Software Design & Architecture

Final Report

By Team 06

John Hunsaker
Thomas Abrahams
Zhangying He
Kena Kamdar

Professor: Dr. Chang-Hyun Jo
Department of Computer Science
California State University, Fullerton

May 2, 2018

Table of Contents

Exercise 1: Quality Attributes

Vision

The Fullerton Chat application will provide a safe space for CSU Fullerton students to share their opinions, audio, video, pictures, and other media with their peers easily, fast, and securely. The vision statement is as follows:

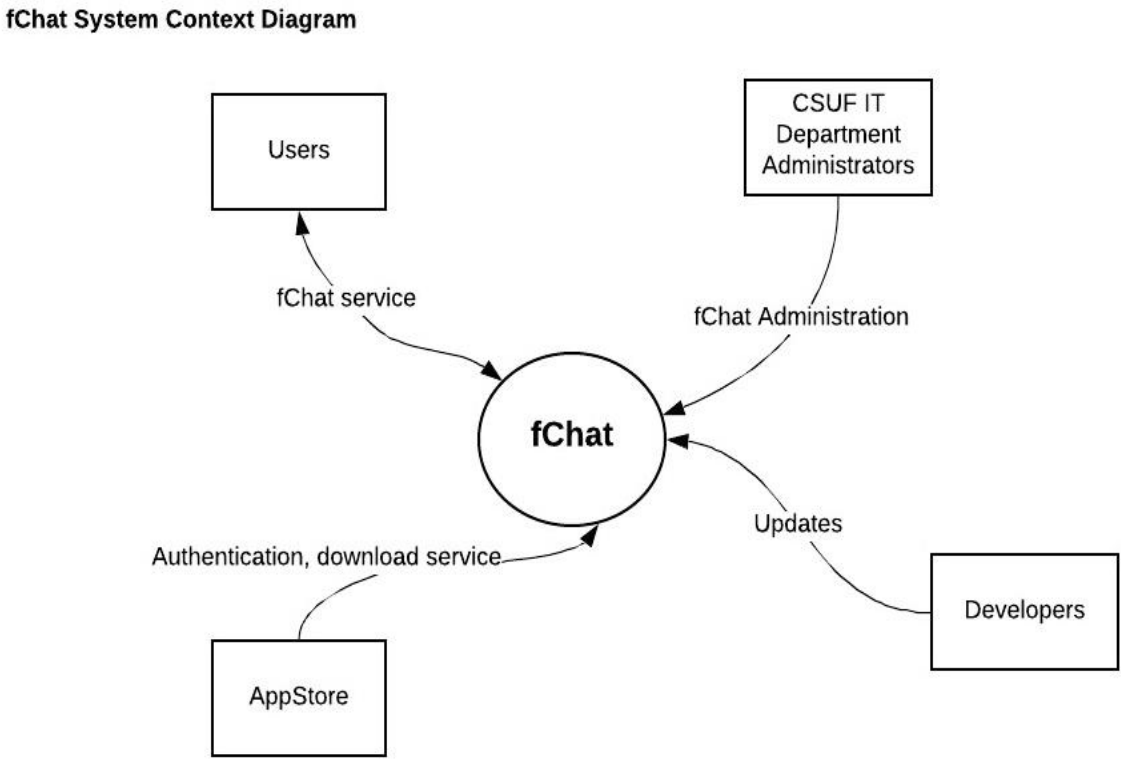
For students at CSU Fullerton who wish to conversate with other students, the Fullerton Chat application is a messaging and media share application that allows peers to share media safely, quickly, and securely. Unlike Facebook, our product is made with the CSU Fullerton student in mind, specifically making it lightweight and low impact, taking up low system resources and removing features that would drain battery life or impact performance on any system.

By reducing the scope and focusing on one particular social group, the application can be made lightweight. It does not need to account for all the features needed by other competitors such as Google+ or Facebook. By only including what is needed for the students in Fullerton, the User Interface is reduced, and the amount of computation client side is also reduced. The design will be modular so that in later versions, the program can be expanded at the users' choice. This application will allow others to message others, share media between other users, allow for the creation of a unique account for each user, to make an individual profile for the user, join and make chat rooms, be able to save a group of friends to allow easy texting and media sharing between those the user trust's, and block individuals that may be disruptive to the application experience. These functional requirements can only be accomplished through a solid base of non-functional requirements. The Fullerton Chat application is designed to be lightweight and expandable for the future to meet student needs.

The non-functional requirements for this application must be able to meet several properties such as: availability, interoperability, modifiability, performance, security, testability, mobility, usability, an portability. In order for a messaging system to be successful, the application has to work near one hundred percent of the time. This is accomplished by making server up time for the application being a number one priority in order to address this issue. Without this, the application will not function as basic authentication will not work. As this will be on multiple platforms, interoperability between different operating systems is an essential attribute that must be maintained by the creation of multiple versions, or by finding a language that will allow for multiple versions to be created from one code base (Web languages, Xamarin etc.). Modifiability will be a day one consideration and the encouragement of a modular program will be used in order to succeed with the addition of new features as needed and to allow for perfective maintenance to be performed with little problem to remaining systems. By looking at competitors' response time and how long it takes for the system to perform tasks, we will be able to determine what is needed in terms of performance. With a lightweight frame that does the essentials really well, our group aims to allow the product to perform quickly for communication, media sharing, and authentication. Regarding accounts, all usernames and passwords will be encrypted and salted within our databases and use modern security standards in order to ensure the information shared by each user is kept secure by the best practices of the software industry. This will lead to testing the application on its' performance, security, and reliability

due to the concrete examples provided by other products within this domain. This testability of the product due to its' modular nature combined with what is considered acceptable within the domain will allow stakeholders to view how effective this product is without creating major problems within development. Usability of the product will be enhanced compared to other products as it only focuses on one group of individuals; CSU Fullerton students. This will remove any other functionality that would not benefit this particular group. Without extra unneeded options, the user interface will need to support less options, allowing for a UI that is minimalistic. With proper artistic choices, the minimalistic user interface will increase the usability of the application. Finally, due to the lightweight design and low resources utilization by the application, every mobile version will be optimized for battery life, allowing mobile users to have an increase in the mobility of the application. During the design phase, different hardware or operating systems will be taken into account due to the modular design, with minimalistic coupling of each module. Combining this with the testability of the system, portability, the ease of effort to port the software to another specification, whether it is different operating systems or hardware will be significantly higher, meaning it will be easier to port the application to other configurations.

System Context Diagram



Technical Environment

The Cyber Intelligence Sharing and Protection Act is a law introduced in November 2011, with the stated goal of giving the United States Federal Government more options and resources to ensure the security of networks against attacks.

Relevant excerpts from the Digital Service Playbook, as laid out from the Office of the Federal Chief Information Officer (policy.cio.gov/web-policy/governance):

- "Establish Integral Digital Governance: A strong governance structure will help agencies develop coherent priorities, set up lines of accountability, and satisfy the public's expectation of the best possible level of service."
- "Use Analytics and User Feedback to Manage Websites and Digital Services: Agencies should use qualitative and quantitative data to determine user goals, needs, and behaviors, and continually test websites and digital services to ensure that user needs are addressed."
- "Provide Access to Government Information on Multiple Devices: Government information and services should be readily available to the public regardless of device."
- "Protect Privacy: The Federal Government necessarily creates, collects, uses, processes, stores, maintains, disseminates, discloses, and disposes of personally identifiable information (PII) to carry out missions mandated by Federal statute."
- "Instructions for submitting a Privacy Act request. Agencies must provide instructions in clear and plain language for individuals who wish to request access to or amendment of their records pursuant to 5 U.S.C. § 552"

Domain Knowledge

The Fullerton Chat application is designed solely for CSU Fullerton students. The application will allow users to share their opinions, audio, video, pictures, and other media with their peers easily, fast, and securely. The primary domain involved is CSU Fullerton, which is a sub-domain of the CSU system, which itself is a sub-domain of the state and federal governments.

The CSU system requires each campus to implement and adopt a program to train employees on safe practices for information security. However, all CSU employees are expected to follow the rules and regulations set forth by the CSU system at large, spelled out in ICSUAM Section 8000 - Information Security. It describes how CSU campuses should establish and information security program, how to perform risk management for information security, what personal information must be kept private, what personnel information must be kept secure, how to manage third-party vendor access to data, how to conduct trainings and awareness for CSU employees, how to manage information assets, information security incident management, and a number of other semi-pertinent topics to this system. CSUF is in compliance with this CSU policy, and even routinely advertises its compliance through activities like employee training programs for information security awareness, depending on their level of access to campus assets. The goal of many of these efforts is to minimize the threat of unauthorized access to information assets containing protected data.

CSUF operates a technology department, called the Division of Information Technology. Should the chat system be implemented within the CSUF Division of Information Technology resources, not only do the aforementioned CSU policy issues apply, but issues of FERPA come into play. FERPA is a federal law, formally titled the Family Educational Rights and Privacy Act (20 USC 1232g), which strictly protects the privacy of student records. Since CSUF receives federal education funds, any record that the school maintains about students becomes protected, including app data hosted by the institution. Once created, these records may only be released to specifically defined individuals, usually the student and designated close relatives, and only for specific purposes. Education records include any information that the school maintains from the student, such as handwriting, computer media, audio, video, and print media. This may be an obstacle to using CSUF computer resources to host or otherwise facilitate the application data, however it may be unavoidable if the app's intended nature is an embodiment of a university-sponsored initiative.

10 System Features

Features

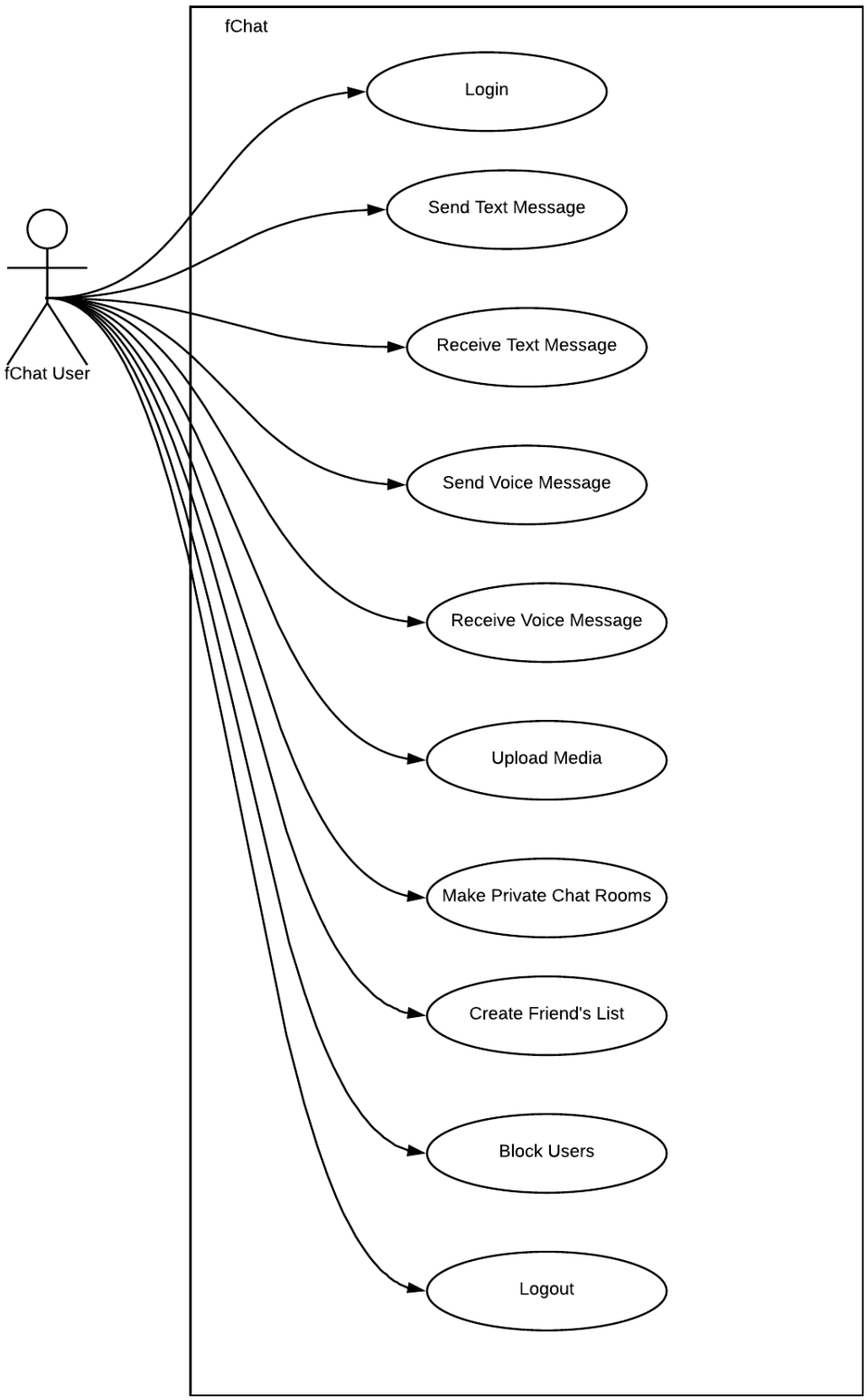
1. Allows users to send text
2. Allows users to receive text
3. Allows users to send voice messages
4. Allows users to receive voice messages
5. Allows users to share pictures among multiple users
6. Allows users to share video among multiple users
7. Allows users to share audio among multiple users
8. Allow a common meeting room between multiple users
9. Allow the user to login for individual profile (Email/Password combination)
10. Create a list of "friends" in order to quickly send data to those on the list
11. Add the ability to allow friends to be added to the particular list
12. Create a profile for that individual
13. Allows user to logout

Functional Requirements

Requirement ID	Requirement Statement	Priority
F01	Allow the user to send a text message to another user	High
F02	Allow the user to receive a text message from another user	High
F03	Allow the user to send a voice message to another user	High
F04	User can send a voice message to another user	High
F05	User are allowed to upload media for other users to observer or interact with.	High
F06	User can make a private room chat room	Medium-High
F07	User must login to access account	High
F08	User can create a friend's list of those they wish to contact.	Medium
F09	User is able to log off their account	High
F10	User can block disruptive individuals from messaging or contacting them	Medium-High
F11	Users are allowed to post hyperlinks to content from authorized providers	High

Use Cases

Use Case Diagram



10 Use Cases - Textual

Use Case Number	1
Use Case Name	Send Text Message
Description	The user (primary actor) sends a message using the application (Fullerton-Chat) to another user who is also using the application (Fullerton-chat) which displays on the screen correctly, accurately, and within a small amount time (dependent on the user's internet connection).

