Project Name: Peer-to-Peer Lending Chatbot

Created by: JuniHers Group 3

Group Members:

Anjana Gupta

Anshika Gupta

Lata Sah

Rizul Gupta

Shambhavi Gupta

Under the mentorship of Sandeep Manthi

Introduction: P2P Lending bot

Me: Hi, I need 500 rupees. Could you lend me some? I'll return it to you soon.

Friend: Sorry, I'm also tight on funds towards the end of the month. I can't lend you any money right now.

Me: Oh, where else can I turn for help?

Managing finances becomes a pressing challenge for students upon entering university. With newfound independence comes the responsibility of managing limited funds, often leading to unforeseen expenses and budget constraints. Many students find themselves struggling to stretch their finances until the end of the month, inadvertently overspending and exhausting their resources prematurely.

Seeking financial assistance from friends is a common recourse, but its success hinges on their financial situation. Meanwhile, turning to external sources for loans raises concerns about transparency and security, dissuading potential lenders from extending help.

However, for those in need of short-term financial support, the confidence in repaying promptly when the new month begins remains steadfast. To address this pervasive issue, we've devised a solution in the form of a peer-to-peer lending bot on Telegram.

Our implementation empowers users within small groups to facilitate lending and borrowing transactions seamlessly. Through transparent agreements and predefined minimal interest rates, the process ensures fair and equitable exchanges while fostering a sense of community trust and collaboration. This feature particularly encourages borrowers outside the friends group to lend money, as they can benefit from a guaranteed return on their investment. By leveraging technology, we aim to alleviate the small short-term financial strain faced by students or any other group like a family members group, or office colleagues group by making the loan process simple yet effective while enabling users to be anonymous throughout a transaction and keeping admin of the group as the central identity.

Understanding P2P Lending

P2P lending, or peer-to-peer lending, is a method of debt financing that allows individuals to borrow and lend money without using an official financial institution as an intermediary. This model directly connects borrowers with potential lenders through online platforms, offering a more streamlined and personalized borrowing experience.

Benefits of our P2P Lending:

1. **Enhanced Accessibility:** The chatbot improves access to credit by enabling users in small groups to lend and borrow money, catering to those who may not qualify for traditional bank loans.

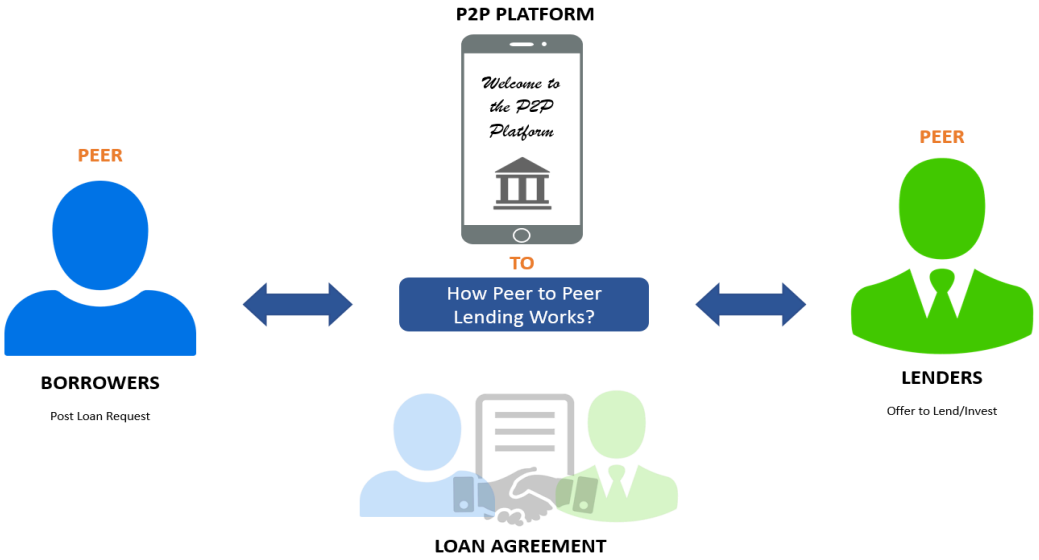
2. **Anonymous Transactions:** Our platform offers anonymous transactions, shielding borrowers from lender scrutiny, avoiding embarrassment over financial difficulties, and repeated requests for funds.

3. **Competitive Interest Rates:** Users can agree upon interest rates that are competitive and mutually beneficial, ensuring fair terms for both lenders and borrowers.

4. **Streamlined Transactions:** The chatbot streamlines the lending process, making transactions easier and faster among group members, thereby saving time and effort.

5. **Transparency and Security:** By operating within a secure platform like Telegram, the chatbot ensures transparency in transactions and provides a secure environment for financial exchanges, fostering trust among users.

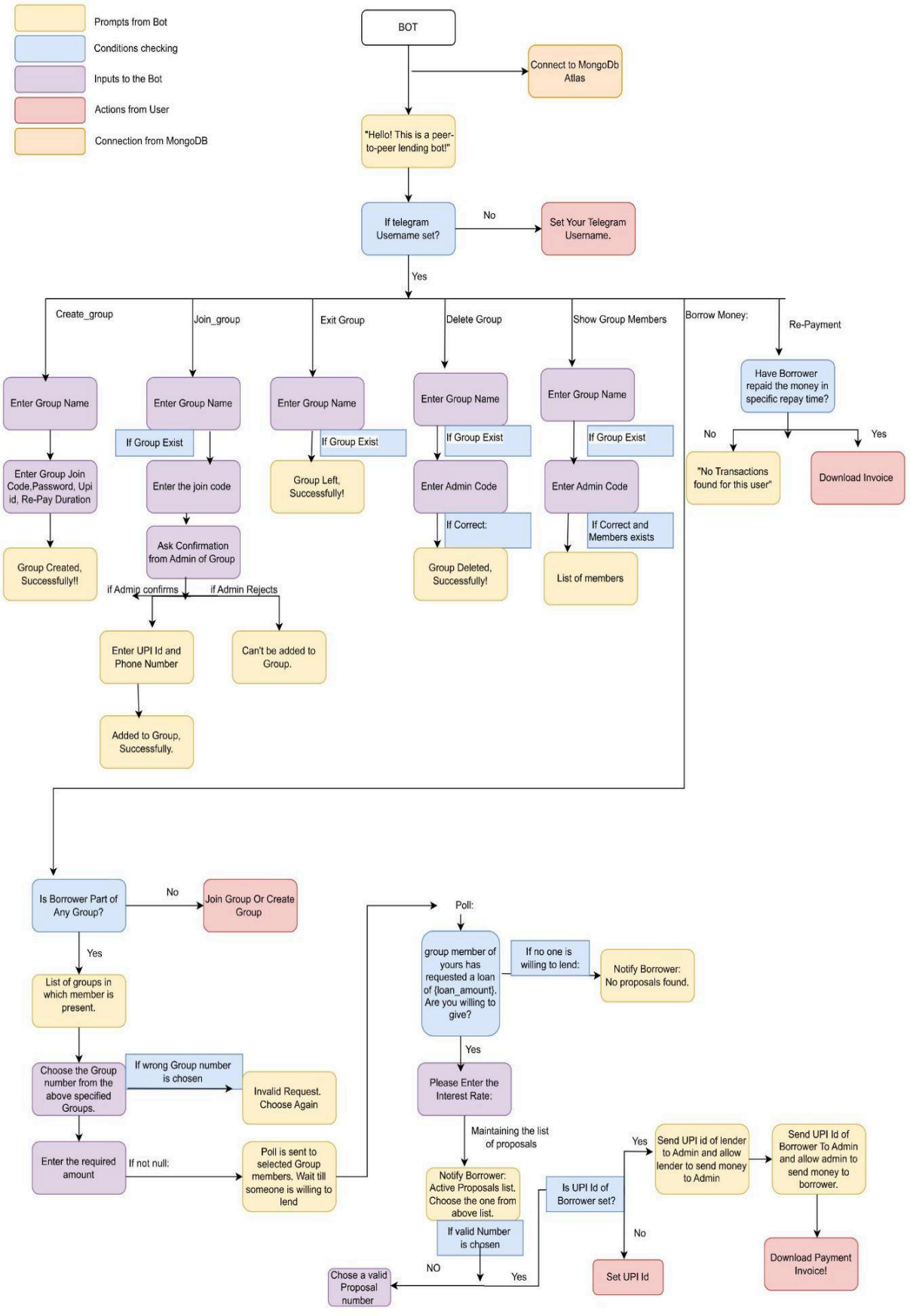
6. **Bot Conversation:** The bot has been trained in natural language, enabling it to seamlessly converse with borrowers and lenders to assist them in the best possible way.



