Deliverable #2 Template

SE 3A04: Software Design II – Large System Design

Tutorial Number: T03
Group Number: G8
Group Members:

- Kyle Jordan Ball McMaster
- Daniel David Franze-Da Silva
- Rosa Chen
- Aidan Edward Froggatt
- Edward Gao

1 Introduction

1.1 Purpose

The purpose of this document is to describe the high-level architecture of the Secure Chat Android Application. The architecture is designed to enable encrypted communication within an organization that handles highly sensitive information. This document is targeted to internal stakeholders within the project scope, such as project managers, software developers, security experts, and other team members. The architectural decisions made in this document builds upon the requirements laid out in Deliverable 1, and depends upon background knowledge in mobile application development and cryptography.

1.2 System Description

The architecture of the Secure Chat Android Application adopts a service-oriented approach, integrated with a robust security model that includes key distribution, authentication, and encryption mechanisms. This structure is designed to provide the security and privacy for Android users, ensuring that all communications are encrypted and users are authenticated through established protocols.

1.3 Overview

In Section 2, we present an Analysis Class Diagram that describes the high-level structure and interactions between different elements of the Secure Chat Android Application. Section 3 describes the architectural design and rational behind the chosen patterns and design descisions. Section 4 describes the Class Responsibility Collaboration (CRC) Cards, which break down the architecture into individual class responsibilities and interactions.