Use Case Number	2
Use Case Name	Receive Text Message
Description	The user (primary actor) is notified that a message has appeared on their screen with a pleasant sound. The message from the other user displays on the screen correctly, accurately, and within a small amount time (dependent on the user's internet connection).

Use Case Number	3
Use Case Name	Send voice message
Description	The user (primary actor) transmits their audio through an input device to send to another user through the application (Fullerton-Chat). The message is heard through audio on the other individuals application.

Use Case Number	4
Use Case Name	Receive Voice message.
Description	The user (primary actor) receives an output through an auditory device to another user through the application (Fullerton-Chat).

Use Case Number	5
Use Case Name	Upload Media
Description	The user (primary actor) is able to upload items to the chat room in order for other individuals to be able to interact with the items.

Use Case Number	6
Use Case Name	Make Private Chat Rooms
Description	The user (primary actor) is able to create a room and password the room in order to prevent unauthorized individuals to gain access to media, chat logs and other items in the chat room.

Use Case Number	7
Use Case Name	Login
Description	The user (primary actor) must password protect their own account that has their own information and customization options for the application.

Use Case Number	8
Use Case Name	Create Friend’s list
Description	The user (primary actor) is able to have a list of contacts that are considered friendly and be easily able to send them messages and media.

Use Case Number	9
Use Case Name	Logout
Description	The user (primary actor) is able to logout to the application on the current system, preventing unintended access on that system.

Use Case Number	10
Use Case Name	Block Users
Description	The user (primary actor) is able to put either create a list of those that can communicate with them or those that cannot through their email.

7 System Quality Attributes

Interoperability

The user initiates by sending the request to other system to have a conversation during normal run time and the system allows the user for the conversation by passing 100% correct information.
Source of stimulus: The request from one system to message another system.
Stimulus: Message being sent from system to another system
Artifacts: Messaging system and database system to check if the individual is available.
Environment: During run time (database system and messaging system)
Response: The messaging system and the database combine in order to send the message to the correct information to the correct system.
Response Measure: The information in the message was **the same to the other system 100**% of the time.

Availability

The user notifies the system about an incorrect response during normal runtime. The System notifies the appropriate entities and the response is created with no downtime.
Source of stimulus: The user
Stimulus: Incorrect Response
Artifact: Processors
Environment: Normal Runtime
Response: System notifies the appropriate entities
Response Measure: No downtime.

Usability

The users initiate messaging in the Fchat system under normal operations and the messages are processed with the latency of 2 seconds.
Source of stimulus: users of Fchat
Stimulus: Initiate messaging
Artifact: Fchat system
Environment: Under normal operations
Response: Transactions are processed
Response Measure: Latency within 2 seconds.

Modifiability

The developer wants to change the UI of the Fchat during the designing phase. The changes are made in 1 day with so side effects.
Source: Developer
Stimulus: Wants to change the UI (Minimization)
Artifact: Code of Fchat
Environment: During the designing phase

Response: In 1 day
Response Measure: Change made causes no ill side effects

Security

The human attacker tries to make an unauthorized attempt to use the system services under normal operations. The system identifies this, and the attacker is denied the service without compromising the system at all.
Source: The human attacker
Stimulus: Tries to make an unauthorized attempt
Artifact: System Services
Environment: Under normal operations
Response: The system identifies the attacker
Response Measure: No compromise at all

Performance

The user initiates the messaging under normal operations. The system processes the messaging with an average latency of 2 seconds.
Source: The user
Stimulus: Initiates the messaging
Artifact: System
Environment: Under normal operations
Response: The system processes the messaging
Response Measure: Average latency of 2 seconds

Testability

The unit tester tests a unit code during the development phase and captures the activity that resulted in the fault with the probability of recovering from that fault by the next time.
Source: The unit tester
Stimulus: Unit code
Artifact: Code Unit
Environment: Development phase
System Response: captures the activity that resulted in the fault
Response measure: Probability of recovering that fault by next time.

Other Quality Attributes: Portability

Portability:

Due to the modularity and testability of the software, it will be significantly easier to port this software to other operating systems due to the significant amount of decoupling of systems and the ease to determine if the program during the porting process will pass all the functional and non-functional requirements.

Exercise 2: Achieving Qualities

Quality Attribute Workshop (QAW)

The whole team spent an afternoon on March 12, 2018 starting from 2:30pm to go through the quality attribute workshop to refine the quality attribute requirements done in the exercise

We performed the following steps. Facilitator is Mandy He, Notetaker is John Hunsaker.

Step 0. Logistics for QAW, target system, QAW team, facilitator

The team agreed to meet in the library at a specific date and time.

The team independently reviewed the QAW steps, as well as the work completed for Exercise 1.

Step 1. QAW Presentation and Introductions

The team greeted each other in the library, 2nd floor conference room.

The facilitator (Mandy), went over the steps of the QAW and reminded the team that the QAW is just to create and prioritize requirements - and designed to refine the quality attribute scenarios from stakeholders.

Step 2. Business/Mission Presentation

Mandy pulled up the Exercise 1 materials on the big screen.

Tom, acting on behalf of the "business", reiterated the highest priority goals:

- The application will send messages between users
- The application will be able to share media between users
- The application will be able to have individual profiles for each user
- The application will be able to have a unique username and password for each user
- The application will be able to make a list for each individual friend
- The application will allow the user to receive recommendations based on their message history.
- Etc…

Step 3. Architectural Plan Presentation

Mandy discussed the Technical Knowledge and the likely-needed technical requirements.

John pulled up the draft of the System Context Diagram on his laptop and gave feedback of what could work with discussed technical requirements, and what would need to be added.

Step 4. Identification of Architectural Drivers

Mandy and Tom suggested the following as the MOST important:

- Availability
- Interoperability
- Security
- Performance

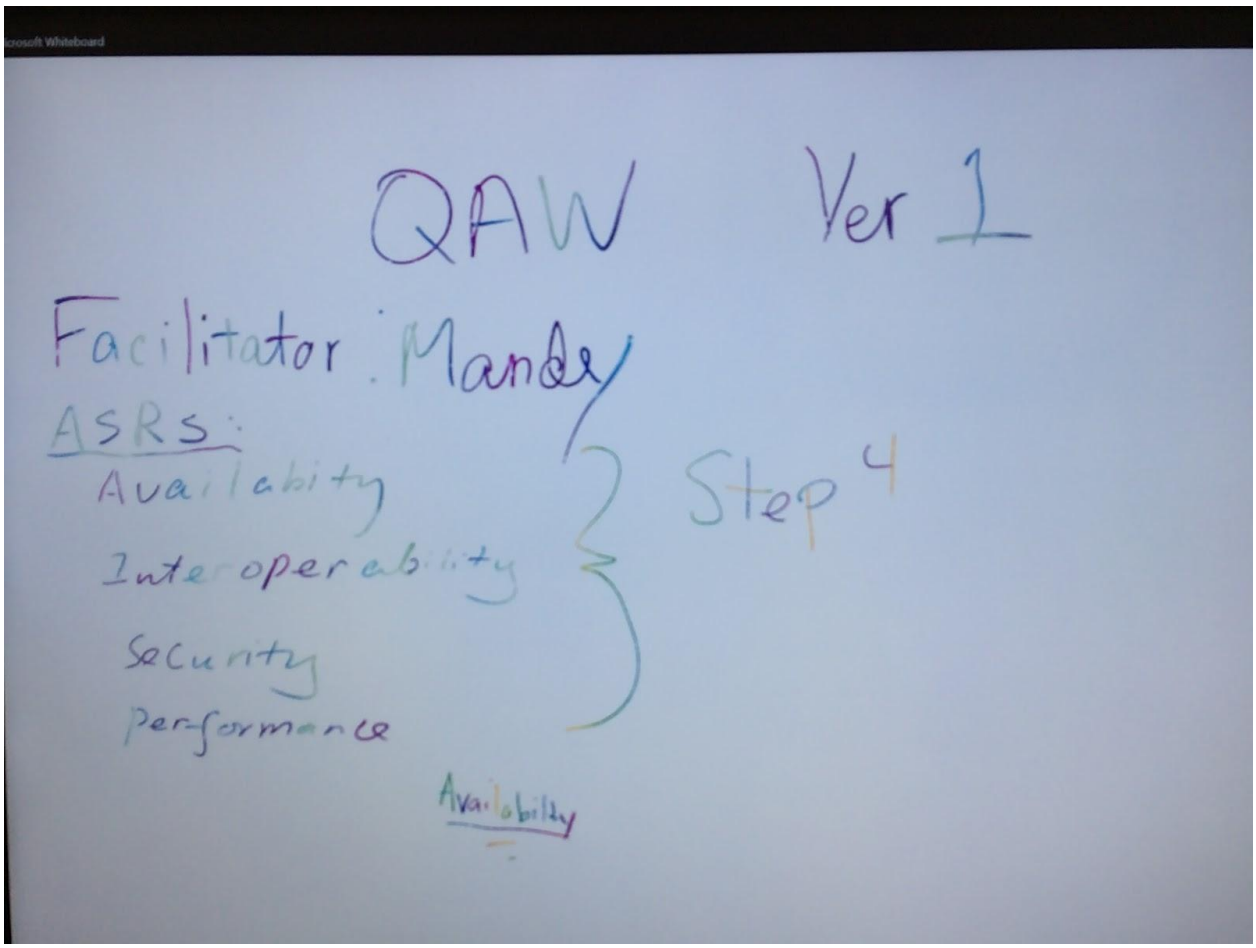


Exhibit A: Picture of digital whiteboard in the midst of Step 4 group discussion, 3:17pm.

Step 5. Scenario Brainstorming

Availability Scenario Idea #1:
Response Measure: No downtime <- Is this feasible given the budget? Change to 99.999%
The rest of the group agreed and decided this should be written out and codified into the QAUT as A01.

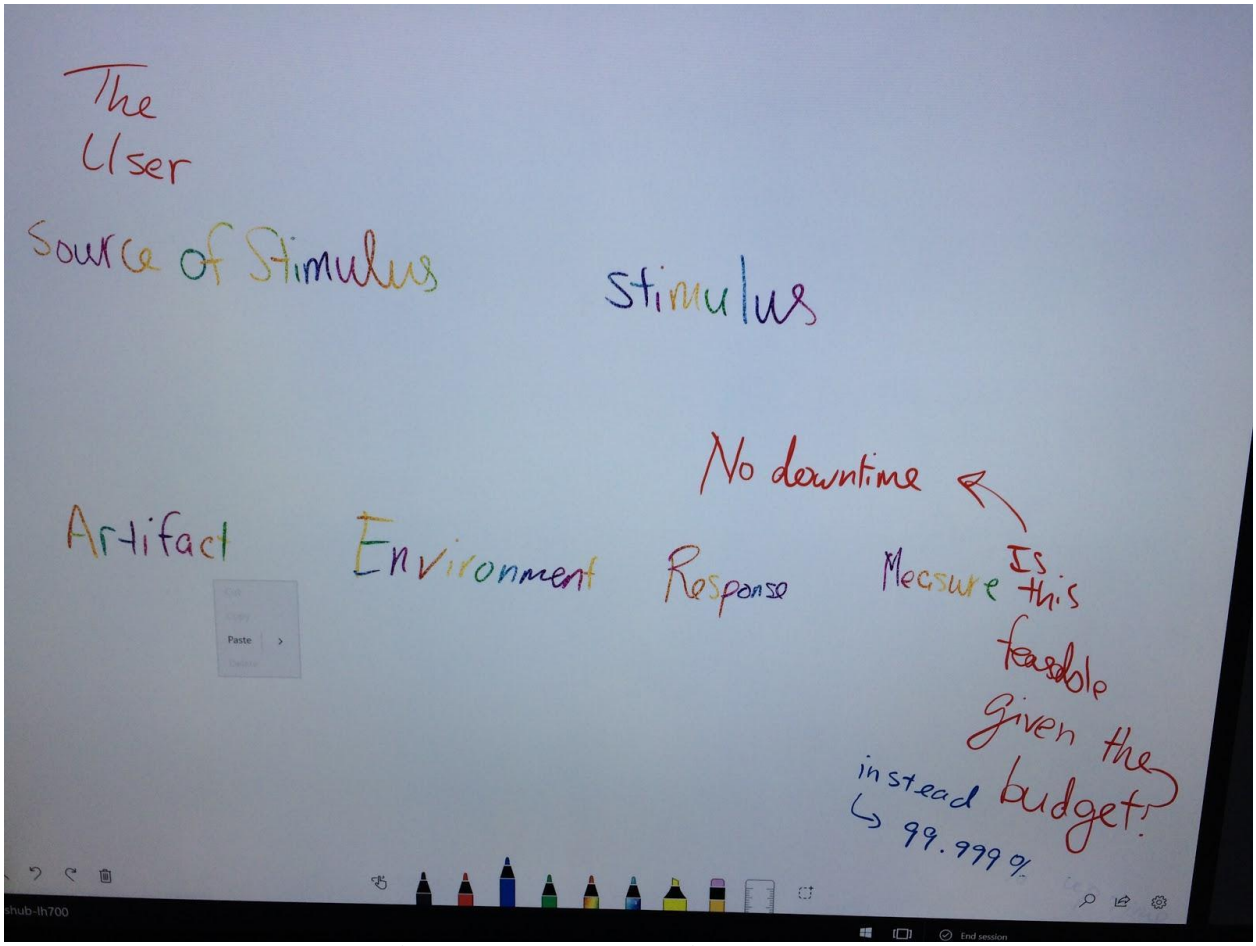


Exhibit B: Picture of digital whiteboard in the midst of Step 5 group discussion, 3:32pm.

