# The Product

This section talks about the methodologies and the design used to build the entire product, i.e., the Language Dependent Messaging App. It starts with the *Design and Project Management* sub-section which discusses the requirements of the app, the product design and the tools used to manage the project and ensure that it is going on time. This section is then followed by the *Development Tools and Methodologies* sub-section which focusses on the programming methods and tools used and the reason behind it. The *Implementation* sub-section illustrates the specific programming tools and resources used to develop the features in the product in chronological order. Finally, the *Result* subsection which demonstrated the final output and how it tackles the problem defined above.

## Design and Project Management:

Before implementing the App, the app was first designed to visualize how the app will function and what will be its basic features. Standard software development process was used while developing the project. It starts with listing the Requirements, then building the Use Case diagram and then the Component Diagram. Activity Diagrams for each functional requirement was also designed at this stage, but it’s shown in Implementation section for simplicity. The modelling diagrams used in this section are all a part of UML (Unified Modelling Diagrams). It is a standard that is used to design the software built.

### Requirements

At first its Functional and Non-Functional requirements were identified and listed. Functional requirements are specific functions and behaviour that a stake-holder has directly asked for and Non-Functional requirements are those that can be used to judge the App.

Following are the Functional Requirements:

**Must Have**:

1. A user must be able to Sign-up to, Login to and Logout from the app

Acceptance Criteria:

* The Sign-up and Login available to the user from the Home screen.
* The Logout option easily available to the user.

1. A user must be able to choose the language they prefer to communicate in.

Acceptance Criteria:

* The user is asked to choose the language only one time during signing up.

1. A user must be able to view all the users who signed up to the app and select one to message.

Acceptance Criteria:

* The users list available to the users after they sign-up or login.
* Each user list item in the list is selectable.

1. A user must be able to view all the messages sent and received from a selected user.

Acceptance Criteria:

* The messages are available in chronological order.
* The messages are available such that it is evident who sent the message.
* The messages are available in the correct language of their choice.

1. A user must be able to send and receive messages in the language they choose.

Acceptance Criteria:

* An area is available where messages can be typed in with a send button.
* A new message received is in the correct language of their choice.
* A new message sent is available in the messages list immediately.

1. When there is a new message and the app is not in the foreground then the user must be sent notification about the message.

Acceptance Criteria:

* Notification permission is taken when the app is first installed.
* The sender and the message are sent in the notification.
* On clicking the notification, it opens the chat screen to the sender.

**Should Have**:

1. A user should be able to set a nick name for themselves that will be available to other users.
2. A user should be able to reset their email, password and language.
3. A user should be able to change their password if they forget it.
4. A user should be able to view the original message sent if the received message is a translated one.

**Won’t have:**

1. The app won’t provide options to send and receive images and files.
2. This app won’t provide option for creating group chats.
3. The app won’t provide option to set the profile picture.
4. The app won’t provide option to make audio or video calls.
5. The app won’t provide options to set user status.
6. The app won’t provide options to block a user.
7. The app won’t provide options to delete chat history.

The Non-Functional Requirements are:

1. Functional Stability
2. Usability:
3. Compatibility
4. Reliability
5. Security
6. Maintainability
7. Portability

### Use Case Diagram

After the functional and non-functional requirements were set, next step was to build the Use Case Diagram. Use Case Diagrams are a part of the Behaviour Diagram under UML. It simply represents the various users and its interaction with the different use cases of the system. Following is the use case diagram built for

