Task Management System

This document details the updated approach, including login functionality, database schema, and time estimates for building a task list web application for FinStack.

Technology Stack:

- Frontend: Angular 2+
- Backend: Python 3.6.x+ with Flask
- Database: MongoDB (considerations for alternatives MongoDB Atlas/Firebase Storage)
- Hosting: netlify (considerations for alternatives heroku)

I will utilize MongoDB as a NoSQL document database. Here's the proposed schema structure:

Users Collection:
- _id (ObjectId): Unique identifier for the user.
- username (String): Username for login (unique).
- password_hash (String): Hashed password for secure storage.

Tasks Collection:
- _id (ObjectId): Unique identifier for the task.
- date_created (Date): Date the task was created.
- entity_name (String): Name of the associated entity.
- task_type (String): Type of the task (enum: "Meeting", "Call", "Video Call").
- time (Date): Date and time the task needs to be completed (optional).
- contact_person (String): Username of the assigned team member.
- note (String): Optional note associated with the task.
- status (String): Status of the task (open/closed).

Features and Implementation:

Login:
A dedicated login page will be implemented on the frontend.
Users will be able to enter their username and password.
Upon login attempt, a POST request will be sent to a Flask endpoint that:

- Validates the username and password against stored credentials (likely hashed passwords) in the Users collection.
- If valid, generates a secure session token and sends it back to the frontend.
- If invalid, returns an error message.
- The frontend will store the received session token securely (e.g., cookies) and include it in subsequent requests to authenticate the user.

Validates the username and password against stored credentials (likely hashed passwords) in the Users collection (accessed through a MongoDB driver library like PyMongo).
If valid, generates a secure session token and sends it back to the frontend.
If invalid, returns an error message.
The frontend will store the received session token securely (e.g., cookies) and include it in subsequent requests to authenticate the user.

Features and Implementation:

- **Create Tasks:**
    - A form will be provided on the frontend to capture task details (date, entity name, task type, time, contact person, note).
    - Upon form submission, a POST request will be sent to a Flask endpoint that validates the data and inserts a new record into the `tasks` table.
- **Assign Tasks:**
    - The "contact person" field in the task creation form will allow selecting a team member from a dropdown menu.
    - The selected contact person will be stored in the database.
- **Edit Existing Tasks:**
    - Clicking on a task will display its details and an edit button.
    - Clicking the edit button will open a pre-filled form for modifying the task details.
    - Submitting the edit form will trigger a PUT request to a Flask endpoint that updates the corresponding record in the `tasks` table.
- **Change Task Status:**
    - Each task will have a visual indicator (e.g., checkbox, button) to mark it as "completed" or "open."
    - Clicking the status indicator will send a PATCH request to a Flask endpoint that updates the `status` field in the `tasks` table.
- **Sort and Filter Tasks:**
    - The application will provide options to sort and filter tasks based on various criteria (team member, task type, status, date, entity name, contact person).
    - Selected filters will be sent to a Flask endpoint that retrieves tasks based on the provided criteria. The backend will utilize database queries with filtering and sorting clauses to retrieve the desired data.
- **Delete Tasks:**
    - Each task will have a delete button to permanently remove it.

    - Clicking the delete button will trigger a DELETE request to a Flask endpoint that removes the corresponding record from the `tasks` table.

Time Estimates:

Phase 1 (By June 23st, 11:59 PM):

- Documenting approach, database schema, and time estimates (**1 Day**).

Phase 2 (By June 28th, 11:59 PM):

Backend Development with Flask and PyMongo (1.5 Days):

- **June 23rd-24th:**
    - Flask project setup, user model and authentication logic
    - Implement Flask endpoints for CRUD operations (Create, Read, Update, Delete) on tasks
    - Integrate MongoDB interaction with PyMongo driver for data persistence

Frontend Development with Angular (2 Days):

- **June 24th:**
    - Learn Angular.js
    - Develop login functionality on the frontend (1 Day)
- **June 25th:**
    - Implement task management features (create, edit, filter, etc.) on the frontend.

Integration and Deployment (1 Day):

- **June 26th:**
    - Integrate backend and frontend for a cohesive application
    - Deploy the application to Netify/heroku

Documentation and Code Commenting (1 Day):

- **June 27th:**
    - Document the approach, code, and deployment process (1 Day)

Buffer: (Optional)

- **June 28th: (0.5 Day) - Unforeseen challenges / Improving Frontend**

Total Estimated Time: 5-6 Days