2 Analysis Class Diagram

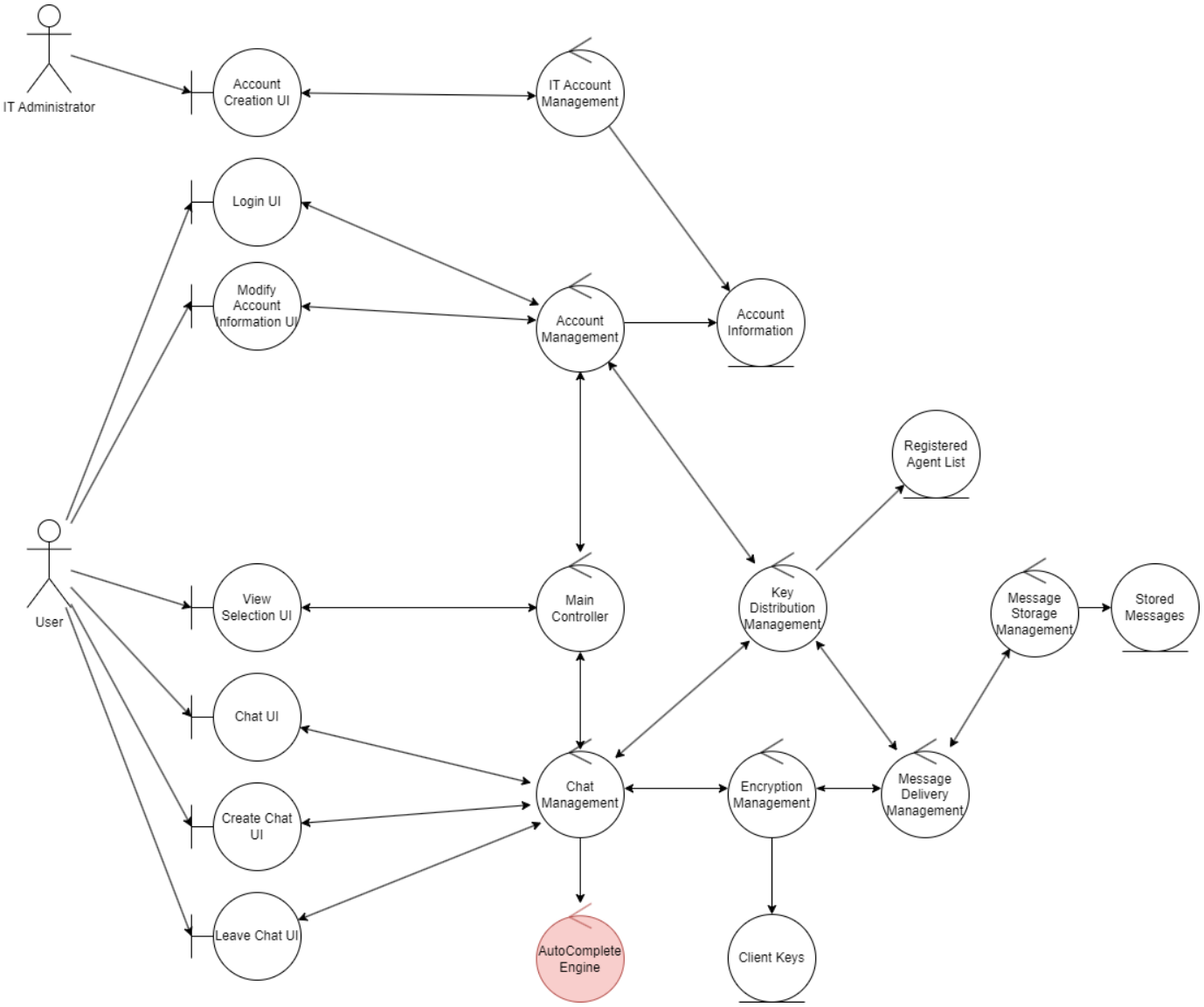


Figure 1. Analysis Class Diagram

3 Architectural Design

3.1 System Architecture

The secure chat application uses the Presentation-Abstraction-Control (PAC) architectural style. The PAC architectural style uses a hierarchal system to decompose complex systems into numerous smaller agents. Each agent consists of three parts which include the presentation, abstraction, and control components. The presentation component is responsible for the graphical user interface seen by the user. The abstraction component is responsible for storing and processing data. Finally, the control component coordinates the communication between the presentation and abstraction components.

In many situations, more than one agent is required to build a system. When introducing multiple agents, agents are sorted into a hierarchy connected by their controller components. The top-level agent oversees fundamental processes which includes the business logic and core data. Intermediate and lower-level agents each perform their own sub-functionality independently. When all agents are combined together, an entire coordinated system is created.

The reason why we chose to use the PAC architecture for the secure chat application is because it provides high cohesion and low coupling. It is a very effective solution for interactive applications with a graphical user interface. By having independent agents, it is easy to break down a complex application into smaller, more manageable components which oversee an independent task. From our analysis class diagram, the Main Client Controller acts as the top level agent, and it is connected to lower level agents through their controllers. The Main Client Controller is connected to the Chat Management and User Account Management systems, which are connected with the remaining lower level agents. This architecture style is beneficial because users will only be interacting with the presentation component (The UI boundary classes). In return, user input through the presentation components will activate appropriate responses from the controller and abstraction components. This aspect of PAC is extremely useful because users should only be interacting with the UI component, allowing the data processing to be encapsulated by the system ensuring better security and structure.

Additionally, using the PAC architecture style supports modification and expansion. Since PAC agents are not dependent on other agents, the system can be easily modified by simply adding or removing agents. This is extremely useful for development of the secure chat application since it can be divided into many smaller subsystems (i.e. The contact management system, the KDC distribution system, etc). Each subsystem can be developed independently and modifying an agent will not cause major changes to the entire system. Futhermore, agents in PAC are able to comunicate dynamically with one another, providing agent reusability, extensibility, multi-tasking, and multi-viewing.

Within the overarching PAC achitecture style, the Chat Management, Account Management, and Contact Management subsystems implement the Repository architecture style. All other subsystems do not have a sub-architecture style. The Repository architecture style was selected due to its widespread usage in database management, efficiently storing large amounts of data in a central data store. In our secure chat application, the Chat Management, Acount Management, and Contact Management subsystems each maintain their own dedicated database for a user account. This choice aligns well with the Repository style's central data storage approach, facilitating data management through specialized data manipulators. As a result, this allows clients to access data while abstracting the underlying details. This supports backup, data integrity, and restore features which are security advantages for our application.

