

**SnapTodo**

**A Minimalist Todo Alarm App**

**Team 13:**

* Abdelrahman Bahaa Eldin Hafez 20201374556
* Abdelrahman Mohamed Essmat 20201374559
* Ali Ashraf Ali Hassan 2103106
* Andrew Botros Ayad Abdelmalek 20201322493
* Deena Fathi Mohamed Fathi 20201447217
* Nour Yasser Hamdy 20201378052

**Project Idea:**

In the hustle and bustle of our daily lives, keeping track of tasks can be a challenge. Enter SnapTodo – a minimalist todo list app designed to simplify your routine. With a singular purpose in mind, SnapTodo allows users to effortlessly save todos and set alarms, creating a streamlined experience for those who prefer a no-frills approach to task management.

Effortless Todo Entry:

SnapTodo prides itself on simplicity. Adding todos is a straightforward process, allowing users to input their tasks quickly and efficiently. The app ensures that capturing your to-dos is a hassle-free experience, eliminating unnecessary complexities.

Quick Alarm Setting:

Recognizing the importance of timely reminders, SnapTodo excels in rapid alarm setting. Users can attach alarms to their todos with just a few taps, ensuring that crucial tasks are never overlooked. The focus here is on efficiency and ease of use.

Intuitive Interface:

Navigating SnapTodo is a delight thanks to its intuitive interface. The app presents a clean layout that emphasizes functionality. No unnecessary distractions – just a user-friendly design that facilitates the seamless management of todos and alarms.

Customizable Alarms:

Acknowledging that personal preferences matter, SnapTodo offers customizable alarms. Users can tailor alarm tones and vibrations to suit their individual styles, adding a touch of personalization to their task reminders.

Daily Overview:

Stay on top of your day with SnapTodo's daily overview feature. A quick glance provides a snapshot of all your todos and associated alarms, helping you plan and manage your time effectively. This feature ensures that you have a clear agenda for the day ahead.

**Sequence Diagram Section:**

The sequence diagram shows the interaction between a user, the Android system, and the database in a to-do list app.

User clicks "Get Started" button:

* Android system navigates to "TodoList" activity.
* Android system fetches to-do list from database.
* Android system updates list UI.

User clicks "+" button:

* Android system navigates to "AddTodo" activity.
* User enters to-do item.
* User clicks "Save" button.
* Android system adds to-do item to database.
* Android system navigates back to "TodoList" activity.
* Android system updates list UI.