Components of Project:

1. **Telebot Integration (telebot):**
    - Integration with the Telegram Bot API using the Telebot library to send and receive messages from users.
2. **Intent Recognition (neural_intents):**
    - Utilizes the neural_intents library for intent recognition, which allows the bot to understand user messages and determine the appropriate action.
3. **Database Interaction (database):**
    - Interaction with a database (MongoDB) to store and retrieve data related to users, groups, loans, and transactions. This includes functions for adding members, creating groups, deleting groups, etc.
4. **Command Handlers:**
    - Handlers for various commands like /start, /hello, /create_group, /join_group, /delete_group, /leave_group, /show_group_members, /borrow_loan, and /bye. Each command triggers a specific action or prompts the user for further input.
5. **Message Handlers:**
    - Handlers for messages that are not associated with specific commands. These include functions like echo_all, which echoes back received messages, and default_handler, which handles messages the bot doesn't understand.
6. **Loan Borrowing Process:**
    - Functions related to the loan borrowing process, including initiating loan requests, processing loan amounts, creating polls for loan approval, handling proposals from lenders, and facilitating the transfer of loan amounts.
7. **Group Management:**
    - Functions for managing lending groups, such as creating groups, joining groups, leaving groups, showing group members, and deleting groups.
8. **User Interaction:**
    - Functions for interacting with users, such as sending greetings, handling acknowledgments, and collecting user input for various processes.
9. **Poll Creation and Handling:**
    - Functions for creating and handling polls within Telegram groups to collect votes from group members.
10. **Data Extraction and Processing:**
    - Functions for extracting and processing data from user messages, such as extracting numeric values from loan requests and interest rates.
11. **Error Handling:**
    - Error handling mechanisms to manage invalid inputs, missing data, and other exceptional conditions gracefully.
12. **Mapping of Intents to Functions:**
    - A mapping dictionary (mappings) that associates recognized intents with corresponding functions, allowing the bot to execute the appropriate action based on user input.

Flowchart:


Prompts from Bot
Conditions checking
Inputs to the Bot
Actions from User
Connection from MongoDB
BOT
Connect to MongoDb Atlas
"Hello! This is a peer-to-peer lending bot!"
If telegram Username set?
No
Set Your Telegram Username.
Yes
Create_group
Join_group
Exit Group
Delete Group
Show Group Members
Borrow Money:
Re-Payment
Enter Group Name
Enter Group Join Code,Password, Upi id, Re-Pay Duration
Group Created, Successfully!!
Enter Group Name
If Group Exist
Enter the join code
Ask Confirmation from Admin of Group
if Admin confirms
if Admin Rejects
Enter UPI Id and Phone Number
Added to Group, Successfully.
Can't be added to Group.
Enter Group Name
If Group Exist
Group Left, Successfully!
Enter Group Name
If Group Exist
Enter Admin Code
If Correct:
Group Deleted, Successfully!
Enter Group Name
If Group Exist
Enter Admin Code
If Correct and Members exists
List of members
Have Borrower repaid the money in specific repay time?
No
Yes
"No Transactions found for this user"
Download Invoice
Is Borrower Part of Any Group?
No
Join Group Or Create Group
Yes
List of groups in which member is present.
Choose the Group number from the above specified Groups.
If wrong Group number is chosen
Invalid Request. Choose Again
Enter the required amount
If not null:
Poll is sent to selected Group members. Wait till someone is willing to lend
Poll:
group member of yours has requested a loan of {loan_amount}. Are you willing to give?
If no one is willing to lend:
Notify Borrower: No proposals found.
Yes
Please Enter the Interest Rate:
Maintaining the list of proposals
Notify Borrower: Active Proposals list. Choose the one from above list.
If valid Number is chosen
NO
Yes
Chose a valid Proposal number
Is UPI Id of Borrower set?
Yes
Send UPI id of lender to Admin and allow lender to send money to Admin
Send UPI Id of Borrower To Admin and allow admin to send money to borrower.
Download Payment Invoice!
No
Set UPI Id

How to try our bot?

Just click the “Start bot now” button below to get started with the bot.

Start bot now

To run your own Telegram bot on your laptop, you'll need to follow these steps, starting from generating an API key (token) from BotFather:

1) Generate API Key (Token) from BotFather:

- Open Telegram and search for BotFather.
- Start a chat with BotFather and use the /newbot command to create a new bot.
- Follow the prompts to choose a name and username for your bot.
- BotFather will provide you with an API token. Save this token securely, as you'll need it to authenticate your bot.

2) Install Required Packages:

- Make sure you have Python installed on your laptop.
- Install the required Python packages. You can use pip to install them:

3)Prepare Your Code:

- Clone the repo from GitHub.
- Then, rename the example.env file as .env and update the API key (TELE_API_KEY) variable in that file with the token you obtained from BotFather.
- Then also update the MONGO_URI there with the mongo_uri you have for your database.

4)Run Your Bot:

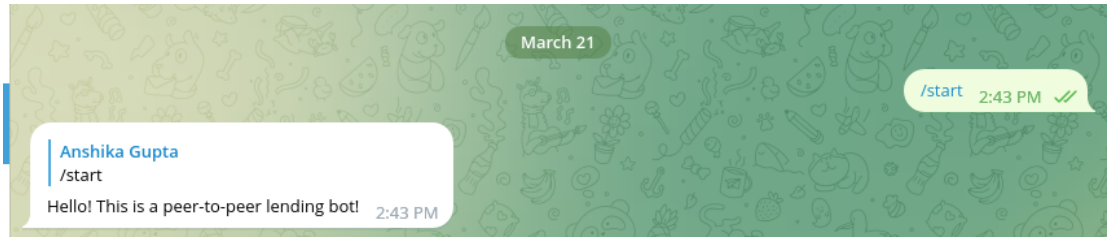
- Open a terminal or command prompt.
- Navigate to the directory where your bot code is located.
- Run your bot script using Python.