All other scenarios: (3:35pm)
As it started to get tedious to write out everything with a digital marker on the big screen, John realized that it would be a lot easier to type the discussions and group decisions directly into the draft Quality Attribute Utility Tree and other documents. John put the documents on the big screen as the group discussed and documents were edited accordingly.

Step 6. Scenario Consolidation

The team reviewed the scenarios constructed earlier, and only had the following consolidation issues:

Availability:
The team decided that A02 and A03 both dealt with faults in similar ways and could be combined.

Security:
The team suspected that S03 and S04 might be too similar and could be written to be less ambiguous.

Step 7. Scenario Prioritization

After a brief discussion, the team unanimously voted the following as the most important in designing architecture:

- A01
- I01
- P01

Step 8. Scenario Refinement

These were many micro-updates to the scenarios; however, these were the two large unresolved points of contention: (3:53pm)

Interoperability:
How will it locate services? Language in the description too vague. Tom to research more and report back to the team at next meeting.

Performance:
Another scenario may be needed for performance. John to consider metrics and report back after researching messaging speeds.

Quality Attribute Utility Tree for the Input of Exercise 3:

	Quality Attributes	Attribute Refinement	(Business Value, Arch Impact) (H/M/L)	Architecturally Significant Requirement (ASR) Scenarios
Utility	Availability	Fault Detection	H,H	If the server goes down, the redundant server immediately goes online. Maximum allowable interrupt is 3 seconds.
	Interoperability	OS Compatibility	H,H	The app runs on iOS, Android, and Windows Phone Operating Systems.
	Modifiability	Profile Upgrades	L,M	Design the User Profile with additional unused fields for future unforeseen purposes.
		Text Upgrades	L,M	Design the text supporting code for a default 500 characters, upgradeable to 1000 characters.
	Performance	Response Time	H,L	Message packets sent to server and received confirmation within 2 seconds during normal conditions.
	Security	Authentication	H,H	In addition to creating profile with Email/Password, user must authenticate profile creation through a link sent to the email they entered.
	Usability	Learning Curve	H,L	The user interface resembles standard text messaging (SMS) or iMessage user interface, with the addition of a menu button.

Architectural Design Relation Table (ADRT) for the Input of Exercise 3:

Case #	Quality Attributes		Tactics	Design Patterns	Architectural Patterns	Implementation on Technology	Concerns	Note
A01	Availability (1)	Detect faults	Ping / echo	ICMP Echo	Peer-to-Peer (OR: Client-Server Pattern)	ICMP (Internet Control Message Protocol); OR Firebase, Android, Java, MongoDb	ping/echo successful, but OS is down so it does not function well e.g. sending message, etc (OR: Server can be a	
A02	(2)	Recover from Faults	Preparation and Repair/ Active Redundancy	Message Data Centralization	Client-Server Pattern	WebAPI,Firebase, Android, Java, MongoDb	Firebase Framework	The unresponsive message server is switched. Without the user's knowledge.
A03	(3)	Recover from Faults	Reintroduction (State Resynchronization)	Reference Data Centralization	Client-Server Pattern	WebAPI	Cost	The application resyncs to the new message server
M01	Modifiability (1)	Reduce coupling	Restrict Dependencies	Abstract Factory Pattern	MVC (Model View Controller)	Java	The complexity may not be worth it for simple user interfaces. The model, view, and controller abstractions may not be	
M02	(2)	Reduce coupling	Refactor	Use Encapsulation	Object Oriented Pattern OR Layered Pattern?	Java	Expertise needed and cost	
M03	(3)	Increase Cohesion	Increase Semantic Coherence	Object Oriented Design	Layered Pattern	Java	Costly	Modules are affected by the same change since they are partially duplicates of themselves
S01	Security (1)	Detect Attacks	Verify Message Integrity	Asymmetric Cryptography	Client-Server Pattern	SSL, TSL	Server can be a performance bottleneck. Decisions about where to locate functionality.	
S02		Detect Attacks	Detect Intrusion	Secure Logger Pattern Structure	SOA	SSL, TSL	Expertise needed and cost. Server can be a performance bottleneck. Decisions about where to locate functionality.	
S03		Resist Attacks	Encrypt Data	Asymmetric Cryptography	Client-Server Pattern	RSA, Blowfish Encryption	Expertise, cost, and securing the private keys. Server can be a performance bottleneck. Decisions about where to locate functionality.	
S04		React to Attacks	Inform Actors	Kill Chain	Client-Server Pattern	Firebase, .NET Framework	Cost, Response Time	
S05		Recover from Attacks	Maintain Audit Trail	Blackboard	Shared Data Pattern	Firebase, .NET Framework	The share-data store may be a performance bottleneck. The shared-data store may be a single point of	
S06		Recover from Attacks	Active Redunsy	Message Data Centralization	Client Server Pattern	Firebase Framework	The complexity may not be worth it for simple user interfaces. The model, view, and controller abstractions may not be	
I01	Interoperability	Locate	Discover Service	Mediator	SOA	Firebase Framework, MongoDb, SQL, Hadoop	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There	
P01	Performance	Control Resource Demand	Increase Resource Efficiency		Multi-Tier Pattern	.NET, Firebase	Substantial up-fron cost and complexity	

Exercise 3: Designing the Architecture

Iteration 1: Decomposition of the Fullerton Chat System into Client and Server Portion

Step 1: Confirm There is Sufficient Requirements Information

The team members representing different stakeholders' views and goals have confirmed that we have sufficient requirements (functional and quality attributes) present to move forward to step 02. These requirements were also prioritized and captured during the stakeholder's Quality Attribute Workshop in exercise 2.

Step 2: Choosing the Decomposition

Since this is the first iteration of Step 2, the decomposition must be done to the entire system.

Step 3: Identify Candidate Architectural Drivers

3.1 Utility Tree Overall (Candidate ASRs)

The Utility Tree was constructed through considerations by stakeholders and developers during our Quality Attribute Workshop (QAW). Certain items were customer required and were highlighted in red. By determining the Architecturally Significant Requirements (ASRs) and mapped them to Quality Attributes of Non-Functional Requirements. The Utility Tree produced based off the entire system is shown in the table on the next page.

ASR #	Number #	Quality Attributes	Attribute Refinement	(Business Value, Arch Impact) (H/M/L)	Architecturally Significant Requirement (ASR) Scenarios (6-parts scenario)
ASR01	A01	Availability	Fault Detection	H,H	The system detects the server is nonresponsive to a user's request during normal operations. The system notifies the operation manager and continue to operate without downtime.
ASR02	A02	Availability	Recover from Faults/Active Redundancy	H,M	The system detects the active operation server goes down during normal operations. The system triggers the redundant server to go online immediately with maximum latency less than 3 seconds.
ASR03	A03	Availability	Recover from Faults/State Resynchronization	H,H	The operator received system notification about a fault detection during normal operations. The operator operates a state resynchronization of the system with no downtime.
ASR04	I01	Interoperability	Discover Service/OS Compatibility	H,H	The system detects users' device operating system (app runs on iOS, Android, and Windows Phone OS) during run time. The system renders the correct version of formatting requirement to display in user device with 100% consistency and accuracy.
ASR05	M01	Modifiability	Reduce Coupling/ Restrict Dependencies	L,M	The developers wish to enhance the security of the system by implementing a new data encryption mechanism during at design time. The modification is made with no side effects within 2 days.
ASR06	P01	Performance	Increase Resource Efficiency	H,H	Users initiate sending message packets under normal operations. The system receives it and processes it within 2 seconds. The system also notifies users message delivered within 2 seconds.
ASR07	S01	Security	Detect Attacks/ Verify Message Integrity	H,H	Users initiate creating new account and/login to existing accounts during normal operations. The system verifies user's username/Email/Password, as well as sends users link to user's email to verify users' identity.
ASR08	S02	Security	Detect Attacks/ Detect Intrusion	H,H	A hacker attempts to delete the files in the system from a remote location during normal operations. The system detects the abnormality of the log in location associated with the account normal location. The system maintains an audit log, notifies the operator, and the correct data is restored within 2 hours.
ASR09	S03	Security	Resist Attack/ Encrypt Data	H,H	The system received users' encrypted message package during normal operations. The system encrypts the data then send out to the intended receiver within 3 seconds.
ASR10	S04	Security	React to Attack/ Inform Actors	H,H	The system detects an attempted attack during normal operations. The system notified the relevant operators and managers about the attack within 2 seconds. They system maintains the audit logs.
ASR11	S05	Security	Recover from Attacks/ Maintain Audit Trail	H,H	The system detects an attack during normal operations. They system keeps a detailed record of user and system actions and their effects.
ASR12	S06	Security	Recover from Attacks/ Active Redundancy	H,H	The system detects an attack that interrupted the re operations during normal operations. The syste triggers the redundant server to go online immedia with maximum latency less than 3 seconds.

ASR13	U01	Usability	Support User Initiative	H,M	The user will be able to work with the client portion of the system after download, and be able to use the app productively during runtime, within 4 minutes of experimentation
ASR14	A04	Availability	Passive Redundancy	H,H	The system passively performs a backup of data from all portions of the server end, including security logs, error logs, and other extraneous data during normal operation. It does this backup every hour, and if a server goes offline within 15 minutes of the last back up, passive recovery will be chosen, and systems will be restored to the previous state within a maximum latency of 3 seconds.

3.1 Utility Tree Choices (Prioritized ASRs)

In our QAW, we reviewed with all stakeholders and prioritized all the candidate ASRs. We finally chose the highest prioritized six ASRs to continue for this iteration, which the customer wants ASR-wise.

ASR #	Number #	Quality Attributes	Attribute Refinement	(Business Value, Arch Impact) (H/M/L)	Architecturally Significant Requirement (ASR) Scenarios (6-parts scenario)
ASR01	A01	Availability	Fault Detection	H,H	The system detects the server is nonresponsive to a user's request during normal operations. The system notifies the operation manager and continue to operate without downtime.
ASR04	I01	Interoperability	Discover Service/OS Compatibility	H,H	The system detects users' device operating system (app runs on iOS, Android, and Windows Phone OS) during run time. The system renders the correct version of formatting requirement to display in user device with 100% consistency and accuracy.
ASR06	P01	Performance	Increase Resource Efficiency	H,H	Users initiate sending message packets under normal operations. The system receives it and processes it within 2 seconds. The system also notifies users message delivered within 2 seconds.
ASR07	S01	Security	Detect Attacks/ Verify Message Integrity	H,H	Users initiate creating new account and/login to existing accounts during normal operations. The system verifies user's username/Email/Password, as well as sends users link to user's email to verify users' identity.
ASR13	U01	Usability	Support User Initiative	H,M	The user will be able to work with the client portion of the system after download, productively during runtime, within 4 minutes of experimentation
ASR14	A04	Availability	Passive Redundancy	H,H	The system passively performs a backup of data from all portions of the server end, including security logs, error logs, and other extraneous data during normal operation. It does this backup every hour, and if a server goes offline within 15 minutes of the last back up, passive recovery will be chosen, and systems will be restored to the previous state within a maximum latency of 3 seconds.

Step 4: Choose a Design Concept that Satisfies Architectural Drivers

4.1 Architectural Design Relation Table (ADRT)

The Architectural Design Relation Table allows for a general overview of the architecture in order to determine what tactics and architectural Patterns would be effective in order to satisfy the architectural drivers. This ADRT is shown in the table below.

ASR #	Quality Attributes	Tactics		Design Patterns	Architectural Patterns	Implementation on Technology	Concerns
ASR 01	Availability (1) A01	Detect faults	Ping / echo	ICMP Echo	Client-Server Pattern	ICMP, Firebase, Android, Java, MongoDb	ping/echo successful, but OS is down so it does not function well e.g. sending message, etc. (OR: Server can be a performance bottleneck)
ASR 04	Interoperability (1) I01	Locate	Discover Service	Mediator	SOA	Firebase Framework, MongoDb, SQL, Hadoop	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR 06	Performance (1) P01	Control Resource Demand	Increase Resource Efficiency	NA	SOA	.NET, Firebase	Substantial up-front cost and complexity
ASR 07	Security (1) S01	Detect Attacks	Verify Message Integrity	Asymmetric Cryptography	Client-Server Pattern or SOA	SSL, TSL	Server can be a performance bottleneck. Decisions about where to locate functionality.
ASR 13	Usability (1) U01	Support User Initiative	Aggregate, Cancel, Undo	NA	Model View Controller	Kotlin, Android, .NET	Split the system into a number of computationally independent execution structures connected by some communications media. Done to provide specific server environments optimized for operational requirements and resource usage. Complexity.
ASR 14	Availability (4) A04	Preparation and Repair	Passive Redundancy	Mediator	Client Server Pattern	SQL, NoSQL, Firebase, .NET, SSL /TLS	Server can be a performance bottleneck. Decisions about where to locate functionality.

4.2 Pros and Cons Matrix

In order to determine what patterns are best for the architectural drivers, the use of a Pros and Cons Matrix will neatly organize what patterns and tactics should be used in order to address the architectural drivers. The Pro's and Con's Matrix may be viewed below.