Structural Architecture Diagram

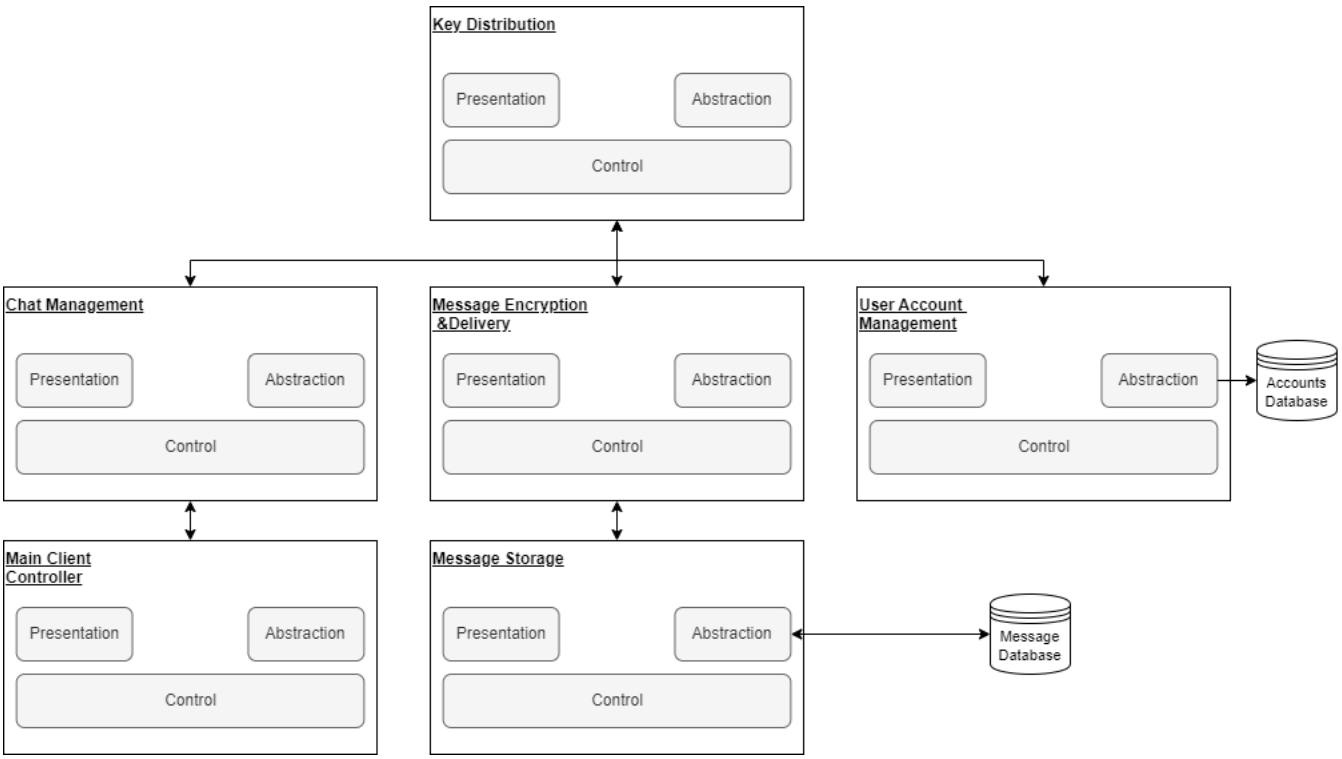


Figure 2. Structural Architecture Diagram

Design Alternatives

When selecting the architecture for our system, we also considered Model-View-Controller (MVC) architecture. MVC is an interaction oriented architecture, which makes it suitable for systems which have a user interface. Although our application does have a user interface, it also has a number of subsystems working to process the messages. Pure MVC is less suitable when using multiple independent agents, which is a key requirement for our system. If we were to use MVC, we would have had to contort our agents to fit in one of the three layers. Using PAC we can allow each agent to be self-contained, which is more natural based on function of the agents.