5) Interact with Your Bot:

- Once your bot is running, open Telegram and search for your bot using its username.
- Start a chat with your bot and test its functionality by sending commands and messages.
- Here's a refined list of flows through which users can interact with your bot:

A. Greeting:

a. /start or /hello: Start a conversation with the bot and receive a warm greeting.



B. Group Management:

a. **Create a group**: Start the process of creating a new group.

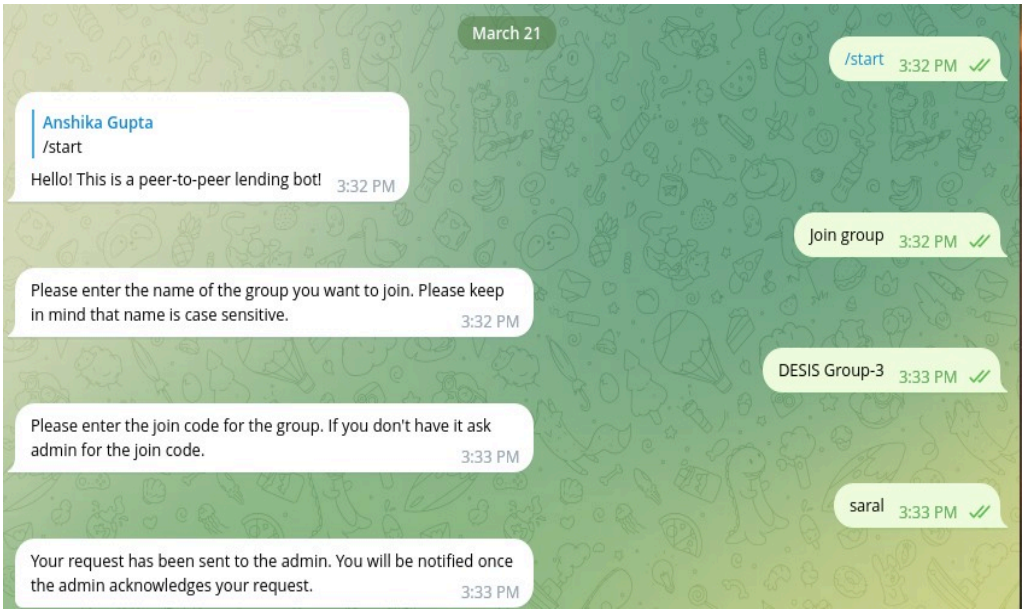
User authentication during the creation of a group is achieved through the following steps:

a)Username Verification: Before creating a group, the user is prompted to set their Telegram username if it's not already set. This ensures that the user is identified by their username within the group.

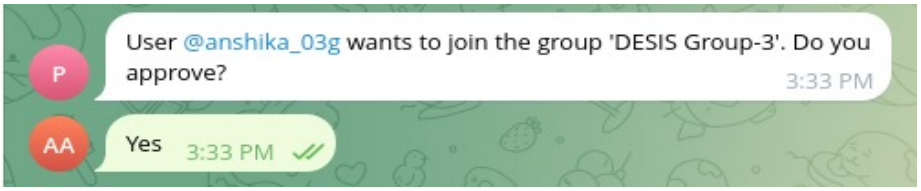
b)Group Name and Join Code: The user provides a group name and a join code during the group creation process. These details are essential for creating and accessing the group.

c)Admin Password: As part of the group creation process, the user is prompted to set an admin password for the group. This password serves as a form of authentication for administrative actions within the group.

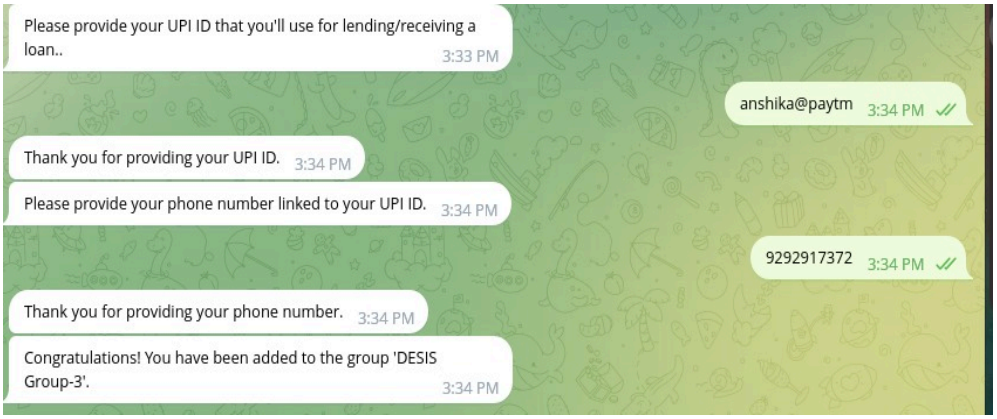
b. Join a group: Initiate joining an existing group.



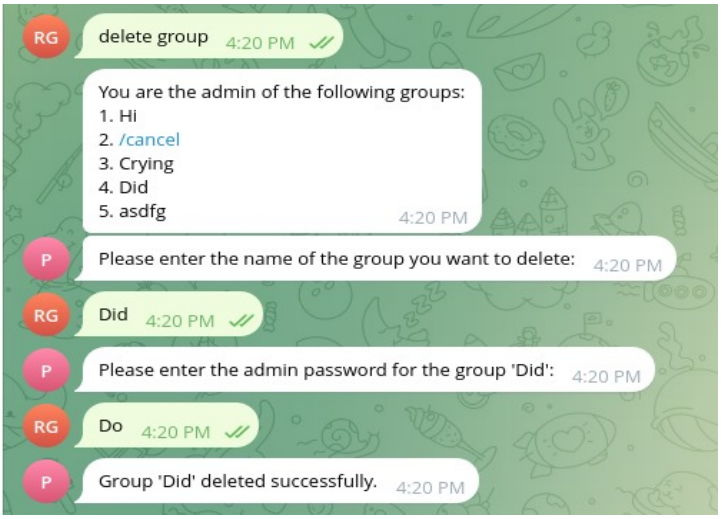
Now an approval message will be sent to the admin of the group asking for the admin's approval to add the particular member to the group



