**MicroChat**

Brady Berner

CST-451 Capstone Project Proposal

Grand Canyon University

Instructor: Professor Mark Reha

Revision: v1.5

Date: 10/1/20

**ABSTRACT**

Communication is vital in today’s world. Companies rely on it to stay operating at break-neck paces. Consumers have grown so used to near-instantaneous communication that it is a must-have for many. However, many of the market applications today have security flaws, reliability issues, or cost consumers to use. This project will provide uses with a messaging application built to be fast, reliable, and secure MicroChat will allow users to easily communicate with each other, potentially from virtually any platform they desire. The application will remain simple to appeal to as many consumers as possible, regardless of demographics. To operate the application, a consumer can go to the website, log in, or create an account, select whom they would like to contact, and begin sending and receiving messages. The application could also serve as a form of consumable entertainment with a feature that would enable users to select whether they would like to start a conversation with another user chosen at random. The application could be expanded upon to allow for encrypted messaging to ensure user security and privacy and support multimedia messages such as pictures, videos, files, et cetera.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 9/27/20 | Brady Berner | Initial draft for review/discussion |
| 10/1/20 | Brady Berner | Revised proposal based on instructor feedback |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**Project Approval**

Professor Mark Reha

**TABLE OF CONTENTS**

[Project Overview and Project Objectives 4](#_Toc52479284)

[Project Overview 4](#_Toc52479285)

[Project Objectives 4](#_Toc52479286)

[Challenges 4](#_Toc52479287)

[Benefits and Opportunities 4](#_Toc52479288)

[Project Scope 5](#_Toc52479289)

[Project Success Measures 6](#_Toc52479290)

[Project High-Level Solution 7](#_Toc52479291)

[Introduction 7](#_Toc52479292)

[Solution 8](#_Toc52479293)

[Proposed Technologies 9](#_Toc52479294)

[Potential Additional Technologies 10](#_Toc52479295)

[Project Controls 11](#_Toc52479296)

[Project Schedule 13](#_Toc52479297)

[Appendix A – References 14](#_Toc52479298)

[Appendix B – Copyright Compliance 15](#_Toc52479299)

# Project Overview and Project Objectives

## Project Overview

Long-distance communication is considered a necessity, whether it comes to business or people’s personal lives. However, many of today’s communication applications have their flaws, such as stability issues, high costs, lack-luster security, or concerns about what the company behind the application might be doing with the user’s data. This project seeks to resolve those issues by creating a reliable, easy to user, secure, transparent, and free messaging application.

## Project Objectives

The resulting application should be accessible to and stable for users 99.99% of the time.

The application should be designed with security in mind to avoid as many potential vulnerabilities as possible.

The application should be intuitive to use. Users should be able to login/register and send a message within five clicks or less.

The application should be transparent and collect the bare minimum amount of information necessary to perform its desired functionality.

## Challenges

The biggest challenges to completing the project successfully include:

* Time management
* Creating a quality product
* Completing a sound technical design
* Ensuring the resulting applications security

## Benefits and Opportunities

Creating a single application capable of solving the majority of issues with current messaging solutions could result in potential revenue. While the application itself should remain free of charge minimally invasive advertisements could be included in the application as a source of income. Creating this application would also result in the development team being trained in new cutting-edge technologies that could benefit future projects.

