balancer interface transfers the information to the load balancer server.

Information flow in Fchat Server through HTTPS between CSUF interface system and Business logic module:

- Flow 1: CSUF system transmits the student's and staff's information to the database through HTTPS.
- Flow 2: CSUF system converts the data into HTTPS packets and transmits the data securely.
- Flow 3: HTTPS is known for secure data transmission.

Information flow in Fchat Server through TCP between User profile management module and the Database communication interface:

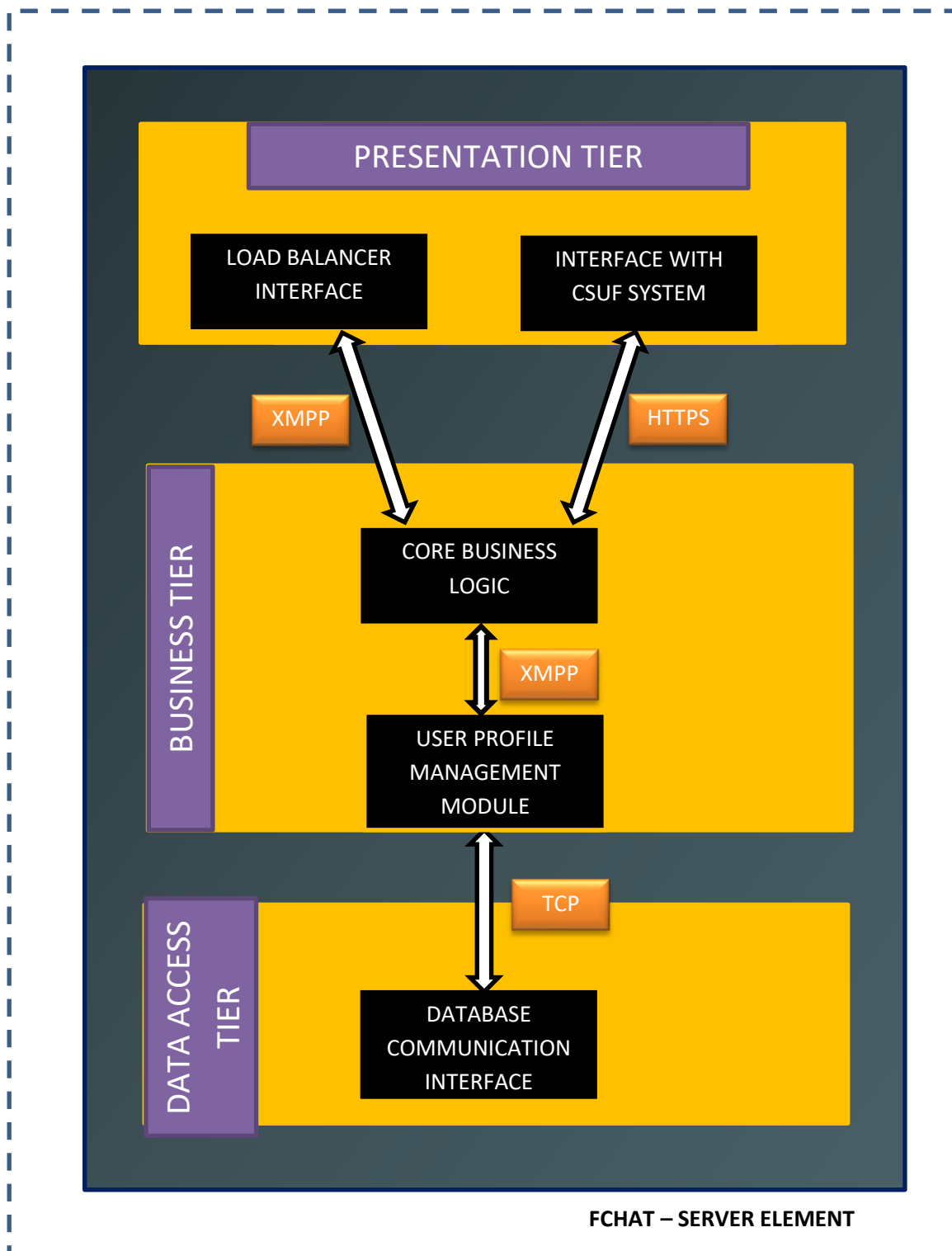
Information flow in Fchat through TCP between Web server and the Database:

- Flow 1: Server element transmits the data to database through TCP
- Flow 2: TCP works in request and reply fashion.
- Flow 3: Before transmitting/remitting data from database, server sends an TCP request and Database responds back with reply TCP.

Table 3.10 explains in detail about the various interfaces in Fchat.

Interoperating elements		Interface
Element 1	Element 2	
Load Balancer Interface	Core Business Logic Module	XMPP
Interface with CSUF system	Core Business Logic Module	HTTPS
Core Business Logic Module	User profile Management module	XMPP
User profile Management module	Database Access interface	TCP

Figure 3.6 describes the interface allocation between various modules in Multi-tier pattern.



STEP 7: VERIFYING AND REFINING REQUIREMENTS

SUMMARY OF ITERATION 2:

- ✓ In iteration 2, we decomposed the server element in Fchat system using Multi-tier pattern.
- ✓ We instantiated 3 tiers – Presentation tier, business tier, data access tier.
- ✓ Presentation tier includes interfaces for communicating with Load balancer and CSUF system
- ✓ Business Tier includes the core business logic module and user profile management module.
- ✓ Database tier consists of database communication interface for communicating with Fchat database.
- ✓ In addition to this, we translated the responsibilities assigned to child elements into functional requirements for the individual elements.
- ✓ Refined the quality attribute requirements for individual child elements.
- ✓ Finally, assigned various interfaces elements between the elements to communicate with each other.

Step 8: CARRYING OUT FURTHER ITERATIONS

As we decomposed the server element at the second phase of architecture, we will be decomposing the client element completely in next iteration.

Iteration 3

STEP 2: CHOOSING CLIENT ELEMENT TO DECOMPOSE:

For the third iteration, client element needs to be decomposed as the final element in the Fchat.

Client element in fchat includes user interface through which user communicates in Fchat. In addition to this, client element also includes message encryption functionality ensuring safety and security for the information processed through Fchat.

STEP 3: - IDENTIFYING CANDIDATE ARCHITECTURAL DRIVERS

We constructed Quality Attribute Utility Tree - Attribute Trade off analysis method (ATAM) to predict the Architectural Significant Requirements (ASR) and candidate architectural drivers.

The four quality attribute requirements are availability, interoperability, performance, modifiability and security.

Table 3.11 QUALITY ATTRIBUTE UTILITY TREE

Quality Attribute	Attribute Refinement	(Business Value, Architecture Impact)	ASR
Availability	No downtime	(H, H)	<p>A-01:</p> <p>The "send message" button in fChat App is nonresponsive because of a bug in the released code. The developers find and fix the bug.</p>
Interoperability	Information exchange	(H, H)	<p>I-01:</p> <p>messages exchange between fChat and csuf system.</p> <p>As a csuf system's client, fChat fetch the student's and professor's profile from csuf's system with 99.9% accuracy.</p>
Security	Data confidentiality	(H, H)	<p>S-01:</p> <p>Data encryption</p> <p>The message from the end-user from be encrypted before network transmission. So even the hacker get the message, the hacker cannot understand the message.</p>
Performance	Speed	(H, H)	<p>P-01:</p> <p>Introduce concurrency</p> <p>During peak operations, the response time of each http request is less than 2 seconds.</p>

Step 4: -CHOOSING A DESIGN CONCEPT:

Table 3.12 ARCHITECTURAL DESIGN RELATIONAL TABLE:

Quality Attribute	Tactics		Architecture Patterns
	Category	Tactics	
Availability	Detect Faults	1. Timestamp 2. Sanity Checking	SOA
	Prevent Faults	Removal from service	
Interoperability	Manage Interfaces	Orchestrate	Module-View-Controller
Security	Detect Attacks	Detect Message Delay	Module-View-Controller
	Resist attacks	Encrypt data	
Performance	Manage resources	Introduce Concurrency	SOA
		Increase Resources	

TABLE 3.13 PROS. & CONS. MATRIX:

ASR	MVC	SOA		
	Pros.	Cons	Pros.	Cons.
A-01	Modification does not affect the entire model.	NONE	Ability to scale operations to meet different demand levels	NONE
I-01	Faster development process.	NONE	Ability to adapt quickly to different external environments	NONE
S-01	Ability to provide multiple views		NONE	NONE
P-01	MVC also supports asynchronous technique, which helps developers to develop an application that loads very fast.	NONE	NONE	There is a performance overhead associated with the SOA middleware.