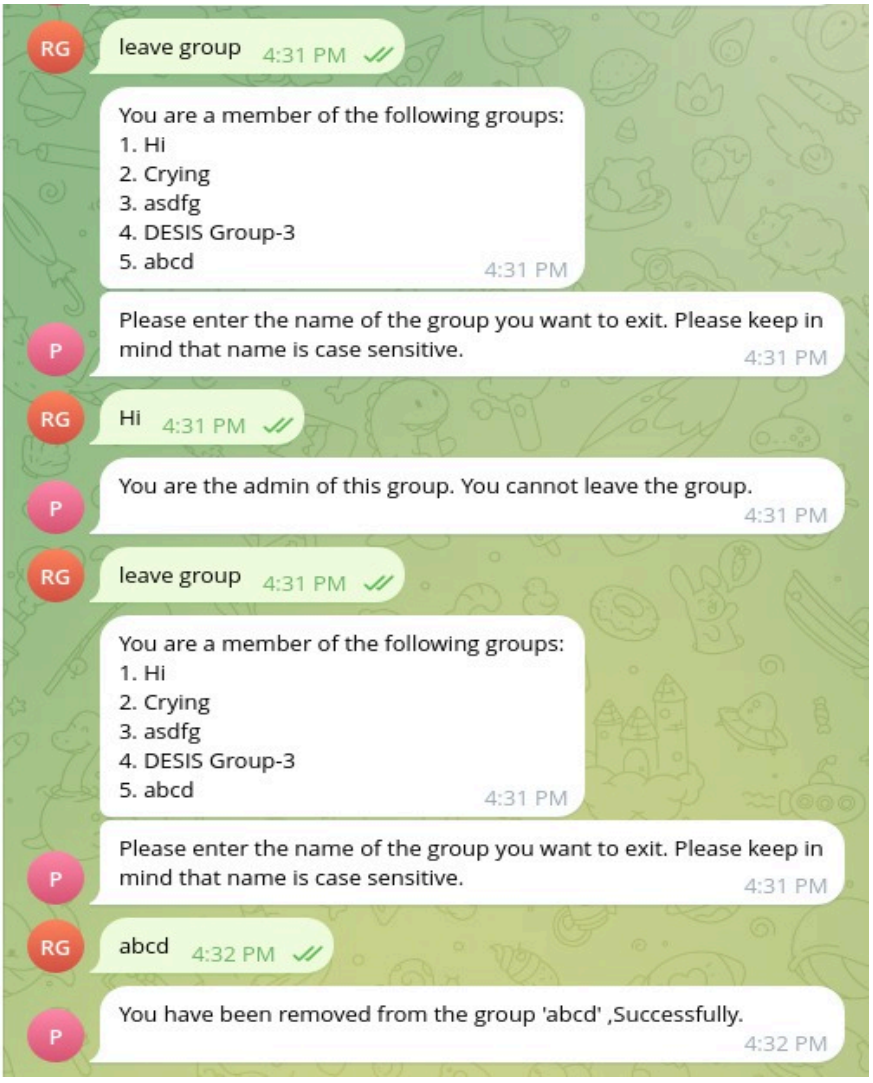
After the request is approved:



c. **Deleting group**: Start the process of deleting an existing group.

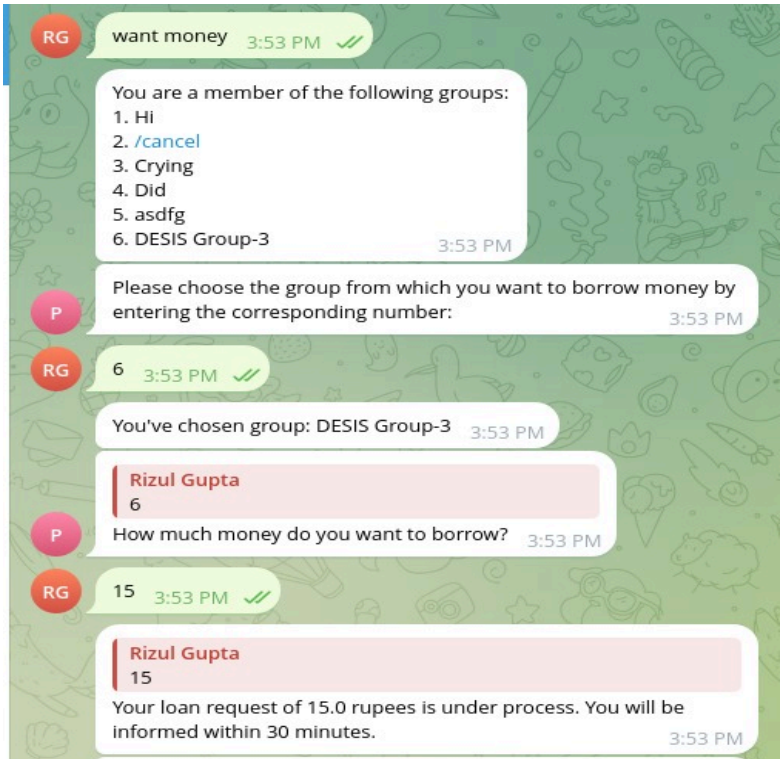


d. **Leaving a group**: Leave a group you're currently a member of.

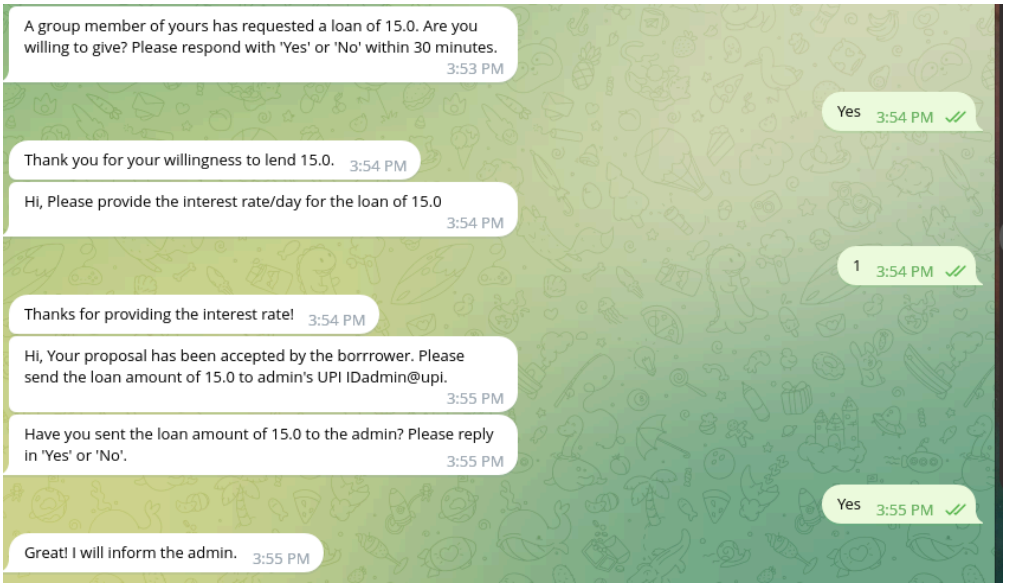


C. Loan Borrowing:

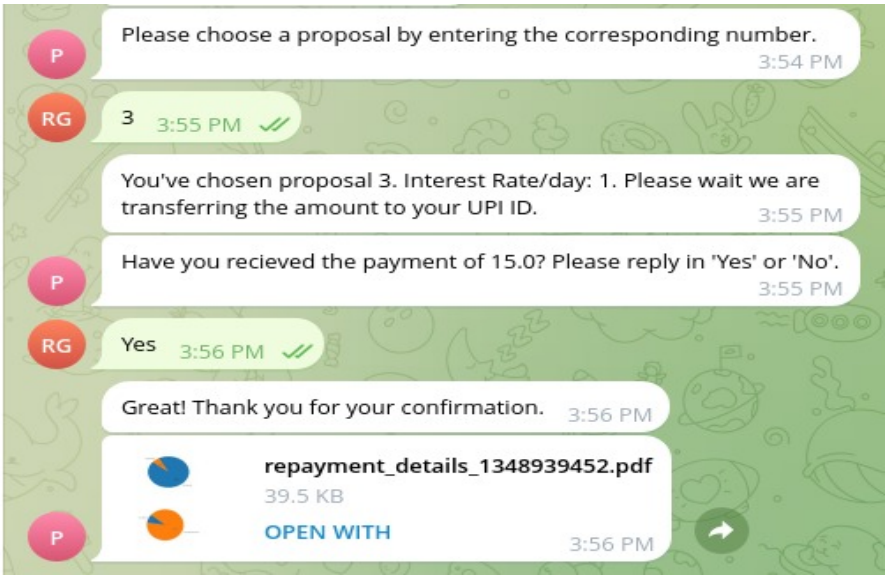
a. Borrow loan: Begin the process of borrowing a loan from a group.



