Flutter Developer Practical Exam

Objective

The objective of this exam is to create a basic task management app using Flutter, showcasing your ability to design, implement features, and work with APIs. You should be able to demonstrate your understanding of:

- Flutter widgets and layouts
- State management
- API integration
- Basic animations (optional)
- App navigation

Instructions

• This exam is to be completed at home.
• You are allowed to use online resources, but you must complete the project yourself.
• You should upload the completed project to a public GitHub repository and provide a README file explaining how to run the project.
• Time frame: You have 7 days to complete this exam from the time of receiving it.

Project Requirements: Task Management App

Create a simple Task Management App with the following features:

1. User Interface

Design a user-friendly and responsive UI for the app. The app should include:

- Home Screen: A list of tasks that shows:
  - Task title
  - Task description
  - Task due date
  - Status (Completed/Incomplete)
- Add/Edit Task Screen: A form that allows users to add or edit tasks with the following fields:
  - Title
  - Description
  - Due Date (use a date picker)
  - Status (Completed/Incomplete)
- Navigation Drawer: Include a drawer for navigation, allowing access to Home, Completed Tasks, and Settings screens.

2. State Management

You can use any state management technique of your choice (e.g., Provider, Bloc, or Riverpod) to manage the app's state.
- Implement local storage (using shared_preferences or SQLite) to store tasks, so tasks persist when the app is closed and reopened.

3. API Integration

Integrate an external API to fetch and display motivational quotes on the home screen.
- You can use this API: [Quotes API](https://type.fit/api/quotes) or any other quotes API of your choice.
- Display a random quote each time the user opens the app.

4. Notifications

Implement local notifications to remind the user of tasks that are due soon (e.g., within 1 hour).
- Use the `flutter_local_notifications` package for this functionality.

5. Bonus Features (Optional)

If you want to showcase additional skills, you can implement:
- Animations: Add a simple animation when navigating between screens or when tasks are marked as completed.
- Dark Mode: Provide a setting to toggle between light and dark mode for the app.

Evaluation Criteria

You will be evaluated based on the following aspects:
1. Code Quality: Clean, well-structured, and readable code.
2. UI/UX Design: The app should be user-friendly and responsive across different devices.
3. Functionality: The app should meet all the basic requirements mentioned above.
4. State Management: Proper use of state management techniques to handle UI and data consistency.
5. API Integration: Correctly fetching and displaying data from the external API.
6. Persistence: Ensure tasks persist after closing and reopening the app.
7. Notifications: Correct implementation of local notifications.
8. Bonus Features: (Optional) Use of animations and dark mode will be considered a plus.

Submission Guidelines

• Create a GitHub repository for the project and push your code there.
• Include a README file with instructions on how to set up and run the project.
• Send the link to your GitHub repository before the deadline.