Conclusion: After all considerations were taken into account the choice for what pattern would be used for the architecture would be a mix of Client-Server and Service Oriented Architecture Patterns.

				Pattern #1	Pattern #2	Pattern #3
ASR Number / QA Number		**Architectural Driver**		**Client-Server Patterns**	**MVC (Model View Controller)**	**SOA (Service Oriented Architecture Pattern)**
ASR01	A01	Availability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers.	This would not be applicable for this ASR.	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet. It is designed to locate services and combine (and dynamically recombine) them into meaningful coalitions while achieving reasonable performance, security, and **availability**.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in the client or in the server) are often complex and costly to change after a system has been built.	The complexity may not be worth it for simple user interfaces. The model, view and controller abstractions may not be good fits for some user interface toolkits.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR04	I01	Interoperability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers. But this is not applicable to interoperability	This would not be applicable for this ASR.	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in the client or in the server) are often complex and costly to change after	The complexity may not be worth it for simple user interfaces. The model, view and controller abstractions may not be good fits for some user interface toolkits.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.

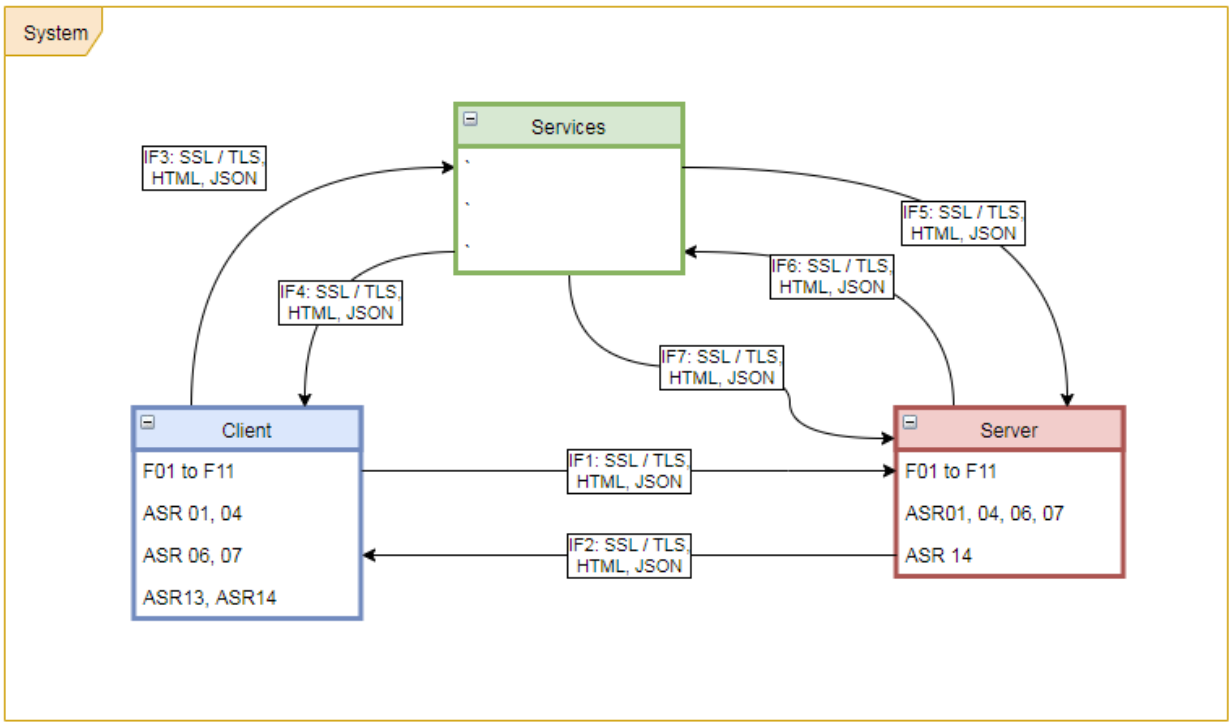
				a system has been built.		
ASR06	P01	Performance	Pros	It improves scalability and availability by centralizing the control of these resources and services, while **distributing the resources themselves across multiple physical servers.**	This would not be applicable for this ASR.	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, security, and availability.
			Cons	Server can be a performance bottleneck. **Server can be a single point of failure**.	The complexity may not be worth it for simple user interfaces. The model, view and controller abstractions may not be good fits for some user interface toolkits.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR07	S01	Security	Pros	It improves scalability and availability by centralizing the control of these resources and services, while **distributing the resources themselves across multiple physical servers.**	This would not be applicable for this ASR.	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, security, and availability.
			Cons	Server can be a performance bottleneck. **Server can be a single point of failure**.	The complexity may not be worth it for simple user interfaces. The model, view and controller abstractions may not be good fits for some user interface toolkits.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR13	U01	Usability	Pros	This is not applicable to this particular case.	User interface functionality be kept separate from application functionality and is still responsive to user input, or to changes in the underlying application's data. It allows multiple views of the user interface be created, maintained, and coordinated when the underlying application data changes.	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, **security,** and availability.
			Cons	Server can be a performance bottleneck. **Server can be a single point of failure**.	Split the system into a number of computationally independent execution structures connected by some communications media. Done to provide specific server environments **optimized for**	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.

					operational requirements and resource usage.	
ASR14	A04	Availability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers.	This would not be applicable for this ASR.	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet. It is designed to locate services and combine (and dynamically recombine) them into meaningful coalitions while achieving reasonable performance, security, and **availability**.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in the client or in the server) are often complex and costly to change after a system has been built.	The complexity may not be worth it for simple user interfaces. The model, view and controller abstractions may not be good fits for some user interface toolkits.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
Conclusion Statement: 1. Client-Server Pattern: A01, A04 2. MVC: U01 3. SOA: I01, P01, S01						

Step 5: Instantiate Architectural Elements and Allocate Responsibilities

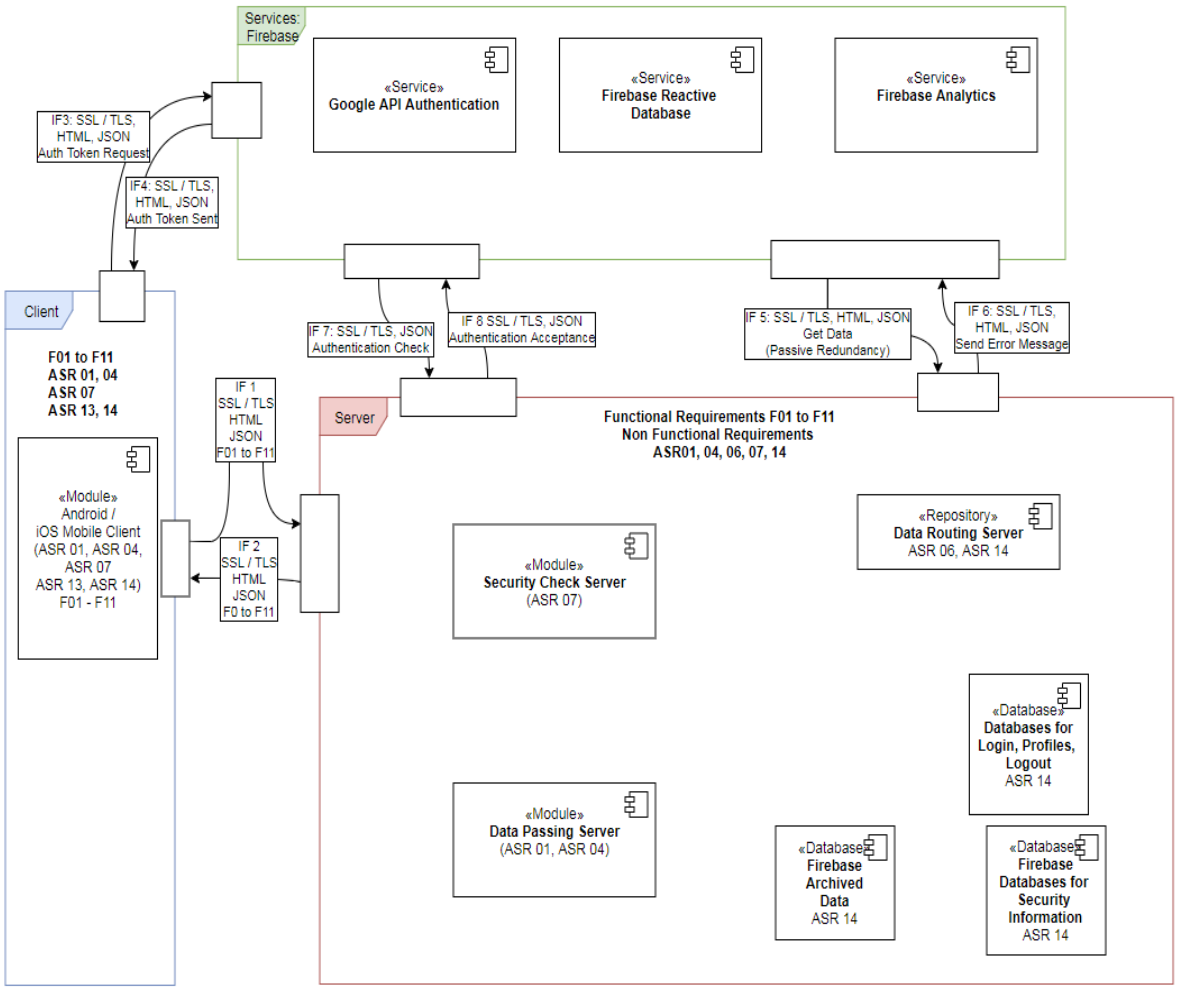
5.1 Decomposition of the System into the Client and Server Allocation, Module View

The system is deconstructed into both client and server portions. After discussing with the development team and architects, the following module view was created and the functional requirements and architectural significant requirements (ASRs) were allocated after discussions. Services are provided by the Firebase framework in order to accomplish several functional and non-functional requirements. (Google)



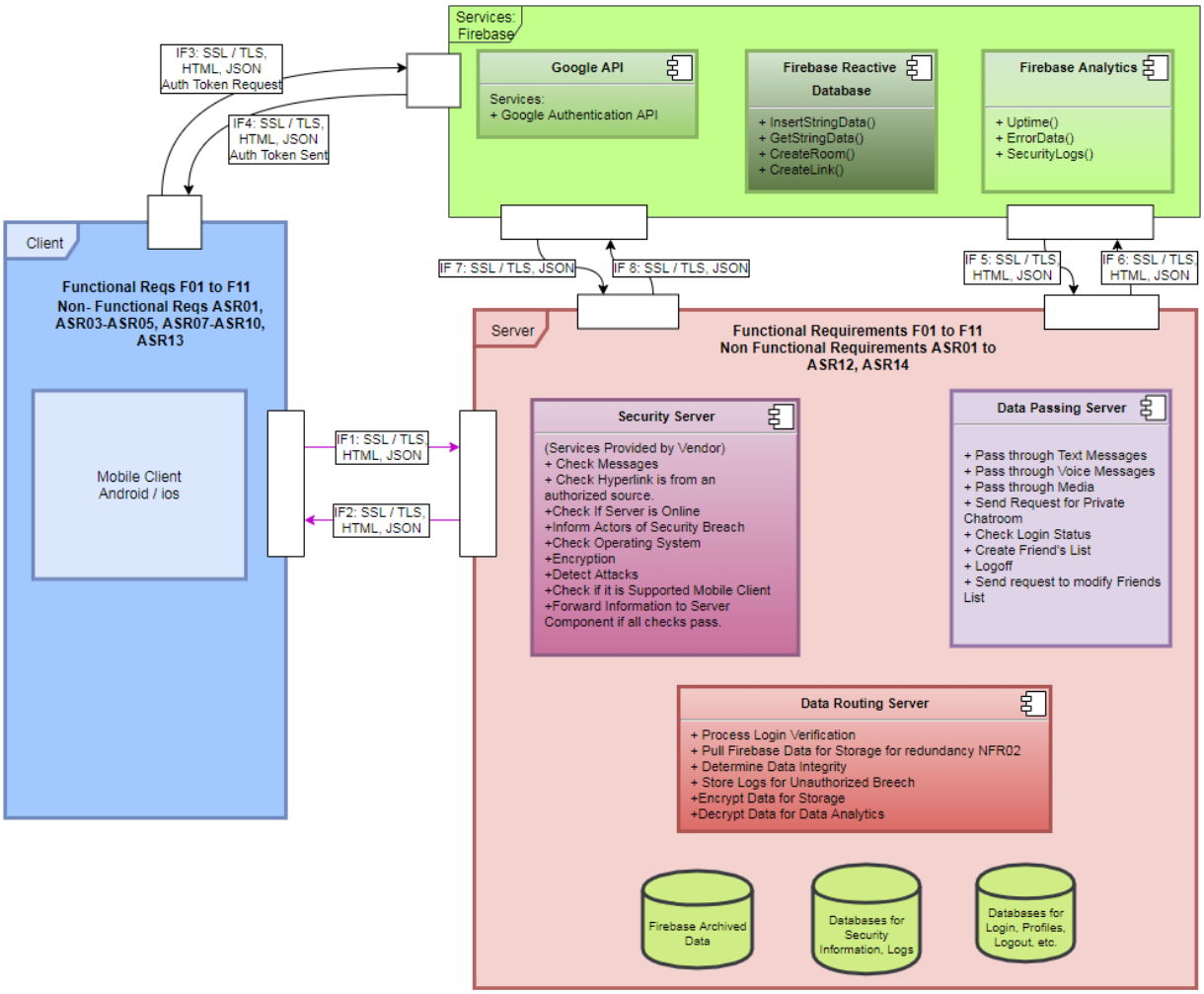
5.2 Decomposition of the System into the Client and Server Allocation, C&C View

The system is deconstructed into both client and server portions. After discussing with the development team and architects, the following component and connector view was created, and the functional requirements and architectural significant requirements were allocated after discussion.