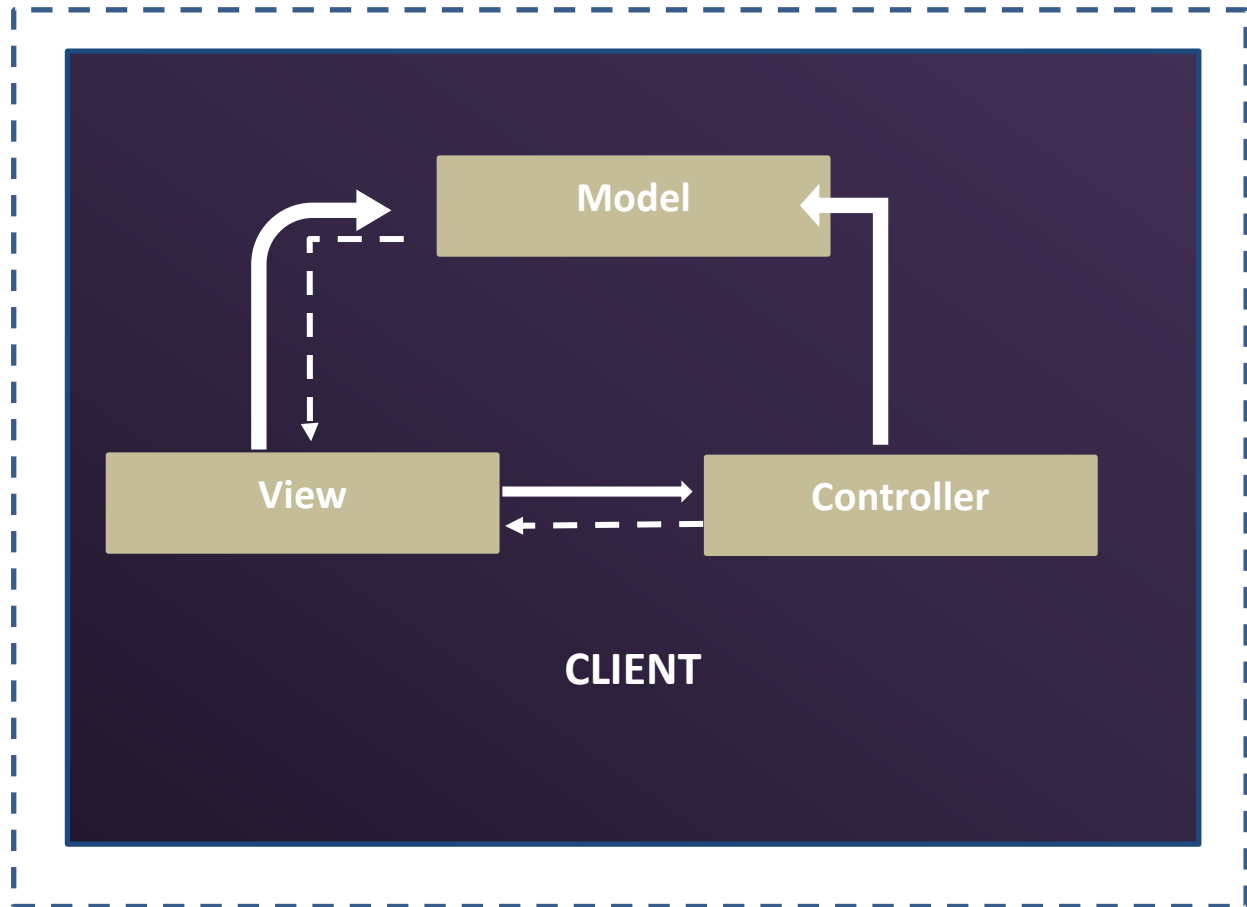
Conclusive Statement from Pros. & Cons. Matrix:



“

Based on the Pros.& Cons. matrix table, we find that **Module – View – Controller pattern fits best** for decomposing the client element. Client element primarily consists of user interface with various screens and application logic. In addition to this, client element also includes message encryption/decryption functionality alongside with proxy server interface. Client element carries out multiple functionalities as a whole, and hence we choose Module-View-Controller pattern to decompose with Server element.

Figure 3.11 describes the basic Module-View-Controller pattern consideration as the result of Pros. & Cons. Matrix table.

Figure 3.11 – Basic Multi-Tier patter consideration for decomposing server element in Fchat.



Client KEY	
Connectors	Description
	Method Invocations
	Events

STEP 5: DESIGN STEP

The module view controller pattern has three components. They are discussed in detail as follows,

CONTROLLER COMPONENT:

- The controller represents data and the rules that govern access to and updates the data. In enterprise software, a controller often serves as a software approximation of a real-world process.
- The controller component includes the interface for communicating with the view component, functionality for encrypting the user's messages/multimedia messages, Interface for communicating with the proxy server.

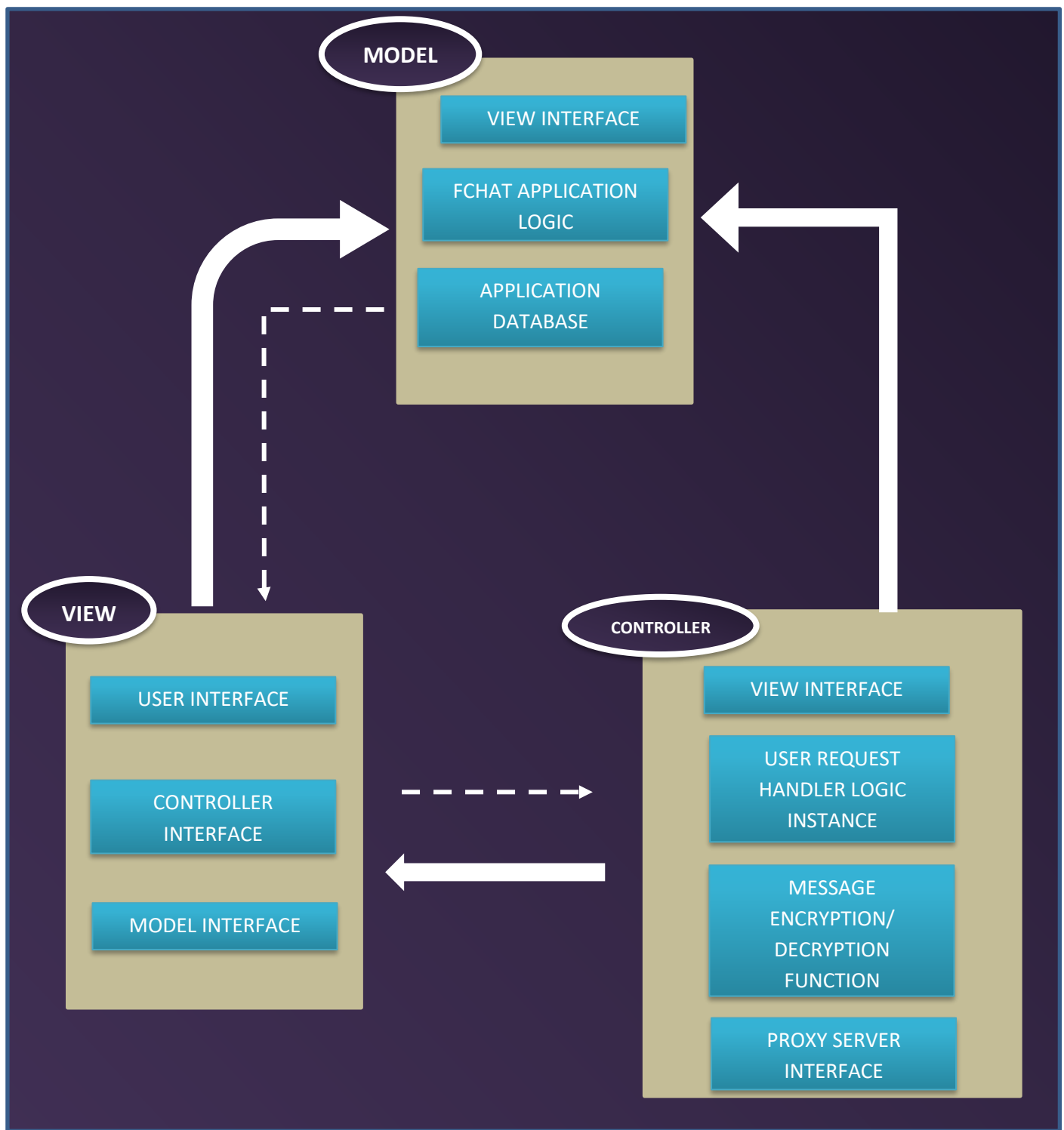
VIEW COMPONENT:

- The view renders the contents of a model. It specifies exactly how the model data should be presented. If the model data changes, the view must update its presentation as needed.
- The user mainly interacts with view component and inputs the message through view component.

MODEL COMPONENT:

- The model translates the user's interactions with the view into actions that the model instance will perform.
- The model component in client component contains the application logic and also includes message encryption functionality ensuring safety and security for the information processed through Fchat.

Figure 3.12 describes the instantiation of Model, View, Controller pattern together in Client Element.



FCHAT – CLIENT ELEMENT

Table 3.14 describes the instantiation of various functional and non-functional requirements associated with server element.

Element	Functional requirements	Non-functional requirements	Constraints
Model	FR5, FR6, FR12, FR11, FR13	NFR4, NFR2	
View	FR1, FR2, FR5, FR6, FR9, FR10, FR11, FR13	NFR4	
Controller	FR1, FR2, FR5, FR6, FR7, FR10, FR13	NFR4, NFR1, NFR3	Constraint No. 2

STEP 6: DEFINING INTERFACE BETWEEN THE INSTANTIATED ELEMENTS:

As we followed Module, View, Controller pattern for decomposing the entire client element, the connectors in MVC pattern will be events and method invocations.

Events:

An Event is defined as the interaction between a service consumer (User) and a service provider (Fchat) is normally initiated by the service consumer as it needs to respond to an event that occurs within the boundary of the service consumer itself.

User's gestures are captured as events and it will be processed in Model component through controller.

Invoking methods:

Invocation is the event of issuing the call to the method; technically - placing the method onto the stack. A method only starts executing after invocation is successful.

Figure 3.13 describes the interface allocation in MVC pattern.

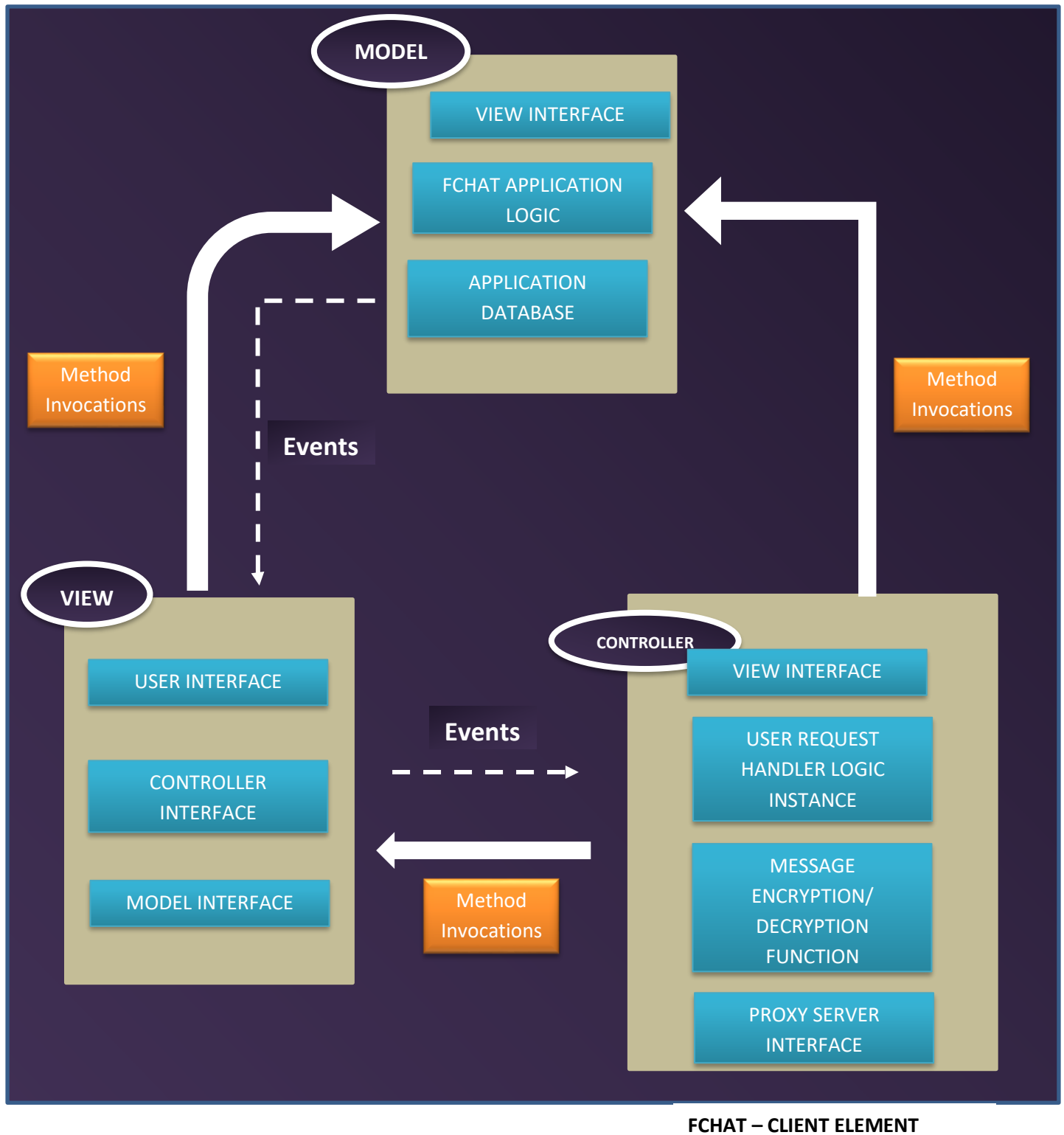


Table 3.15 explains in detail about the various interfaces in Fchat.

Interoperating elements		Interface
Element 1	Element 2	
Model	View	Event
View	Controller	Event
Controller	Model	Event

STEP 7: VERIFYING AND REFINING REQUIREMENTS

SUMMARY OF ITERATION 3:

- ✓ In iteration 3, we decomposed the client element in Fchat system using MVC pattern.
- ✓ In addition to this, we translated the responsibilities assigned to child elements into functional requirements for the individual elements.
- ✓ Refined the quality attribute requirements for individual child elements.
- ✓ Finally, assigned various interfaces elements between the elements to communicate with each other.

Step 8: CARRYING OUT FURTHER ITERATIONS

As we decomposed entire Fchat system, including client element and server element, we conclude that we completed deriving the Fchat architecture.

DOCUMENTING THE ARCHITECTURE:

As we have performed all the ADD steps and derived an architecture for Fchat application, we plan to document the architecture for further purposes.

- We have chosen customer and End user as the main stakeholders who are mainly interested in reading our architecture document.

CHOOSING THE VIEWS:

- **Step 1: Building stakeholder/view table:**

Based on our research, as Fchat is mainly concerned about Availability, Interoperability, Security and Performance and Fchat is mainly constructed based on Component and Connector pattern, we are choosing component and connector view and allocation views as our primary focus for documenting the Fchat.

We have chosen component and connector views and Allocation view that includes some details for the customers and end users.

