**Software Requirements Specification**

for

**BANKING BOT**

**Version 1.0**

**January 6, 2022**

**Prepared by:**

**POOJA. G – s222314216**

**IRAM FATHIMA – s222576806**

**JAY NILAWEERA – s219302799**

**AJAY BENIWAL – s222554334**

TRELLO BOARD LINK FOR BANKING BOT -- <https://trello.com/b/CrJgy0X9/bb>

**Table of contents**

BANKING BOT

1. **Introduction** 
   1. Purpose 3
   2. Document Conventions 3
   3. Intended Audience and Reading Suggestions 3
   4. Product Scope 3
   5. Reference 3
2. **Overall Description**
   1. Product Perspective 4
   2. Product functions 4
   3. User Classes and Characteristics 5
   4. Operating environment 6
   5. Design and Implementation Constraints 7

**3 External Interface Requirements**

**3.1** User Interfaces 7

3.2 Hardware Interfaces. 9

3.3 Software Interfaces 10

3.4 Communications Interfaces 11

**4 System Features**

**4.**1 Login Process 12

4.2 Design Module 14

4.3 Architecture 14

4.4 Data base design 15

4.5 Speech recognition 16

4.6 Designing Dialogs specific 17

**5 Other and non-functional specification**

5.1 Performance Requirement 20

5.2 Security Requirement 20

5.3 Safety Requirement 20

5.4 Usability 20

5.5 Reliability 21

5.6 Availability 21

5.7 Scalability 21

5.8 Maintainability 21

5.9 Portability 23

5.10 Localization 23

5.11 Regulatory Requirement 23

**6 Appendix 1: Sprint 1 Review**

6.1 Git Pulse and Git contributor’s screenshot 23

6.2 Screenshot of Trello board 24

6.3 Document with a breakdown of all tasks completed by each member.

6.4 A series of screenshot of current state of application 26

1 Introduction

**1.1 Purpose**

Project’s main goal is to create an online banking application for customers. All banking work is done manually in the current system. To withdraw or deposit money, the user must go to the bank. Today, it is also hard to find account information for people who have accounts in the banking system. Depending on the bank’s policies, the bank personnel and/or customers can utilize the Banking Bot application to check their transactions or bank statements etc.

* 1. **Document conventions**

The document was created using the RRL (Review of Related Literature), Methodology, Evaluation, and Recommendations

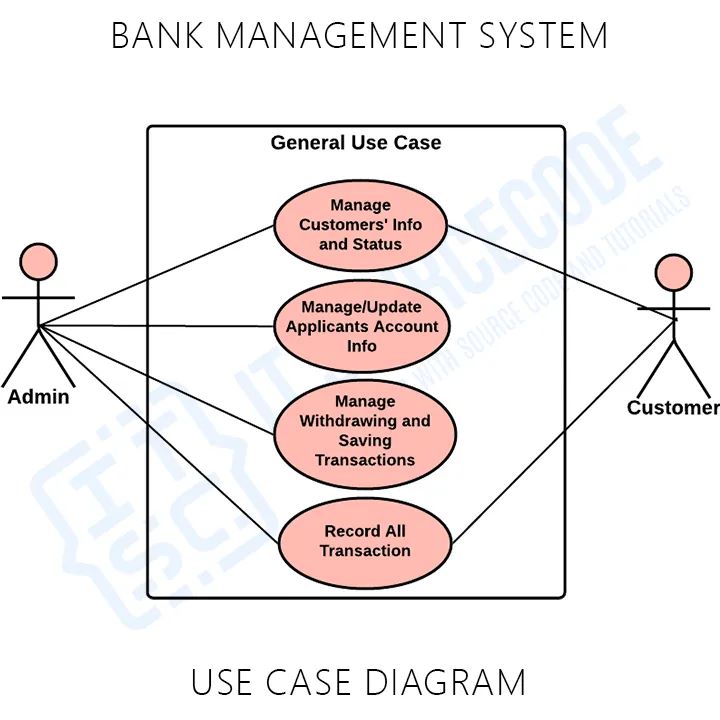
* 1. **Intended Audience and Reading Suggestions**
* Typical Users: Bank Customers, Bank employees and Authorised Persons
* Developers: IT Source code
* Testers: Who would perform UI tests.
* Programmers: who want to improve the application and make the process easier for the intended users.
  1. **Product scope.**

The Banking Bot project is a program that keeps track of a client’s bank account. This project demonstrates the operation of a banking account system and covers the essential functions of banking software.

* 1. **Reference**.
* Google chrome: itsourcecode.com

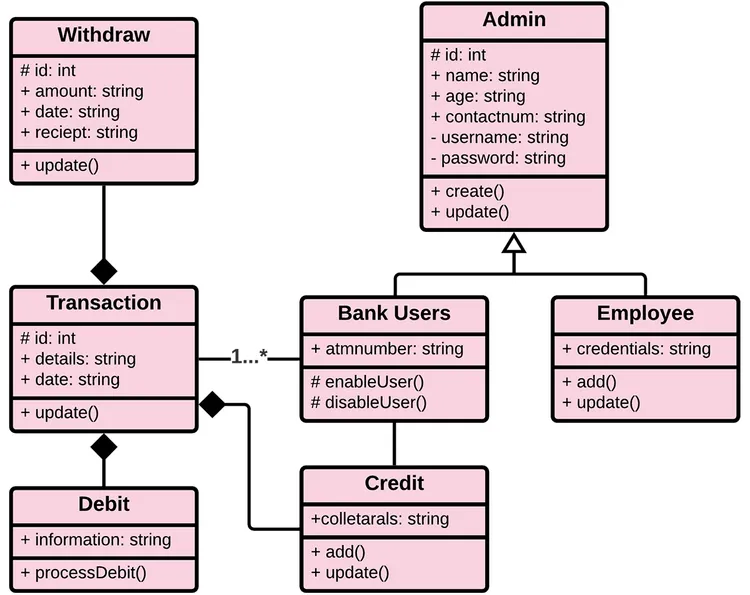
1. **Overall Description.**

2.1 Product Perspective



* 1. **Product Functions.**
* Sign up page.
* First Name
* Last Name
* Date of Birth
* Mobile Number
* Pin code
* State
* Acc no.
* PIN
* Redirect to OTP
* Login Page
* Acc No.
* OTP
* Create Account – The user needs to create new account first to get a unique account number to login.
* Login – After creating an account, the user needs to login first to enable to access the system.
* Manage Account – The user can view his/her account status such account number, current balance in your account, and the account type.
* Money Transfer – The user can transfer money from one bank account to another bank account.
* Loan application – The user can request a loan to the bank.
  1. **User Classes and Characteristics.**