5.3 Decomposition of the System into the Client and Server Allocation, Allocation View

The system is deconstructed into both client and server portions. After discussing with the development team and architects, the following allocation view was created, and the functional requirements and architectural significant requirements were allocated after discussion. This is a very detailed view meant for developers and architects.



5.4 Allocation of Requirements

After deciding the architecture, the responsibilities were allocated within the instantiations, with the assistance of architects and developers.

Client		Server	
Allocation of Drivers	**Description**	**Allocation of Drivers**	**Description**
F01	Allow the user to send a text message to another user	F01	Allow the user to send a text message to another user
F02	Allow the user to receive a text message from another user	F02	Allow the user to receive a text message from another user
F03	Allow the user to send a voice message to another user	F03	Allow the user to send a voice message to another user
F04	User can send a voice message to another user	F04	User can send a voice message to another user
F05	User are allowed to upload media for other users to observer or interact with.	F05	User are allowed to upload media for other users to observer or interact with.
F06	User can make a private chat room	F06	User can make a private chat room
F07	User must login to access account	F07	User must login to access account
F08	User can create a friend's list of those they wish to contact.	F08	User can create a friend's list of those they wish to contact.
F09	User is able to log off their account	F09	User is able to log off their account
F10	User can block disruptive individuals from messaging or contacting them	F10	User can block disruptive individuals from messaging or contacting them
F11	Users are allowed to post hyperlinks to content from authorized providers	F11	Users are allowed to post hyperlinks to content from authorized providers
ASR01	The System is able to Detect Faults	ASR01	The System is able to do Fault Detection
ASR04	The System is able to Discover Service/OS Compatibility	ASR04	The System is able to Discover Service/OS Compatibility
ASR07	The System is able to Detect Attacks/ Verify Message Integrity	ASR06	The System is able to Increase Resource Efficiency
ASR13	The System is able be more useable through Aggregate, Cancel and Undo Tactics	ASR07	The System is able to Detect Attacks/ Verify Message Integrity
ASR14	The System will be able to have some sort of preparation and repair, through passive redundancy	ASR14	The System will be able to have some sort of preparation and repair, through passive redundancy

Step 6: Define Interfaces for Instantiated Elements

6.1 Define interfaces for instantiated elements

Interface	From Elements	Interface	To Element
IF1	Client: Mobile Client Android & iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF2	Server Portion: Security Server	HTML, SSL / TLS, JSON	Client: Android Application & iOS Application
IF 3	Client: Mobile Client Android & iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF 4	Server Portion: Security Server	HTML, SSL / TLS, JSON	Client: iOS Application
IF 5	Services: Firebase	HTML, SSL / TLS, JSON	Server
IF 6	Server	HTML, SSL / TLS, JSON	Services: Firebase

Step 7: Verify and Refine Requirements and Make Them Constraints for Instantiated Elements

In this step, the team verified that the element decomposition outputs have meet the requirements and design constraints, responsibilities have been allocated, and ready to move to the next iteration of decomposition.

Documentation a Software Architecture

During the requirements collection and prioritization process in exercise 1 and exercise 2 QAW, we have produced many views in the architecture design process. In this phase of documentation, we choose and combine the views by following the three-step of choosing views.

Choosing the Views

Step 1: Stakeholder and View Table

1.1 Build a Stakeholder / View Table

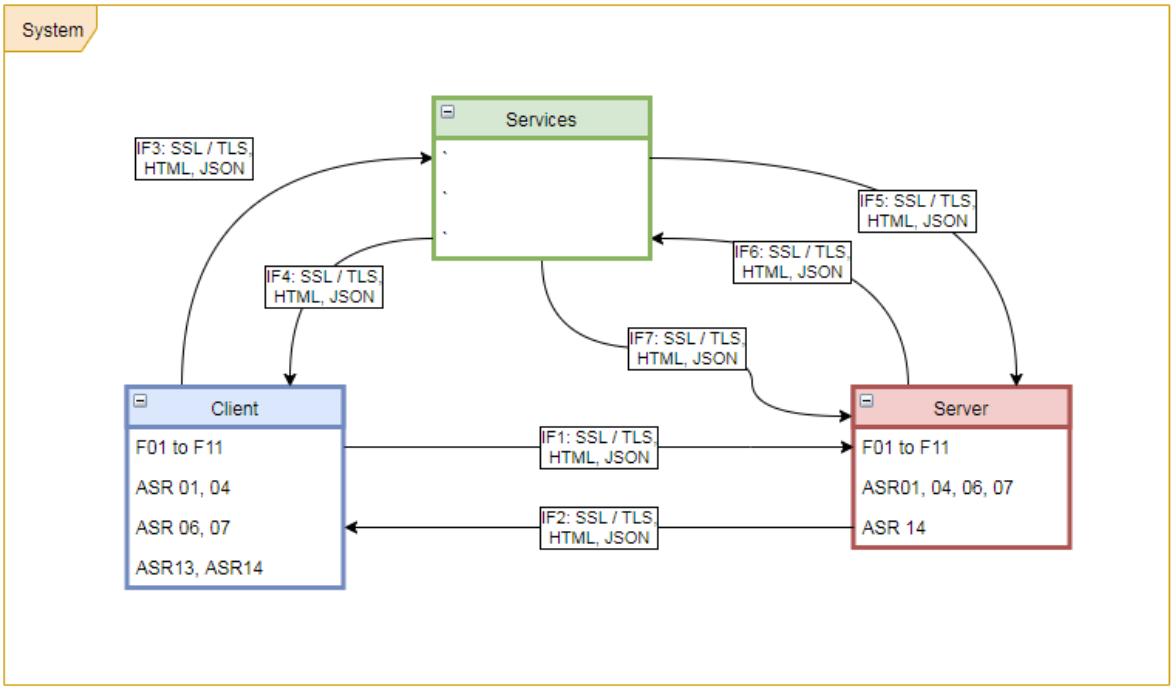
	Module Views				C & C Views	Allocation Views	
Stakeholders	**Decomposition**	**Uses**	**Class**	**Layer**	**Various**	**Deployment**	**Implementation**
Project Manager	s	s		s		d	**o**
Member of Development Team	d	d	d	d	d	s	s
Testers and Integrators		d	d		d	s	s
Maintainers	d	d	d	d	d	s	s
Product Line Application Builder		d	s	s	s	s	s
Customer					s	o	
End User					s	s	
Analyst	s	s	s	s	s	d	d
Infrastructure Support	s	s		s		d	d
New Stakeholder or Investor	a	a	a	a	a	a	a
Current and Future Architect	d	d	d	d	d	d	s
Legend							
d = detail architecture							
s = some details							
o = overview of information							
a = any amount of information							

1.2 Determine the readers of these views

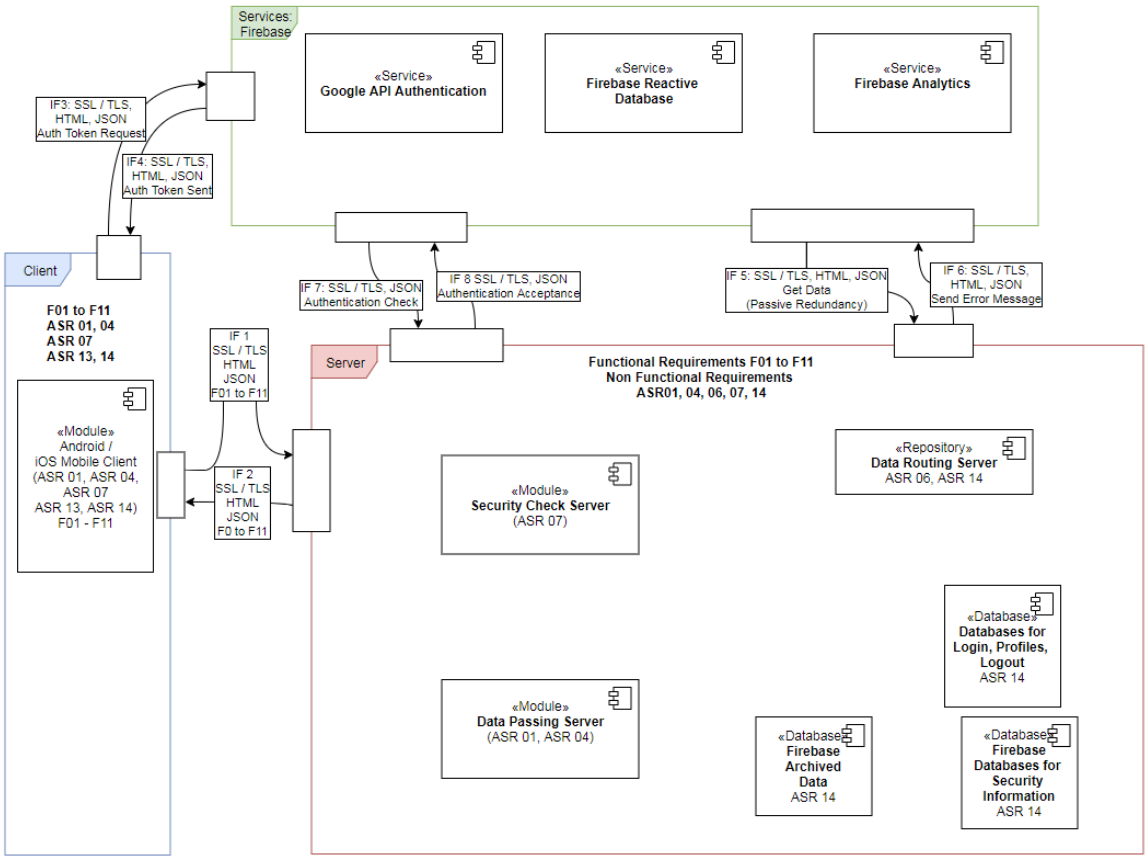
The two readers to be targeted will be project managers and analysts, and the views that we will attempt to build based off these individuals will be the module views, component and connector views, and the allocation views for the first and second iterations. The first iteration of the decomposition of the entire system are shown below.

Step 2: Combine Views for Stakeholders.

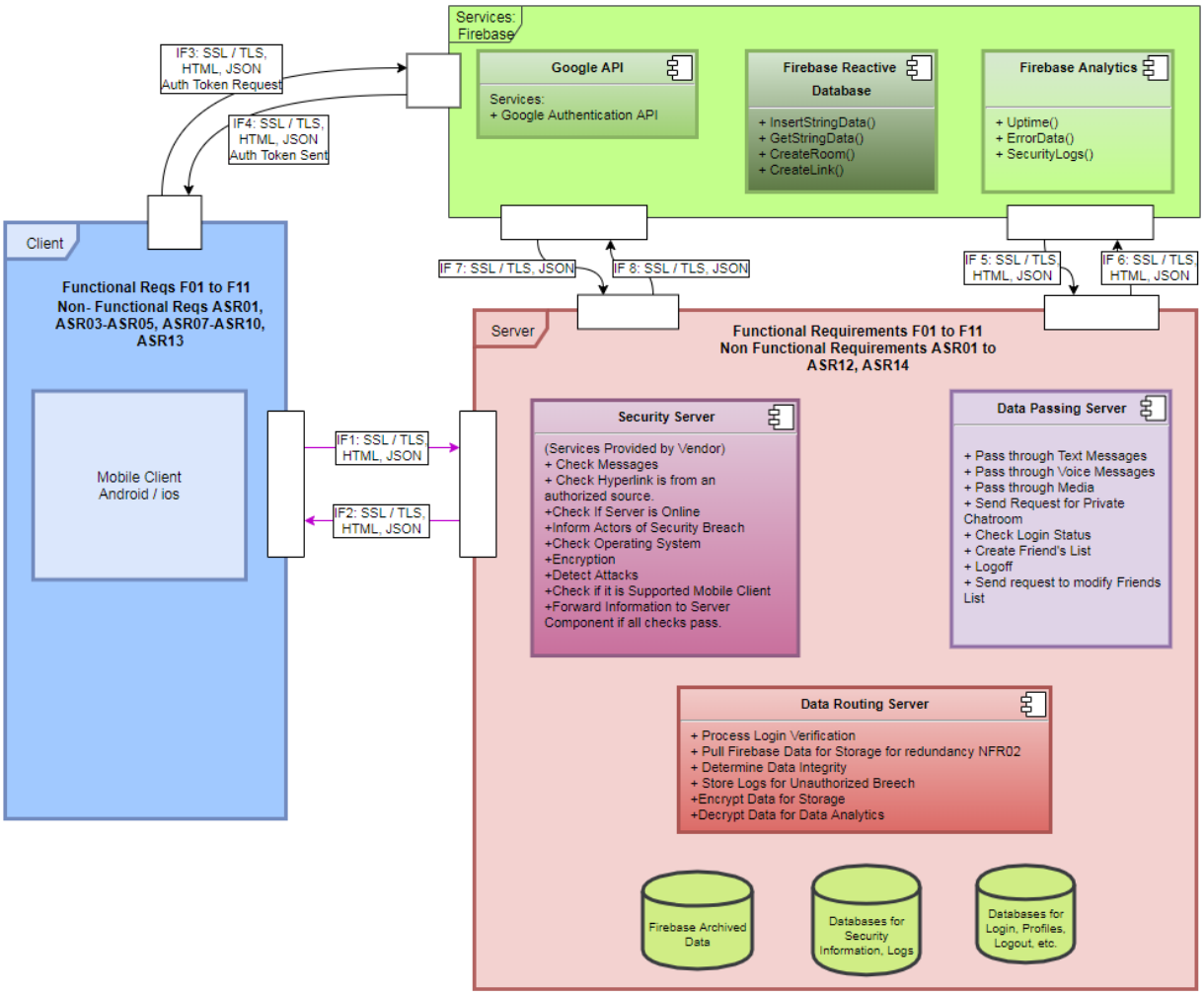
2.1 Module view of the decomposition of the entire system



2.2 Component and Connector View of the Entire System



2.3 Allocation View of the Entire System



Step 3: Prioritize and Stage

The views constructed for the prioritize and staging are the allocation view, component and connector, and module views provide more than enough information and the diagrams are located above in step 2. These provide more than enough details in order to satisfy analysts and project managers.