A request will be sent to each member of the group asking their willingness to lend money



Now borrower will choose from the available options



A repayment invoice along with the loan amount, repayment amount, repayment duration, and interest will be sent to the borrower.

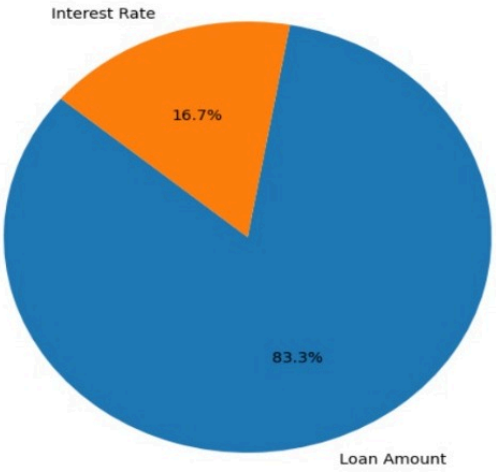
Please find your repayment details in this invoice.

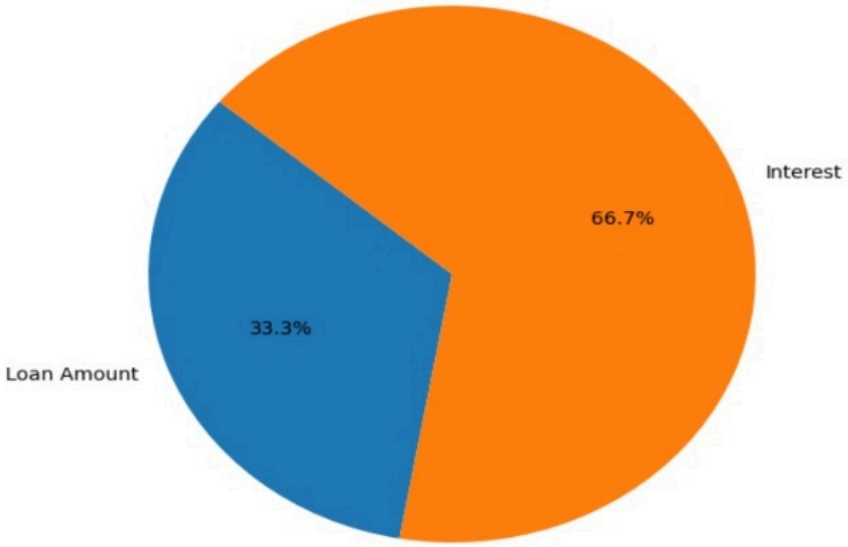
Loan amount: $10.0

Interest rate per day: 2.0%

Repay time: 10.0

Repayment amount: $30.0

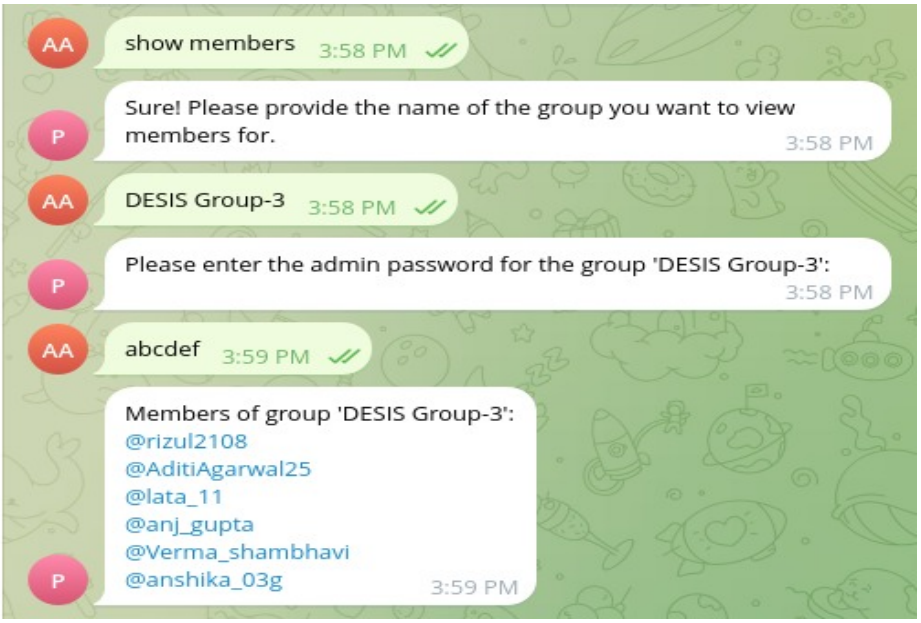




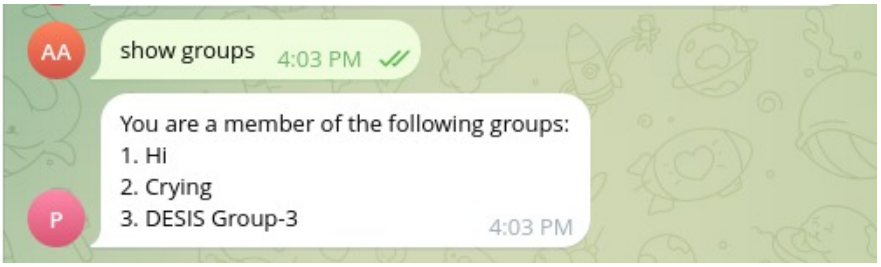
Here admin is acting as a bridge between the transactions between the lender and the borrower.

D. Group Information:

a. Show group members: Display the members of a specific group.



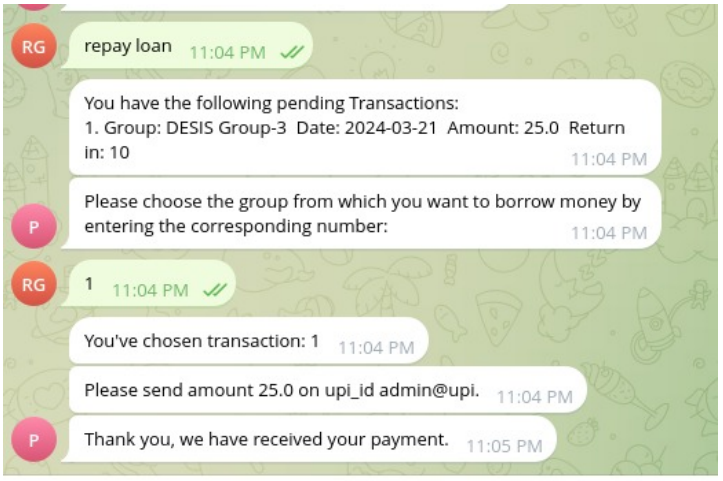
b.Show groups: Display all the groups of which we are members.