The Data Flow Diagram (DFD) represents the flow of data and the transformations in the [Banking](https://itsourcecode.com/free-projects/python-projects/loan-management-system-project-in-django-with-source-code/) Bot. It discusses the overall definition of [input](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi_ysPk-ZDwAhUGZt4KHcuDC_MQFjAJegQIHhAD&url=https%3A%2F%2Fwww.merriam-webster.com%2Fdictionary%2Finput&usg=AOvVaw0g06VShmc70be5dN9jQlyU), [processing](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiqhNv1-ZDwAhUNO3AKHZtmDXEQFjAIegQIChAD&url=https%3A%2F%2Fbrainly.in%2Fquestion%2F12916842&usg=AOvVaw0wRLbTkI0mBTVvzoTGsATz), and [output](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjg_KaJ-pDwAhWMA4gKHfB2CEAQFjACegQIAxAD&url=https%3A%2F%2Fwww.computerhope.com%2Fjargon%2Fo%2Foutput.htm&usg=AOvVaw2USkqsJIMZ7QFHHM1LyIfi).  
  
The banking bot application DFD has three levels explaining the content of the data flow diagram.



**2.4 Operating Environment.**

* JavaScript
* A webserver to host the application.
* A fast internet connection with good bandwidth that enables quick response from the application.
* An efficient web browser that allows the user to access the webpage.
* Chrome
* Firefox
* Safari
* Microsoft edge

**2.5 Design and Implementation Constraints.**

This application is a simple web based.

* It is very easy to understand and use.
* Talking about the application it involves the fundamental features required in a bank.
* The customer can use all the available capabilities without any confusion or any restriction.
* It is too simple to use, the account holder can easily understand how to navigate into the application without any difficulty.

**3 External Interface Requirements**

**3.1 User Interfaces**

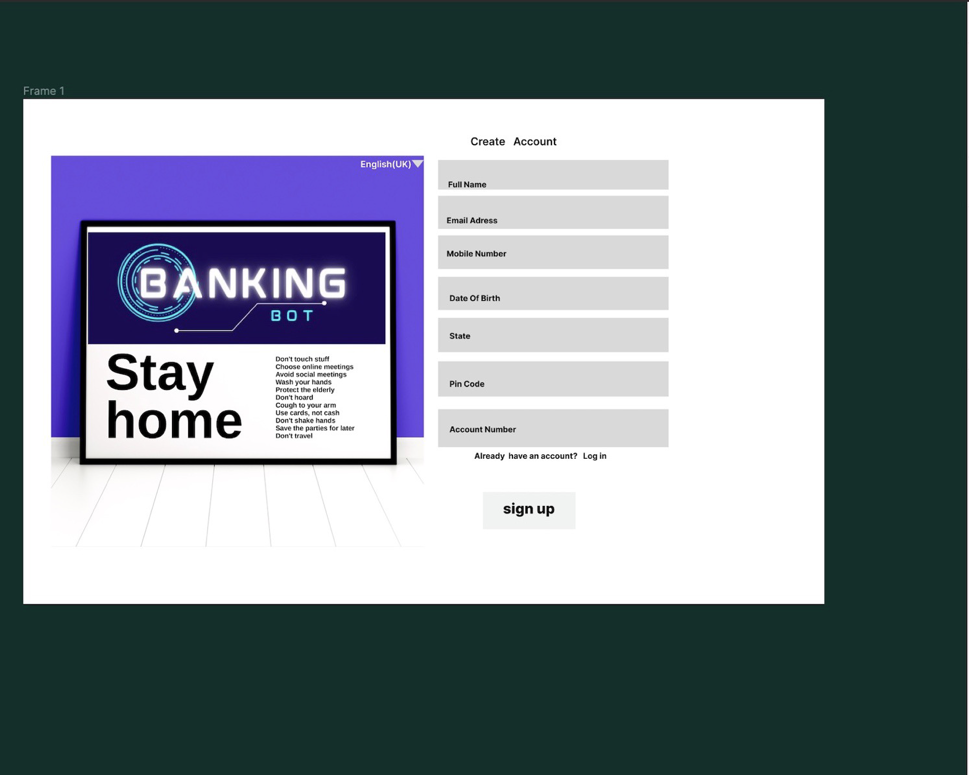
The User interface of banking BOT is very easy to understand and Flexible to use by the customer. The first page of the App consists of home page/Sign up page. The Sign-up page is to register or create an account in the App. The Sign-up page consists of the entries for the

* Full name
* Email Address
* Mobile Number
* Date of Birth
* State
* Pin Code
* Account number

Once the App takes the information of the Customer it stores in the Data Base. and sends a One Time Code to the register mobile number. After signing in the app takes the customer to the next page which consists of

* Username
* Password
* OTP.

The customer has to choose the password and username and give the correct OTP to proceed.



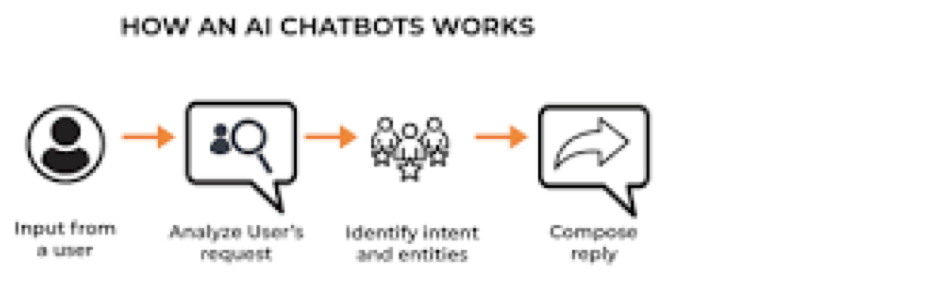
SIGN UP PAGE



LOGIN PAGE

**3.2 Hardware Interfaces**

* PC running Linux-Ubuntu to act as server to host Banking Bot locally
* Processor – intel core i5
* Android Device



* **Frontend and backend**

The frontend was developed using the Recast. Ai framework which includes a collaborative platform for building, hosting and deploying bots. Recast. Ai is an end-to-end collaborative bot platform for developers by developers. All bot tools are integrated here to enable auto-discovery of entities. We use NLP, machine learning. API connectivity is via PHP.