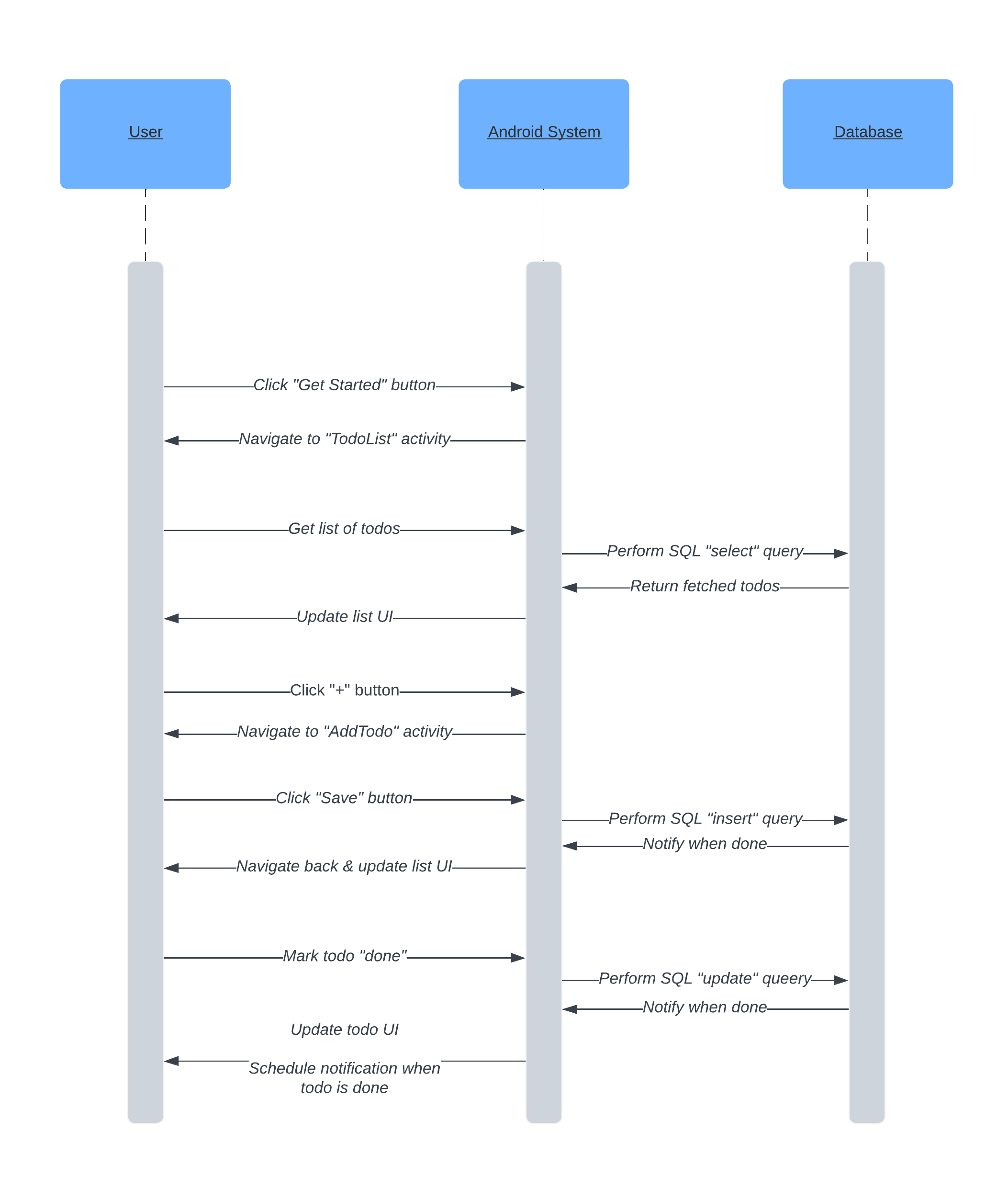
User marks to-do item as done:

* Android system updates to-do item in database.
* Android system updates to-do UI.

Android system schedules notification for to-do item:

* Android system sends notification when to-do item is due.

The sequence diagram shows that the user and the Android system interact with the database to perform various operations, such as getting a list of to-dos, adding a new to-do item, and marking a to-do item as done. The database notifies the Android system when each operation is done, so that the Android system can update the UI accordingly.

****

**Data Transfer Section:**

Welcome Screen Activity:

This activity launches only once when the application is opened for the first time ever, It then directs you to the main To-do list activity that contains all your current to-dos.

* This logic is handled by the SharedPreferences which basically allows you to save and require data in the form of keys and values and provides a simple method to read and write them
* Hence we make use of a “first\_launch” flag that is set to false when the application is firstly opened by the user to hide the welcome screen in later launchers.

To-do List Activity:

If it's not the first time opening the app, this should be the main activity which opens when launching the application.

* It then fetches the current to-dos set earlier from the “database” to be displayed for the user.
* It also updates the “database” about the state of the to-do item
* (Checked Or Unchecked).
* It contains a “add to-do” button that directs you to the add todo activity.

Add To-do Item Activity:

This activity is opened when attempting to create a new to-do item by clicking on the “add to-do” button.

* When adding a new to-do item, the “Save” button below then inserts and stores a new to-do item inside the “database” to be later fetched inside the to-do list activity.

To-do List View Model:

It controls the dataflow between the database and the to-do list main activity.

* Fetches the todos from the database and displaying them in the to-do list main activity.
* Transfers the updates of the to-do items state from the main activity to the database.

Add To-do Item View Model:

It controls the dataflow between the Add to-do item activity and the database.

* Transfers the names, dates and times of the new to-do items to the database to be stored there.