Table 3.16 Stakeholder – View table

Stakeholders	Views	
	Component and connector	Allocation View
	Various C & C views	Deployment view
Customer	Some Details	Overview information
End user	Some Details	Some Details

- **Step 2: Combining views:**
 - From the table we could see that, customer and end user requires some details about the architecture.
 - As per the rule, marginal details and similar information can be combined. Hence, we are combining the deployment views for both Customer and End users together as single view.
 - We are also combining the component and connector views for both customer and end users, both stakeholders share same views.

Table 3.17 The updated Stakeholder/view table

Stakeholders	Views	
	Component and connector	Allocation View
	Various C & C views	Deployment view
Customer	Some Details	
End user	Some Details	

Step 3: Prioritize and Stage

Fchat is mainly constructed based on Client and Server pattern which is a branch of component and connector pattern. We are prioritizing component and connector pattern to document first, as C&C pattern explains the architecture in a much insightful way.

Because C&C views all show runtime relations among components and connectors of various types, they tend to combine well. Different (separate) C&C views tend to show different parts of the system, or tend to show decomposition refinements of components in other views.

BUILDING THE DOCUMENTATION PACKAGE

The documentation package has five sections, those sections are as follows:

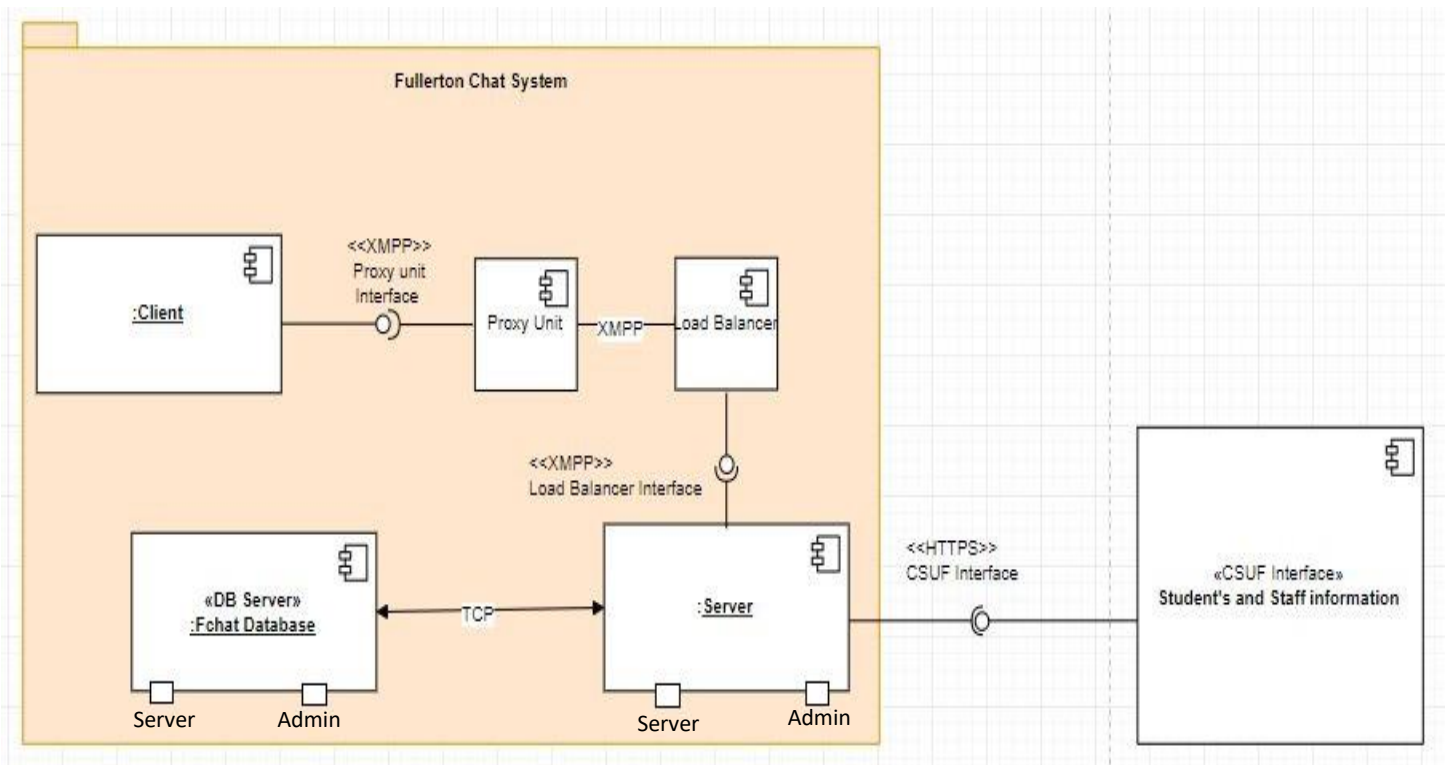
Section 1: Primary Presentation:

Fullerton_Chat' (fchat) falls under Instant Messaging domain. Instant messaging is a set of communication technologies used for text-based communication between two or more participants over the Internet or other types of networks.

Fchat architecture has been build primarily from Client – Server architectural pattern. Fchat makes use of additional components such as proxy server, database and load balancer for satisfying the Non-functional and functional requirements.

Figure 3.14 displays the primary presentation for Fchat.

Figure 3.14 Primary Presentation of Fchat:



Section 2: The Element Catalog

Elements and their properties:

Fchat system primarily consists of 5 main components, they are as follows,

SERVER:

- Server element possess the maximum weightage of comprising majority of functional and non-functional requirements associated with it.
- server element is interfacing with load balancer, database, and CSUF interface which are the sub-ordinate elements for Fchat.
- Server element holds the business logic for processing the incoming XMPP request from the client end.
- Server element demands availability, performance, interoperability in most of the cases.
- Server element has the drastic impact on the customer's business

CLIENT COMPONENT:

- Client element in fchat includes user interface through which user communicates in Fchat.
- Client element also includes message encryption functionality ensuring safety and security for the information processed through Fchat.
- Client element acts as the trigger for Fchat, it carries the user's input messages through XMPP.

- This XMPP request will be processed by the server.

LOAD BALANCER:

- A load balancer acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that *maximizes speed and capacity utilization* and ensures that no one server is overworked, which could degrade performance.
- If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

DATABASE:

- Database acts as a data-repository for Fchat.
- Database stores the user profile information, the dashboard activities of all servers associated with Fchat system.
- In general, as Fchat is an instant messaging application, NoSQL database such as Apache Cassandra is preferred.

CSUF INTERFACE SYSTEM:

- The students or staff's information is seamlessly shared between CSUF system and Fchat.
- Fchat can only access students or staff's information from CSUF system via invoking CSUF system interfaces.

PROXY SERVER:

- A proxy server is a server that sits between a client application, such as a Web browser, and a real server.
- Proxy server intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.
- In Fchat, we preferred to instantiate *Client end proxy server* and *Server end proxy server*.
- In Fchat, due to high security considerations, *High Anonymity Proxy server* is preferred in both client and server.

Relations and their properties:

XMPP:

- XMPP is abbreviated as eXtensible Message Presence Protocol
- XMPP is originally developed as messaging platform, and the primary communication of this protocol is short messages between server and client or between client and client.
- XMPP is reactive to user's presence and status
- Client component converts the user input message content into XMPP packets.
- These XMPP packets are processed in the server.

HTTPS:

- HTTP Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP) for secure communication over a computer network, and is widely used on the Internet.
- Fchat system uses HTTPS as the communication medium to invoke the Student's and staff's data from CSUF system.

TCP:

- The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP).
- Fchat uses TCP for communicating with Fchat Database.
- TCP generally follows request and reply mechanism.

Element Interfaces:

LOAD BALANCER INTERFACE

- Server component consists of Load Balancer Interface for communicating with the Load balancer.
- Through Load balancer interface the server receives the incoming XMPP requests.

PROXY INTERFACE:

- Client component consists of Proxy Server Interface for communicating with the Proxy Server
- Through Proxy Server interface the client component sends and receives the XMPP requests.