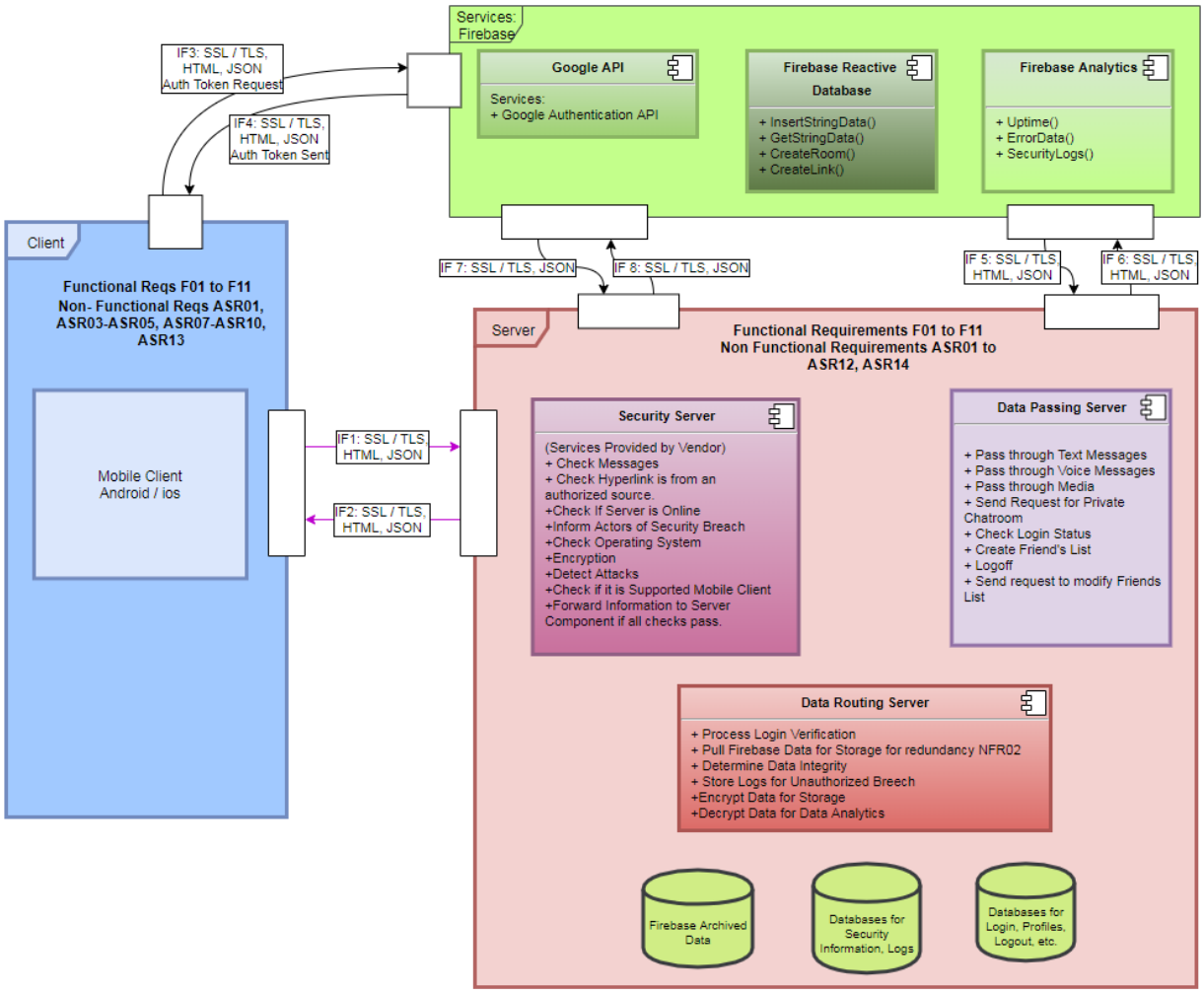
Construct the associated artifacts for each view of your selection (as the textbook suggested for each of 3 views).

The Artifacts for Utility Tree, ADRT, and PCM were constructed for each of the ADD v2, to make the artifacts necessary for the construction of the views. This does not have all the documentation requirements as it still requires the "Template for a View" as described by *Software Architecture in Practice* by Len Bass, Paul Clements, and Rick Kazman. This template is shown below.

Template for a View

Section 1: Primary Presentation

Allocation View of Entire System



Section 2: Element Catalog

Section 2.A. Elements and Their Properties

Element	Properties
Client: Android / iOS Client	This is the client that provides the user with a user interface that allows for meaningful work to be done and gives the user the ability to call upon the functionality within the system.
Server	Server module contains the Security Server, which checks to see if messages are authenticated and are safe to distribute in the

	database or to a client. The Server Module also contains the Data Passing Server, which determines where the data is to be routed either back to the client through the security server or to the data routing server, which stores or retrieves information. It acts as an intermediary that informs the system if the data routing server is unavailable.
Services: Firebase	This Firebase Framework is a separate service that is not coded by our engineers, but instead used by the client or server modules in order to accomplish the goals needed. They do not have functional requirements that we have programmed but rather the client and server modules have the code that delivers or authenticates to the framework. (Google)

Section 2.B. Relations and Their Properties

Interface	Relation	Properties
IF 1	The relation is of the Client to the Server	This is how the client is able to send data (messages, hyperlinks, voice, media, etc.)
IF 2	The relation is of the Server to the Client	This is how the client is able receive data (servers are up, retrieve messages, profile data, etc.) from the server.
IF 3	The relation is of the Client and the Services: Firebase modules.	This is how the client is able to authenticate with the Service: Firebase, and send other data for use by the Server portion later on...
IF 4	The relation is of the Services: Firebase to the Client modules	This is how the Services: Firebase, sends the information back, (Authentication) to the client.
IF 5	This is the relation of the Services: Firebase to the Server modules	This allows data to be pulled from the Services:Firebase to the Server module. Data is sent from the Services: Firebase portion to the Server (Data Routing Server) .
IF 6	This is the relation of the Server to the Services: Firebase	This allows data to be sent from the Server (Data Routing Server) module to the Services: Firebase.
IF 7	This is the relation of the Services: Firebase to the Server modules	This allows data to be pulled from the Services:Firebase (Authentication) to the Server (Security Server) module. Data is sent from the Services: Firebase portion to the Server.
IF 8	This is the relation of the Server to the Services: Firebase	This allows data to be sent from the Server module to the Services: Firebase.

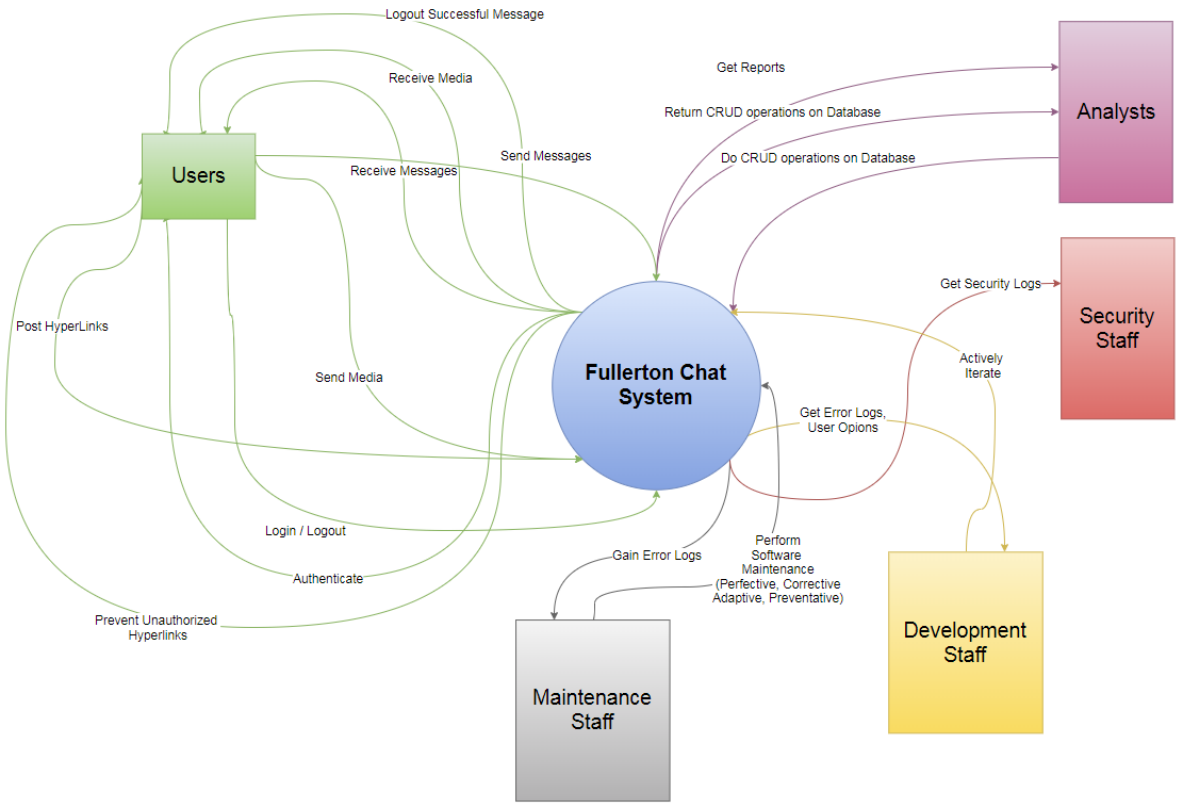
Section 2.C. Element Interfaces 45r

Interface	From Elements	Interface	To Element
IF1	Client: Mobile Client Android & iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF2	Server Portion: Security Server	HTML, SSL / TLS, JSON	Mobile Client: Android Application & iOS Application
IF 3	Client: iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF 4	Server Portion: Security Server	HTML, SSL / TLS, JSON	Client: iOS Application
IF 5	Services: Firebase	HTML, SSL / TLS, JSON	Server
IF 6	Server	HTML, SSL / TLS, JSON	Services: Firebase

Section 2.D. Element Behavior

Interface	From Elements	Behavior	To Element
IF1	Client: Mobile Client Android & iOS Application	Sends Data to the Server (Security portion) that is needed to satisfy the Functional Requirements, and Architecturally Significant Requirements	Server Portion: Security Server
IF2	Server Portion: Security Server	Receives Data from the Server (Security portion) that is needed to satisfy the Functional Requirements, and Architecturally Significant Requirements	Mobile Client: Android Application & iOS Application
IF 3	Client: Mobile Client Android & iOS Application	Get authorization token from authentication.	Firebase Services
IF 4	Firebase Services	Send authentication token if credentials are valid to the client.	Client: iOS Application
IF 5	Services: Firebase	Send information to the server as a passive redundancy in order to allow the data from the client to be backed up.	Server
IF 6	Server	Send information to the Firebase Framework in case of incomplete data, or corrupted data.	Services: Firebase

Section 3. Context Diagram



Section 4. Variability Guide

The items that can be changed are the client-side mobile devices, as if the development team wants to make a web portal application that will run on Linux or windows computers and how many database types will be needed, NoSQL or SQL.

However, the items that must stay the same are the Firebase framework and the need for a security server.

Section 5. Rationale

The reason for the need of security is the information that clients give up in order to use this program is dependent on the fact that their data will be kept confidential. By losing the client's trust due to data breaches and the fact that we want to expand the product into the education market could prove very difficult if our security is not enough to withstand reasonable attacks. If we eventually get the blessing of educational institutions the system must be able to protect FERPA data, which must remain rigorous, or will cost the company fines or even state investigations.

The Firebase framework must remain rigid, due to the dependence of the prototypes and the knowledge of the staff in regard to this framework. Google is considered a reputable source with products that provide robust performance and security, which are needed for this project to succeed, if another architect tries to switch the framework, it is highly recommended of repeating the steps of ADD v2 from the beginning of the system architecture.

Template for Documentation Beyond Views

Section 1. Documentation Roadmap

Scope and Summary

This document will provide a general overview of the entire system for stakeholders who do not need an entire in-depth understanding of the Fullerton Chat Application, through the Module View. The Component and Connector View and the Allocation Views are meant to provide a detail and in-depth understanding respectively of the system. It shows the interfaces and how each module interacts with each other. However, this was documented with the intention to present to project managers and analysts mainly. More in-depth documentation is part of the beginning portion iteration 3 of this document, which goes through ADD v2.

How the Documentation is Organized

The documentation is organized in several portions to best describe the process of designing software architecture learned from this class. It contains Exercise 1: Quality Attributes, Exercise 2: Achieving Qualities, Exercise 3: Designing the Architecture, Lesson Learned, References, and other relevant appendixes. Exercise 1 and Exercise 2 are mainly to capture the requirements particularly quality attribute requirements that will heavily impact architecture design, and they are very straightforward.

In Exercise 3 designing the architecture, it starts with the ADD v2 process by the first iteration of decomposing the whole system, and the artifacts needed in order to complete the first iteration of the entire system, which splits it into Client, Server, Services: Firebase portions. Through the use of a Utility Tree, ADRT, and Pros and Cons Matrix, the views can be instantiated, and interfaces can be defined.

After this ADD v2 decomposition, the Documenting a View portion which is broken up into primary presentation, element catalog, context diagram, variability guide, and rationale. This finishes the first iteration of the document, followed by the Template for Documentation Beyond views.

