Plant Care Reminder System

🧠 Objective:

Create a web application where users can add their plants, set watering schedules, and receive reminders for plant care activities (e.g., watering, fertilizing, pruning). The system should allow users to store details about each plant, including its name, species, care instructions, and watering frequency.

Tech Stack:

- **Frontend**: Angular
- **Backend**: .NET Core Web API
- **Database**: SQL Server (with stored procedures)
- **API Testing**: Postman

1️⃣ Database Design & Stored Procedures

📘 Tables:

1. **Users**
    a. **UserId** (PK, INT, IDENTITY)
    b. **UserName** (VARCHAR)
    c. **Email** (VARCHAR)
    d. **PasswordHash** (VARCHAR)
2. **Plants**
    a. **PlantId** (PK, INT, IDENTITY)
    b. **UserId** (FK → Users)
    c. **PlantName** (VARCHAR)
    d. **Species** (VARCHAR)
    e. **WateringFrequency** (INT, Days) – How often the plant needs to be watered
    f. **FertilizingFrequency** (INT, Days) – Frequency for fertilizing the plant
    g. **LastWatered** (DATE) – The last date the plant was watered
    h. **LastFertilized** (DATE) – The last date the plant was fertilized
    i. **CareInstructions** (TEXT)
3. **Reminders**
    a. **ReminderId** (PK, INT, IDENTITY)
    b. **UserId** (FK → Users)
    c. **PlantId** (FK → Plants)
    d. **ReminderType** (VARCHAR) – e.g., "Watering", "Fertilizing"
    e. **ReminderDate** (DATE) – The date when the reminder should occur

📚 Stored Procedures:

1. **sp_AddPlant** – Inserts a new plant into the database.
    a. **Inputs**: @UserId, @PlantName, @Species, @WateringFrequency, @FertilizingFrequency, @CareInstructions
2. **sp_GetPlantsByUser** – Retrieves all plants for a specific user.
    a. **Inputs**: @UserId
3. **sp_AddReminder** – Inserts a reminder for a plant.
    a. **Inputs**: @UserId, @PlantId, @ReminderType, @ReminderDate

4. **sp_GetRemindersByDate** – Retrieves reminders for the current day or a specified date.
    a. **Inputs**: @UserId, @Date
5. **sp_UpdateLastWatered** – Updates the last watered date for a specific plant.
    a. **Inputs**: @PlantId, @LastWatered
6. **sp_UpdateLastFertilized** – Updates the last fertilized date for a specific plant.
    a. **Inputs**: @PlantId, @LastFertilized

2️⃣ Backend – .NET Core API

📁 API Endpoints:

Method	Endpoint	Description
POST	/api/plants/add	Adds a new plant to the user's plant collection (calls **sp_AddPlant**)
GET	/api/plants?userId=1	Retrieves all plants for a specific user (calls **sp_GetPlantsByUser**)
POST	/api/reminders/add	Sets a reminder for a plant (calls **sp_AddReminder**)
GET	/api/reminders/today?userId=1	Retrieves today's reminders for the user (calls **sp_GetRemindersByDate**)
PUT	/api/plants/water	Updates the last watered date for a plant (calls **sp_UpdateLastWatered**)
PUT	/api/plants/fertilize	Updates the last fertilized date for a plant (calls **sp_UpdateLastFertilized**)

📌 Backend Notes:

- Use ADO.NET to call stored procedures.
- Validate that the input for watering and fertilizing dates is correct and not in the future.
- Implement authentication for users to ensure only registered users can access their data.
- Handle errors gracefully (e.g., invalid input, database issues).
- Implement business logic to calculate the next watering or fertilizing dates based on the user's settings and current date.

3️⃣ Frontend – Angular

🧩 Pages & Components:

1. **Dashboard Page**
    a. **Upcoming Reminders**: Show a list of the user's plant care reminders for the day, including watering and fertilizing.
    b. **Add New Plant**: Button to open the form for adding new plants.
2. **Plant Details Page**
    a. **Plant Information**: Displays details about the plant, including species, care instructions, and watering/fertilizing frequency.
    b. **Set Reminder**: Allows the user to set a reminder for watering or fertilizing.
    c. **Last Watered & Fertilized Dates**: Display the last dates for these activities and allow the user to update them.
3. **Reminder Page**
    a. **Reminder List**: A list of all the reminders for the plants, with a button to mark them as completed.
    b. **Date Filter**: A date picker to filter reminders by date.
4. **Reminder Notification**
    a. **Popup/Toast Notification**: Display a reminder when it's time to water or fertilize a plant.

🔗 API Integration:

- Use **Angular HttpClient** to call the backend API endpoints.
- Handle loading states while waiting for the backend to respond (use loading spinners).
- Implement form validation in the frontend (e.g., required fields, valid frequency input).
- Use **Angular** components for UI elements such as buttons, dialogs, and notifications.
- Use **RxJS** for reactive programming to manage asynchronous API calls and state changes (e.g., for reminders).

4️⃣ Bonus (Optional)

- **Reminder Notifications**: Implement email or push notifications for plant care reminders.
- **Graphical Timeline**: Display a timeline or calendar view showing upcoming watering and fertilizing dates for each plant.
- **Plant Health Tracking**: Allow users to track the health of their plants over time by marking them as healthy, under-watered, or over-watered.

Suggested Timeline (10 Days)

Day	Task
1	Set up the database schema (tables, stored procedures)
2-3	Implement backend API for adding plants, setting reminders, and updating watering/fertilizing dates
4	Test backend API using Postman and refine stored procedures
5-6	Build Angular components for managing plants and reminders
7	Implement reminder notifications and reminder list page
8	Integrate the frontend with the backend API (add plants, set reminders)
9	Finalize the user interface and testing
10	Final testing, bug fixes, and polish the application