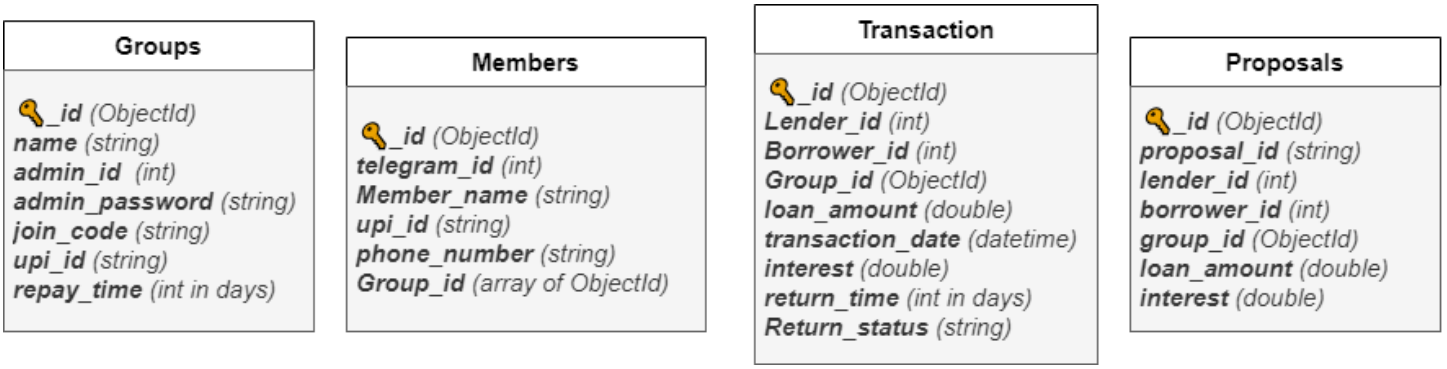
E. Additional Features:

a. Sending any message: Allows users to interact with the bot and get responses.
b. Implicit intents: The bot understands various intents like greetings, loan borrowing, group management, etc., providing a more interactive experience based on the user's input.

F. Loan Repayment:



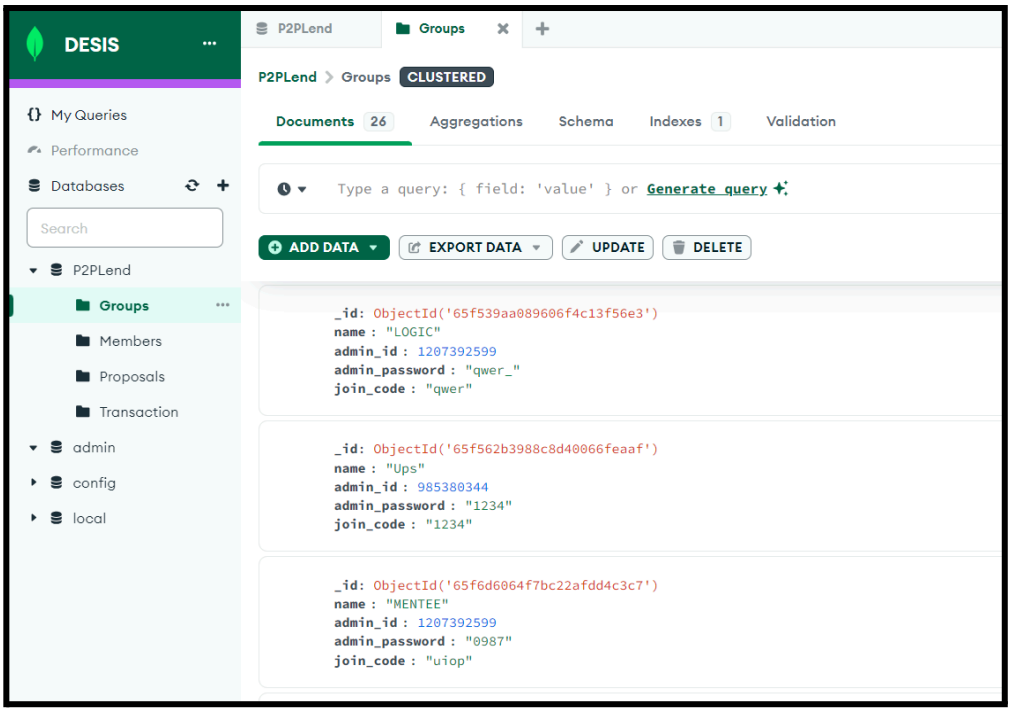
Database:



- **Groups**

  Lists all the existing groups made by admins with our bot

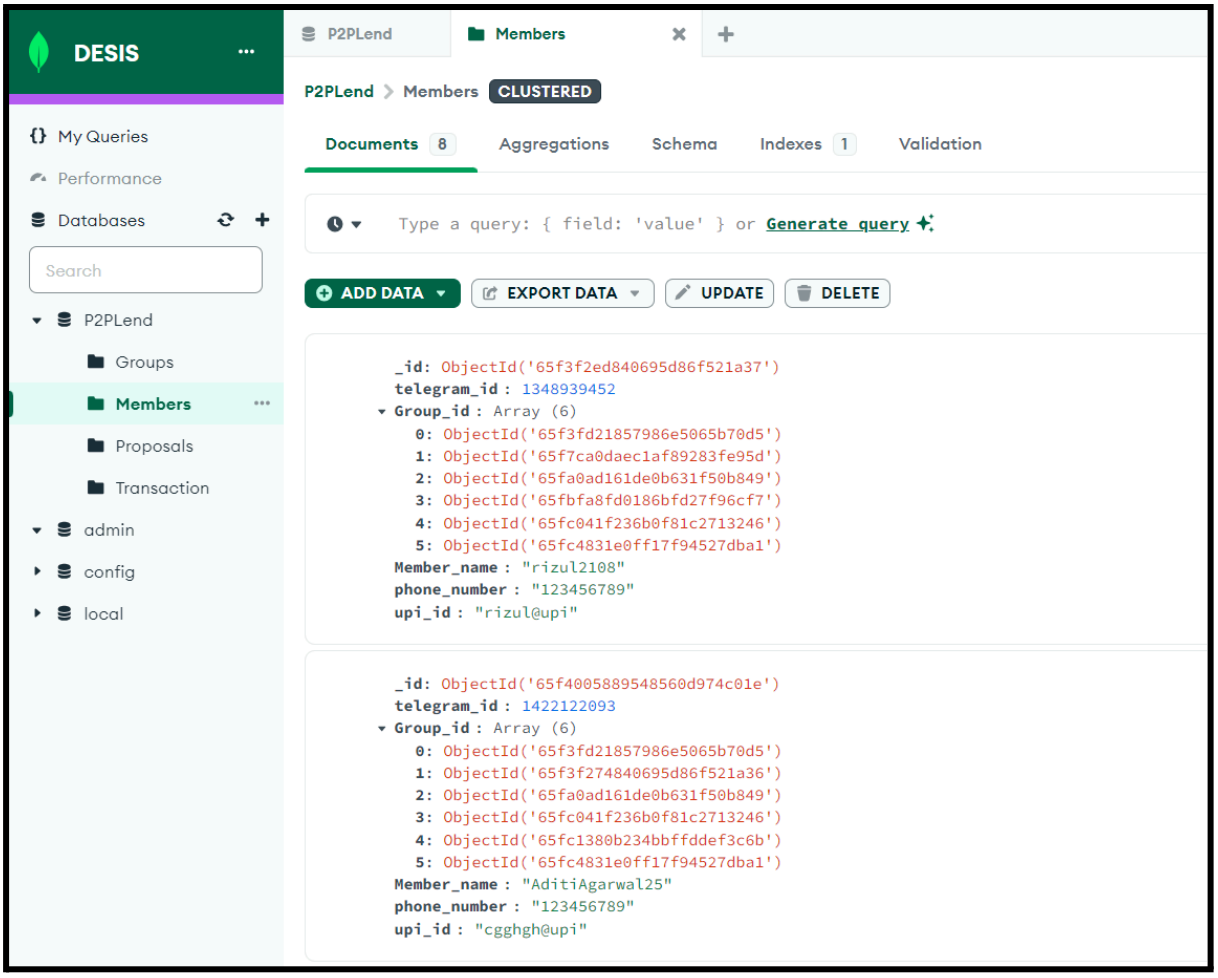
  - ***_id*🔑:** unique id for 'Groups' collection
  - ***name*:** unique and case sensitive provided by the user
  - ***admin_id*:** telegram chat id of admin (remains same for all his conversations)
  - ***admin_password*:** set by admin at the time of group creation and used for admin's authentication while he exercises his administrative actions like group deletion
  - ***join_code*:** password set by admin for any non-member to join the group
  - ***upi_id*:** admin's upi_id which is the intermediary for any transaction
  - ***repay_time:*** maximum time set by admin for which any borrower can borrow money



