UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS301 - Software Engineering

Atbash

Software Requirements and Design Specifications

September 19, 2021

Item: Capstone Project - Demo 1

Team Name: Bit by Bit

Team Members:

Name	Surname	Student Number
Liam	Mayston	u19027801
Dylan	Pfab	u19003961 *
Connor	Mayston	u1906936
Joshua	Reddy	u19196042
Targo	Dove	u15020275

** - indicates team leader*

Contents

1 Introduction

1.1 Project Owner

Mr Reinhardt Eiselen
Mr Matthew Gouws
Mr Peter Rayner
Amazon Web Services

1.2 Purpose

The purpose of this document is to present an overview of the proposed Atbash system. This document will describe the target audience, user-interface, functionality and requirements (hardware, software and business). This document is intended for EPI-USE, the stakeholders, and the developers who will implement the system.

1.3 Vision

Our vision is to create a secure, fast and easy to use messaging application that will enable secure communication for all users.
Atbash is a messaging application, where the privacy of messages is the top priority. Message content is only visible to the sender and the recipient and can never be seen, even by the owners. Security is vital for this application to work and the encryption has to be top tier. Atbash uses the signal encryption with additional functionalities built on. Atbash allows users to send messages without the ability of 3rd parties knowing who the sender and receivers are. This makes Atbash unique as it completely hides all messages and if there was even any form of contact between its users. It is truly a secure messaging platform.

1.4 Objectives

- Provide a Mobile app for sending/receiving messages
- Provide peer to peer encryption, where only persons communicating can see messages
- Ensure the system is able to scale automatically as demand grows

2 User Characteristics

The user should be operating a smartphone to download and use the application. The user should have access to the internet and understand how to access and use the messaging services. The Atbash system will be used by the following users:

2.1 Regular User

- Does not require any technical knowledge of the system or other similar systems in general.
- Is familiar with their smartphone and with messaging as a concept
- Is not familiar with the concept of end-to-end encryption
- A person who wants to message others securely and privately.
- A person who does not want any of their personal details or messages to be view able by the application owners.

3 User Stories

3.1 Receive notifications

As the user, I expect to be told when I have received a message whether I am online or offline.
Solution - The system will notify the user when they have received a message when they are online or offline. When they are online it will display in their chat page. When they are offline they will receive a notification and then when they login and open the application it will be displayed in their chat page.

3.2 Customise profiles

As the user, I want to be able to create a unique profile
Solution - The system will allow the user to change their profile picture, status and display name.

3.3 Communicate privately with individual users

As the user, I expect that my conversations will be private
Solution - The system will be end-to-end encryption, which means that no one except

the person you are messaging and yourself will be able to view the contents of the messages.

3.4 View all current chats

As the user, I want to be able to view all of the conversations that I am currently participating in
Solution - The application will allow you to view your conversations on a chats page so that the user doesn't lose previous messages, which might be important.

3.5 Integrate contacts and Atbash

As the user, I want to be able to find my contacts from my phone
Solution - The system will allow you to add your contacts list to Atbash, by integrating the contacts list to the application. This will allow you to find your friends and message them on Atbash.

3.6 Add a new contact

As the user, I want to be able to add a new contact on Atbash
Solution - The system will allow you to add a new contact on Atbash and it will add the contact information to your contact list.

3.7 Edit a contacts details

As the user, I want to be able to edit my contacts details.
Solution - The system will allow the user to edit their contact's display name and phone number.

3.8 Delete a contact

As the user, I want to be able to delete certain contacts from Atbash.
Solution - The system will allow the user to delete their contacts on demand.

3.9 Message handling

As the user, I want to be able to forward, like, edit and reply to messages.
Solution - The system will allow the user to perform these tasks for more intuitive messaging.

3.10 Blocking contacts

As the user, I want to be able to block unwanted contacts.
Solution - The system will allow the user to block as well as unblock any number.

3.11 Sharing options

As the user, I want to be able to set preferences as to what personal information Atbash will share.
Solution - The system will allow the user to prevent sharing of their profile such as their display picture, status, birthday and read receipts.