CSUF INTERFACE:

- Server component consists of CSUF Interface for communicating with the CSUF system
- Through CSUF interface the server component invokes the student's and staff's information.

SECTION 3: CONTEXT DIAGRAM

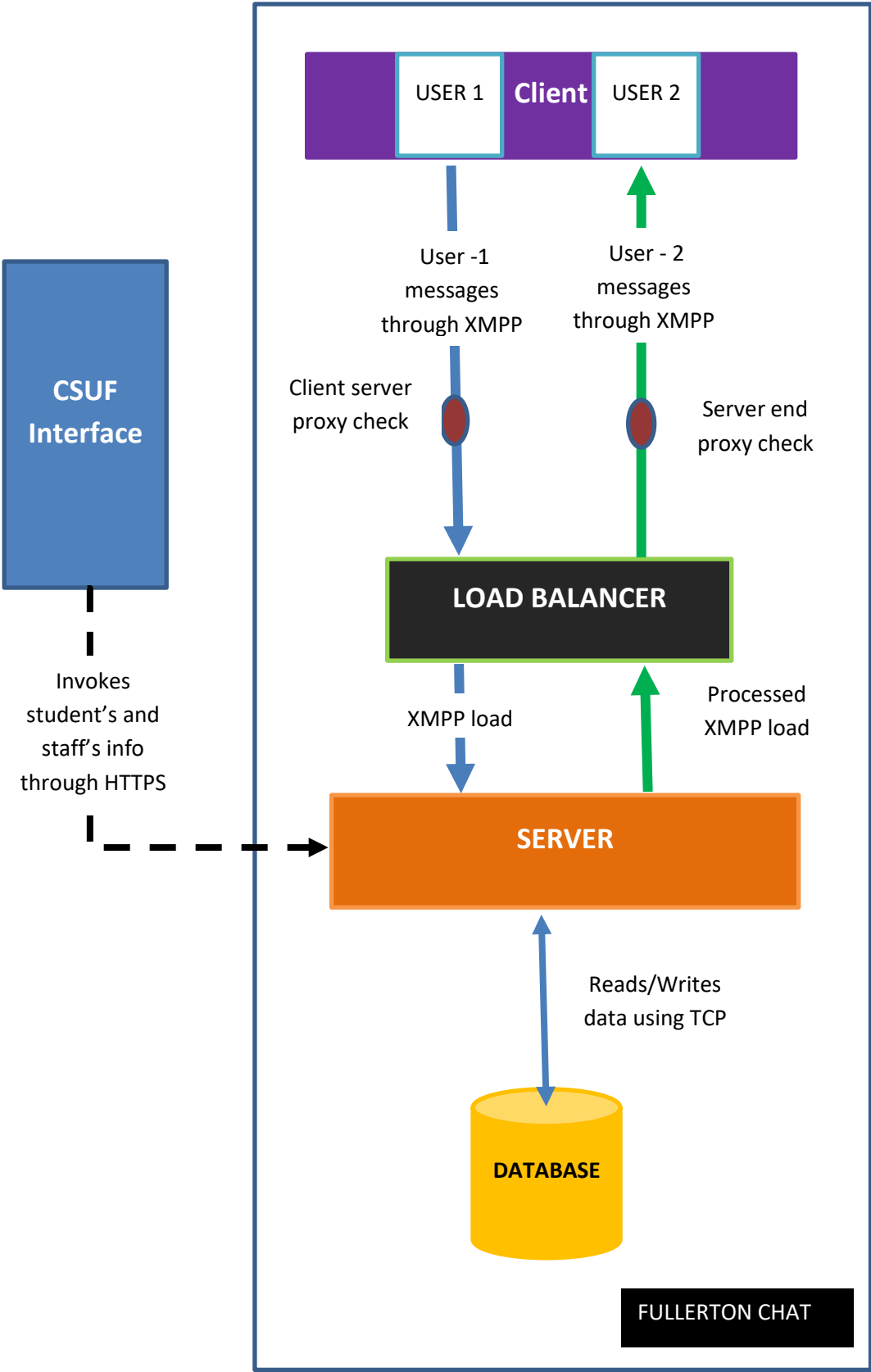
- The context diagram describes the normal functioning of Fchat.
- End users are the clients for Fchat.
- The client component in Fchat is divided into user 1 and user 2, where various users send and receive messages.



Working Strategy:

- User 1 initiates the chat with user 2.
- Client converts user 1's message into XMPP packets and transmits to the Server.
- During XMPP packet transmission, proxy will be checked and the load balancer provides the XMPP load to the server, based on the server's response time
- Server processes the XMPP packets and transmits them to User 2.

- The communication time between user 1 and user 2 in Fchat is less than 1 second during normal operations.

Figure 3.15 CONTEXT DIAGRAM OF FCHAT



Fchat KEY	
Data flow	Description
	Inter-connectivity within Fchat
	External connectivity with Fchat

SECTION 4: VARIABILITY GUIDE

Add/Remove users in group chat:

- Adding and removing various users in group chat will be the one of the primary important features of Fchat.
- End user tend to add their friends and remove their friends frequently depending on the need.
- Client component initiates the request and Server component processes the request, all the group details will be stored in Fchat database under user's profile.

Block users/ Change display pictures:

- Block users/ Change display pictures for end users in Fchat will be the one of the primary important features of Fchat.
- End user tend Block users/ Change display pictures frequently depending on the need.
- Client component initiates the request and Server component processes the request, all the details will be stored in Fchat database under user's profile.

CSUF interface:

- Fchat frequently invokes the student's and staff's information from CSUF system.
- If there is a change in CSUF system, we need to update our Fchat system depending on the need.

Fchat Database:

- We need to consider expanding Fchat database, if the Fchat subscriber count increases.
- Fchat is generally targeted for CSUF student's and staffs.

Section 5: Rationale:

- Fchat is an instant messaging application.
- We carried out ADD v.2 for developing the Fchat architecture.
- We conducted Quality Assurance Workshop (QAW), Attribute Design Relation Table (ADRT) and Pros. & Cons. Matrix (PCM) for figuring out the architecture pattern.
- We finally concluded that Client Server pattern fits best for instant messaging applications.
- Client-Server pattern has the ability to distribute the roles and responsibilities to various systems.
- As Client-Server pattern is a branch of component and connector family, Fchat is expressed in Component and connector view.
- We also find that; component and connector view expresses the quality attributes of Fchat in a pellucid fashion.

DOCUMENTING DEPLOYMENT VIEW:

As mentioned earlier, we planned to document deployment view of Fchat.

Elements

The various software elements in Fchat are,

Elements	Description
Client	Client element in fchat includes user interface through which user communicates in Fchat. Client element also includes message encryption functionality ensuring safety and security for the information processed through Fchat.
Server	Server element possess the maximum weightage of comprising majority of functional and non-functional requirements associated with it.
Fchat DB	Database stores the user profile information, the dashboard activities of all servers associated with Fchat system.
Load balancer	A load balancer acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that <i>maximizes speed and capacity utilization</i> and ensures that no one server is overworked, which could degrade performance.
Proxy DB	Client elelemt consists of Proxy Server Interface for communicating with the Proxy Server

environmental element—computing hardware

Environmental elements	Description
LAN	Internet connectivity is necessary for normal functioning of Fchat. As Fchat is an instant messaging application, without internet connectivity Fchat wont be available to the user.

