Secure Communication Platform Functional Specification

27th November 2020

Bachelor Of Science (Honours)
Software Development

Liliana O'Sullivan
C00227188

Paul Barry
Project Supervisor

Institiúid Teicneolaíochta Cheatharlach
INSTITUTE *of* TECHNOLOGY CARLOW
At the Heart of South Leinster

Table of Contents

1. Table of Figures

2. Introduction

This document will showcase the functional and non-functional requirements for the development of this project. It will include diagrams to showcase or explain concepts that where appropriate.

3. Project Description

This is a research project to develop a server that exposes an Application Programming Interface (API) to a conversation-facilitating platform. The ultimate goal is to enable developers to create their own clients that can tap into the API. The platform sets out to ensure user's conversations have technology-backed security in place. The platform is attempting to provide an additional choice of a security-oriented platform to end-users. The server nor a third party eavesdropping on the conversation will have the capability of determining the conversation in place.

The API server will follow and document using the OpenAPI standard as set by the Linux Foundation. Any features provided by the platformed are chosen based on the mass-reach it applies to users. The API functionalities can be loosely classified into the use case shown in Figure 1.

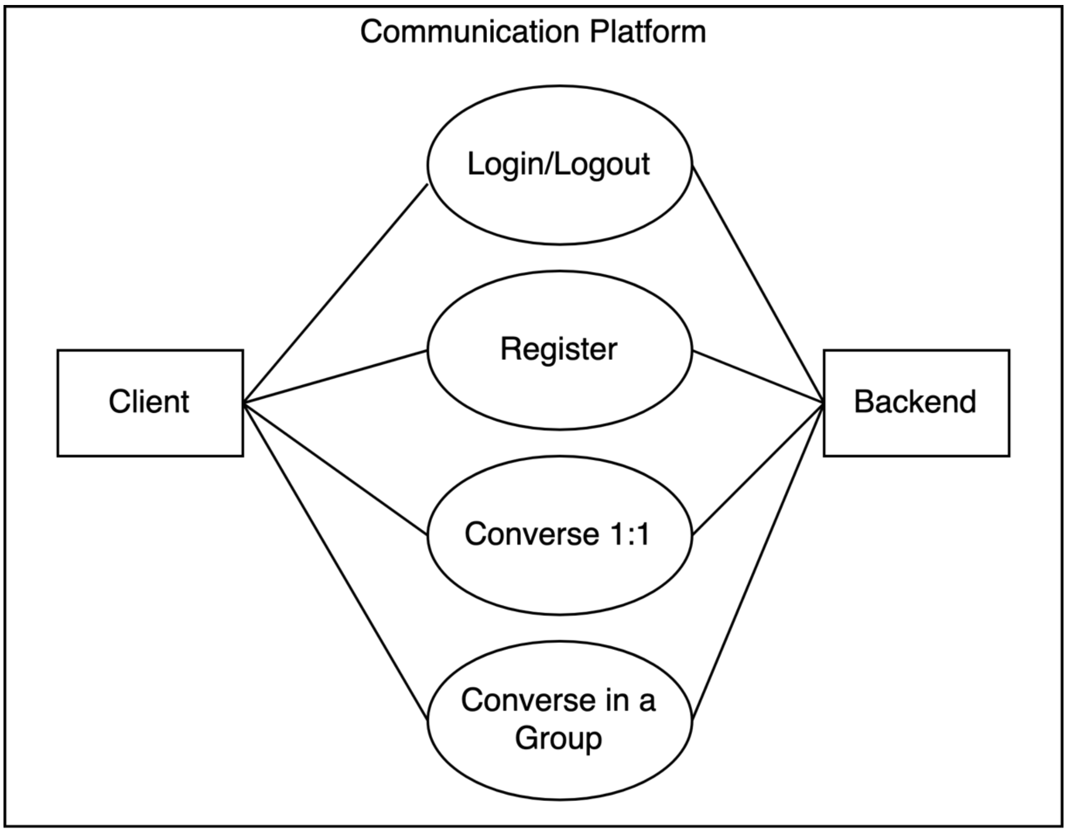

Figure 1 Use Case

Use Case	Login
Actors	Client, Backend
Pre-conditions	An account has been registered
Brief Description	This use-case begins once a client requests to log in to an existing account
Main Success Scenario	1. The client forwards credentials to log in to an account 2. The backend verifies the existence of an account with the provided credentials. 3. The backend logs the client in and responds with success
Alternatives	2a. The credentials provided are not associated with an account 2.1 An error is returned to the client 2.2 The client forwards new information 2.3 The backend checks the new credentials 2.4 The backend logs the client in, replies in acknowledgement.
Post-conditions	A client is logged into the system.