4 Functional Requirements

The requirements of the system models the functionality that the Atbash system offers and should allow for the following functionality:

R1 : The system should allow the user to register an account and link their phone number

R2 : The system should allow the user to login securely

R3 : The system should allow the user to change their display name, status (online/offline/busy/away/custom), and profile picture

R4 : The system should allow the user to add contacts

R5 : The system should allow the user to send private messages to their contacts

R6 : The system should send push notifications if the application is not open

R7 : The system should should encrypt all outgoing messages

R8 : The system should not store any messages permanently on the external database, only temporarily for the case of a recipient being offline

R9 : The system should use end-to-end encryption to secure all messages

5 System Class Diagrams

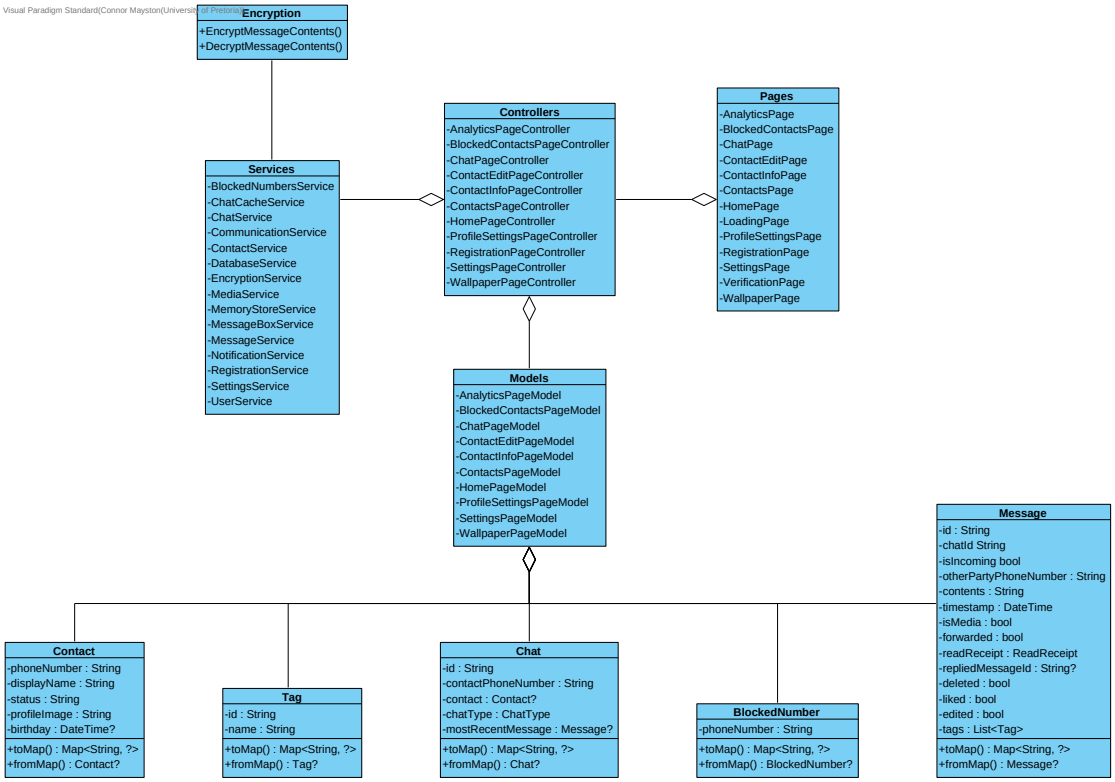

Figure 1: Diagram showing the Mobile Class Diagram for the Atbash system

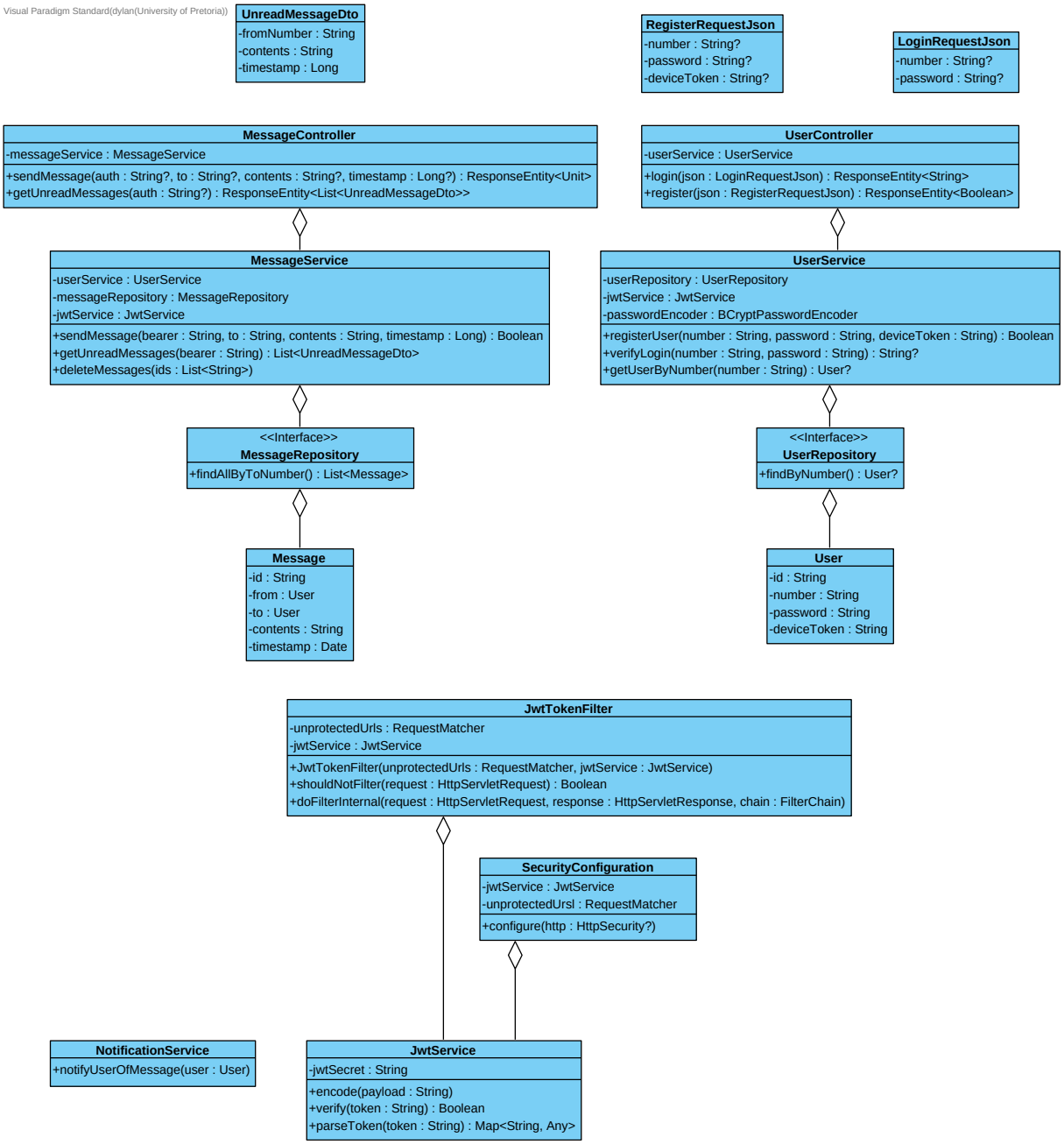

Figure 2: Diagram showing the Server Class Diagram for the Atbash system

5.1 Description

The Atbash system consists of only a few subsystems. The user subsystem is the database of all the users currently registered on the application. Users have many activities they can do such as: sending a message, creating a chat and updating their settings (notifications included). The messaging subsystem allows for the sending and receiving of messages. These messages that are sent are also end-to-end meaning that no one will be seeing your conversations with other people (this also includes Atbash). When messages are sent they are temporarily stored on a database and then sent to the specified user and then deleted. The only time messages are stored is if a user

is currently offline. In this case when the user logs in, the message will be sent and deleted on the database. Messages on the users devices are stored on a local database so that they can keep chat history. Most of this system will also be using Amazon Web Service tools such as DynamoDB.

6 Quality Requirements

The Atbash system has a number of Architectural quality requirements that directly co-relate with the most important aspects of the project.

Top 5 Quality requirements:

6.1 Security

Security is the most important quality attribute for the Atbash system. The system needs to be completely end to end encrypted and no one should be able to read the contents of the message other than the sender and receiver.

- This means the application should make use of the current state of the art standards for secure communication.
- This means that the server should not be able to read or decrypt any information in a message package that is not critical for delivering the message. i.e. The server should not be able to extract any information other than the sender and receiver numbers.
- The server should not store or collect any information other than what is necessary to provide its intended functionality. e.g. Message packets should be deleted once delivered and encryption keys must be deleted once fetched from the server.
- All communications with the server should be encrypted, critical messages should be signed to prevent tampering and message contents between two users should also be encrypted.

Justification: This addresses the core system requirements to enable peer to peer encryption and follow the best security practices. Encryption is guaranteed as the encryption algorithms used would take an exponential amount of time to crack. (For example, AES-256 would take 2.29×10^{32} years to crack.)