Database:

The database is basically a storage space that stores the to-do items being added by the user.

* It is fetched to display the to-do items of the user to the to-do list main activity.
* Keeps track of the to-do list item state being updated by the user via the to-do list main activity.
* Stores the new to-do items being created by the user via the add to-do items activity

A diagram of a company

Description automatically generated

**UML Section:**

The UML class diagram shows the classes and interfaces in a to-do list app.

Classes:

* Todo: Represents a to-do item. Has attributes for id, todo text, due date, and completion status.
* TodosRepository: Provides an abstraction layer for accessing the to-do list database.
* TodosDao: Defines the methods for interacting with the to-do list database.
* TodosDatabase: A Room database that stores the to-do list data.

Interfaces:

* TodosDao: Defines the methods for interacting with the to-do list database.

Relationships:

* TodosRepository has a composition relationship with TodosDao. This means that the TodosRepository owns the TodosDao instance and is responsible for its lifecycle.
* TodosRepository has a dependency on TodosDatabase. This means that the TodosRepository needs to be able to access the TodosDatabase in order to function.

The UML class diagram provides a high-level overview of the classes and interfaces in a to-do list app. It can be used to understand the relationships between the different components of the app and to design a well-structured and maintainable codebase.

A close-up of a computer screen

Description automatically generated

**Use-Case Section:**  
The use case diagram shows the different actors and use cases in a to-do list app. It provides a high-level overview of the different ways that a user can interact with a to-do list app. It also can be used to identify the different features that the app should have and to design a user interface that is easy to use and understand.

Actors:

* User: A person who uses the to-do list app to create, manage, and complete to-dos.

Use cases:

* Create a to-do list: The user creates a new to-do list.
* Add a to-do item to a to-do list: The user adds a new to-do item to an existing to-do list.
* Edit a to-do item in a to-do list: The user edits an existing to-do item in a to-do list.
* Mark a to-do item as done in a to-do list: The user marks an existing to-do item as done in a to-do list.
* Delete a to-do item from a to-do list: The user deletes an existing to-do item from a to-do list.
* View a to-do list: The user views a to-do list.
* Search for a to-do item: The user searches for a to-do item.

Relationships:

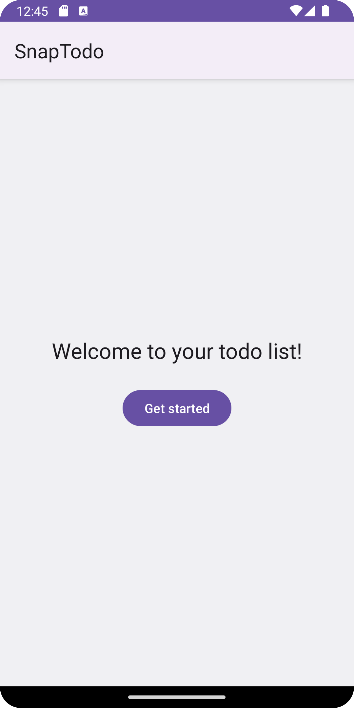
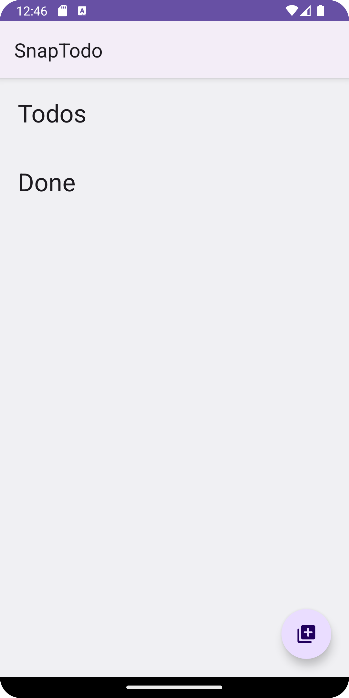
* A diagram of a to-do list

  Description automatically generatedThe User actor can interact with all of the use cases.

