SMART - BRIDGE INTERNZ ORAGANIZATION

SAMHITHA
DEGREE COLLEGE

TEAM MEMBERS :-

1.T.Nagini (200927109079)
2.K.K.N.Ratnam(200927109063)
3.Ch.Chandhini (200927109052)
4.D.S.Divya sri (200927109054)
5.T.Veerababu (200927146106)

TODO APP

INTRODUCTION:

A TODO APP is a software application designed to help users organize and manage their tasks and to-do lists . It allows user's to create ,prioritize ,and track tasks ,set dead lines ,and mark tasks as completed .TODO APPS are commonly used for personal and professional productivity to stay organise and ensure tasks are completed on time.

Creating a TODO app involves several steps. Here's a high-level overview of the process:

1. Define the Requirements: Determine the features and functionalities you want in your TODO app, such as task creation, task completion, priority levels, due dates, etc.

2. Choose a Development Platform: Decide whether you want to develop a web app, mobile app, or desktop app. Choose the appropriate programming languages and frameworks for your chosen platform.

3. Design the User Interface: Create wireframes or mockups of your app's user interface to plan the layout and interactions

4. Set Up a Development Environment: Install the necessary development tools and software to start coding your TODO app.

5. Implement Task Management: Build the core functionality for creating, updating, and deleting tasks. Store task data in a database.

6. Add Task Prioritization and Due Dates: Implement features to assign priority levels and due dates to tasks.

7. Implement Task Completion: Allow users to mark tasks as completed and update their status accordingly.

8. User Authentication (Optional): If you want to add user accounts and secure data, implement user authentication.

9. Testing: Thoroughly test your TODO app to ensure it works correctly and fix any bugs or issues.

10. Deployment: Choose a hosting platform for your app (if it's a web app) or submit it to app stores (if it's a mobile app).

11. Monitor and Update: Continuously monitor your app's performance and user feedback. Regularly update and improve the app based on user needs.

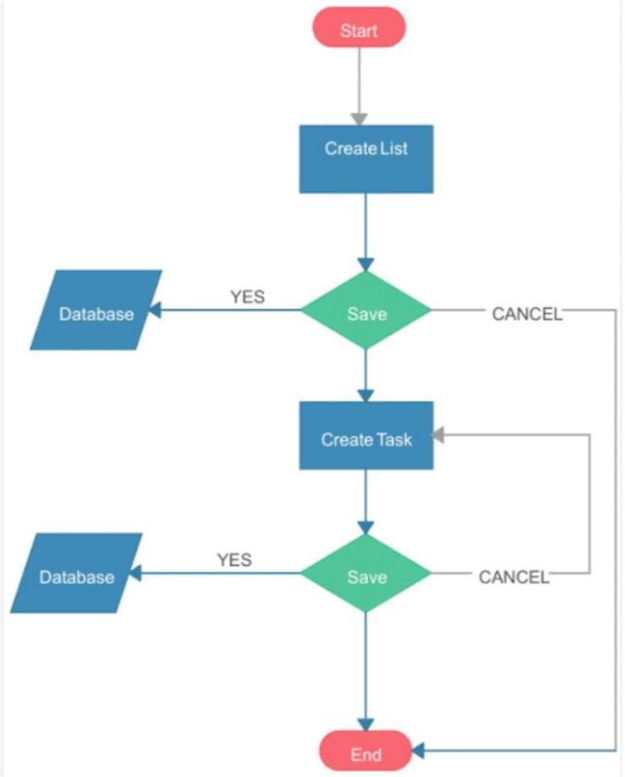
Pre requisites:

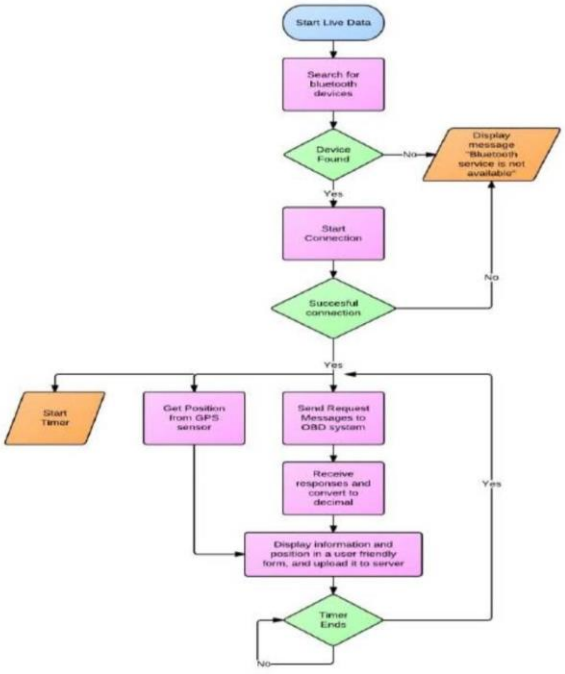
Before we begin with our project, we would like you to know the prerequisites for the same so you can understand it. We have made our project using Java programming, but Android even supports Kotlin now.