6.2 Portability

The system needs to be fully operational on both IOS and Android and have the same look and feel on both systems.
Justification: This ensures that the application can be used on and to communicate with as many mobile devices as possible. The application should be able to run on at least two major OS's. (Apple and Android)

6.3 Availability

The core functionality of the front-end should still be usable even when there is no internet connectivity (for example new messages can be created and sent and will be queued and sent when internet connectivity is restored). The system should be available at least 99% of the time. Server up time is a good way to test this.
Justification: This will ensure intermittent and weak connections will not degrade the user experience and make using the application a frustrating experience. This will also ensure that application will always be usable when required.

6.4 Performance

Instant messaging applications are common place today and users have come to expect responsive applications that provide instant results:

- Navigation to different screens should take no longer than 1 second.
- User input should result in a visible change in the interface within 500ms.
- When opened the application should load within 2 seconds.
- Queries to the server should result in a response within 3 seconds.
- When both devices have a strong internet connection, a sent message should be received on the recipient device within 5 seconds.
- When more intense processing is required (such as for initial setup), the user must be notified of the processing and it should not take more than 10 seconds.

Justification: This will ensure the application meets user expectations and will improve the users experience.

6.5 Scalability

The chosen architecture should allow for both vertical and horizontal scale-out so that the application would be able to easily scale to a user base of 50,000 users within a month and process up to 50,000 requests per second up to a maximum of 250,000 requests per second. This is quantifiable mathematically using the servers maximum threshold.

Justification: This will ensure that the system can easily scale to meet higher demands.

These last 2 quality requirements are just as important but less useful for the functional requirements of the system.

6.6 Usability

Usability is one of the most important quality attributes. Computer literacy cannot be assumed as anyone could use this app however it can be assumed that users have used a messaging application before. Functionality should be as intuitive as possible and provide instructions where necessary. The system needs to be easy to use and understand by anyone who picks it up. Error messages should be self-explanatory and as much as possible of the input validation should be done on the client.

- users should not find any aspects of the system cumbersome, non-intuitive or frustrating,
- the use of colors should not be evasive and distracting,
- user interfaces should not be cluttered and hard to use.

Justification: This ensures the application will be able to service a wide ranges of users with varying abilities and technical know-how and will ensure that using the application is a pleasant experience. The user should have a grasp of the application within 30 minutes.

6.7 Testability

All services offered by the system will be testable through unit tests and integration tests. This will consist of:

- automated unit tests testing components in isolation using mock objects, and

- automated integration tests where components are integrated within the actual environment.

Unit tests will test whether the system functions and classes yield correct, predictable results. Integration tests should test that all needed services from other subsystems are available and work together.
Acceptance testing will be used to determine whether the system meets the use cases and functional requirements described in this document. This will be done by verifying:

- that the service is provided if all pre-conditions are met (i.e. that no exception is raised except if one of the pre-conditions for the service is not met), and
- that all post-conditions hold true once the service has been provided.

Justification: This will ensure most errors produced when updating the code can be quickly identified and located. This will greatly improve the quality of the resultant software. At least 90% of the system should be tested.

7 Subsystems

Below follows a description of the various subsystems that the system is composed of. Together with this, the scope of each subsystem is represented by a use case diagram. For each use case a service contract can be designed which details preconditions, post conditions, in-variants and interactions that each use case has in the system.

7.1 User

The user subsystem involves all the actions a user may take when operating Atbash, including login and registration, updating account settings and starting new chats or adding contacts.

7.1.1 Use Cases

The use cases of the user subsystem are shown in Figure 2, showing all the functionality a user should be able to do in the system.

User Subsystem Use Cases

U1 **Messaging**

U1.1 **Start new chat:** The user should be able to start a new chat.

U1.2 **Open chat:** The user should be able to open an existing chat.

U1.3 **Send message:** The user should be able to send a message to their contacts.

U1.4 **Encrypt messages:** The users messages should be encrypted.

U2 **Administration**

U2.1 **Register Account:** The user should be able to register an account.

U2.2 **Delete Account:** The user should be able to delete their account.

U2.3 **Update Settings:** The user should be able to update their settings.

U2.3.1 **Update display name:** The user should be able to update their display name.

U2.3.2 **Update birthday:** The user should be able to update their birthday.

U2.3.3 **Update privacy:** The user should be able to adjust privacy settings.

U2.3.4 **Update sharing status:** The user should be able to update their sharing statuses.

U2.3.5 **Update profile picture:** The user should be able to pick a profile picture.

U3 **Notification**

U3.1 **Receive push notifications:** The user should be able to receive push notifications.

U3.2 **Receive Message** The user should be able to receive messages from their contacts.