We also considered using the Batch Sequential Architecture. This is because we have some data that travels sequentially from the Chat Management system to the Message Storage system. Two key factors we valued in our choice of architecture was the ability to support concurrency and interactive interfaces. Since Batch Sequential does not accomodate these two features, we decided to not go through with implementing this style.

We did not consider any additional architecures for the system architecture as human interaction is a key requirement for our system.

3.2 Subsystems

- Main Client Controller
- Chat Management
- Message Encryption & Delivery
- Message Storage

- Key Distribution
- User Account Management
- Contact Management

Main Client Controller: controls the user interface and control flow for the client application. It allows the user to navigate between messaging and account management.

Chat Management: manages the display and composition of messages. When typing messages, it displays all the necessary UI components and message history to the user. It also generates automatic predicted responses to incoming messages. When a message is sent, it prepares the message to be passed to the Message Encryption and Delivery subsystem.

Message Encryption & Delivery: has several responsbilities surrounding the sending and storage of messages. Specifically, it communicates with the Key Distribution subsystem to ensure it has updated keys for message encryption. With those keys, this subsystem will encrypt outgoing messages and decrypt incoming messages. It will then check with the contact management system that the sending user is permitted to send to the intended target. If so, it will coordinate the delivery of the message. When the message has been sent, it will store the message with the Message Storage subsystem.

Message Storage: is responsible for the storage of messages. This system receives messages from the Message Encryption & Delivery subsystem and stores these messages into a secure database. It also has a user interface for the IT team to request formatted reports of chat logs.

Key Distribution: responsible for managing the generation and management of keys. After keys have been generated, they will be stored in the Registered Agent List and sent to the Encryption Management subsystem to be distributed to users.

User Account Management: manages the interaction between the user and the application. User Account Management is responsible for presenting the UI for the login page and modifying account information which is stored in Account Information. It also communicates with the Contact Management subsystem to retrieve data of available contacts.

Contact Management: responsible for adding, removing, and storing contacts. Sends contact information to the User Account Management to allow users to edit their contacts. Also receives information from the Message Encryption and Delivery subsystem to ensure that messages are delivered to the correct contact.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: Chat UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Knows chat message history • Displays chat messages • Handles message field input • Handles message send button click	• Chat Management Controller

Class Name: View Selection UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Main Controller • Displays chat selection buttons • Handles navigation button click	• Main Controller

Class Name: Respond to Contact Request UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Contact Management Controller • Knows contact request information • Displays contact request information • Handles contact acceptance button click	• Contact Management Controller

Class Name: Manage Contact UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Contact Management Controller • Knows contact list • Displays contact list • Handles contact addition button click • Handles contact removal button click	• Contact Management Controller

Class Name: Account Selection UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Account Management Controller • Knows account information • Displays account information • Handles account modification fields input • Handles account modification save button click	• Account Management Controller

Class Name: Login UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Account Management Controller • Handles login fields input • Handles login button click	• Account Management Controller

Class Name: Account Creation UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows IT Account Management Controller • Knows account information • Handles account creation fields input • Handles account creation button click	• IT Account Management Controller

Class Name: Report Generation UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Message Storage Management Controller • Handles report generation fields input • Handles report generation button click • Displays report generation results	• Message Storage Management Controller

Class Name: IT Account Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Account Creation UI • Knows Account information • Creates new accounts for users	• Account Creation UI • Account information

Class Name: Account Information (Entity)	
Responsibility:	**Collaborators:**
• Knows Account Management Controller • Knows IT Account Management Controller • Contains information regarding personal account information	• Account Management Controller • IT Account Management Controller