# Project Scope

|  |  |  |
| --- | --- | --- |
| Functionality | | |
| Functionality | Description | In Scope |
| Login | Users should be able to log in to the application with their registered account or using another approved account such as a GitHub or Google account | Yes |
| Registration | Users should be able to register for an account if they do not have one | Yes |
| Messaging | Users should be able to send and receive messages to and from any other user registered for the application, users should also be able to delete sent messages | Yes |
| Contacts | Users should be able to add other users they commonly converse with as contacts | Yes |
| Random Messaging | Users should be able to request a random chat partner and then be paired with another user requesting the same to begin a conversation without two users being contacts | Yes |
| Conversation Storage | Users conversations should be preserved and stored by the application for users to access | Yes |
| Conversation Removal | Should a user wish they should be able to delete a conversation and should both members of the conversation remove it all messages in the conversation should be deleted | Yes |
| Encrypted Messaging | All messages sent and received through the application should be encrypted using AES | No |
| Mobile Application Capability | The project should be expanded upon to allow for the development of a mobile application in the future. (Building an API Gateway) | No |
| Mobile Application | Users should be able to access all features of the application they are able to via the web application from the mobile application | No |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Work Breakdown Structure | | | | | | | | |
| ID | Task | Dependencies | Status | Effort Hours | Start Date | Planned Completion | Estimate to Completion | Actual Completion | |
| 1 | Lean Canvas | - | Complete | ~1hr | 9/21/20 | 9/27/20 | - | 9/27/20 | |
| 2 | Project Proposal | Lean Canvas | Complete | ~6hrs | 9/21/20 | 9/27/20 | - | 9/28/20 | |
| 3 | Project Proposal Revision | - | Complete | ~5hrs | 9/29/20 | 10/1/20 | - | 10/1/20 | |
| 4 | Proofs of Concept | - | Active | - | 10/5/20 | 11/29/20 | < 72hrs | - | |
| 5 | Requirements Analysis | Project Proposal | Active | - | 10/19/20 | 11/1/20 | < 8hrs | - | |
| 6 | Final Architectural Plan | Requirements Analysis & Proofs of Concept | Pending | - | 11/16/20 | 11/29/20 | < 12hrs | - | |
| 7 | Development | Final Architectural Plan | Pending | - | 12/14/20 | 12/20/20 | < 24hrs | - | |

# Project Success Measures

|  |
| --- |
| Project Completion Criteria |
| 1 – All in scope feature should be completed before the end of CAPSTONE 2 |
| 2 – All deliverables should be completed and handed in on time according to the project schedule |
| 3 – All desired technologies and patterns were successfully learned and implemented |
| 4 – Application be able to pass a vulnerability test |
| 5 – At least some out of scope features were brought into scope and completed |
| 6 – Successful implementation of a microservice architecture |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Assumptions and Constraints | | | | | |
| ID | Description | Comments | Type | Status | Date Entered |
| 1 | React Deployment | Currently it is assumed that React deployment will not need any additional technologies. | Assumption | Active | 9/27/20 |
| 2 | MongoDB Querying | Currently it is assumed that Spring Boot will have built in implementation that allows for relatively easy querying of Mongo DB databases | Assumption | Active | 9/27/20 |
| 3 | Time Between Milestones | There is an extremely limited amount of time in between milestones with a large number of tasks to be completed in each | Constraint | Active | 9/27/20 |

# Project High-Level Solution

## Introduction

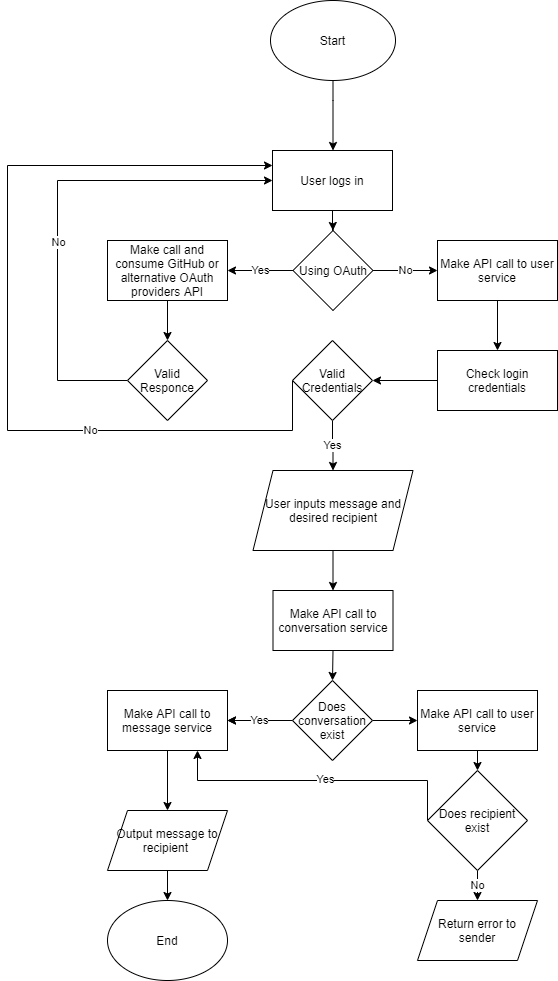
The primary purpose of this project is to realize an application that utilizes a microservice based architecture from design to realization. Compared to a traditional monolithic software architecture microservices take the traditional monolith and divide it horizontally across. The same applies to upgrading/expansion, while traditionally more code would be added on to the existing code base creating an ever-expanding application adding to microservices is instead done by adding on small amounts to existing microservices or creating a new service. With each service being separate however there are some major variations to consider such as the fact that each service needs its own database. Since each service needs its own database and the services are only loosely connected to each other the benefit of relational databases are mostly lost, meaning that non-relational databases are usually the go to solution as their drawbacks are already present so the design is left only really taking their benefits. With all this in mind below is a figure depicting a simple application built using a microservice based architecture.