* **NodeJS**

Node.js is an open source, cross platform runtime environment for executing JavaScript code. Node is used extensively for server-side programming, making it possible for developers to use JavaScript for client side and server-side code without needing to learn an additional language. Node is mostly referred as development framework, but it is strictly a JavaScript runtime.

* **Machine learning**

Machine learning is a central part of artificial intelligence. This allows the computer to enter self-learning mode without being explicitly programmed. These computer programs can learn, grow, change, and evolve on their own when exposed to new data.

* **NLP**

Natural Language Processing (NLP) is the branch of computer science and artificial intelligence concerned with the interaction between computers and human (natural) language, especially the programming of computers to effectively process large amounts of natural language data. is.

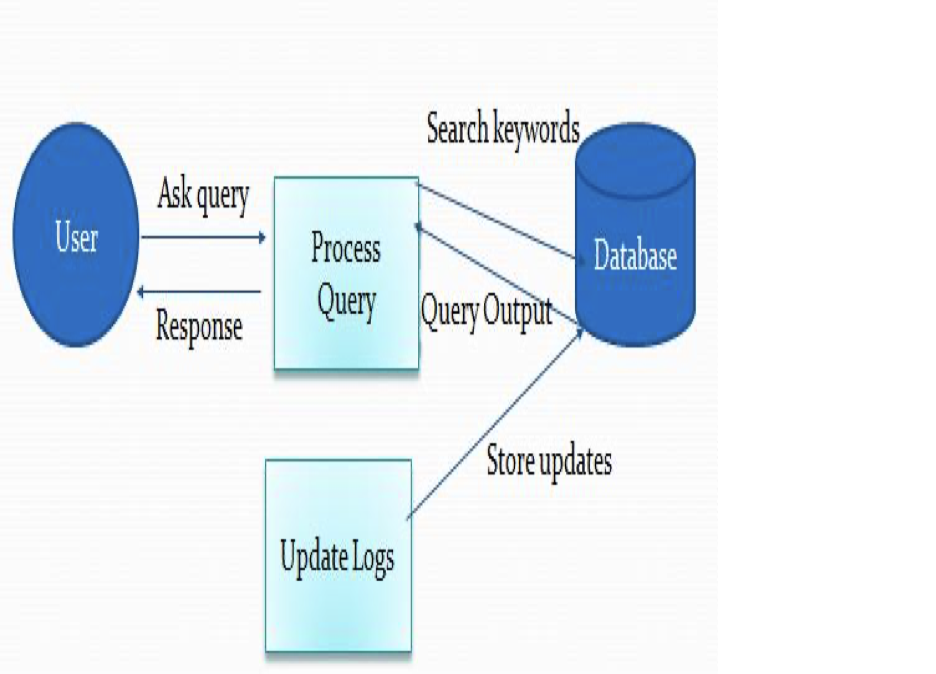
* **Mongo DB**

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables.

* **JSON**

JSON (JavaScript Object Notation) is a minimal human-readable format for structuring data. It is mainly used to transfer data between the server and her web application.

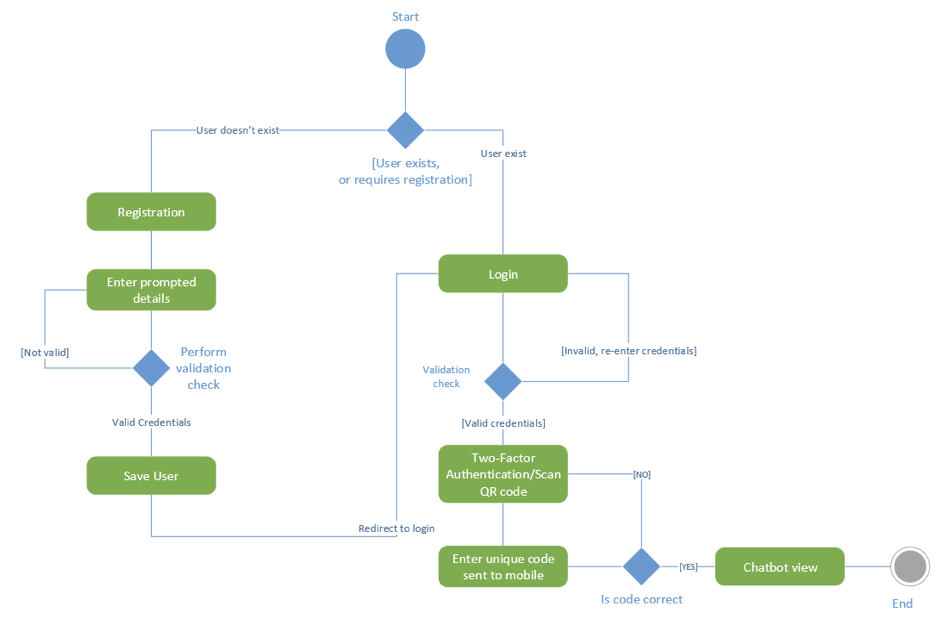
Alternative to XML. The two main parts of JSON are keys and values. Together they form a key-value pair.

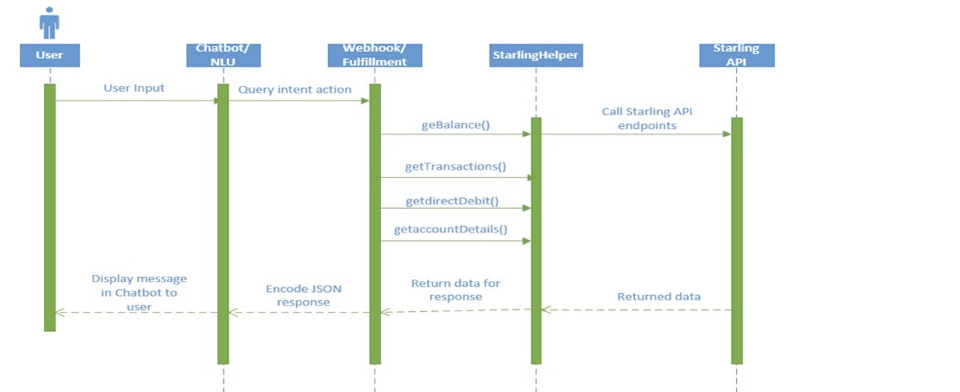


**3.4 Communications Interfaces**

**Communications**

The sequence diagram shown highlights the process of a user requesting bank information via a Banking bot and outlines the basic processes involved. Webhook receives user input in her JSON format from Banking bot NLU engine (Dialog flow). The webhook determines the actions of predefined intents posted by NLU by parsing her incoming JSON. If the action matches a defined bank operation, the webhook makes the appropriate method call to the TrueLayerClientStarlingHelper class and calls the True Layer Starling API to return data to the webhook and generate a for answer. When the webhook receives the data returned by the API, it encodes the data into JSON. The JSON is sent back to the NLU as a response and displayed to the user via the Banking bot view.