Please consider the following things that you must have good hands with-

1. Object Oriented Programming Concepts
2. Java Programming / Kotlin
3. Extensible Markup Language
4. Android Studio
5. Android Virtual Machine

FLOW CHART :-





PROJECT STRUCTURE :-

Setting up the project structure for a TODO app involves organizing the codebase and resources in a systematic way. Here's a basic project structure outline:

TODO-App/

```
|-- app/
|   |-- src/
|   |   |-- main/
|   |   |   |-- java/
|   |   |   |   |-- com.example.todoapp/
|   |   |   |   |   |-- activities/
|   |   |   |   |   |   |-- MainActivity.java
|   |   |   |   |   |-- adapters/
|   |   |   |   |   |   |-- TaskAdapter.java
|   |   |   |   |   |-- data/
|   |   |   |   |   |   |-- models/
|   |   |   |   |   |   |   |-- Task.java
|   |   |   |   |   |   |-- repositories/
|   |   |   |   |   |   |   |-- TaskRepository.java
```

```
|   |   |   |   |   |-- utils/
|   |   |   |   |   |   |-- DateUtils.java
|   |   |   |   |   |-- viewmodels/
|   |   |   |   |   |   |-- TaskViewModel.java
|   |   |   |-- res/
|   |   |   |   |-- layout/
|   |   |   |   |   |-- activity_main.xml
|   |   |   |   |-- values/
|   |   |   |   |   |-- strings.xml
|   |   |-- test/
|   |   |   |-- java/
|   |   |   |   |-- com.example.todoapp/
|   |   |   |   |   |-- TaskViewModelTest.java
|   |-- build.gradle
|-- gradle/
|   |-- wrapper/
|   |   |-- gradle-wrapper.properties
|-- build.gradle
|-- settings.gradle
```

|-- gradle.properties

|-- local.properties

|-- gradlew

|-- gradlew.bat

Design and implement the user interface:

As an AI language model, I can provide a general outline for designing and implementing the user interface (UI) of a TODO app. However, keep in mind that actual UI design and implementation might require additional details and considerations based on your specific requirements and platform (Android or iOS). For this example, I'll outline the steps for an Android app using XML layout files:

1. *Design the UI*:

   - Create a wireframe or sketch of the app's main screen, where users can view and manage their tasks.

   - Plan the layout, including elements like task list, task item, add task button, etc.

- Decide on the color scheme and typography to maintain a consistent visual style.

2. *Create XML Layouts*:

- Open the `activity_main.xml` layout file in the `res/layout` directory.

- Design the main screen UI by using LinearLayout, RecyclerView, TextView, EditText, and Button elements as needed.

- Define the task item layout (e.g., `task_item.xml`) to display individual tasks in the Recycler View.

3. *Implement Main Activity*:

- Create `MainActivity.java` in the `activities` package.

- Initialize the Recycler View and its adapter.

- Set up the Recycler View with a Layout Manager to display the list of tasks.

- Implement the logic to fetch tasks from the repository and pass them to the Recycler View adapter.

4. *Implement Task Adapter*:

- Create `TaskAdapter.java` in the `adapters` package.

- Extend Recycler View .Adapter and implement required methods.

- Inflate the `task_item.xml` layout in `on Create View Holder`.

- Bind task data to the views in `on Bind View Holder`.

- Implement item click listeners to handle task interactions.

5. *Handle Add Task*:

- Add an Edit Text and Button for adding new tasks to the layout.

- Implement logic to handle the "Add Task" button click.

- When the button is clicked, retrieve the task from the Edit Text, create a new Task object, and add it to the repository.

6. *Implement Task Model*:

- Create the `Task.java` class in the `data/models` package.

- Define properties for a task, such as ID, title, description, due date, etc.

7. *Implement Task Repository*:

- Create `TaskRepository.java` in the `data/repositories` package.

- Implement methods to handle CRUD operations for tasks (e.g., add Task, get All Tasks, delete Task).