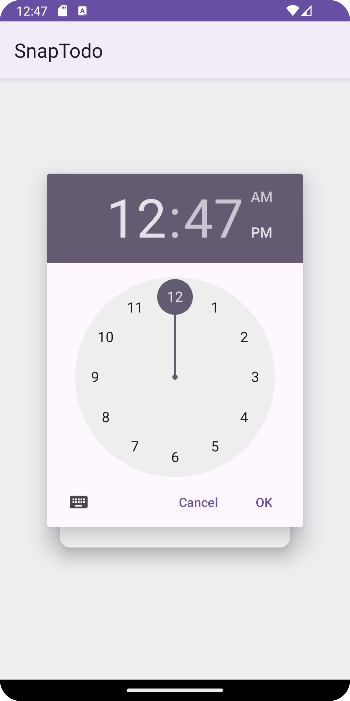
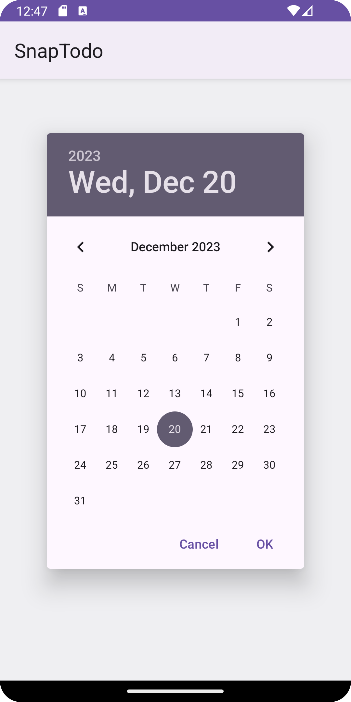
**Initial Layout Section:**

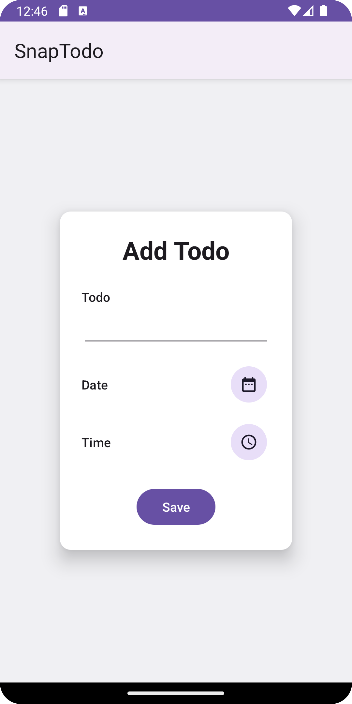
Welcome screen: Empty Todo Screen:

Shown only once when first time launch. Has FAB to add todo.

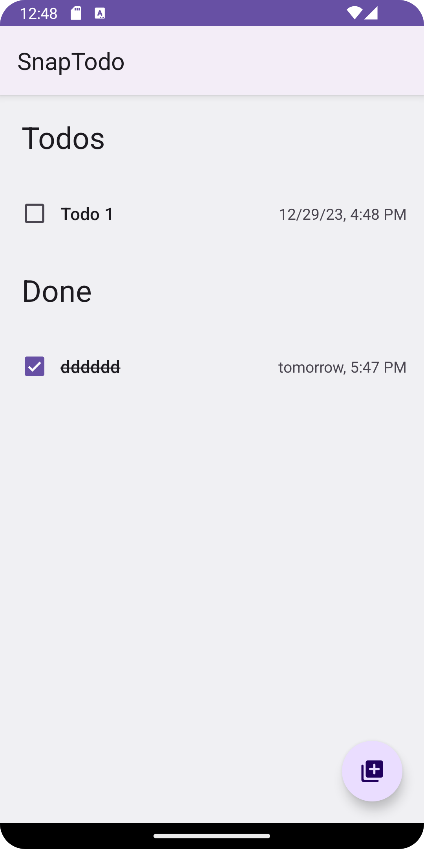


Add Todo Screen: When Pressing Date/Time:

When FAB is pressed redirected to it. Floating menu appears above.



Full Todo Screen:

This Is how it will look with todos in it, also they’re arranged by date and time.