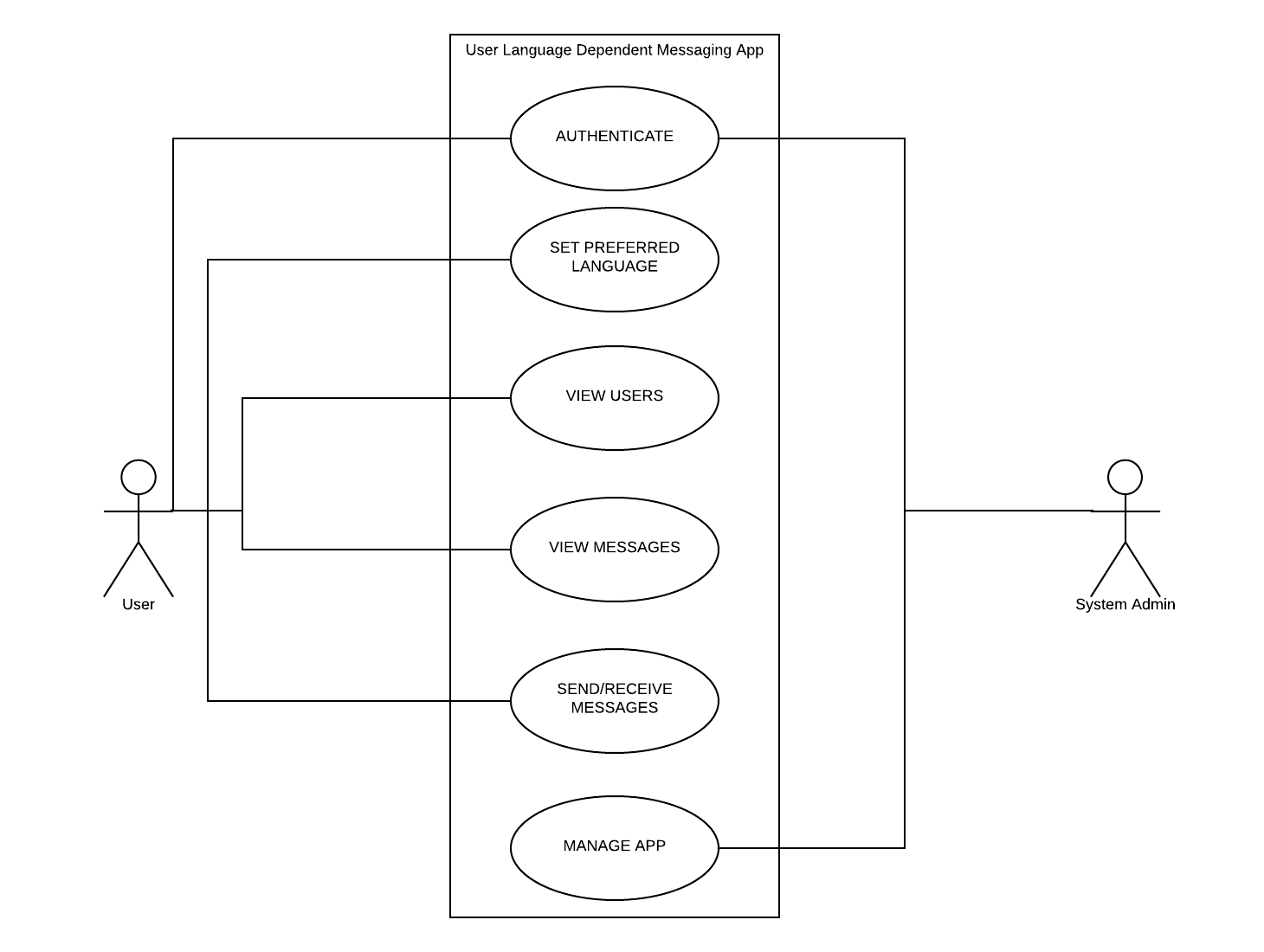


Figure: Use Case Diagram of Language Dependent Messaging App

### Component Diagram

Next is the Component Diagram. This diagram forms a part of the Structure Diagram of UML. It shows how components are joined together to form the software as a whole. The following is the Component Diagram used to design the various components of the app.