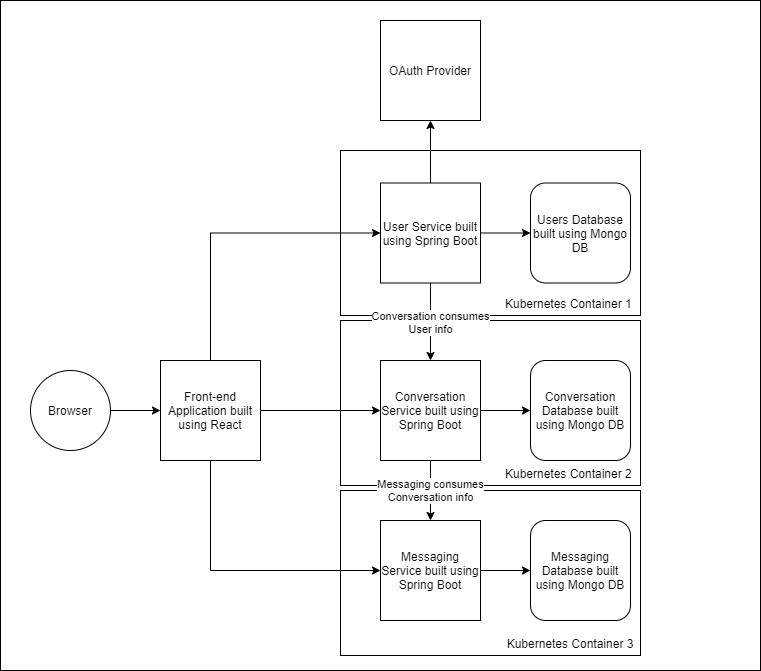


(Richardson, 2020)

Looking at this figure shows how REST APIs are typically used to tie the architecture together making the application even more flexible. Some of the biggest challenges with a microservice based architecture however has to do with its inherent higher level of complexity. This complexity creates issues like maintaining data consistency between each services database, implementing some form of service discovery to allow services to communicate with one another when necessary, and many other unique issues.

## Solution

When looking at this project’s proposed solution there are several aspects to look at starting with the basic functionality of the application. At the most basic logic the application needs to follow the logic depicted in the included flowchart to be considered a successful messaging service. With this functionality being relatively minimal the difficult of the project lies mainly in its architectural complexity as well as the variety of technologies being used.

In order to properly take advantage of a microservice architecture the application will be divided up into three main services, the user service, the conversation service, and the messaging service. The user service will handle typical user info and logging in and out of the application. The conversation service will keep a record of what users are talking to each other which is vital to being able to preserve conversations in an application without a relational database with those established connections. Lastly the messaging service will handle the actual receiving, storing, and then distributing of user messages to one another. While each of these services is simple enough on its own the fact that each service needs to communicate with at least one other service increases the complexity of the application as well as the necessary solution. Ideally the application will be divided into two layers, the front-end user interface, and the backend where the API, application logic, and databases lie. As such the basic architecture would be along the lines of what is depicted in the figure below. 

## Proposed Technologies

* React
  + Version 16.13.1
* Spring Boot
  + Version 2.4.0 M3
* MongoDB
  + Version 4.4
* Kubernetes
  + Version 1.19.2
* OAuth 2.0
* Material-UI
* IntelliJ
* WebStorm
* GitHub
* Java 8

## Potential Additional Technologies

* Websocket
* Spring Cloud
  + Version 2.2.5
* Swift
* Advanced Encryption Standard (AES)