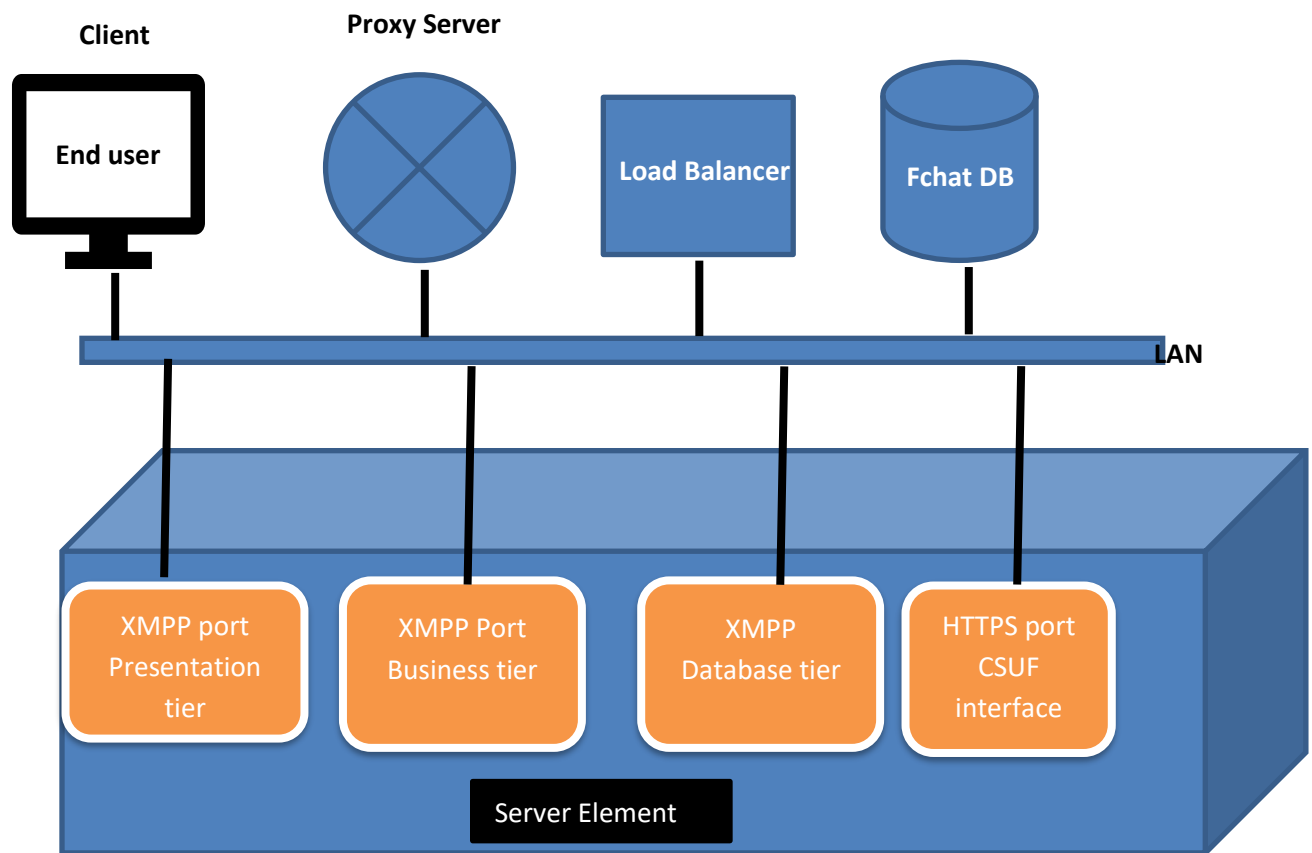
User's device	User installs Fchat application from website or from google play store
Server computer	Server computer holds the business logic and the interface connectivity with other devices
Database server machine	Database server holds the logic for holding the user profile information and other information of Fchat
XMPP port	XMPP port is available in every device through which the user's message flows. XMPP port carries XMPP packets from client to server
HTTPS port	HTTPS ports helps the Fchat system to invoke the student's and staff's information
TCP port	TCP port helps the server element to communicate with Fchat DB.

RELATIONS:

Table 3.18 describes the relations between various elements in Fchat.

Elements		Relations
Element 1	Element 2	
Client	Proxy Server	XMPP
Proxy Server	Load Balancer	XMPP
Load Balancer	Web Server, Spare Web Server	XMPP
Web Server, Spare Web Server	Database	TCP
Web Server, Spare Web Server	CSUF Interface	HTTPS

Figure 3.16 Deployment view of Fchat



DOCUMENTING BEYOND THE VIEWS:

SECTION 1: DOCUMENTATION ROADMAP

SCOPE AND SUMMARY:

This document discusses about the architecture of Fchat. Fchat is an instant messaging application, that is constructed using ADD v2. The intended target readers for this document are customers and end users. We have constructed Fchat using client-server pattern (family of component and connector) and hence, we have provided the detailed information for various components and connectors in Fchat.

We have also included the deployment view of Fchat, which discusses about the message transmission between various users in Fchat. We have not provided the module view of Fchat, as our stakeholders Customers and End users demand only component and connector view and deployment view of Fchat.

ORGANIZED WAY OF THE DOCUMENT:

- This document consists of the primary presentation, which includes the component and connector view of Fchat.
- The element catalog describes about the various information and properties about components and connectors, relations between component and connectors.
- Context diagram explains about the functionality of Fchat, it depicts the working strategy of Fchat in an insightful way.
- A variability guide shows how to exercise any variation points that are a part of the Fchat architecture shown in component and connector, deployment view.
- Rationale section explains the conclusive decision for arriving to architecture design, choice of patterns and the choice of views.

HOW STAKEHOLDERS CAN USE THE DOCUMENT?

- The document is mainly framed for the customers and end users.
- Customers and end users share equal weightage as we have combined the views.
- We have depicted component and connector view and deployment view for both customer and end user.
- Document is framed to explain the various components in fchat and the deployment procedure of Fchat.

SECTION 2: HOW VIEW IS DOCUMENTED?

- We have followed the view template for documenting the architecture.
- This document consists of the primary presentation, which includes the component and connector view of Fchat.

- The element catalog describes about the various information and properties about components and connectors, relations between component and connectors.
- Context diagram explains about the functionality of Fchat, it depicts the working strategy of Fchat in an insightful way.
- A variability guide shows how to exercise any variation points that are a part of the Fchat architecture shown in component and connector, deployment view.
- Rationale section explains the conclusive decision for arriving to architecture design, choice of patterns and the choice of views.

SECTION 3: SYSTEM OVERVIEW:

- Fchat is an instant messaging application which helps users to communicate with each other.
- Fchat flexibly runs on both mobile phones and computers.
- Fchat allows users to send and receive text and voice messages, and to share pictures, audio and video among multiple users.
- Fchat must be able to run between different devices with different operating systems.
- Fchat follows XMPP as the primary communication medium for sharing messages. For security reasons, we have adapted encryption and decryption functionality for sharing XMPP packets.
- Fchat system is fast enough when exchanging texts and sharing audio and video messages, during normal operations the responsive time of Fchat for sending and receiving messages between two users is less than two seconds.

SECTION 4: MAPPING BETWEEN VIEWS

- With reference to our main stakeholders customer and end user we have constructed component and connector view, deployment views for Fchat.
- Component and connector views explains the various components and provides the functionality of Fchat system.
- Deployment view explains the operation of Fchat.
- Both ‘component and connector’ and deployment view can be mapped together, as component and connector view explains in detail about various components and connectors. However, deployment view explains the operating conditions of Fchat with to those components.

SECTION 5: RATIONALE

We find that, Fchat system is mainly developed from Attribute driven Design method version 2. As per the customer’s specifications, Fchat architecture was instructed to build with Availability, Interoperability, Security and Performance quality attributes. Hence, we tried to document the Fchat which mainly discusses those views in common.

While picking those views we figured out that, Component and connector view, deployment view together focuses mainly on the quality attributes mentioned by the customer, hence we planned to document those views for Fchat.