**4 System Features**

**4.1 Login Process**

Registered users can log in. Once the credentials are sent to the database, the user can.

Present your own QR code implemented with Google's two-factor authentication.

A code is generated and sent to the user's mobile her device.

• Banking Her bot allows users to view information about their accounts such as: B. Savings, Loans,

Current account.

• Banking bots allow users to

User email.

• Banking bot is integrated with True Layer Starling API that collects data about banks.

Bank accounts.

• Banking bots support user requests and perform appropriate actions, such as scheduling appointments.

event. with a financial advisor.

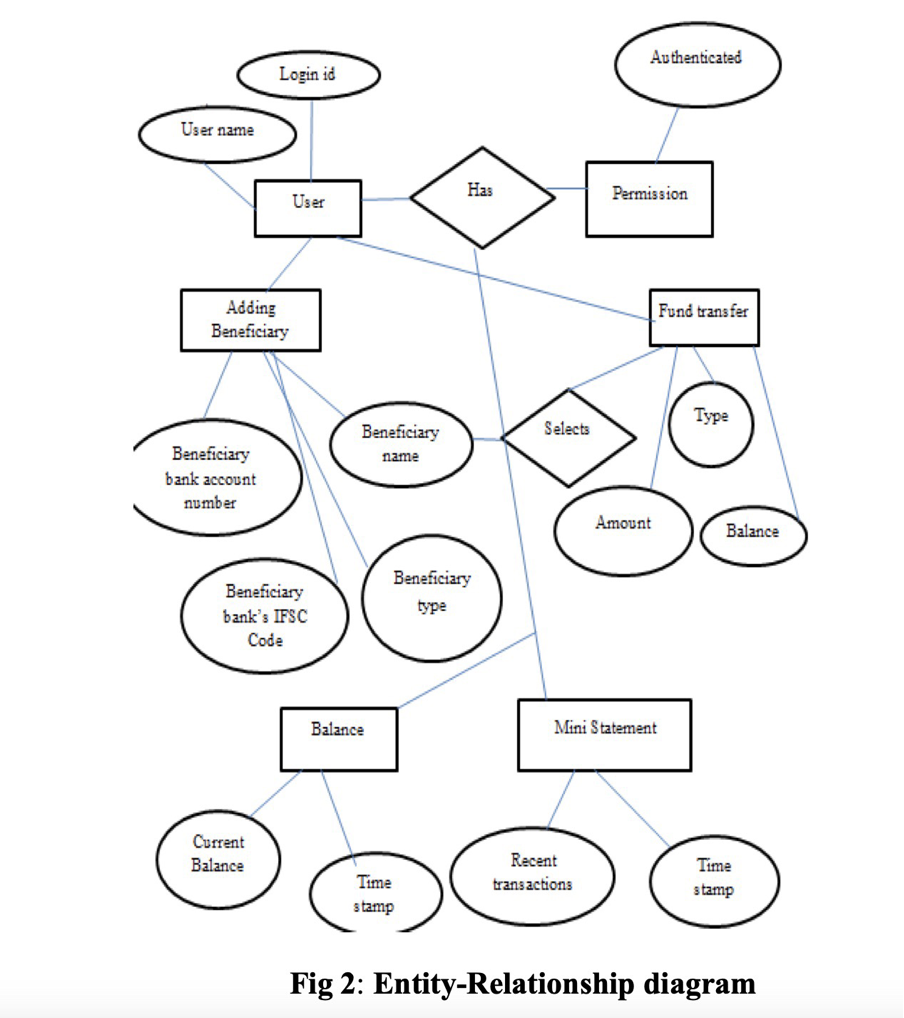
• User can talk to her banking bot using voice or text commands.

Understand what users are saying with natural language understanding provided.

By integrating the Dialogflow API.

• Banking bots should be able to maintain conversational state even when context is unclear.

Through previous messages and conversations.



**4.2 The design module:**

Module 1: Adding beneficiary Adding beneficiaries for a user.

Link with fund transfer mechanism

Module 2: Fund transfer Initiate fund transfer

Fund transfer to beneficiary with different modes

Module 3: Balance Enquiry Authentication

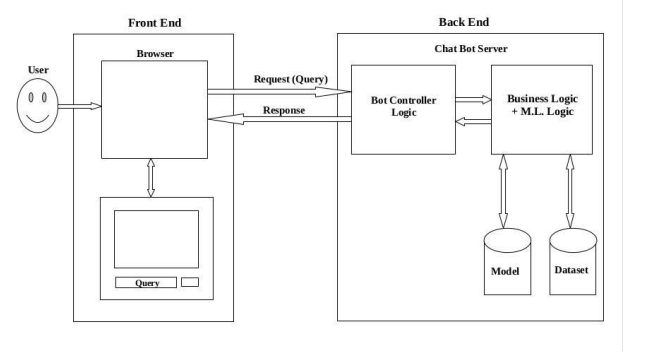
Display the current balance.

Module 4: Mini statement

Display the mini statement.

Track the recent transactions and display them.