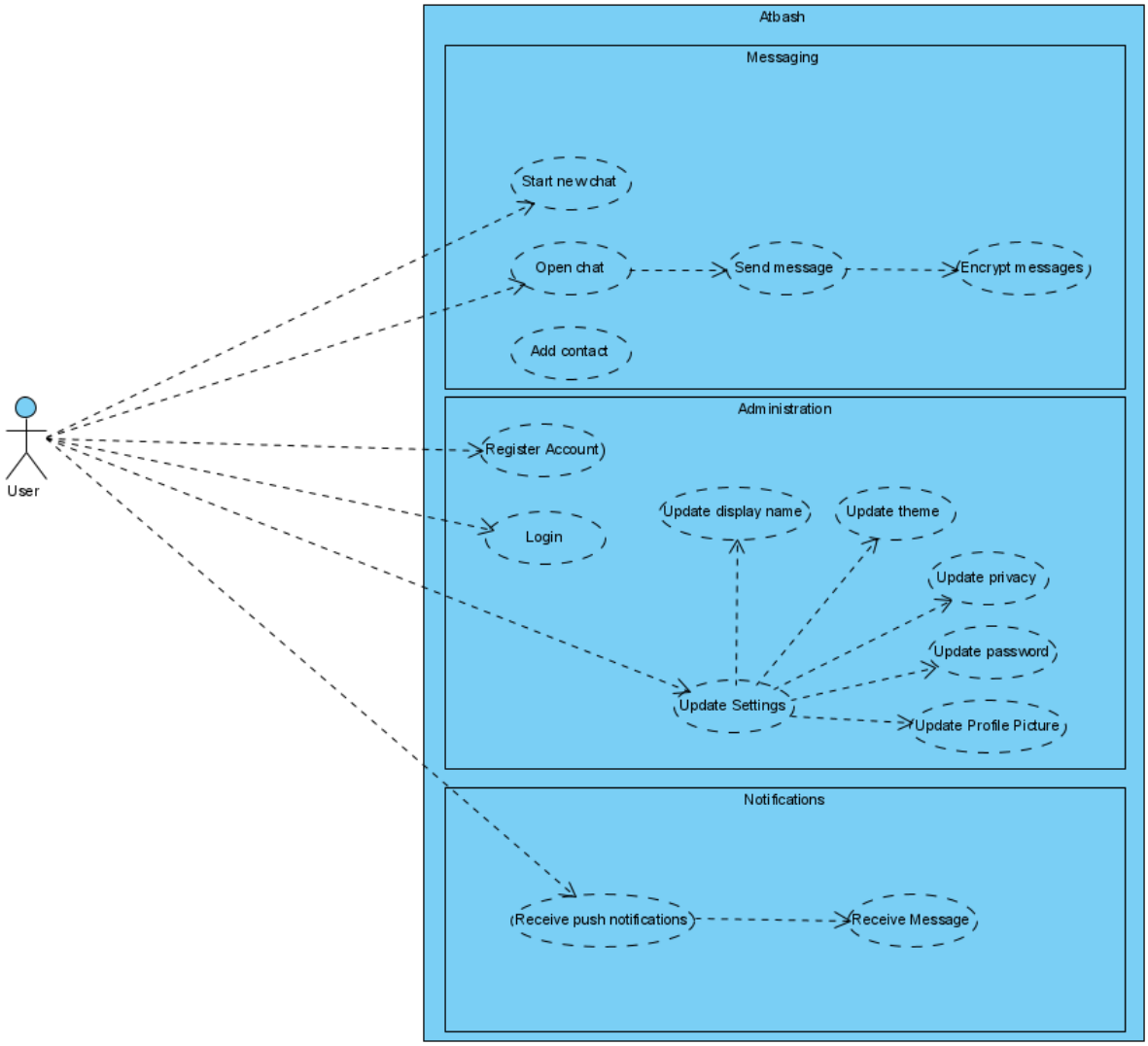

Figure 3: Diagram showing the use cases for the user subsystem

8 Traceability Matrix

UC/R	Priority	R1	R2	R3	R4	R5	R6	R7	R8	R9
U1.1	5				X	X				
U1.2	5					X				
U1.3	5					X	X			
U1.4	5						X	X	X	X
U2.1	5	X								
U2.2	5		X							
U2.3	5			X						
U2.3.1	4			X						
U2.3.2	3			X						
U2.3.3	5			X						
U2.3.4	5			X						
U2.3.5	3			X						
U3.1	5									X
U3.2	5								X	X
Priority		5	5	4	5	5	5	5	5	5

Figure 4: Trace-ability Matrix Diagram