CREATING A TELEGRAM BOT

Done By PRATHAM M

Requirements :

- [ ] Install Telegram App in Mobile/Laptop.
- [ ] Login to telegram using phone number
- [ ] Install node.js latest version (Run JavaScript Everywhere)

Steps to be followed:

1. **Open Telegram**: Open the Telegram app on your device or desktop.

2. **Search for BotFather**: In the Telegram search bar, type "BotFather" and select the BotFather bot from the search results.

3. **Start BotFather**: Click on the "Start" button or send a message to BotFather to initiate a conversation.

4. **Create a New Bot**: Send the **/newbot** command to BotFather to create a new bot. BotFather will guide you through the process and ask you to provide a name for your bot.

5. **Choose a Name**: Choose a unique name for your bot. This name will be displayed to users when they interact with your bot.

6. **Choose a Username**: After providing a name, BotFather will ask you to choose a username for your bot. This username must be unique and end with the word "bot" (e.g., **my_event_bot**). If the username is available, BotFather will confirm your choice.

7. **Receive Bot Token**: Once you've chosen a username, BotFather will provide you with a token for your bot. This token is a string of letters and numbers that you'll use to authenticate your bot with the Telegram API.

8. **Copy Bot Token**: Copy the bot token provided by BotFather. This token is essential for your bot to interact with Telegram's Bot API.

9.Open VS code create file bot.js

10.**CODE:**

```
const TelegramBot = require('node-telegram-bot-api');


const bot = new TelegramBot('Replace ur API ', { polling: true });
```

```
const upcomingEvents = [
    {
        id: 1,
        name: "Event 1",
        date: "2024-05-15",
        time: "18:00",
        location: "Location 1",
        description: "Description 1",
        attendees: []
    },
    {
        id: 2,
        name: "Event 2",
        date: "2024-05-20",
        time: "19:30",
        location: "Location 2",
        description: "Description 2",
        attendees: []
    }
];

bot.onText(/\/start/, (msg) => {
    const chatId = msg.chat.id;
    bot.sendMessage(chatId, 'Welcome to the Event Management Bot! How can I assist you?');
});

bot.onText(/\/list_events/, (msg) => {
```

```
    const chatId = msg.chat.id;

    if (upcomingEvents.length > 0) {
        let message = "Upcoming Events:\n\n";
        upcomingEvents.forEach((event) => {
            message += `Event ID: ${event.id}\n`;
            message += `Name: ${event.name}\n`;
            message += `Date: ${event.date}\n`;
            message += `Time: ${event.time}\n`;
            message += `Location: ${event.location}\n`;
            message += `Description: ${event.description}\n`;
            message += `Attendees: ${event.attendees.length}\n`;
            message += `RSVP: /rsvp ${event.id}\n\n`;
        });
        bot.sendMessage(chatId, message);
    } else {
        bot.sendMessage(chatId, "There are no upcoming events.");
    }
});

bot.onText(/\/create_event/, (msg) => {
    const chatId = msg.chat.id;
    const userId = msg.from.id;

    bot.sendMessage(chatId, 'Please enter the name of the event:');
    bot.once('message', (eventNameMsg) => {
        const eventName = eventNameMsg.text;
```

```
bot.sendMessage(chatId, 'Please enter the date of the event (YYYY-MM-DD):');
bot.once('message', (eventDateMsg) => {
    const eventDate = eventDateMsg.text;

    bot.sendMessage(chatId, 'Please enter the time of the event (HH:MM):');
    bot.once('message', (eventTimeMsg) => {
        const eventTime = eventTimeMsg.text;

        bot.sendMessage(chatId, 'Please enter the location of the event:');
        bot.once('message', (eventLocationMsg) => {
            const eventLocation = eventLocationMsg.text;

            bot.sendMessage(chatId, 'Please enter a description for the event:');
            bot.once('message', (eventDescriptionMsg) => {
                const eventDescription = eventDescriptionMsg.text;

                const newEvent = {
                    id: upcomingEvents.length + 1,
                    name: eventName,
                    date: eventDate,
                    time: eventTime,
                    location: eventLocation,
                    description: eventDescription
                };
                upcomingEvents.push(newEvent);

                const eventSummary = `Event created successfully!\n\n${JSON.stringify(newEvent, null, 2)}`;
                bot.sendMessage(chatId, eventSummary);

                bot.sendMessage(userId, `Your event "${eventName}" has been created successfully!`);
```

```
                });
            });
        });
    });
});

bot.onText(/\/rsvp (.+)/, (msg, match) => {
    const chatId = msg.chat.id;
    const eventId = parseInt(match[1]);
    const event = upcomingEvents.find(event => event.id === eventId);

    if (event) {
        const keyboard = {
            inline_keyboard: [
                [
                    { text: "Attend", callback_data: `attend_${eventId}` },
                    { text: "Maybe", callback_data: `maybe_${eventId}` },
                    { text: "Decline", callback_data: `decline_${eventId}` }
                ]
            ]
        };
        bot.sendMessage(chatId, `Event Details:\n\nName: ${event.name}\nDate: ${event.date}\nTime: ${event.time}\nLocation: ${event.location}\nDescription: ${event.description}`, {
            reply_markup: JSON.stringify(keyboard)
        });
    } else {
        bot.sendMessage(chatId, "Event not found.");
```

```
    }
});

bot.on('callback_query', (callbackQuery) => {
    const chatId = callbackQuery.message.chat.id;
    const eventId = parseInt(callbackQuery.data.split('_')[1]);
    const userId = callbackQuery.from.id;
    const rsvpAction = callbackQuery.data.split('_')[0];

    let responseMessage = "";

    switch (rsvpAction) {
        case "attend":
            responseMessage = `You have RSVP'd to attend the event with ID ${eventId}.`;

            const event = upcomingEvents.find(event => event.id === eventId);
            if (event) {
                event.attendees.push(userId);

                if (event.attendees.length >= 3) {
                    const attendees = event.attendees.join(',');
                    bot.createGroupChat(chatId, attendees)
                        .then((groupChat) => {
                            bot.sendMessage(userId, `Congratulations! You've been added to the group chat for attendees of the event "${event.name}".`);
                        })
                        .catch((error) => {
                            console.error("Error creating group chat:", error);
                        });
```

```
            }
        }
            break;
        case "maybe":
            responseMessage = `You have RSVP'd to maybe attend the event with ID ${eventId}.`;
            break;
        case "decline":
            responseMessage = `You have RSVP'd to decline the event with ID ${eventId}.`;
            break;
        default:
            responseMessage = "Invalid RSVP action.";
    }

    bot.sendMessage(userId, responseMessage);
});

bot.on('message', (msg) => {
    const chatId = msg.chat.id;
    bot.sendMessage(chatId, 'You said: ' + msg.text);
});
```

10. **Replace Token in Code**: In your **bot.js** file, replace the placeholder **'YOUR_BOT_TOKEN'** with the actual bot token you received from BotFather.

11. **Save Changes**: Save your **bot.js** file after replacing the token.

12 . In terminal locate it to location of folder where the file is stored and install the packages .

- **npm install node-telegram-bot-api**
- **npm update node-telegram-bot-api**

13 . To Run the code follow these steps :

- [ ] npm init
- [ ] node bot.js

14 . Open telegram app: BotFather will be give u link to open ur bot and start running the bot

15 . Commands to run the bot :

- /start
- /list_events
- /create_event
- /rsvp 1 or rsvp id
- /stop