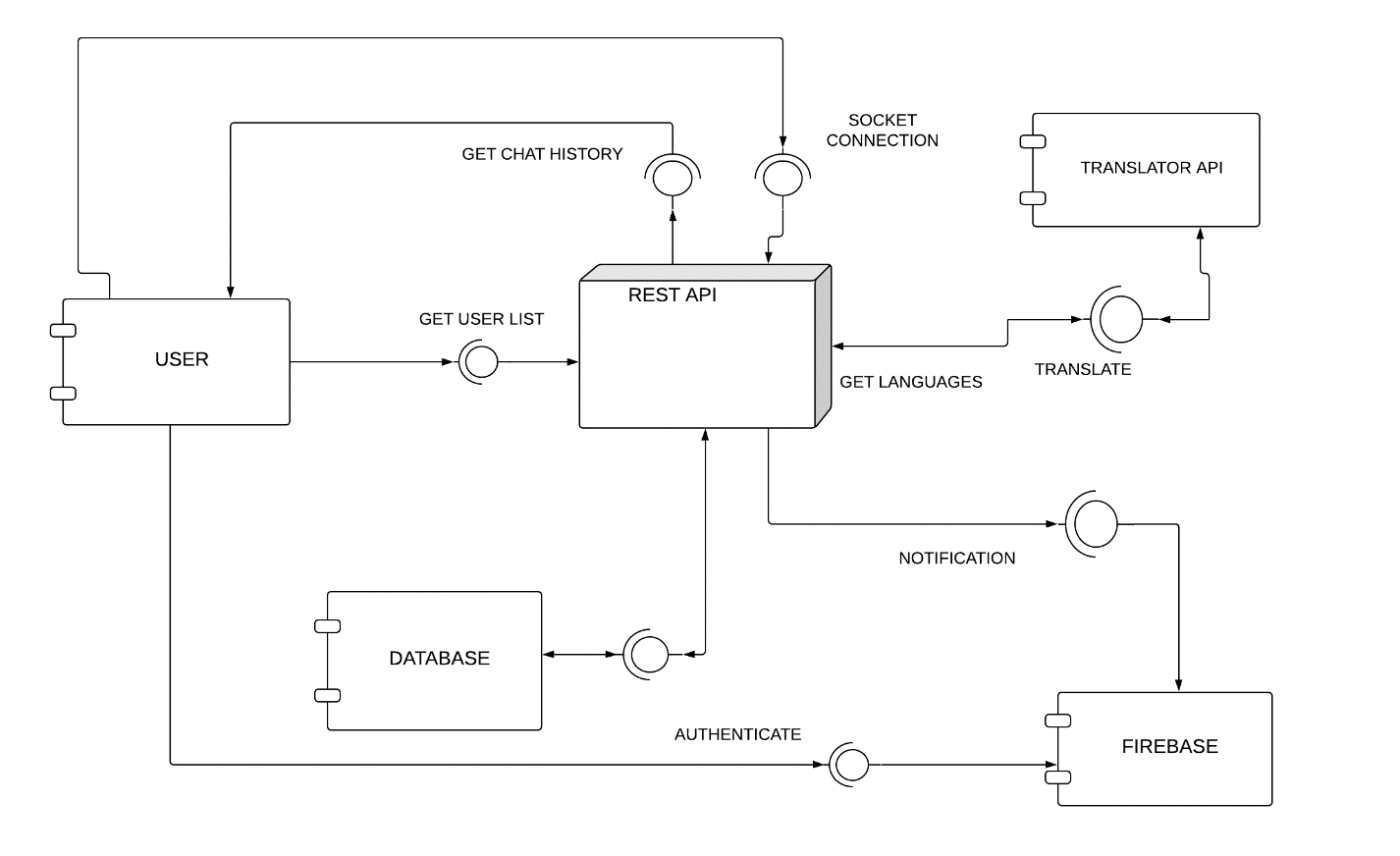


Figure: Component Diagram of Language Dependent Messaging App

### Project Management

Gantt Chart was used as a tool for project management. It was invented by Henry Gantt. It is a part of bar chart where activities are represented in the y-axis and time in the x-axis. Following is the Gantt Chart created for this project.