The Template for Documentation Beyond Views, describes how the documentation was designed and why the processes were used for the first iteration. It is then followed by a second decomposition of ADD v2, with each rationale for each artifact described during each step.

In closing, the documentation is divided into ADD v2 decomposition of the first iteration, Documenting a View Template, Documentation Beyond Views Template, and the ADD v2 decomposition of the second iteration of the Fullerton Chat Application.

View Overview

The view documentation is taken from the Documentation Beyond Views Template from Software Architecture in Practice Third Edition, by Len Bass, Paul Clements, and Rick Kazman.

This is template is broken up into several parts:

1. Primary Presentation - This displays the elements and relations of a view. It should include the
2. Information you want to convey about the system.
3. Elements Catalog - This is divided into four parts with the express concern of detailing at elements shown in the primary presentation. The four parts are:
    1. Elements and Their Properties
    2. Relations and Their Properties
    3. Element Interfaces
    4. Element Behavior
4. Context Diagram - This shows how the system or part of the system shown in this view relates to its environment. The purpose of the context diagram is to show the scope of the view being analyzed.
5. Variability Guide - This shows how to exercise any variation points, items that the architect will allow to change, that are a part of the architecture shown in the view.
6. Rationale - This is the portion that explains why the architecture must be why it is and how it was decided upon.

In the Case of Decomposition of the Entire System, First Iteration, Module View

The system is deconstructed into both Client, Server and Services: Firebase portions. After discussing with the development team and architects, the following module view was created, and the functional requirements and architectural significant requirements were allocated after discussion. Not that the service portion does not have our own code in the Services, but rather the Client and Server use the services provided by the Services: Firebase module. This means that we do not include those requirements that are working with services in the Services: Firebase module as we do not code for the Services:Firebase side, but rather our other modules have code that access them. Therefore, there are no Architecturally Significant Requirements or Functional Requirements Assigned directly to Services: Firebase.

How stakeholders can use the documentation

The stakeholders can use the documentation for module view for a general overview of the entire first decomposition of the system as it is separated into the Client, Server, and Firebase: Services modules. The component and connector view and the allocation view provide an in depth understanding of the system that is meant not for the general new-stakeholder or investor, but rather analysts, managers, testers, and developers. This in-depth understanding will allow these individuals to understand and make actual decisions in how major portions should be implemented, instead of a general understanding of the system.

Section 2. How a View is Documented

The items documented are done before each step on ADD v2. for the second iteration. However, for the first Iteration, the template in the book, Software Architecture in Practice Third Edition, by Len Bass, Paul Clements, and Rick Kazman, is used to document the view. Specifically, in the Section known as "Template for a View".

Section 3. System Overview

The system is meant to serve students in college by providing them a suite of tools used in order to coordinate and study for projects without any distractions, using a lightweight framework and a secure system in which they can share text messages, media, and voice messages. They can partition themselves into group and arrange meetings without the constant distractions in other messaging programs.

Section 4. Mapping Between Views

The association between the views for the first iteration show the same parts, being Client, Server, and Services: Firebase, from a general overview with the module view, to a more in-depth view from the component and connector view, to an allocation view that shows what each component is able to do. This why they have the same items associated with them and provides an easy understanding for what is needed for what type of stakeholder as it goes from general stakeholder to developer in order of the depth needed.

Section 5. Rationale

The documentation was chosen to display how iterations of ADD v2 and the artifacts generated during the ADD v2 process can be used to design a software architecture step by step. The views were chosen based of the individuals who would use the information (project managers, analysts). The "Beyond the Views" and the "Template for a View" were taken from the book, Software Architecture in Practice Third Edition, by Len Bass, Paul Clements, and Rick Kazman, which is a well-regarded academic source, and is utilized in industry. With well-regarded academic sources and industry standards for architecture like ADD v2, the documentation has been created with a high standard of quality.

Section 6. Directory – index, glossary, acronym list

- Module View – An architectural view that decomposes a system into modules
- Component and Connector View – An architectural view that decomposes a system into component and connector.
- Allocation View – An architectural view that describes the mapping of software items to elements of an environment that the system is developed or in which it occurs.
- SSL – Secure Sockets Layer
- TLS – Transport Layer Security
- JSON – JavaScript Object Notation
- NoSQL – Non-Relation Database
- SQL – Relation Database, Structured Query Language
- Firebase Framework – A mobile and web application framework owned by Google

Iteration 2: Decomposition of the Fullerton Chat System's Server Component into Security Server, Transfer Server, and Data Server.

Step 1: Confirm There is Sufficient Requirements Information

During iteration 1, the team members has collected and prioritized sufficient information to move forward to iteration 2.

Step 2: Choosing the Decomposition

The stakeholders decided to decompose the Fullerton Chat System's Server Component for this iteration. Since this is the second iteration of Step 2, the decomposition chosen was the most important portion, which controls the data, the routing, encryption, and must be designed accurately and be given the most amount of care.

Step 3: Identify Candidate Architectural Drivers

3.1 Utility Tree

The Utility Tree was constructed through considerations by stakeholders and developers. Certain items were customer required and were highlighted in red. These have many similarities to the previous decomposition but does not have all the same architecturally significant requirements. By determining the Architecturally Significant Requirements and mapped them to Quality Attributes of Non-Functional Requirements. The Utility Tree produced based off the entire system is shown in the table below.

ASR #	Number	Quality Attributes	Attribute Refinement	(Business Value, Arch Impact) (H/M/L)	Architecturally Significant Requirement (ASR) Scenarios (6-parts scenario)
ASR01	A01	Availability	Fault Detection	H,H	The system detects the server is nonresponsive to a user's request during normal operations. The system notifies the operation manager and continue to operate without downtime.
ASR02	A02	Availability	Recover from Faults/Active Redundancy	H,H	The system detects the active operation server goes down during normal operations. The system triggers the redundant server to go online immediately with maximum latency less than 3 seconds.
ASR03	A03	Availability	Recover from Faults/State Resynchronization	H,H	The operator received system notification about a fault detection during normal operations. The operator operates a state resynchronization of the system with no downtime.
ASR04	I01	Interoperability	Discover Service/OS Compatibility	H,H	The system detects users' device operating system (app runs on iOS, Android, and Windows Phone OS) during run time. The system renders the correct version of formatting requirement to display in user device with 100% consistency and

					accuracy.
ASR05	M01	Modifiability	Reduce Coupling/ Restrict Dependencies	L,M	The developers wish to enhance the security of the system by implementing a new data encryption mechanism during at design time. The modification is made with no side effects within 2 days.
ASR06	P01	Performance	Increase Resource Efficiency	H,H	Users initiate sending message packets under normal operations. The system receives it and processes it within 2 seconds. The system also notifies users message delivered within 2 seconds.
ASR07	S01	Security	Detect Attacks/ Verify Message Integrity	H,H	Users initiate creating new account and/login to existing accounts during normal operations. The system verifies user's username/Email/Password, as well as sends users link to user's email to verify users' identity.
ASR08	S02	Security	Detect Attacks/ Detect Intrusion	H,H	A hacker attempts to delete the files in the system from a remote location during normal operations. The system detects the abnormality of the log in location associated with the account normal location. The system maintains an audit log, notifies the operator, and the correct data is restored within 2 hours.
ASR09	S03	Security	Resist Attack/ Encrypt Data	H,H	The system received users' encrypted message package during normal operations. The system encrypts the data then send out to the intended receiver within 3 seconds.
ASR10	S04	Security	React to Attack/ Inform Actors	H,H	The system detects an attempted attack during normal operations. The system notified the relevant operators and managers about the attack within 2 seconds. They system maintains the audit logs.
ASR11	S05	Security	Recover from Attacks/ Maintain Audit Trail	H,H	The system detects an attack during normal operations. They system keeps a detailed record of user and system actions and their effects.
ASR12	S06	Security	Recover from Attacks/ Active Redundancy	H,H	The system detects an attack that interrupted the regular operations during normal operations. The system triggers the redundant server to go online immediately with maximum latency less than 3 seconds.
ASR14	A04	Availability	Passive Redundancy	M,H	The system passively performs a backup of data from all portions of the server end, including security logs, error logs, and other extraneous data during normal operation. It does this backup every hour, and if a server goes offline within 15 minutes of the last backup, passive recovery will be chosen, and systems will be restored to the previous state within a maximum latency of 3 seconds.

Step 4: Choose a Design Concept that Satisfies Architectural Drivers

4.1 ADRT

The Architectural Design Relation Table allows for a general overview of the architecture in order to determine what tactics and architectural Patterns would be effective in order to satisfy the architectural drivers. This ADRT is shown in the table below.

ADRT (Second Iteration)

ASR #	Case #	Quality Attributes	Tactics		Design Patterns	Architectural Patterns	Implementation on Technology	Concerns
ASR01	A01	Availability (1)	Detect faults	Ping / echo	ICMP Echo	Client-Server Pattern	ICMP (internet Control Message Protocol); OR Firebase, Android, Java, MongoDb	ping/echo successful, but OS is down so it does not function well e.g. sending message, etc. (OR: Server can be a performance bottleneck)
ASR02	A02	Availability (2)	Recover from Faults	Preparation and Repair/ Active Redundancy	Message Data Centralization	Client-Server Pattern	WebAPI,Firebase, Android, Java, MongoDb	Firebase Framework
ASR04	I01	Interoperability (1)	Locate	Discover Service	Mediator	SOA	Firebase Framework, MongoDb, SQL, Hadoop	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR06	P01	Performance (1)	Control Resource Demand	Increase Resource Efficiency		SOA	.NET, Firebase	Substantial up-front cost and complexity
ASR07	S01	Security (1)	Detect Attacks	Verify Message Integrity	Asymmetric Cryptography	Client-Server Pattern or SOA	SSL, TSL	Server can be a performance bottleneck. Decisions about where to locate functionality.
ASR09	S03	Security (3)	Resist Attacks	Encrypt Data	Asymmetric Cryptography	Client-Server Pattern	RSA, Blowfish Encryption	Expertise, cost, and securing the private keys. Server can be a performance bottleneck. Decisions about where to locate functionality.

4.2 Pros and Cons Matrix

In order to determine what patterns are best for the architectural drivers, the use of a Pros and Cons Matrix will neatly organize what patterns and tactics should be used in order to address the architectural drivers. The Pro's and Con's Matrix is in the table on the next page.
Conclusion: After all considerations the choice for what pattern would be used for the architecture would be a mix of Client-Server and Service Oriented Architecture Patterns.

Pros and Cons Matrix (Second Iteration)

				Pattern #1	Pattern #2
ASR Number / QA Number		**Architectural Driver**		**Client-Server Patterns**	**SOA (Service Oriented Architecture Pattern)**
ASR01	A01	Availability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers.	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet. It is designed to locate services and combine (and dynamically recombine) them into meaningful coalitions while achieving reasonable performance, security, and **availability**.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in the client or in the server) are often complex and costly to change after a system has been built.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR02	A02	Availability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers.	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet. It is designed to locate services and combine (and dynamically recombine) them into meaningful coalitions while achieving reasonable performance, security, and **availability**.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in the client or in the server) are often complex and costly to change after a system has been built.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR04	I01	Interoperability	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers. But this is not applicable to interoperability	It supports interoperability of distributed components running on different platforms and written in different implementation languages, provided by different organizations, and distributed across the Internet.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure. Decisions about where to locate functionality (in	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated

				the client or in the server) are often complex and costly to change after a system has been built.	with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR06	S01	Security	Pros	It improves scalability and availability by centralizing the control of these resources and services, while **distributing the resources themselves across multiple physical servers.**	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, security, and availability.
			Cons	Server can be a performance bottleneck. **Server can be a single point of failure**.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR07	S02	Security	Pros	It improves scalability and availability by centralizing the control of these resources and services, while **distributing the resources themselves across multiple physical servers.**	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, security, and availability.
			Cons	Server can be a performance bottleneck. **Server can be a single point of failure**.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.
ASR09	S04	Security	Pros	It improves scalability and availability by centralizing the control of these resources and services, while distributing the resources themselves across multiple physical servers.	It is designed to locate services and combine them into meaningful coalitions while achieving reasonable performance, **security,** and availability.
			Cons	Server can be a performance bottleneck. Server can be a single point of failure.	SOA-based systems are typically complex to build. You don't control the evolution of independent services. There is a performance overhead associated with the middleware, and services may be performance bottlenecks, and typically do not provide performance guarantees.