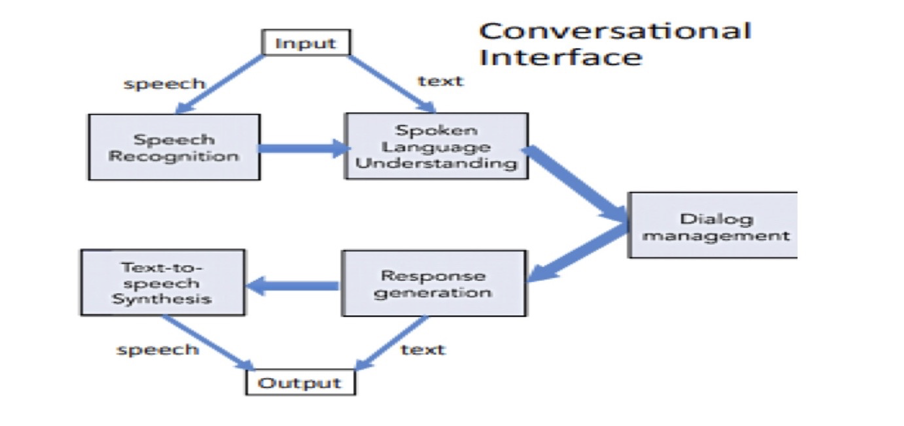
**4.3 Architecture**



In Banking bot MVC architectural development, as you can see in the above diagram, we have two main architectural segments those are Bot controller logic and Business logic. Business logic interface with Model segment and the Database segment. In terms of Rest API, Ai Services running in the back-end logic facilitates the HTTP requests and HTTP responses. Mongo dB related CRUD query requests are sent to the database, generated from HTTP methods as we use Rest API. In this development we will be using Get, Post, Delete, Put and Patch HTTP methods.  Frontend and Backend both will be written in native JavaScript. We will be using Ajax for the data access We will be using materialized for the cascading in the UI. This will be host in locally however there is provision for cloud hosting.

Historically, bots have only supported one adjacent pair, also known as one-shot conversations. However, modern bots maintain multiple neighbour pairs, remember state and context between conversations, and have the ability to correlate relevant data from different neighbour pairs. This is the bot's ability to keep the conversation going. The banking Bot consists of her four main parts:

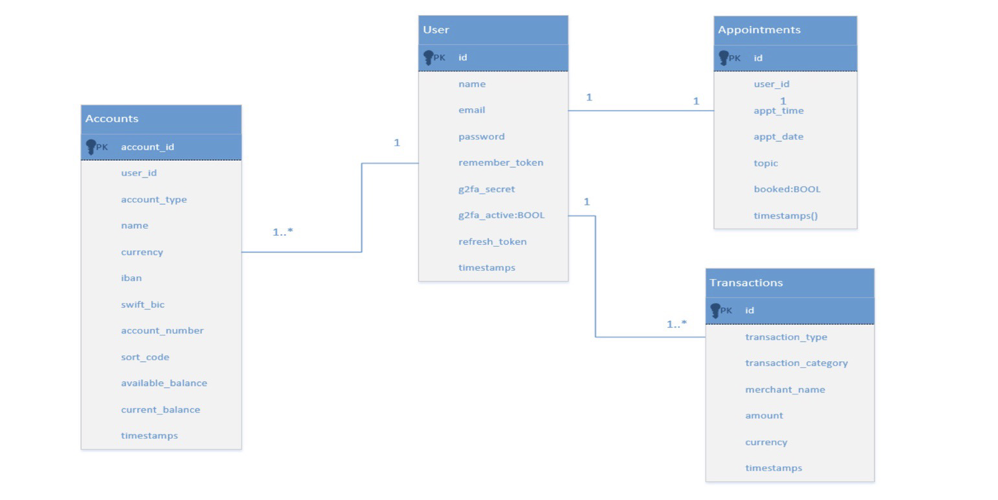
Frontend, knowledge base, backend and corpus are training data. The frontend is responsible for enabling communication between bots and users. NLU uses artificial intelligence techniques to identify the intent and context of user input. The corresponding response is generated from the user's intent. A knowledge base defines knowledge for Banking bots created within NLU and supported by a backend. The backend applies the domain corpus to create the knowledge base.

Input can be fed to the bot in the form of text or voice. The input is sent to the dialog management system (his NLU in this case). The NLU determines the appropriate response and changes the bot's state accordingly to take the required action. Banking bots generate text and voice responses in both text and voice formats.

**4.4 Database Design**

Data is stored locally in the MongoDb database and is small to save disk space. Each table in the database is represented using a "model" that allows you to define and manipulate the logic and relationships between database objects.

The proposed database is multiplicity defined. The users table contains data about the users of our application and whether the user has been verified by her google 2 factor authentication. The Appointments table contains data about a user's appointment.



This includes information provided by the user about the purpose of the appointment. The busy attribute is set to a Boolean value to determine if the user already has an appointment. Accounts and Transactions

**4.5 Speech recognition**

Users can interact with Banking bots using voice commands and talk to bots using the banking bot device's built-in microphone.

The HTML5 Speech Recognition API is implemented in web apps for speech recognition and synthesis.

Voice Recognition

Dialogflow then takes the decrypted language and converts it into a structured JSON object can be parsed. This will be used as a text input. JSON response is given entity match Score, also called confidence score. This score represents how good the NLU engine is Mapped user input to intents defined in the console . good score Between 0 and 1, where 1 is an exact match.

**4.6 Designing Dialogs**

Banking bots should provide users with consistent choices to reduce the effort required to interact with them. This can be achieved using buttons and images.

follow up question.

Follow-up questions should be specific to the context of the conversation. An example of this is when a user wants to request a currency conversion. the bot asks.

What amount and currency do you want to convert? "

Cognitive load

Conversation design is an important aspect to consider when designing a bot, as the focus of the interaction is between the user and the banking bot and is conducted via natural language. To achieve that; the corresponding dialog is designed via the Dialogflow console using follow-up and fallback intents. When invoked by a user, bots have a specific intent that effectively ends the conversation, regardless of the context of the conversation. A dialog is a collection of words and phrases used to respond to user input. the dialog is.

user:

"Please book an appointment with me"

Styled with human-like phrases to respond to users and sound more natural. Another design aspect will include fall-back intents. If a user utterance cannot be matched to an intent, the bot will respond with a message stating it did not match the input to an intent and prompt the user to repeat the query or provide more details. The dialog is also designed using follow-up intents, these provide a more natural conversational flow when interacting with the Banking bot. Follow-up intents extract more information from users as and when required, for instance; a user may say.