## Development Tools and Methodologies used

After the design of the App is completed, the tools and methodologies to be used to implement it was decided. This section talks about all the frameworks and libraries used in this project, including coding style. It starts with the software development methodology - Waterfall used and its lifecycle. It then leads on to discussions about the technologies used such as Mobile app development technology, REST API development technology, design patterns used etc.

### Software Development Life Cycle (SDLC)

SDLC is a process used to design, develop and test high quality software. The process is followed in the form of model which define the software development process. The model followed in this project is the Incremental Model. In this model the product is designed, developed and tested incrementally. The product is decomposed into smaller portions and each portion are developed one at a time in the order of priority. In the first increment, working version of the product is released and then in the subsequent releases, features are added to the first increment.

The reason why this model was chosen was because of its advantages. The main advantage of this model is that it allows dividing the project into small iterations and thus have control on designing and developing them. Specific deadlines can be fixed on each iteration and development can be tracked. This model generates a working model in the beginning of the project and testing and debugging each iteration is easier.

As it is evident from the Gantt Chart explained above, the development of this project was done according to the Incremental model. In the Requirements section, the main requirements of the project were first identified. Then prior to implementation the design of the overall project was completed to understand the architecture of the project and then it is divided into smaller increments such as Login, 1-2-1 communication etc. During development, one iteration was completed and then features were added on top of it after satisfactory testing of the previous iteration was completed.

Some other software development models which are very common are the Agile Methodologies and Waterfall Model.

In Agile methodology, the software is built incrementally using short time frames from 1 – 4 weeks so that it can align with the changing Business requirements. The reason why this methodology was not used is because it provides best practices when working in a team of at least 3 – 4 members. Moreover, in Agile Methodology planning is done before each development cycle where the effort in that development cycle is assessed prior to its start and as this project was completed by me as the single developer it was not possible to estimate the effort required accurately.

In Waterfall Model, also referred to as *linear-sequential life cycle model* the software is divided into smaller units called phrases. Each phrase needs to be complete before the starting of the next phrase. The output of one phrase acts as the input of the next phrase sequentially. The phrases in the model are: Requirement Analysis, System Design, Implementation, Testing, Deployment of System and Maintenance. This model is only used when the requirements are fixed and clear, the technology is clear and abundant expert resources are available for completing this project. These uses make this model unsuitable for applying to this project as the model requires fixed requirements and technology which was not the case for this project.

### Design Patterns

Design patterns are general, reusable solution to a commonly occurring problems. These are the best practices followed by experienced object-oriented software developers. Some of the most common design patterns are Singleton, Adapter, Iterator, Command, Factory Method and Modal View Controller (MVC) pattern. In this project the design pattern MVC is implemented. The Modal deals with the data related logic, the View deals with the User Interface logic and the Controller deals with the business logic. Its main advantage is that it allows code reusability and easy code maintenance. For the Modal, NoSQL database MongoDB is used. For the View, React Native is used and for the Controller, a REST API is used.

### Programming Style

In this project, JavaScript is the programming language that is mainly used. Standard coding conventions were used while developing the App. The code is kept simple and easy to read. The naming of the variables and the functions are done using Camel Case naming. Hard coding of values was avoided as much as possible, especially while creating the UI. Also, comments are provided wherever necessary for easy maintainability of the code.

In addition to proper coding convention, defensive programming was followed as well. It is a form of defensive design that makes sure that the software keeps working when the user is diverted from the happy path. Defensive Programming is handled by appropriate error handling and by sending response status of 4xx if the user sent wrong request or 5xx if the server fails handling the request.

### Source Control

Source Control or Version Control is a medium to track and manage the different versions of code and documents. The most common tool used for version control is Git. It is a distributed Version Control system. In this project GitHub is used for version control which is owned by Microsoft and provides version control functionality of Git in addition to its own.