Class Name: Account Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Login UI • Knows Modify Account information UI • Knows Main Controller • Knows Account Information • Performs page navigation • Performs modifications to account information	• Login UI • Modify Account information UI • Main Controller • Account Information

Class Name: Contact List (Entity)	
Responsibility:	**Collaborators:**
• Knows Contact Management Controller • Contains information regarding a users contacts	• Contact Management Controller

Class Name: Contact Management (Controller)	
Responsibility:	**Collaborators:**
• Know Account Management Controller • Knows Manage Contact UI • Knows Respond to Contact Request UI • Knows Contact List • Knows Message Delivery Management Controller • Adds and removes contacts • Verifies sender can message recipient	• Account Management Controller • Manage Contact UI • Respond to Contact Request UI • Contact List • Message Delivery Management Controller

Class Name: Main Controller (Controller)	
Responsibility:	**Collaborators:**
• Knows View Selection UI • Knows Account Management Controller • Knows Chat Management Controller • Handles page navigation	• View Selection UI • Account Management Controller • Chat Management Controller

Class Name: Create Chat UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Knows contact list • Handles click event "Create" button • Handles selection of personnel to begin chat with	• Chat Management Controller

Class Name: Leave Chat UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Handles click event "Leave" button	• Chat Management Controller

Class Name: Record Audio UI (Boundary)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Handles click event of voice record button • Handles audio event of microphone recording • Handles playback of audio and click event to confirm	• Chat Management Controller

Class Name: Chat Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Chat UI • Knows Create Chat UI • Knows Leave Chat UI • Knows Record Audio UI • Knows AutoComplete Engine • Knows Audio Processing Engine • Knows Encryption Management • Knows Main Controller • Performs chat navigation • Processes chat modification • Process the interaction of sending and receiving messages	• Chat UI • Create Chat UI • Leave Chat UI • Record Audio UI • AutoComplete Engine • Audio Processing Engine • Encryption Management • Main Controller

Class Name: AutoComplete Engine (Controller)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Generate word completion suggestions • Generate word recommendation suggestions	• Chat Management Controller

Class Name: Audio Processing Engine (Controller)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Knows Encryption Management Controller • Process audio files	• Chat Management Controller • Encryption Management Controller

Class Name: Client Keys (Entity)	
Responsibility:	**Collaborators:**
• Knows Encryption Management Controller • Contains information regarding active encryption/decryption keys	• Encryption Management Controller

Class Name: Encryption Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Chat Management Controller • Knows Audio Processing Engine Controller • Knows Key Distribution Management Controller • Knows Message Delivery Management Controller • Knows Client Keys • Performs the encryption and decryption of messages	• Chat Management Controller • Audio Processing Engine Controller • Key Distribution Management Controller • Message Delivery Management Controller • Client Keys

Class Name: Key Distribution Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Encryption Management Controller • Knows Registered Agent List • Generates fresh keys for communication • Processes the interaction between registered agents and keys	• Encryption Management Controller • Registered Agent List

Class Name: Registered Agent List (Entity)	
Responsibility:	**Collaborators:**
• Knows Key Distribution Management Controller • Contains information regarding the registered agents in a communicating session	• Key Distribution Management Controller

Class Name: Message Delivery Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Encryption Management Controller • Knows Contact Management Controller • Knows Message Storage Management Controller • Performs the authentication of message sending	• Encryption Management Controller • Contact Management Controller • Message Storage Management Controller

Class Name: Message Storage Management (Controller)	
Responsibility:	**Collaborators:**
• Knows Message Delivery Management Controller • Knows Report Generation UI • Knows Stored Messages • Performs storage of message histories • Performs report generation	• Message Delivery Management Controller • Report Generation UI • Stored Messages