SECTION 6: DIRECTORY:

Acronym	Description
Fchat	Fullerton Chat
ADD	Attribute Driven Design
XMPP	EXtensible Message Presence Protocol
TCP	Transmission Control Protocol
HTTPS	Hyper Text Transfer Protocol
CSUF	California State University Fullerton
IP	Internet Protocol

LESSONS LEARNED

- **Lesson learned by Deepak**

- Experience on designing architecture:

- I have been following professor's class works regularly. I find software architecting is one of the challenging jobs and I'm shaping myself to be a successful software architect in these forth coming days. I feel very blessed to have software architecture in my graduate coursework. After understanding the basics of architecture, ADD v2, TOGAF and architecture description, I started Homework#2*

- It was really challenging to build an architecture for Fchat, as I dint have much technical background, I had to explore a lot before beginning it, we planned to come up with some catchy features. We visualized a rough architecture before beginning ADD v2.*

- Later, we performed QAW and once we framed Utility Tree, PCM and ADRT we find that designing architecture was easy, as we had ASRs with which guided us to be confident with our work.*

- Learning from Homework#2

- Understood and had a hands-on practice in implementing an architecture. I feel pretty confident with building a new architecture. I advocate using ADD method to perform architecture design.*

- About team

- I took charge of leading a fabulous team. All my team members are elder to me in terms of age and experience. It was challenging to extract work from my team members. I allotted tasks for individuals and we set up homework discussion every-week in college library. My team members supported me in all problematic situations and we together shared our views on software architecture. I was the only Indian and I'm new to United States, I never been away from my country. However, I became a good friend to my teammates in short span.*

- Future works:

- As I designed an architecture, I feel to implement Fchat in realtime, so that I can owe the ownership of a real time software. While implementing Fchat, I also feel to learn programming languages, which will help to shape up my career. I would like to explore cloud architecture patterns and also have some plans to learn AWS cloud platform.*

- **Lesson learned by ChongBei Wang**

From HW2, I learned the following things: ADD2, Utility Tree, PCM, ADRT, and architectural patterns.

- **ADD2**

ADD (Attribute-Driven Design). ADD is an approach to define a software architecture based on software's quality attribute requirements. The inputs for ADD are functional requirements, design constraints, and quality attribute requirements. The output of ADD is a system design in terms of the roles, responsibilities, properties, and relationships among software elements. Add is a recursive process that we need to decompose a system or system element each time and make sure each element satisfies its driving quality attribute requirements. There are usually 7 steps in ADD2, and the main work-products for each step is as the following:

- ✓ *In step2, we need draw system context diagram.*
- ✓ *In step3, we need to select 5 or 6 high-priority requirements as the candidate architectural drivers and construct quality attribute utility tree that represents the overall quality of the system.*
- ✓ *In step4, we need to build PCM (Pros and Cons Matrix) and ADRT (Architectural Design Relation Table).*

- **Utility Tree**

Using Utility tree is a good approach to identify, document and prioritize quality attributes. The root node in the tree is labeled Utility, then followed by quality attributes, quality attributes concerns, priority and scenarios. The most important part of the utility tree is scenarios because the scenarios help us understand the quality attributes needed and make these goals both concrete and measurable.

- **PCM**

We use PCM (Pros and Cons Matrix) to evaluate candidate patterns. We put the quality attributes to the left of the table, then write down the pros and cons of architectural pattern for each attribute.

- **Architectural patterns**

I learned some architectural patterns in HW2, such as service-oriented architecture pattern, multi-ties pattern, layer pattern, peer-to-peer pattern, publish-subscribe pattern. Each pattern is not perfect and has its own benefits and costs. We should choose the patterns based on your individual quality attributes and scenarios.

- **What you like to study more related to this in the future.**

In the future, I want to learn how to implement different architectural patterns in my project. For example, I want to build my personal website. I think model-view-controller is a good choice for my website. I will learn NodeJS and its framework such as VUE to implement MVC pattern in my personal website.

HW2 lesson learned by Guangyi Shang

- *Your experience with architectural design through HW*

Through the software architecture HW 2, I learned to find quality attribute from the requirement document and communication with stakeholders. The Quality Attribute Workshop is very important for gathering stakeholders to make group interview and discuss the quality attribute, identify the Architectural Drivers, brainstorm to find the scenario, make consolidation and prioritization. Also, I have learned to perform ADD method by constructing Quality Attribute Utility Tree, Architectural Design Relation Table and creating pros and cons matrix table to get the architectural decision of selecting patterns and tactics. Following ADD method, we decompose the big software into parts, then instantiate architectural elements and allocate responsibilities, define interfaces and verify. After one iteration, we continue the second iteration and the third iteration. I learn how to make architectural design one step by one step.

- *What you learn from HW (and from this class)*

Through the homework and class, I have learned the process of designing architecture for a system. Understand and practice how to find ASRs and candidate patterns through Utility Tree, ADRT and PCM. And using ADD method to perform architecture design.

- *What you like to study more related to this in the future.*

I would like to study more about common architecture design patterns and tactics. Compare with them and find the advantage and disadvantage of each patterns and the proper kind of system of each pattern. And I also plan to read more architecture design document of other common software or system to learn how other experienced software architect design architecture.

- *What your team learned as a group*

I learned the importance of collaboration and communication in a team. For example, organize team meeting, explain the requirement in homework exercise, listen the team members' idea, satisfy team member, then assign individual tasks, discuss for different opinions, and learn from other's advantages.

HW2 lesson learned by Xiaosong Lu

Your experience with architectural design through HW

Software architecture is a very important and difficult part of software engineering. Although professors are very careful, but practice through HW is the best way to improve our understanding of software architecture concepts and methods. Learn and practice how to find ASR and architecture patterns through Utility Tree, ADRT and PCM. And use the ADD method to perform the architectural design. I also learned to split a complex project into many small parts, and then through the teamwork step by step to complete the entire project. This methodological approach to learning through theory and then completing a predetermined project is our most valuable experience.

What you learn from HW (and from this class)

In the previous HW1 and exercises, we learned how to standardize the system from the original idea of a project into the original architecture, but also understand the meaning and methods of some professional terms, such as ADRT, ADD, PCM, etc. These skills have been enhanced and improved in HW2. Through the learning of the software architecture HW2, we have particularly enhanced our understanding of ADD: how to obtain the required quality attribute requirements from stakeholders through appropriate collection methods at first, identify the architecture drivers, find the corresponding scenarios, and most importantly The priority is based on their respective factors. Through the ADD method, we first establish an evaluation matrix to distinguish the advantages and disadvantages of the method. Then we can split a large project into a number of small parts, and then decompose each small part. After a few iterations, we can step by step building the software architecture.

What you like to study more related to this in the future.

For me personally, I would like to read more about designing software engineering industry cases to consolidate the knowledge and methods I have learned in class 545. I also plan to become familiar with various methods, architecture patterns, and architecture strategies by gradually completing more small projects on my own. Only through continuous practice can we better understand various methods, the advantages and disadvantages of various models, and work in the future. In order to more efficiently choose the most appropriate method and mode.

What your team learned as a group

Through this semester team project cooperation, we learned how to improve the efficiency of

team work: Our four members of the team have completely different professional backgrounds: computer science, software engineering, electronic engineering and mechanical engineering; from two countries: India and China; have different work backgrounds and work habits. At the beginning of our communication, there were many obstacles. Understanding each other's views was like a bad business analyst getting product requirements from an unprofessional customer as difficult as it was, but as mutual understanding gradually increased, we found each other. The effective communication method greatly enhances the efficiency and accuracy. According to each person's strengths and weaknesses, the corresponding tasks and workload are allocated. In the end, we successfully completed all the tasks for the current semester. These valuable experiences will play a multiplier role in our future career life.

REFERENCE LIST

- [1] ISO/IEC/IEEE 42010-2011(E), Systems and software engineering — Architecture description
- [2] Wiki: [Attribute-driven-design](#)
- [3] [ADD](#)
- [4] A paper on A COMPARISON OF ENTERPRISE ARCHITECTURE FRAMEWORKS Lise Urbaczewski, Stevan Mrdalj, Eastern Michigan University.
- [5] Presentation on “Quality Attribute Design Primitives and the Attribute Driven Design Method” by Len Bass, Mark Klein, and Felix Bachmann.
- [6] Documenting component and connector views :
https://resources.sei.cmu.edu/asset_files/TechnicalReport/2004_005_001_14387.pdf
- [7] Overview of component and connector views
<https://sites.google.com/site/softwarearchitectureinpractice/9-documenting-software-architecture/c-component-and-connector-c-c-views>
- [8] <https://www.slideshare.net/udayslideshare/whatsapps-architecture>

TEAM CHARTER

COURSE TITLE	CPSC 545 Software Design & Architecture	All team members participated in the creation of this charter and agree with its content. Date 1/31/2018
INSTRUCTOR	Dr. Chang-Hyun Jo	
COURSE DATES	1/22/2018 – 5/11/2018	

Team Members (Contact Information)