User:

“Please Book me an appointment”

The Banking bot is aware it needs more information from the user such as date, time and phone number and will therefore use follow-up intents to extract the required information from the user. Dialogflow recommend designing a linear dialog for interactions where data present in the conversation is extracted to help the users achieve their goal, this is applied to the

design specification is applied to the bot as it is a tasked-based interaction bot that consist of banking domain knowledge. User utterances can be pronounced in a variety of ways.

In this example, the banking bot extracts the required entity parameters “Credit” and “Account” and returns a custom her response to the user.

"Currency Conversion Dialog"

The simplified layout makes it more intuitive and efficient for users compared to other forms of traditional user interfaces (DZone, 2018). Users can ask questions directly to the Banking bot using input fields or by tapping the device's microphone. As soon as the icon is tapped, the device's microphone is activated to alert the user that recording has started, and the Banking bot's response is immediately displayed on the screen. Google Assistant uses a specific colour for each message. or

user:

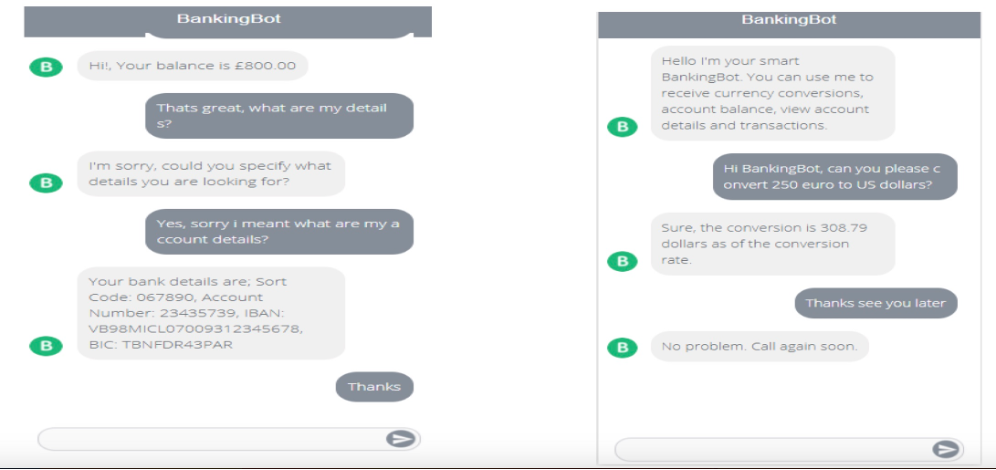
"Show me your balance?"

user:

"What is my balance?"

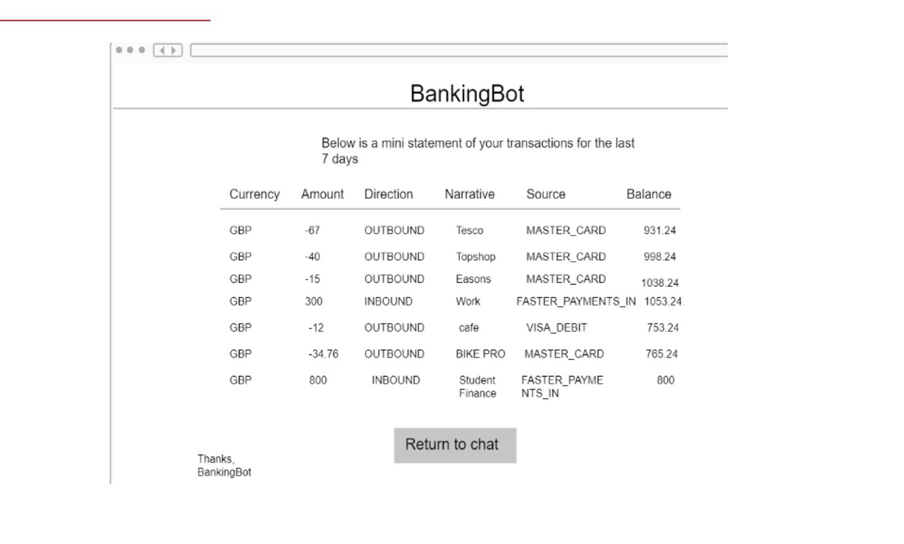
user:

"Could you tell me your account balance?"



In this example, the Banking bot extracts the required entity parameters "Credit" and "Account" and returns a custom response to the user.

Users can view transactions on a regular basis without having to navigate to another website. As soon as the user requests to view the transaction, an email will be sent to her email address provided during registration.



**5. Other and Non-Functional Requirement**

**5.1 Performance Requirement**

* The users need to have an active internet connection to access the application over the internet.
* The user’s device needs to be installed with a web browser and other necessary tools such as the location services feature to run the application efficiently.
* System can withstand many users at a particular given time.
* Stable performance is required from the application even when user traffic is peaking.
* Fast response from the web server should be provided to the user for every action performed by the user.
* The system should run on multithread and be free from any deadlocks to increase the performance of the system.

**5.2 Security Requirement**

* The application needs to be secured with HTTPS protocol, which protects the user and server data integrity.
* SSL certificates will be used to secure and transport users’ credentials and browsing data.
* The application will not grant access until the user creates a strong password. For example, a strong password contains a minimum number of characters, capital letters, and symbols.
* The webserver needs to be scanned regularly for possible vulnerabilities and viruses, that could affect the working of the application.
* System will require users to create an account to access applications that store their information and profile.
* The application will be using the database to store information, which needs to be secured against SQL injection etc. that can threaten the integrity of the database.

**5.3 Safety Requirement**

* The application management team should ensure adequate backup of data in case of loss.
* Both data and software should be backed up periodically.
* The application should pass through testing periodically to ensure its quality and effectiveness of the application.
* If a user finds any potential vulnerabilities or bugs, he must be able to report the same to the developer team.

**5.4 Usability**

* The web application interface will be easy to use and user-friendly.
* Users can easily navigate the application interface.
* With high usability, users can easily distinguish each feature of the application.
* A quick guide tutorial will be added to the application, where users can learn how to operate the basic features of the application.

**5.5 Reliability**

* The system must be very reliable due to the importance of data and the destruction an incorrect or incomplete data could do.
* The system must perform without failure 96% of the time.
* Reliability will always be determined and improved after assessing the probability of failure and the time between critical failures.