Conclusion Statement:

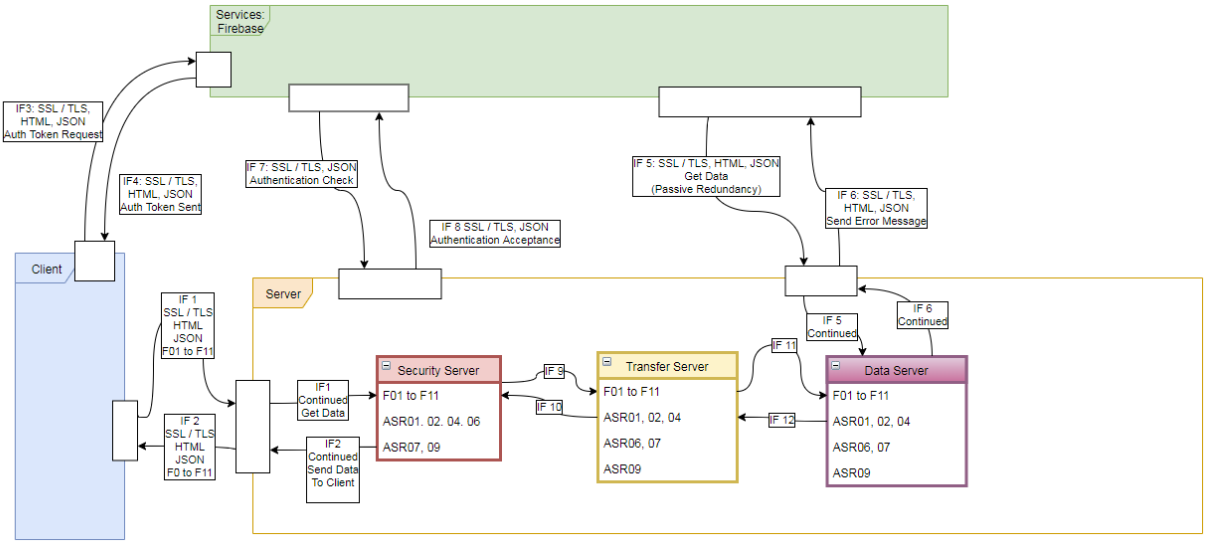
1. Client-Server Pattern: A01, A02, S04

2. SOA: I01, S01, S02

Step 5: Instantiate Architectural Elements and Allocate Responsibilities

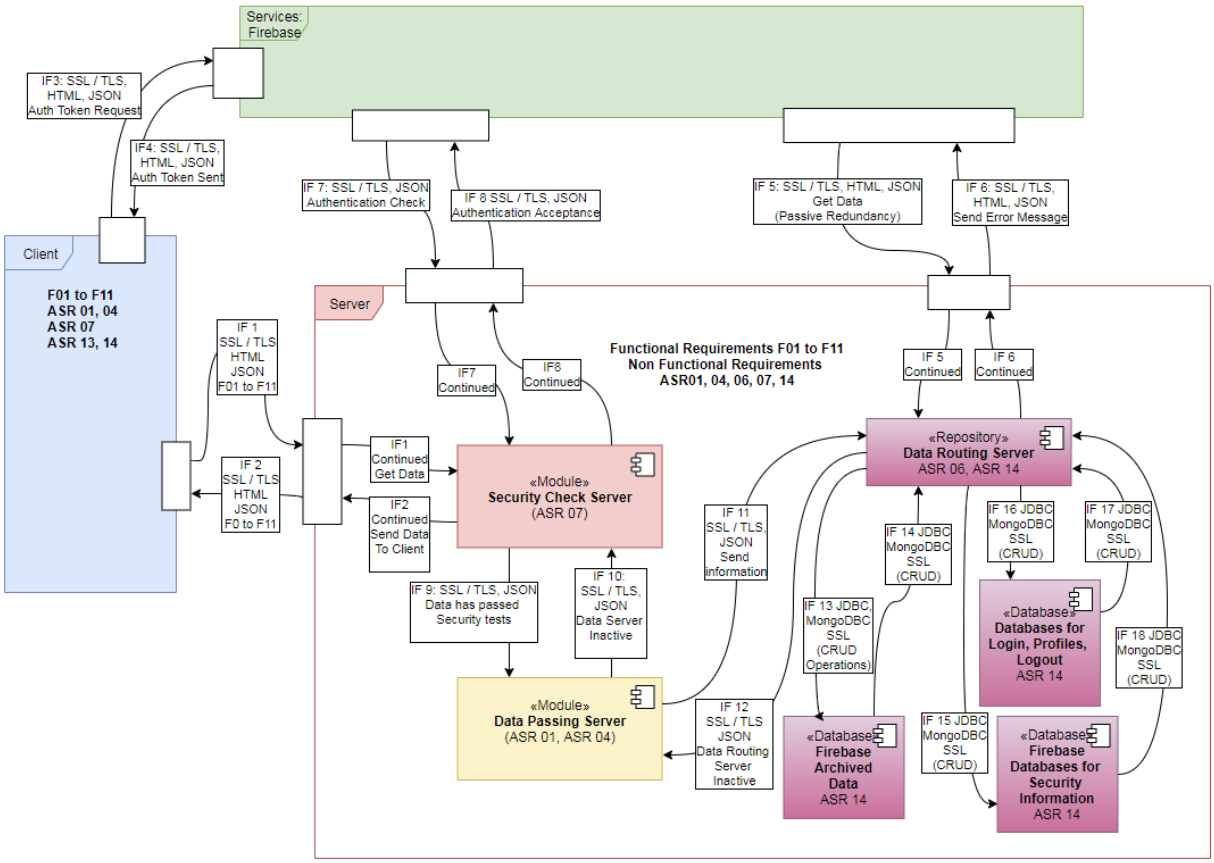
5.1 Decomposition of the Server Portion into the Security, Transfer, and Data Servers, Module View

The server portion is deconstructed into security server, transfer server, and data server. After discussing with the development team and architects, the following module view was created, and the functional requirements and architectural significant requirements were allocated after discussion.



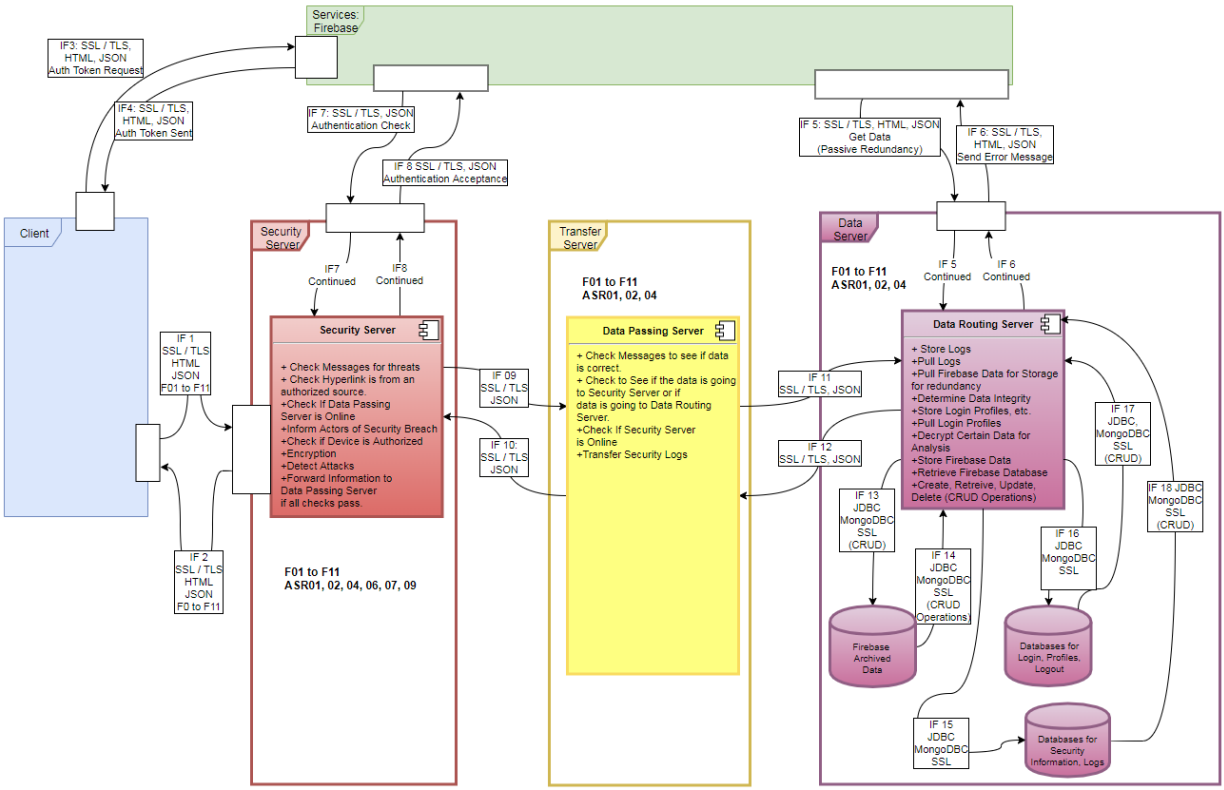
5.2 Decomposition of the Server Portion into Security, Transfer, and Data Servers, C&C View

The server portion is deconstructed into security server, transfer server, and data server. After discussing with the development team and architects, the following component and connector view was created, and the functional requirements and architectural significant requirements were allocated after discussion.



5.3 Decomposition of the Server Portion into the Security, Transfer, and Data Servers, Allocation View

The server portion is deconstructed into security server, transfer server, and data server. After discussing with the development team and architects, the following allocation view was created, and the functional requirements and architectural significant requirements were allocated after discussion. This is a very detailed view meant for developers and architects.



5.4 Allocation of Requirements

After deciding the architecture, the responsibilities were allocated within the instantiations, with the assistance of architects and developers.

Security-Server	
Allocation of Drivers	**Description**
F01	Allow the user to send a text message to another user
F02	Allow the user to receive a text message from another user
F03	Allow the user to send a voice message to another user
F04	User can send a voice message to another user
F05	User are allowed to upload media for other users to observer or interact with.
F06	User can make a private chat room
F07	User must login to access account
F08	User can create a friend's list of those they wish to contact.
F09	User is able to log off their account
F10	User can block disruptive individuals from messaging or contacting them
F11	Users are allowed to post hyperlinks to content from authorized providers
ASR01	The System is able to do Fault Detection
ASR02	The System is able to Recover from Faults/Active Redundancy
ASR04	The System is able to Discover Service/OS Compatibility
ASR06	The System is able to Increase Resource Efficiency
ASR07	The System is able to Detect Attacks/ Verify Message Integrity
ASR09	The System is able to Encrypt Data

Transfer-Server	
Allocation of Drivers	**Description**
F01	Allow the user to send a text message to another user
F02	Allow the user to receive a text message from another user
F03	Allow the user to send a voice message to another user
F04	User can send a voice message to another user
F05	User are allowed to upload media for other users to observer or interact with.
F06	User can make a private room chat room
F07	User must login to access account
F08	User can create a friend's list of those they wish to contact.
F09	User is able to log off their account
F10	User can block disruptive individuals from messaging or contacting them
F11	Users are allowed to post hyperlinks to content from authorized providers
ASR01	The System is able to do Fault Detection
ASR02	The System is able to Recover from Faults/Active Redundancy
ASR04	The System is able to Discover Service/OS Compatibility

Data Server	
Allocation of Drivers	**Description**
F01	Allow the user to send a text message to another user
F02	Allow the user to receive a text message from another user
F03	Allow the user to send a voice message to another user
F04	User can send a voice message to another user
F05	User are allowed to upload media for other users to observer or interact with.
F06	User can make a private room chat room
F07	User must login to access account
F08	User can create a friend's list of those they wish to contact.
F09	User is able to log off their account
F10	User can block disruptive individuals from messaging or contacting them
F11	Users are allowed to post hyperlinks to content from authorized providers
ASR01	The System is able to do Fault Detection
ASR02	The System is able to Recover from Faults/Active Redundancy
ASR06	The System is able to Increase Resource Efficiency
ASR07	The System is able to Detect Attacks/ Verify Message Integrity
ASR14	The System has Passive Redundancy

Step 6: Define Interfaces for Instantiated Elements

6.1 Define interfaces for instantiated elements

Interface	From Elements	Interface	To Element
IF1	Client: Mobile Client Android & iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF2	Server Portion: Security Server	HTML, SSL / TLS, JSON	Client: Android Application & iOS Application
IF 3	Client: Mobile Client Android & iOS Application	HTML, SSL / TLS, JSON	Server Portion: Security Server
IF 4	Server Portion: Security Server	HTML, SSL / TLS, JSON	Client: iOS Application
IF 5	Services: Firebase	HTML, SSL / TLS, JSON	Server: Data Server
IF 6	Server	SSL / TLS, JSON	Services: Firebase
IF 7	Services: Firebase	SSL / TLS, JSON	Server: Security Server
IF 8	Server: Security Server	SSL / TLS, JSON	Services: Firebase
IF 9	Server: Security Server	SSL / TLS, JSON	Server: Transfer Server
IF 10	Server: Data Passing Server	SSL / TLS, JSON	Server: Security Server
IF 11	Server: Transfer Server	SSL / TLS, JSON	Server: Data Server
IF 12	Server: Data Server	SSL / TLS, JSON	Server: Transfer Server
IF 13	Server: Data Server – Data Routing Server	JDBC, MongoDBC, SSL	Server: Data Server – Firebase Archived Data
IF 14	Server: Data Server – Firebase Archived Data	JDBC, MongoDBC, SSL	Server: Data Server – Data Routing Server
IF 15	Server: Data Server – Data Routing Server	JDBC, MongoDBC, SSL	Server: Data Server – Databases for Security Info, Logs
IF 16	Server: Data Server – Data Routing Server	JDBC, MongoDBC, SSL	Server: Databases for Login, Profiles, Logout
IF 17	Server: Databases for Login, Profiles, Logout	JDBC, MongoDBC, SSL	Server: Data Server – Data Routing Server
IF 18	Server: Data Server – Databases for Security Info, Logs	JDBC, MongoDBC, SSL	Server: Data Server – Data Routing Server

Step 7: Verify and Refine Requirements and Make Them Constraints for Instantiated Elements

In this iteration, the whole team verified that the element decomposition has met the requirements and design constraints, and responsibilities have been allocated accordingly.

References

Wojcik, Rob., Bachmann, Felix., Bass, Len., Clements, Paul., Merson, Paulo., Nord, Robert., & Wood, William. (2006). Attribute-Driven Design (ADD), Version 2.0 (CMU/SEI-2006-TR-023). Retrieved February 15, 2018, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8147

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd ed.). N.p.: Addison-Wesley Professional.

CSUIAM Section 8000 - Information Security. (2010, April 19). Retrieved from http://www.calstate.edu/icsuam/documents/Section8000.pdf

Information Security Policies, Standards, and Guidelines (n.d.). In *Policies, Standards, and Guidelines - Division of Information Technology | CSUF*. Retrieved April 30, 2018, from http://www.fullerton.edu/it/iso/policy/

Protecting the Privacy of Student Education Records. (1997, March). Retrieved from https://nces.ed.gov/pubs97/web/97859.asp