- **Members**

  List all the members in all the groups

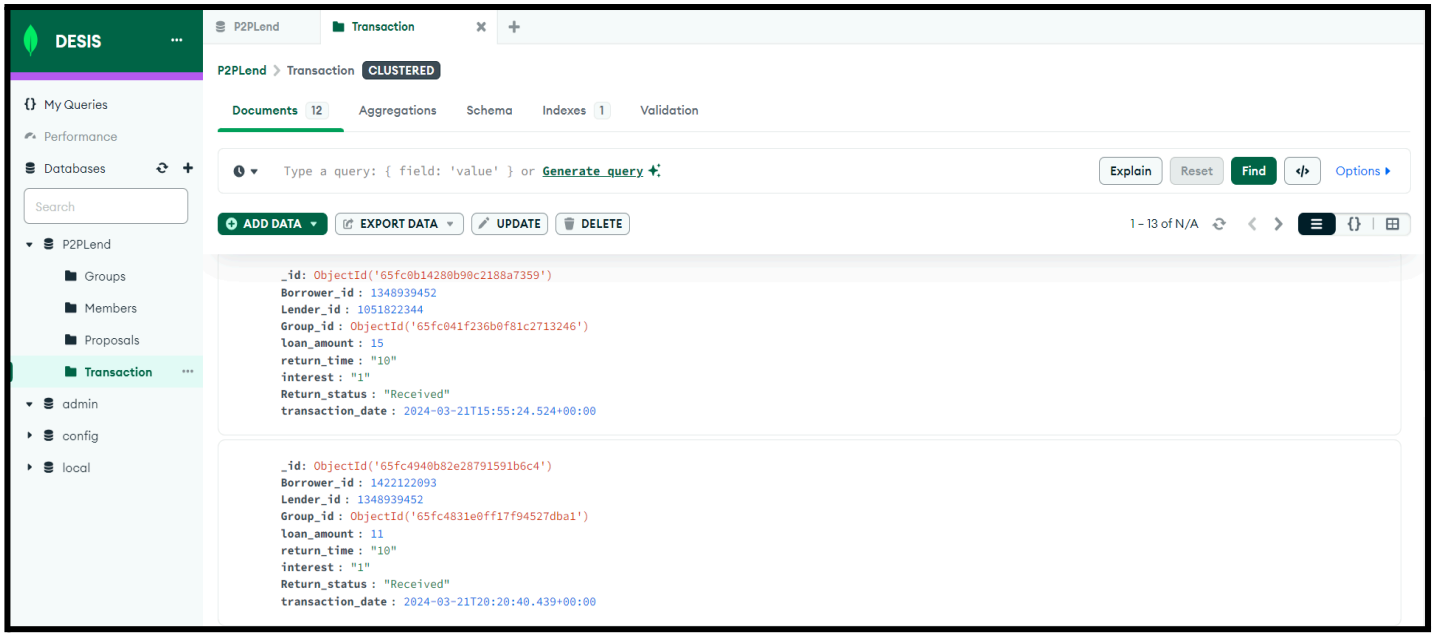
  - ***_id*🔑:** unique id for ‘Members’ collection
  - ***telegram_id:*** telegram chat id of user’s interaction with bot(constant for all conversations)
  - ***Member_name:*** username set on telegram(mandatory to set for bot interaction)
  - ***upi_id*:** member’s upi_id which is used to make transactions
  - ***phone_number:*** provided by the user when he joins his first group
  - ***Group_id:*** Array of all IDs of all groups a user is part of



- **Transaction**

  Lists all the transactions made so far

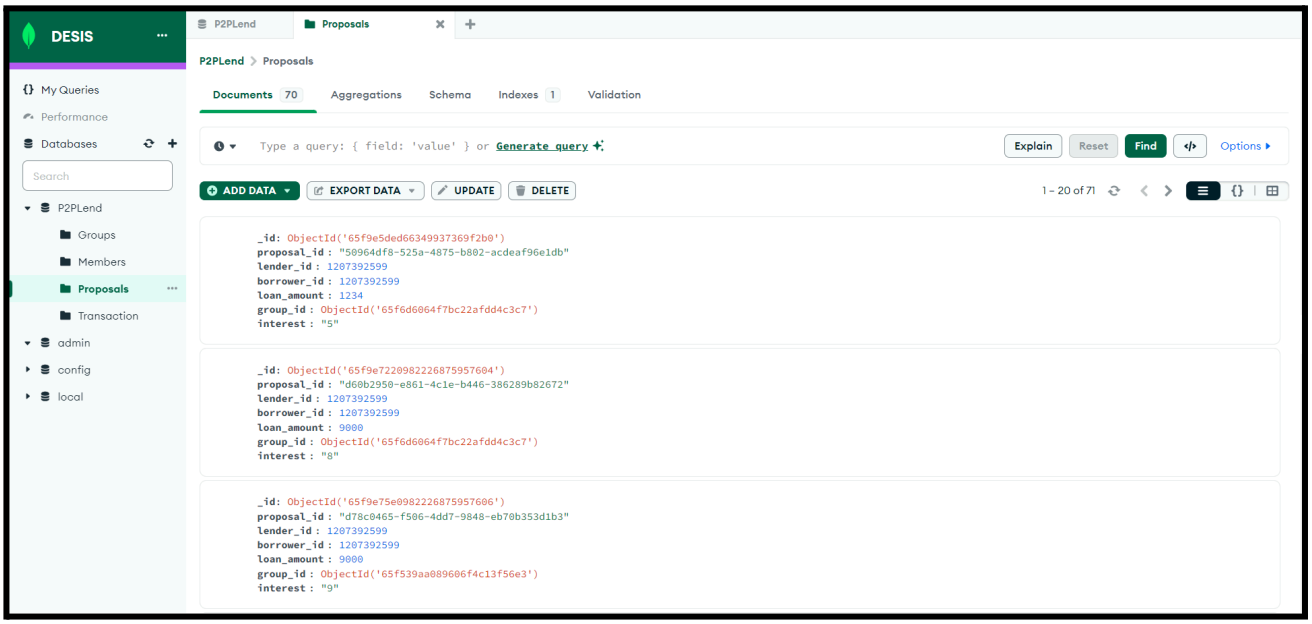
  - **_id🔑:** unique id for ‘Transaction’ collection
  - ***Lender_id:*** lender’s ‘telegram_id’ from ‘Members’ collection
  - ***Borrower_id:*** borrower’s ‘telegram_id’ from ‘Members’ collection
  - ***Group_id:*** ‘_id’ of the group under which the transaction is occurring taken from the ‘Groups’ collection
  - ***loan_amount:*** the principal amount borrowed by a borrower
  - ***transaction_date:*** the time and date at which the transaction was registered
  - ***interest:*** interest/day demanded by the lender
  - ***return_time:*** time in which the borrower is/was required to repay the lender with interest
  - ***Return_status:*** “Received” if the borrower has repaid the money else “Pending”