### Programming Tools and Methods used

In this project a large number of programming tools are used to develop the App. Following is a list of the main tools used and the reason why they were chosen:

* Mobile App Development:

For mobile app development React Native was chosen. It is an open source framework built by Facebook. It is used to built Apps in JavaScript which is compatible with both Android and iOS operating system. It helps developers reuse code to run software in both mobile app and web. The reason why this framework was chosen for building the app is because even thought the initial project proposal was to build an Android App, using this framework gives the opportunity to expand the app to iOS operating system as well, thus bringing it closer to wider audience.

* WebSocket Integration

After confirming to use React Native, the next technology that was confirmed was to use WebSocket protocol for real-time communication. This protocol provides a full-duplex communication over a TCP connection, i.e., allowing a bidirectional communication between the client application and the web browser. This allows user of this app to have true real time conversation rather than the traditional way, where the client application sends request to the server for new messages over an interval and the server replying to it depending on the presence of new message.

The library used in this project to implement WebSocket protocol is Socket.IO. It is a JavaScript library. At first, Socketio library of python language was considered to handle the real-time conversation but due to the lack of online support for this library in comparison to the JavaScript Socket.IO library, the later was finally used.

* Server-Side Development

For server sider development Express.js is used. It is a free and open source web application framework for Node.js. It is based in JavaScript. It provides a rich set of features to develop web and mobile application. Initially, Flask micro framework of python language was considered for server-side development, but after changing WebSocket library from Socketio in python to Socket.IO in JavaScript the language to develop the server side was shifted from Python to JavaScript. It provides easier integration and faster development as the entire project is built in only one language-JavaScript.

* Database Development

For storing the data of the users and the messages sent and received a local database is used. The type of database used is MongoDB which is a type of NoSQL database. The main advantages of this database are

* High Scalability: NoSQL uses horizontal scalability. It is adding more machines to handle large volume of data whereas Vertical scalability is adding more resources to one machine to handle the large volume of data.
* High Availability: NoSQL database has auto replication feature making it highly available.

This project requires a database that can store large volumes of data and for long time. Thus, NoSQL database is the best option.

The other type of database available is SQL (Structured Query Language) database. It deals with Relational Database. SQL follows vertical scaling hence when it is used for large volumes of data, the system gets slower.

* Firebase

Firebase is Google’s own mobile app development platform that allows business to rapidly grow along with providing features such as Cloud Firestore, ML Kit etc that help building the app. In this project Firebase is used for authentication and notification.

Firebase authentication provides a number of authenticating options such as email, phone number, social media login etc. As this project is a messaging app, these options provide a number of opportunities for future implementations like adding the user phone number. Moreover, for easier user authentication, Firebase can also be used to provide social media login such as Google login and Facebook Login.

For notification service in the app, which allows the user to be informed of a new message, Firebase Cloud Messaging (FCM) service is used. Sending a notification to an Android app can only be done via FCM. Moreover, it is also a cross platform messaging platform that allows sending notification to iOS. In addition to FCM being a cross-platform messaging service, this service allows testing of notification to client applications by sending notifications from the Firebase console. Thus, making testing of background notification service in the app very simple and easy.

* Translator API Integration

For translating a message from one language to another, a translator API is utilized. The main APIs available for Translation are:

* Google Translate API:

The Google Translate API is a translator API provided by Google. It allows dynamic translation between thousands of language pairs. It is easy to integrate and provides <500,000 free character translation per month. Originally Google API used Statistical Machine Translation but in 2016 it transitioned into Neural Machine Translation. After the usage of the free characters, translating 500,001-1billion character cost $20 per million character and using Language Detection cost $20 per million character. It has 104 supported language.

* Microsoft Translator API:

Microsoft Translator API is provided by Microsoft Azure which is Microsoft’s cloud computing platform. This API provides 2million characters per month for free and it includes language detection as well. It uses Neural Machine Translation. It provides a many payment options such as pay as you go where it costs £7.454 per million characters. It supports a total of 63 languages.

* Amazon Translate API:

Amazon Translate API is a service provided by Amazon Web Service which is cloud computing platform provided by Amazon. It provides language identification along with translation. It provides 2million characters per month free in its free tier and costs $15 per million character. It also uses Neural Machine Translation. It supports a total of 25 languages.