Use Case	Register
Actors	Client, Backend
Pre-conditions	None
Brief Description	This use-case begins once a client is launched and is about to make its first request.
Main Success Scenario	4. The client forwards required information to register to the backend 5. The backend checks the information, such as ensuring an account does not currently exist. 6. The backend creates an account and responds to the client
Alternatives	2a. The information entered is already tied to an account 2.1 An error is returned to the client 2.2 The client forwards new information 2.3 The backend checks the new credentials 2.4 It creates the account and replies with success
Post-conditions	A newly created account exists on the database

Use Case	Converse 1:1
Actors	Client, Backend
Pre-conditions	The client is logged in to an existing account
Brief Description	This use-case begins once a client requests to converse with an existing account.
Main Success Scenario	1. The client sends a request desiring to converse, specifying a particular account 2. The backend verifies the request by ensuring the client requesting to converse is logged in and verifies the cryptographic integrity. 3. The backend forwards the desire to the client requested.
Alternatives	2a. The clients request failed cryptographic integrity. 2.1 An error is returned to the client 2.2 The client forwards a new request 2.3 The backend verifies the request. 2.4 The backend upholds the request and returns a success
Post-conditions	None

Use Case	Converse in a group
Actors	Client, Backend
Pre-conditions	The client is logged in to an existing account
Brief Description	This use-case begins once a client requests to converse with a group
Main Success Scenario	1. The client sends a request desiring to converse with a request-included list of accounts 2. The backend verifies the request by ensuring the user requesting to converse is logged in and verifies the cryptographic integrity 3. The backend forwards the request to all the specified accounts on the list
Alternatives	2a. The credentials provided are not associated with an account 2.1 An error is returned to the client 2.2 The client forwards new information 2.3 The backend checks the new credentials 2.4 The backend logs the client in, replies in acknowledgement.

Post-conditions	None

4. Users

The project is explicitly targeted towards developers; however, the resulting platform ultimately targets end-users. The technical side of the project, building the API server and producing the appropriate documentation, is targeted towards technically minded developers. The developers are primarily concerned with an intuitive and well-documented API.

Documentation is essential for expressing expectations and behaviour to developers without delving into source code to examine its inner workings. Documentation dictates how a developer should interact with the API, with expected responses from the server.

The end-user of the platform is not specifically targeting an individual end-user or a specific use-case. The platform as a whole aims to provide generic tools to converse that can accommodate multiple use-cases. The generic approach is taken in an attempt to maximise user reach. The platform will differentiate itself through the security-conscious design. While the platform aims to be accessible to most users, even if the user is not technically savvy, the individual experience provided will be created ultimately by the client used.

The platform is aimed to be intuitive and accessible. The APIs provided will be high-level, aiming to provide abstractions to the cryptography and technical aspects behind the interface.

5. FURPS+

FURPS is an acronym for Functionality, Usability, Reliability, Performance and Supportability. The software in question is discussed with information related to each caption. The '+' is appended for any requirements that do not neatly fit into the previously mentioned classifications.

5.1 Functionality

This refers to the main feature-set provided by the software. The following are considered core functionality to the application.

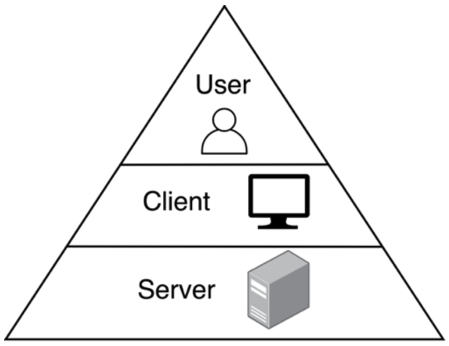

Figure 2 Hierarchy Triangle

The project-server aims to be the fundamental logic on which client applications are built upon. The end-user will not have direct contact with this server, instead using a client to create the end-user experience. This client can come in many forms, for example, a mobile application running on an Android or iOS device, a Desktop user interface on a Linux operating system or a web-based client. A simplified hierarchy can be seen in Figure 2.

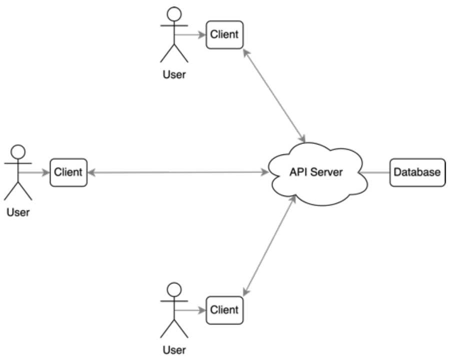

Figure 3 Overview

Once a client 'interfaces' with the API, it should enable users to connect and converse with each other. A visual demonstration can be seen in Figure 3. A client is likely to implement many user-friendly features that are not controllable by the server. The API will be documented, specifically targeted at a technical audience, to aid in the development of a client.

5.1.1 Account Management

The server must provide an interface capable of registering an account, logging in and logging out.

5.1.2 Conversation Capabilities

The server must be capable of enabling conversations. This is one of the fundamental purposes of the platform. A logged-in user should have the ability to initiate a conversation with another user of the platform.

5.2 Usability

Usability is about the user experience of the application. This generally is in the form of the user interface, such as the aesthetics or responsiveness. Usability also encapsulates any documentation or support resources made available.