**5.6 Availability**

* The system is 99.9% available to the user and can be accessed at any time i.e., 24X7.
* The system must be up 24 hours every day to accommodate users.
* In the case of scheduled maintenance, the users will be pre-notified about the downtime for the application.
* In normal operating conditions, a request generated by a user for a service should be handled within a few seconds.

**5.7 Scalability**

* To handle the growth of users and requests, scalability should be increased by using any two methods i.e., horizontal scalability and vertical scalability.
* To curb the new data, more storage space and servers can be accessed using cloud computing technology based on demand profiling.
* New database schema and functions can be used to improve the process of gathering and storing information to the web application based on user traffic.
* New features can be added to the web application based on the review from the customers.

**5.8 Maintainability**

* There should be design documents to define the internal working of the software.
* The code for the application should be well maintained with comments in the source code to explain the feature of the function for easy upgradation of code during Maintenance.
* MTTRS (Mean time to restore system) following a system failure should not be more than 30 minutes.
* Different versions of the product should be maintained in case of upgradation of the application software.
* Source control or version control software must be used such as Git, SVN, TFS, etc. for managing, maintaining, and correcting defects or making changes in the code in an adequate and efficient manner.
* Maintenance should be cost-effective and easy.

**5.9 Portability**

* There are no portability requirements, the application runs on JavaScript, and works with the same efficiency on all platforms.
* The application will be highly compatible and works on all different operating systems i.e. windows, Linux, and macOS. The web application is a cross-platform application.

**5.10 Localization**

* The web application is primarily based in Australia and will use the Australian dollar as the only currency.
* The date format for the application will be as follows: dd/mm/yyyy
* The application will be running on AEST (Australian Eastern Standard Time) for all the processes and functions in the application.
* The web application will be using only Standard English Language to communicate with the users.

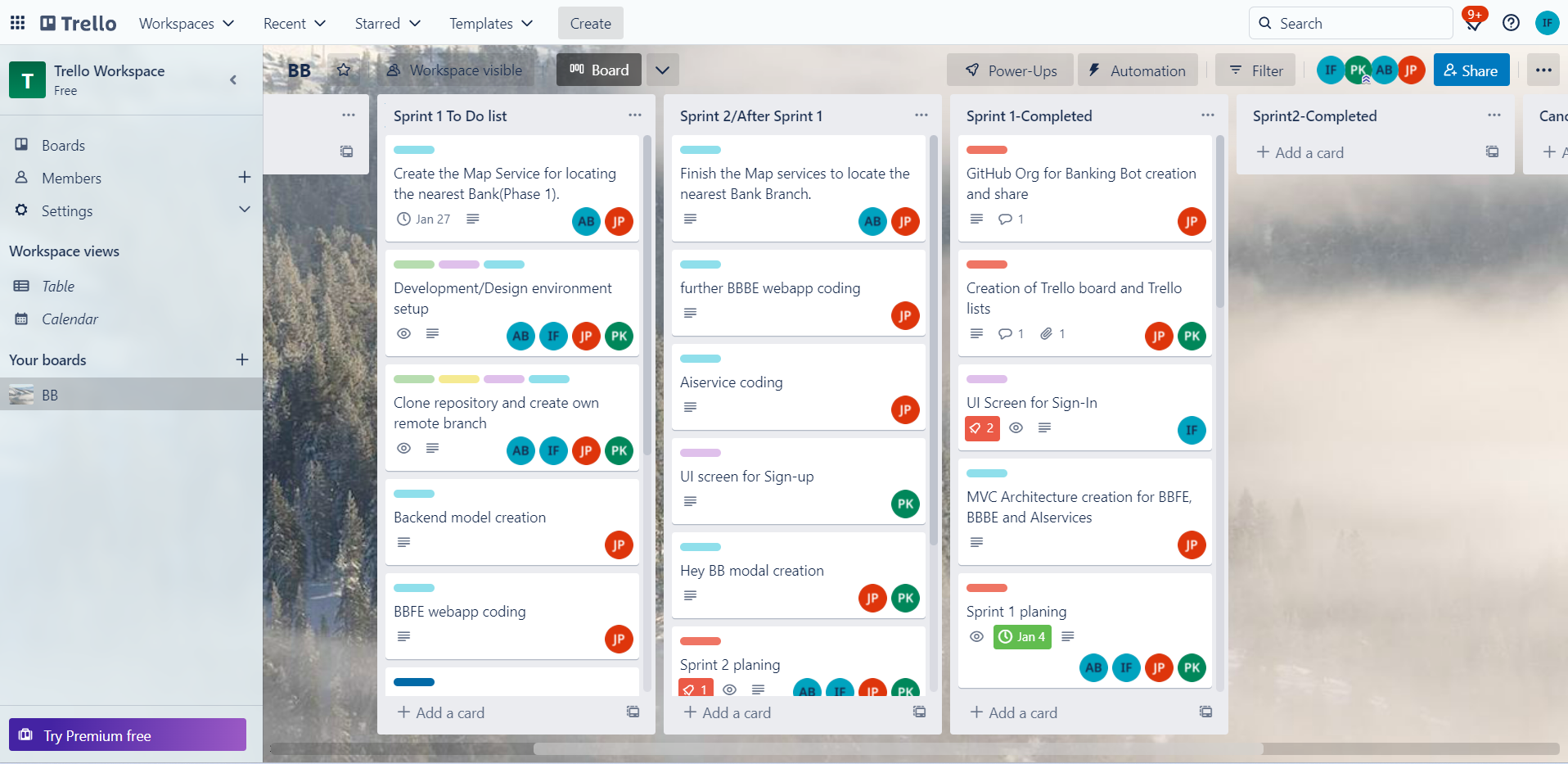
**5.11 Regulatory Requirement**

* The application complies with the privacy policies drafted by the government.
* The data gathered from users will not be shared with third parties and will not be used for any kind of activity except for banking bot.