# Project Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Management | | | | |
|  | **Risk Probability** | **Risk Impact** |  |  |
| **Event Risk** | **(high, medium, low)** | **Risk Mitigation** | **Contingency Plan** |
| What is the risk? | What is the probability? | What is the impact if the risk occurs? | What can be done to minimize the risk? | What can be done to minimize the impact of the risk? |
| Learning Spring Boot | Low | Should I fail to learn spring boot creating the entire back end as envisioned would not be possible | Spend an adequate amount of time not only learning Spring Boot but completing relevant proofs of concept | Switch to Spring MVC or a similar back end technology that is already familiar |
| Learning React | Low | Should I fail to learn React creating the envisioned front end would not be possible | Spending an adequate amount of time learning React, as well as completing relevant proofs of concept | Switch to another front-end language that is already familiar |
| Learning Mongo DB | Low | Should I fail to learn Mongo DB the current plans for the applications database would not be possible | Spend an adequate amount of time learning Mongo DB and work on not only proofs of concept just focusing on Mongo but also on Mongo interacting with Spring Boot | Switch to a more familiar relational database language like MariaDB or switch to a known language like MySQL |
| Service Discovery | Medium | Without implementing service discovery, the application’s services would be unable to interact with one another crippling the majority of the application’s functionality | Spend an adequate amount of time discovering, researching, and implementing a service discovery solution | Redesign the application so that services work entirely independent of each other or switch the entire application back over to a monolithic architecture. |
| CQRS Solution | Medium | Without implementing a CQRS or API composition solution it would not be possible to query from multiple databases at a time decreasing the functionality of the application or increasing the complexity. | Spend as much time as necessary to find, learn, and implement a CQRS or API composition solution to enable cross database queries | Switch to a relation database like MariaDB or MySQL |
| Saga Pattern | Medium | Without implementing the saga pattern into the application there is a good chance that discrepancies could occur between databases | Spend as much time as possible attempting to research and implement a saga pattern solution or spend time regularly checking for and fixing database discrepancies | Regularly check the databases for discrepancies and fix them manual. |
| OAuth 2.0 | Low | Should I be unable to implement OAuth it would no longer be possible to use it as an account/API authentication solution | Spend time looking through GitHub’s OAuth documentation and learning/completing proofs of concepts implementing the technology | Switch to traditional account management and using traditional API keys |
| Websocket | Medium | If I am unable to learn/implement websocket protocol into my design, then I will need to use an alternative such as a REST API or event subscriber | Spend time researching and potentially working on proofs of concept to allow for implementation | Fall back to a known technology such as REST APis |
| CQRS Solution | Medium | If a CQRS solution is not implemented, it will not be possible for a single service to get information from multiple databases potentially limiting the applications functionality | Spend time research and complete a proof of concept implementing a CQRS Solution | Create multiple calls and aggregate the returned data at a higher level of the application |
| Saga Pattern | Medium | If a Saga Pattern is not implemented than it is very possible that there could end up being discrepancies between databases | Spend time researching and implementing the Saga Pattern into the CQRS proof of concept | Manually maintain the databases by regularly checking for discrepancies and resolving them |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Issues Log | | | | | | | |
| **ID** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| 1 | Project Proposal Revision | Work on proof of concepts is slightly delayed | Complete the revision as quickly as possible and devote extra time to proofs impacted when possible | High | 9/28/20 | 9/29/20 | 10/1/20 |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Change Control Log | | | | | | | | | |
| **ID** | **Change Description** | **Priority** | **Originator** | **Date Entered** | **Date Assigned** | **Evaluator** | **Status** | **Date of Decision** | **Included in Rev. #** |
| 1 | Project Proposal Revision | High | Prof. Reha | 9/29/20 | 9/30/20 | Brady Berner | Complete | 10/1/20 | 1.5 |
| 2 |  |  |  |  |  |  |  |  |  |

# Project Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Schedule | | | | | |
| **Task** | **Start Date** | **Target End Date** | **Actual End Date** | **Status** |
| Project Proposal | 9/21/20 | 9/27/20 | 9/28/20 | Complete |
| Project Proposal Revision | 9/29/20 | 10/5/20 | 10/1/20 | Complete |
| Proofs of Concept | 10/1/20 | 11/29/20 | - | Active |
| Requirements Analysis | 10/19/20 | 11/1/20 | - | Active |
| Final Architectural Plan | 11/16/20 | 11/29/20 | - | Pending |
| Development | 12/14/20 | 12/20/20 | - | Pending |

# Appendix A – References

Richardson, C. (2020). Microservice Architecture Diagram. https://microservices.io/i/Microservice\_Architecture.png.

# Appendix B – Copyright Compliance

Net yet applicable.