The API will be constructed using the OpenAPI Specification. This is a specification overseen by the OpenAPI Initiative, which is part of the Linux Foundation. The specification is language-independent and attempts to define the structure of APIs to ensure human understanding without the need to access the source code or documentation behind it.

5.3 Reliability

Reliability refers to the stability and availability of the application. This can be measured by multiple factors such as Failure Frequency or Mean Time Between Failures.

The API will be using Transmission Control Protocol (TCP) for network connections. TCP is more reliable, in relation to guarantees on deliverance, over the use of User Datagram Protocol (UDP). This is due to the recipient checking the arrival of the send information and sending a request for information that did not arrive.

5.4 Performance

This refers to the speed and scalability of the application. It is about the expected response times of the software, the anticipated throughput and its resource consumption.

- A code profiler will be applied in the development of the platform

5.5 Supportability

This classifies the maintainability, flexibility and compatibility of the application. This can refer to an applications ability to incorporate multiple localisations, its ability to operate on multiple platforms or its ability to be maintained in the long term. Due to the developer-centric nature of the application and the general trend of conversational platforms evolving, it is essential to ensure maintainability in this project. The following targets have been set to be met:

- The API will be documented targeted towards a technical audience
- The code must be sufficiently commented
- The code must have a high degree of test coverage

5.6 +

The '+' aims to capture any non-functional requirements that did not fit into the previous sections. It can also be used to capture application-specific requirements.

5.6.1 Security

As security is the focus of this project, it has high requirements for security. It is required that user data, as it flows from a user's computer to its endpoint, should be enciphered and accessible exclusively to the end-users. This includes the server host, who should only have access to the cypher text. The conversation data should be secure in transit and at rest. In-transit data is information that has not reached the intended recipients. Data at rest is information that is stored by the server-host, such as information in the database. This is showcased in Figure 4.

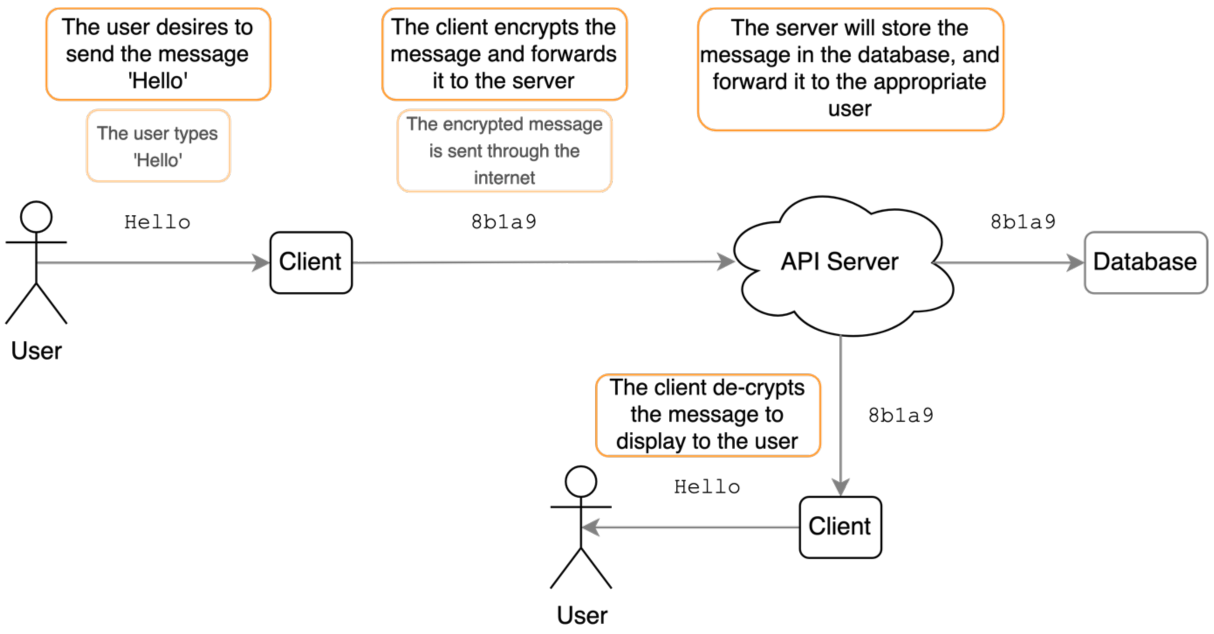
The user desires to send the message 'Hello'
The user types 'Hello'
The client encrypts the message and forwards it to the server
The encrypted message is sent through the internet
The server will store the message in the database, and forward it to the appropriate user
Hello
Client
8b1a9
API Server
8b1a9
Database
User
The client de-crypts the message to display to the user
8b1a9
Hello
Client
User

Figure 4 End-To-End Encryption

Each cryptographic algorithm implemented within the system will require a meticulous evaluation and, finally, carefully concluded upon. A static analysis tool must be run to ensure the following of programming best practices. Static analysis tools aid in development by performing automated analysis on code to find bugs, security vulnerabilities, improving performance and following programming language best practices.

Configuration options will allow encrypted or clear-text conversations.