**6. Appendix1: Sprint 1 Review:**

**6.1 Git pulse and git contributor’s screenshot:**

**6.2 Screenshot of Trello Board**

****

**Link:** [BB | Trello](https://trello.com/b/CrJgy0X9/bb)

**6.3 Document with a breakdown of all tasks completed by each member.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pooja** | **Jay** | **Ajay** | **Iram** |
|  | **Map service for location** | **Map service for location** |  |
| **Deployment** | **Deployment** | **Deployment** | **Deployment** |
| **Clone repository** | **Clone repository** | **Clone repository** | **Clone repository** |
| **Create Git Commands** |  |  | **Create Git Commands** |
| **Backend Model Creation** | **Backend Model Creation** | **Backend Model Creation** | **Backend Model Creation** |
| **Prototyping and wireframe** | **BBFE web app coding** | **Prototyping and wireframe** | **Prototyping and wireframe** |
| **Sign up model Creation** |  |  | **Sign up model Creation** |
| **Create Docker Virtualization** | **Create Docker Virtualization** | **Create Docker Virtualization** | **Create Docker Virtualization** |
| **Verification Code model** | **Verification Code model** | **Verification Code model** | **Verification Code model** |
| **SRS Document update** | **SRS Document update** | **SRS Document update** | **SRS Document update** |

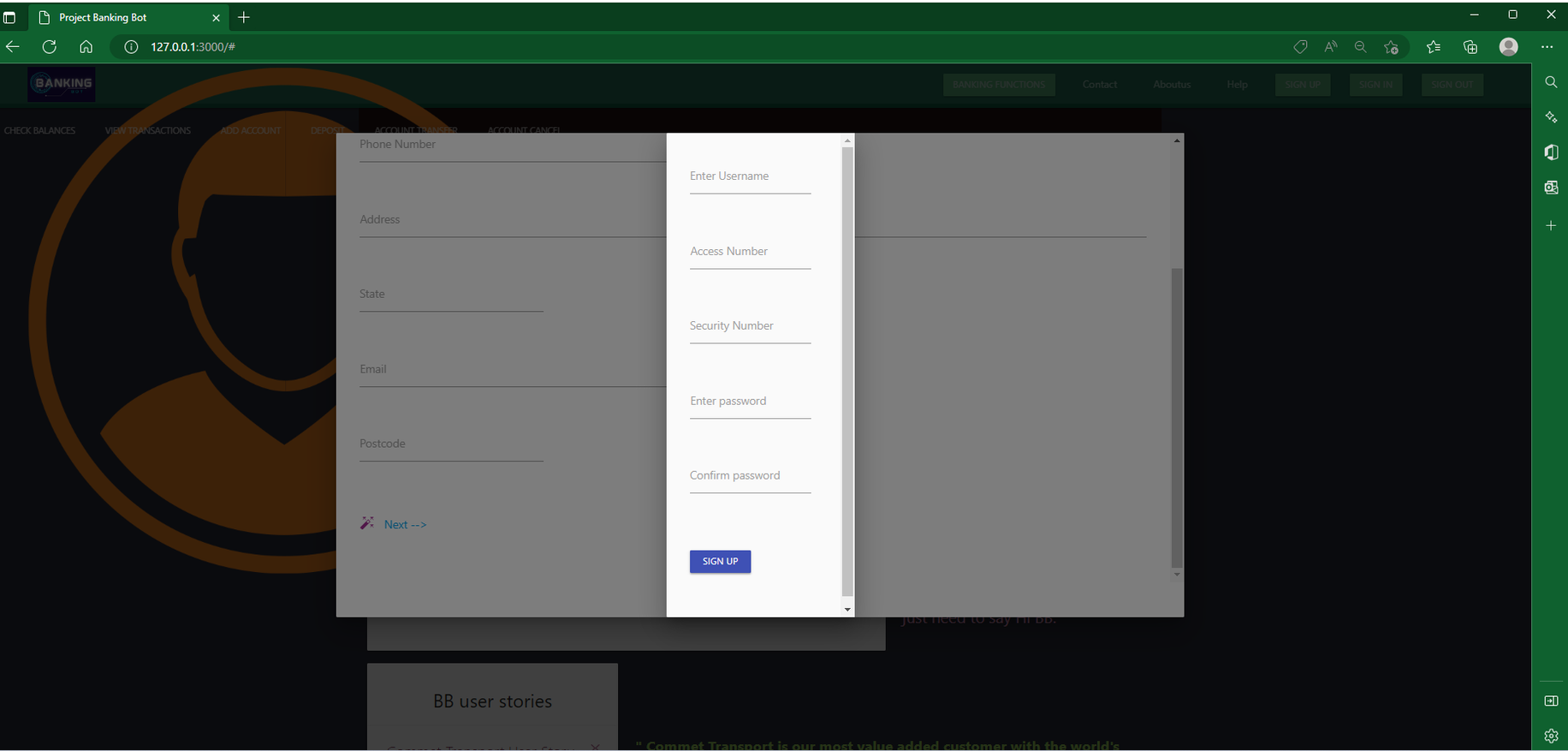
**Pooja:** She is the person who has generated this idea of creating an application called Banking Bot to ease the banking facilities to the customers . Pooja is giving her best by participating in all the project related meetings, work assignments like deployment, UI design, preparation of SRS document, Creating docker Virtualization, in creation of login and sign pages also in prototyping and wireframe, creating git repositories and git commands, Back end model creation and even more in future she is spending almost of 2.5 hours each day on this project. We have equally divided the project tasks and development related works.

**Jay:**  jay is the backbone of our project and he is the one who has decided and divided all the tasks into parts and allocated it to each person from the team so that nobody feels a burden . jay has also worked on the development, deployment, coding, UI testing, SRS document, creating git repo and git commands, creation of trello board. He is also working on the wireframe and prototyping, backend model creation of the project and creating API’s and locating maps as per the project requirements. Jay is also spending 2.5 hours per day on the project.

**Ajay:** Ajay is also the crucial member of our project Banking Bot as he is also working hard on the development, deployment, docker virtualization, sign up model creation , map service for location, Sprint 1 planning and execution and even completion, backend model creation adding cards on Trello board. Ajay is also working on the API’s he is contributing 2.5 hours per day on this project and making sure that he is completing the work allocated to him on time and reporting and attending all project related meeting without fail.

**Iram:** Iram is also a hardworking and important member of the group as she is also working on development, creation of application logo, theory part of the project, SRS document update, sign up model creation, docker virtualization, git repo and git codes, prototyping and wireframe, backend model creation, verification code model. She is also working for 2.5 hours each day on the project and attending the project related meetings and submitting the allocated work on time.

**6.4 A series of screenshots of the current state of application.**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Graphical user interface, website

Description automatically generated**

**Graphical user interface, website

Description automatically generated**