- **Proposals**

  Lists all the proposals made by all the potential lenders and the interest rates demanded by them from which borrower chooses one

  - ***_id*** 🔑**:** unique id for 'Members' collection
  - ***Proposal_id:*** unique for each loan request by the borrower, that is, same for all proposals under one loan request (used to identify all proposals for one loan request)
  - ***lender_id:*** lender's 'telegram_id' from 'Members' collection
  - ***borrower_id:*** borrower's 'telegram_id' from 'Members' collection
  - ***group_id:*** '_id' of the group under which the loan is requested taken from 'Groups' collection
  - ***loan_amount:*** the principal amount requested for a loan by a borrower
  - ***interest:*** *interest/day demanded by that particular lender*



Challenges we faced:

- Initially, we encountered difficulties in integrating various components of the project to ensure seamless interaction among the backend, frontend, and database, ultimately delivering the desired results through the Telegram messaging system.

- Implementing advanced features such as a button system, where users must choose between 'Yes' and 'No', presented challenges. Our bot, trained with neural intents, sometimes struggled to differentiate between intents and choices, leading to confusion.

- Designing databases, connecting MongoDB tables, and retrieving required data through queries proved to be a challenging task.

- Handling multiple business logic flows, such as loan borrowing, group creation and joining, borrower payments, confirmation requests, and their subsequent testing, posed the greatest challenge. Testing necessitated at least three devices—one acting as an admin, and the others as lender and borrower, respectively.

- Creating pie charts using Matplotlib and embedding them in PDF invoices, along with loan details such as amount, interest rate, repayment time, and repayment amount, and sending them to borrowers via Telegram, was itself a daunting task.

- Managing multiple device inputs for lending, returning money, and other flows posed a significant challenge, particularly in keeping track of Telegram IDs and applying the necessary logic in the code to identify the sender of incoming messages.

What we learned through this project:

From this project, we learned several key concepts and techniques:

- **Bot Development**: We learned how to develop a Telegram bot using the Telebot library in Python. This includes handling various messages, commands, and user interactions.
- **Natural Language Processing (NLP):** We utilized the neural_intents library to implement natural language understanding capabilities for our bot. This allowed us to recognize user intents and extract relevant information from their messages.
- **User Authentication:** The project demonstrates how to implement user authentication mechanisms within a bot application. This includes verifying user identities, managing permissions, and enforcing security measures.
- **Database Integration:** We integrated a database (MongoDB) to store and manage user data, group information, and other relevant details. This enables the persistence of data and efficient retrieval when needed.

- **Group Management:** The bot supports various group management functionalities, such as creating groups, joining groups, leaving groups, and deleting groups. These features are essential for collaborative and organized communication within communities.
- **Error Handling:** The project incorporates mechanisms to gracefully handle invalid inputs, unauthorized access attempts, and other exceptional scenarios. Proper error messaging helps guide users and maintain the integrity of the bot's functionality.
- **Asynchronous Processing:** The bot utilizes asynchronous processing techniques to handle multiple user interactions simultaneously. This ensures responsiveness and scalability, especially in scenarios where the bot serves a large number of users concurrently.
- **Feedback and Interaction:** Through prompts, messages, and interactive components (e.g., reply keyboards), the bot facilitates seamless communication with users. Feedback mechanisms help users understand the bot's actions and provide input as needed.

Overall, this project is a practical example of building a functional Telegram bot with user authentication and group management capabilities, leveraging NLP and database integration for enhanced functionality and user experience.

Libraries and packages used:

- **telebot**: This library is used for interacting with the Telegram Bot API. It provides an interface for sending and receiving messages, handling commands, and managing updates from Telegram users.
- **neural_intents**: This library is used for natural language understanding (NLU) tasks. It allows the bot to recognize user intents and extract relevant information from messages using machine learning models.
- **sys**: This is a built-in module in Python used to access system-specific parameters and functions. In this project, it's used to configure the standard output stream to handle UTF-8 encoding.
- **re**: Another built-in module, `re`, is used for regular expression operations. It's used here for extracting numeric values from user messages.
- **database**: Although not a specific library, `database` likely refers to a custom module or script responsible for database interactions. It could be using a database library like `Pymongo` for MongoDB integration, but the specific library isn't mentioned in the code provided.
- **Numpy:** NumPy is a fundamental Python library for numerical computing, providing powerful tools for handling arrays and matrices.
- **ReportLab:** A Python library used for creating PDF documents programmatically and to generate dynamic and customized PDF files with text, images, graphics, and tables.
- **Matplotlib:** For plotting expected savings plot, i.e., a bar graph showing the difference in current and expected expenditures.
- **PyMongo:** It provides a powerful interface for interacting with MongoDB databases from Python applications.

Future Scope:

- **Telebot to Autonomous App**: Starting with Telebot, our trajectory involves evolving into an autonomous app with its dedicated bot. This transition will optimize user experience, granting independence and fostering growth in our peer-to-peer lending platform.
- **Razorpay Integration**: Currently, an admin serves as an intermediary, but our future roadmap involves integrating Razorpay for secure transactions, and maintaining user anonymity while ensuring safe and reliable peer-to-peer lending operations.
- **Automated Member Verification:** In the future, we plan to integrate automatic member verification for institution-based groups, lessening the admin workload. This enhances security by eliminating join code vulnerabilities, ensuring only eligible members join, and maintaining the integrity of the group's purpose and privacy.
- **Currency Friendly**: Presently, transactions are in Indian rupees, restricting operations to India. In the future, we envision a currency-flexible system, expanding our platform globally. This adaptation broadens accessibility, enabling users worldwide to participate in peer-to-peer lending seamlessly.
- **Credit Scoring and Monitoring**: Develop a proprietary credit scoring system that evaluates borrower creditworthiness based on a variety of factors, including financial history, income stability, and debt-to-income ratio.

References:

- https://pymongo.readthedocs.io/en/stable/tutorial.html
- https://www.mongodb.com/basics/create-database-for-python-app#:~:text=To%20get%20started%20using%20MongoDB,type%20of%20account%20you%20need
- https://pytba.readthedocs.io/en/latest/

Acknowledgments:

We would like to express our heartfelt gratitude to the DESIS Ascend Educare 2023 team for giving us the wonderful opportunity to put the knowledge and skills that we have gained from the program to test through this project. We'd also like to extend our sincere gratitude to our team mentor for their invaluable guidance, support, and expertise throughout this project. Their insights and advice have been instrumental in our project, and we are truly grateful for their dedication and commitment. Thank you for sharing your knowledge and for being an inspiring mentor.