# Planarr



**GitHub Username**: Plasius

## Description

Planarr is a productivity to-do list app that keeps things simple, empowering the user. Set up your daily schedule, plan every part of your day. Thanks to it's cloud based storage, you won't forget a task ever again.

## Intended User

The app is intended to be used by people looking to keep track of daily chores and tasks. The ideal user would keep all their plans for the future organized in the app, while not missing out on any recurring daily routines.

## Features

The app should:

- Have the ability to add, edit, complete and delete tasks
- Daily reminder via notifications about tasks (Use of Intentservice to check Database and build notification)
- Identify the user via Firebase Auth service

# Planarr

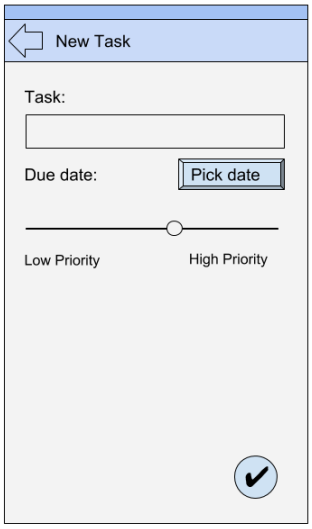
- Store tasks on the cloud (Firebase Realtime Database)
- Provide a widget showing the current tasks

## User Interface Mocks

### Screen 1 - TaskListActivity

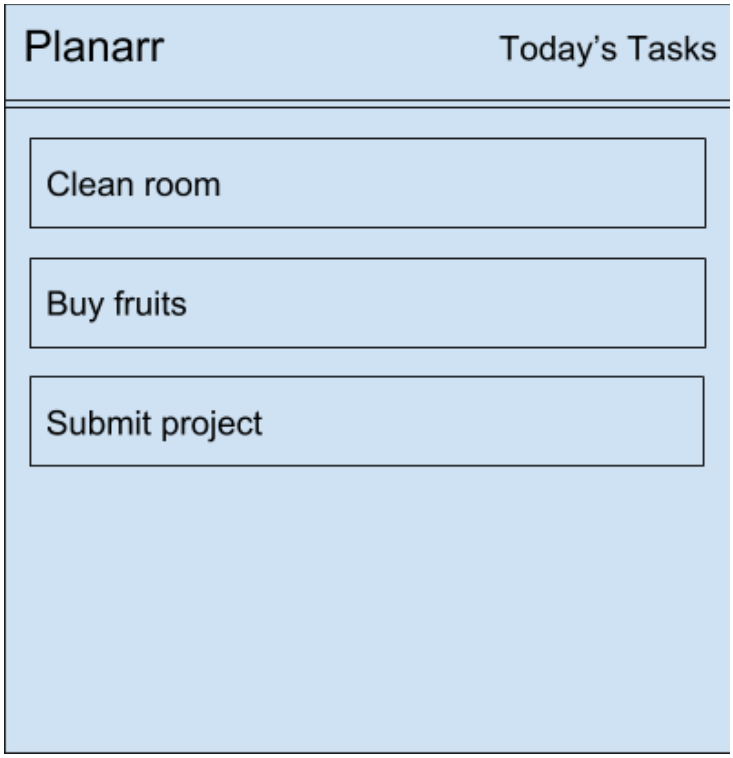

Main screen of the app representing the tasks that need to be done, ordered by priority.

### Screen 2 - TaskActivity




The activity to add new and edit existing tasks.

### Screen 3 - Widget




A resizable widget with a minimum size of 2x2

# Planarr

## Key Considerations

- application will be written solely in the Java Programming Language
- Andorid Studio (version 3.1.3), Gradle (version 4.9)
- Resources will be stored in their respective xml files in the values folder (colors.xml, strings.xml, styles.xml etc.)
- Careful attention to accesibility design guidelines

**How will your app handle data persistence?**

The app will use a Firebase Realtime Database to store the tasks, relying on its cache to store and save new tasks offline.

**Describe any edge or corner cases in the UX.**

Once the user creates a task the back button will loose it's functionality to go back to the the task creation activity. Same method will be used for the login activity, leaving the user to press the sign-out button to return to the log-in screen. After which no back button functionality will be provided.

**Describe any libraries you'll be using and share your reasoning for including them.**

ButterKnife (version 8.8.1)- to ease the EditTexts' access in UI heavy screens like the TaskActivity.

**Describe how you will implement Google Play Services or other external services.**

The 2 implemented services are:

- Firebase Realtime Database
- FireBase Auth

After the inital project setup in the console and integration in the project, the services will be implemented following the Google provided tutorials.

## Next Steps: Required Tasks

**Task 1: Project Setup**

- Set up Firebase Console
- Integrate SDK
- Enable Auth
- Enable Realtime Database

**Task 2: Implement UI for Each Activity and Fragment**

- Create LoginActivity with corresponding xml
- Create TaskListActivity with corresponding xml
- Create TaskActivity with corresponding xml
- Create item_task.xml item UI

**Task 3: Set up Auth**

- Implement Auth, using the real time database for debugging advantages
- Provide and implement a sign-out button in the TaskListActivity menu

**Task 4: Implement Task system**

- Create a “contract” class Task.java to represent the transmitted data.
- Provide storing and retrieving functions for the cloud based on UserID
- Code TaskActivity with the option to be launched for creating new and editing existing tasks
- Inflate the TaskListActivity with the new tasks
- Provide Delete functionality upon swiping

**Task 5: Add widget**

- Provide a home screen widget with similar functionality as the TaskListActivity’s list