Out of the 3 translate APIs listed above, Microsoft Translator API is used in this project. It provides a good range of languages for translation and 2M characters per month for free along with language detection. It also allows easy integration to JavaScript server along with extensive official documentation availability. In addition to these features, signing up to this service do not require registering any financial cards.

## Implementation

The Language Dependent Messaging App is a complicated app developed using a number of tools and libraries. This section explains the implementation of the app and how it was developed. For simplicity the section will follow the navigation flow of the app shown in the figure below and explain the development of each part with first describing the functional part and then the UI part. The fulfilment of each functional requirement will be highlighted at the same time.

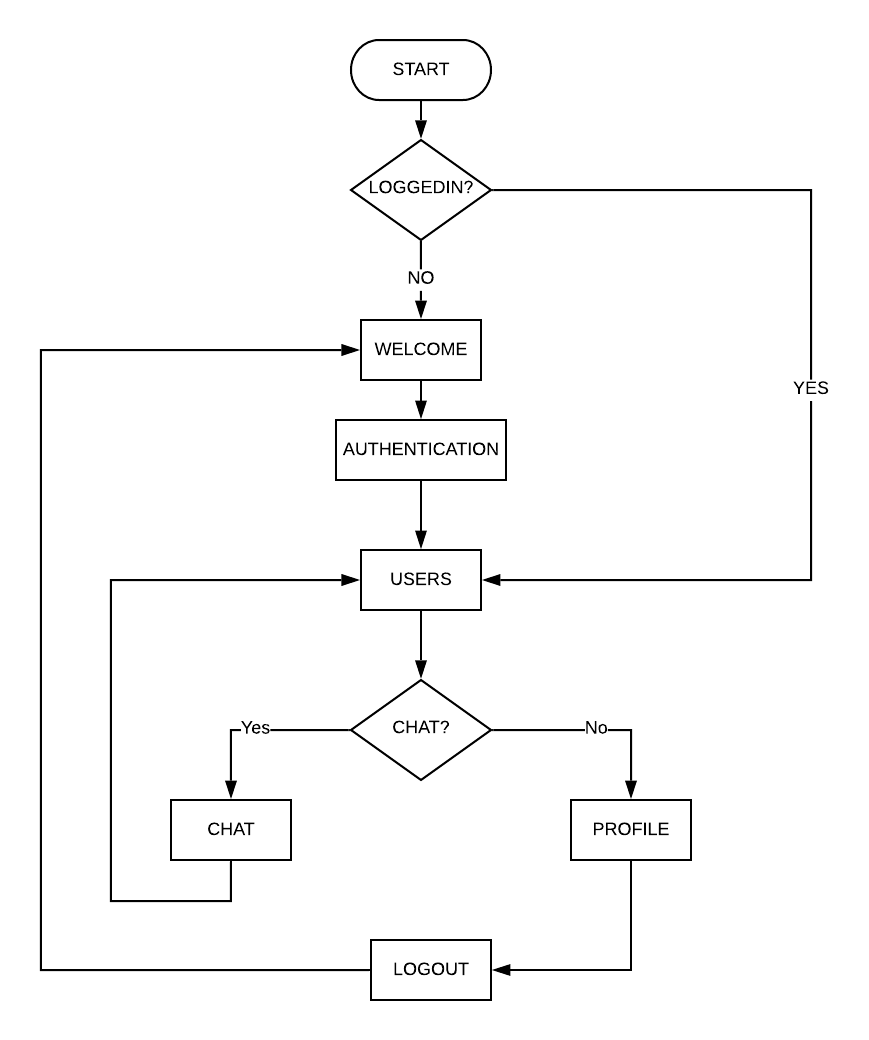


Figure: Navigation in the App

## Result

# References

Tutorials Point, n.d. *SDLC - Waterfall Model.* [Online]   
Available at: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm  
[Accessed 30 08 2019].