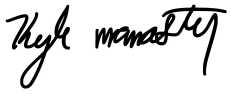
Class Name: Stored Messages (Entity)	
Responsibility:	**Collaborators:**
• Knows Message Storage Management Controller • Contains message histories between communicators	• Message Storage Management Controller

A Division of Labour

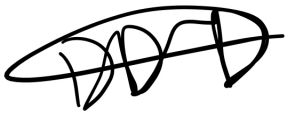
Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

Section	Team Member	Delivered on	Reviewed by	Reviewed on
1.1	Edward	2024-03-07	Kyle, Daniel	2024-03-10
1.2	Edward	2024-03-07	Kyle, Daniel	2024-03-10
1.3	Edward	2024-03-07	Kyle, Daniel	2024-03-10
2 Brainstorm & First Draft	Team	2024-03-04	Kyle	2024-03-04
2 Final Draft	Kyle	2024-03-07	Rosa	2024-03-07
3.1 (excluding diagram and rejected architectures)	Rosa	2024-03-09	Kyle	2024-03-10
3.1 (diagram and rejected architectures)	Kyle	2024-03-09	Rosa	2024-03-09
3.2 (first 4 subsystems)	Kyle	2024-03-09	Rosa	2024-03-09
3.2 (remaining 3 subsystems)	Rosa	2024-03-09	Kyle	2024-03-09
4	Daniel and Aidan	2024-03-10	Daniel	2024-03-10
Record Audio UI	Daniel	2024-03-10	Daniel	2024-03-10
Leave Chat UI	Daniel	2024-03-10	Daniel	2024-03-10
Create Chat UI	Daniel	2024-03-10	Daniel	2024-03-10
Chat Management	Daniel	2024-03-10	Kyle, Daniel	2024-03-10
Auto Complete Engine	Daniel	2024-03-10	Daniel	2024-03-10
Auto Processing Engine	Daniel	2024-03-10	Daniel	2024-03-10
Client Keys	Daniel	2024-03-10	Daniel	2024-03-10
Encryption Management	Daniel	2024-03-10	Daniel	2024-03-10
Key Distribution Management	Daniel	2024-03-10	Daniel	2024-03-10
Message Delivery Management	Daniel	2024-03-10	Daniel	2024-03-10
Registered Agent List	Daniel	2024-03-10	Daniel	2024-03-10
Message Storage Management	Daniel	2024-03-10	Daniel	2024-03-10
Stored Messages	Daniel	2024-03-10	Daniel	2024-03-10
Chat UI	Aidan	2024-03-08	Daniel	2024-03-10
View Selection UI	Aidan	2024-03-08	Daniel	2024-03-10
Respond to Contract Request UI	Aidan	2024-03-08	Daniel	2024-03-10
Manage Contact UI	Aidan	2024-03-08	Daniel	2024-03-10
Modify Account Information UI	Aidan	2024-03-08	Daniel	2024-03-10
Login UI	Aidan	2024-03-08	Daniel	2024-03-10
Account Creation UI	Aidan	2024-03-08	Daniel	2024-03-10
IT Account Management	Aidan	2024-03-08	Daniel, Rosa	2024-03-10
Report Generation UI	Aidan	2024-03-08	Daniel	2024-03-10
Account Management	Aidan	2024-03-08	Daniel, Rosa	2024-03-10
Account Information	Aidan	2024-03-08	Daniel, Rosa	2024-03-10
Contact List	Aidan	2024-03-08	Daniel	2024-03-10
Contact Management	Aidan	2024-03-08	Daniel	2024-03-10
Main Controller	Aidan	2024-03-08	Daniel	2024-03-10

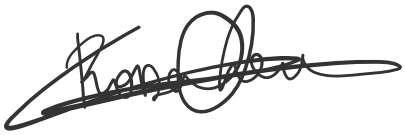
We certify the above information is correct and complete.



Kyle Jordan Ball McMaster



Daniel David Franze-Da Silva



Rosa Chen

Aidan Edward Froggatt

Edward Gao