NAME	ADDRESS (CITY, STATE, COUNTRY)	PHONE	CELL	EMAIL
Deepak Prasanna Mayanattanmy Yagneswaran	Fullerton, CA	6572536335	6572536335	dee7@csu.fullerton.edu
Xiaosong Lu	Irvine, CA	9497714564	9497714564	Xiaosong.lu@csu.fullerton.edu
Chongbei Wang	Diamond Bar, CA	9096558628	9096558628	chongbeiwang@csu.fullerton.edu
Guangyi Shang	San Gabriel, CA	6266208116	6266208116	guangyi.shang@csu.fullerton.edu

Team Member Skill Inventory (Areas individual members can contribute)

Deepak Prasanna Mayanattanmy Yagneswaran	<ul style="list-style-type: none"> • Project Management • Web programming, Database design and development • SAP developer and administrator • Full stack developer
Xiaosong Lu	<ul style="list-style-type: none"> • MS Word Knowledge • ADD Knowledge/ • Programming Languages (Java, Python) • Machine Learning
Chongbei Wang	<ul style="list-style-type: none"> • J2EE(Java8, Spring, Springboot, Mybatis and so on) • Database(Mysql, Oracle, Redis) • Frontend(HTML, CSS, Javascript) • Programming Languages (Java) • Backend Experience
Guangyi Shang	<ul style="list-style-type: none"> • Solid working knowledge of OS (Solaris, Linux, Windows) • Database operation and maintenance (Sybase, Oracle, SQL) • Workstations, Storage, IP Networking, Virtualization. • Veritas High availability system. • Familiarity with operations and maintenance for Network Management System in telecom domain.

	<ul style="list-style-type: none"> • Basic knowledge in C and Python programming language.
--	---

Team Goals (Project goals, team process goals, quality goals, etc.)

<ul style="list-style-type: none"> • Understand the software architecture knowledge such as software quality attributes, Software architecture life cycle, software architecture and business, etc. • Be familiar with the progress of a software architecture project and prepare for your future career • Enhance teamwork, especially the efficiency of cross-cultural team cooperation, team cohesion • Learn from other teams and make up for each other's weaknesses (both hard skill and soft skill) • Complete a high quality team paper for final assignments

Team Roles (Define roles of each member to achieve goals)

Deepak Prasanna Mayanattanmy Yagneswaran	<ul style="list-style-type: none"> • Project Manager, Responsible for assigning tasks to team members, supervise the completion of tasks, and help solve the problem • Identify the tasks to be completed, the project's timeline and schedule. • Document any ideas and key points, and share with other team members. • Remind team members to submit assignments by deadline. • Ensure team members know the purpose of the team and the overall goal. • Focusing team members on the purpose and task of the project.
Xiaosong Lu	<ul style="list-style-type: none"> • Communicate with professors and define the goals and tasks of the project • Team member management, responsibility assignment, and other team exchanges and cooperation • Clears organizational barriers that may impact the team.
Chongbei Wang	<ul style="list-style-type: none"> • Team Charter • Title Page • PCM, Utility tree, ADRT. • Make sure each teammate has a chance to present his opinion during the meeting. • Make sure the meeting solves the problems each player encounters and the problems the entire project is experiencing • Remind team of their progress and ask for input.
Guangyi Shang	<ul style="list-style-type: none"> • QAW discussion for figure out ASRs and scenarios. • Perform ADD method for iteration 1 and 2, build UT/ADRT/PCM. • Review condidated pattern and pattern selection analyze and discuss.. • Act as one of the stakeholders to review the output design document.

	<ul style="list-style-type: none"> • Team Evaluation document preparation share by google doc and reminder for filling in. • Review document from other team members, and give suggestion.
--	--

Ground Rules (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

- All team members must attend each team meeting on time.
- All team members expect participation and commitment.
- The team will discuss current and upcoming projects or tasks (chat or conference call) at least weekly (initial schedule: Wednesday-5pm).
- Team members will be notified in advance if they can not attend the scheduled meeting.
- Membership team members should check their e-mail at least daily to maintain their best.
- Member team members should reply to the email within 24 hours.
- Membership team members will be completed within two weeks prior to the team's mission deadline.
- Team All team members will be responsible for some of their projects and are expected to complete and try their best to do their job.
- Interrupting notifications may result in the emergency team failing to complete the due date or meeting. The rest of the team will do their best to participate in the team's mission.
- Each semester, different people are nominated for the leadership team.
- Team The team must always be open, clear and effective communication.
- Assist teammates in need.
- Respect the suggestions of other members and adopt constructive criticism and encouragement to maintain a positive, honest and open atmosphere.
- Do not copy. Each member must be responsible for avoiding / preventing plagiarism.

Time Commitments/Availability (Pacific Time)

Deepak Prasanna Mayanattanmy Yagneswaran	<ul style="list-style-type: none"> • On Campus Wednesday 3:45-5:00 • On-Line M-Sun, 8pm-9pm (can make other arrangements as necessary)
Xiaosong Lu	<ul style="list-style-type: none"> • On Campus Wednesday 3:45-5:00. • On-Line M-Sun, 8pm-9pm (can make other arrangements as necessary)
Chongbei Wang	<ul style="list-style-type: none"> • On Campus Wednesday 3:45-5:00. • On-Line M-Sun, 8pm-9pm (can make other arrangements as necessary)
Guangyi Shang	<ul style="list-style-type: none"> • On Campus Wednesday 3:45-5:00. • On-Line M-Sun, 8pm-9pm (can make other arrangements as necessary)

Conflict Management (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

- In order to avoid conflicts, clear roles and responsibilities must be assigned to avoid confusion.
- If a team member does not perform the task, the team leader will talk to the member and try to solve the problem.
- If there is a conflict, please submit them to the entire team so that everyone can help solve the problem in a peaceful and good manner.
- All team members must resolve conflicts within the team as soon as possible.

Risk Management (What are potential barriers to the achievement of these goals?)

- List the opportunities or possibilities of loss or danger in the project.
- Identify risks.
- Assess risk (likelihood, consequence, occurrence, urgency, manageability, dependency, etc.)
- How to avoid / prevent risks?
- How to manage risk? How to reduce the risk? How to manage risk?
- Risk mitigation plans to reduce risk before it becomes a problem. (To reduce risk)
- (In response to the risk) o Describe actions that the project (and team) might take to deal with the problem that has occurred.)

Team Evaluation Criteria (List evaluation criteria that will be used to evaluate team members objectively.)

- Evaluate objectively by objective evidence as defined here (e.g., team meeting log, documents, email record, etc.), not by subjective opinions.
- Scoring is based on their own ratings and others scoring, the total score for the sum of all scores
- Scoring based on each person's role and responsibilities
- According to each person's workload, each stage (each HW) give stage score, in the final project to give a final project score
- Depending on the quality of each team member's work and whether it is done on time, each stage (each HW stage) give a stage score, in the final project to give a final project score
-

Team Evaluation

CPSC 545 – Spring 2018						
Self-Evaluation of Team						
Submission Date:	05/03/2018					
See instructions below. Do not mess with the cells with red text. They are formulas.						You can customize the form (by adjusting the number of columns and rows, total, and average) according to your team's situation. However, total must be [100 x number of members]
Members	Xiaosong Lu	Deepak Prasanna Mayanattanmy Yagneswaran	Guangyi Shang	Chongbei Wang	Total	Comments on Your Evaluation on Team
Evaluators						
Xiaosong Lu	80	110	100	110	400	
Deepak Prasanna Mayanattanmy Yagneswaran	98	100	100	102	400	Planning and execution needs improvement
Guangyi Shang	100	100	100	100	400	Everyone is nice and kind. And we can improve more in team communication and cooperation.
Chongbei Wang	95	115	95	95	400	Cooperation needs improvement
Total	373	425	395	407	1600	
Max	400	400	400	400	400	
Average	93.25	106.25	98.75	101.75	400.00	
Percent	93.25%	106.25%	98.75%	101.75%	100.00%	
Signature	song	Deepak	Shang	chongbei		

Comments on Your Score Earned from Team						
--	--	--	--	--	--	--