8. *Implement Task View Model (Optional)*:

- Create `TaskViewModel.java` in the `view models` package (if using MVVM architecture).

- Implement the View Model to handle data between Main Activity and Task Repository.

9. *Apply Styles and Themes*:

- Define styles and themes in the `res/values` directory to maintain a consistent look and feel across the app.

10. *Test and Iterate*:

- Test the app thoroughly to ensure all functionalities work as expected.

- Iterate on the design and implementation based on user feedback and to fix any bugs.

Remember, this is a basic outline, and you may need to modify or extend it depending on your app's complexity and requirements. Additionally, it's essential to follow Android development guidelines and best practices while designing and implementing the UI

Implement task management functionality:

```
<!DOCTYPE html>
<html>
<head>
  <title>Todo List</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <link href="style.css" rel="stylesheet">
</head>
<body>
```

```
<div class="container mt-5">
    <h1 class="mb-4">Todo List</h1>
    <div class="form-group">
      <input type="text" class="form-control"
id="todoInput" placeholder="Enter a new task">
    </div>
    <button type="button" class="btn btn-primary"
onclick="addTodo()">Add Task</button>
    <ul class="list-group mt-3" id="todoList">
      <!-- Todo items will be added here dynamically
-->
    </ul>
  </div>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery
/3.6.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.co…
[2:35 pm, 27/07/2023] Chandhana (Dc): let todos
= [];
```

```
function addTodo() {
  const inputElement =
document.getElementById("todoInput");
  const task = inputElement.value.trim();
  if (task !== "") {
    todos.push(task);
    inputElement.value = "";
    renderTodos();
  }
}
function deleteTodo(index) {
  todos.splice(index, 1);
  renderTodos();
}
function renderTodos() {
  const todoListElement =
document.getElementById("todoList");
  todoListElement.innerHTML = "";
todos.forEach((task, index) => {   const
listItemElement = document.createElement("li");
```

```
listItemElement.className = "list-group-item d-flex justify-content-between align-items-center";
listItemElement.innerHTML = `
  ${task}
   <button type="button" class="btn btn-danger btn-sm" onclick="deleteTodo(${index})">Delete</button>
    `;
todoListElement.appendChild(listItemElement);
  });
}
// Render initial todos (if any)
renderTodos();
[2:35 pm, 27/07/2023] Chandhana (Dc): body {
    font-family: Arial, sans-serif;
    height:100vh;
    background-image: url(images/to-do-4483048.jpg);
    background-repeat: no-repeat;
    background-size:cover;
```

```
    overflow: hidden;
}
.container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
}
h1 {
    text-align: center;
    color: #030202;
   font-family:system-ui, -apple-system,
BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue',
sans-serif
}
.input-container {
    display: flex;
```

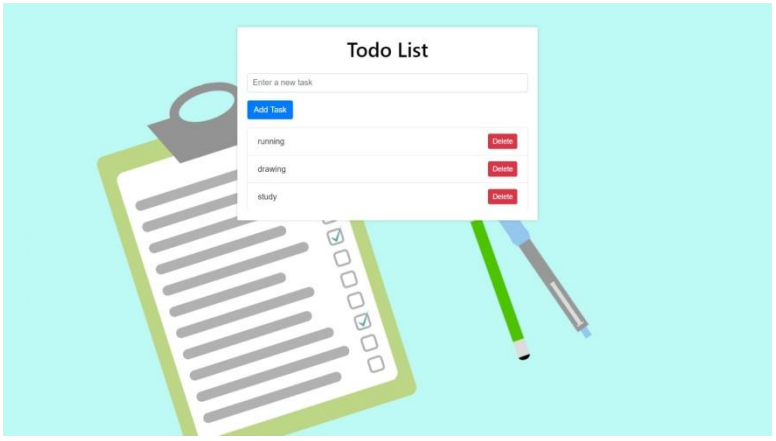
```
    margin-bottom: 10px;
}
input[type="text"] {
    flex: 1;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
```

```
ul {
    list-style: none;
    padding: 0;
}
li {
    padding: 10px;
    border-bottom: 1px solid #ccc;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
li:last-child {
    border-bottom: none;
}
input[type="checkbox"] {
    margin-right: 10px;
}

.delete-btn {
```

```
    color: #cc0000;
    background: none;
    border: none;
    cursor: pointer;
    font-size: 16px;
    padding: 0;
    display: none;
}
li:hover .delete-btn {
    display: block;
}
```

Output : -




Thank You!