

How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: “**Capstone_Stage1**”
3. Replace the text in green

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

GitHub Username: [ramyhelow](#)

My Daily Planner

Description

Daily Planner allows you to be more productive and organized in your everyday life. Quickly create tasks, plan new projects or just write something down in your notes. Thanks to the Firebase cloud sync, you can be productive on all of your android devices.

Intended User

People around age 18-40 who are focused on doing more every day and organizing their life.

Features

- Create tasks
- Add details to tasks
- Group tasks into projects
- Quickly filter your tasks for today or tomorrow
- Look through finished tasks in archive
- Write your thoughts in notes
- Sync your data across multiple android devices
- See your tasks for today on the home screen, thanks to the widget

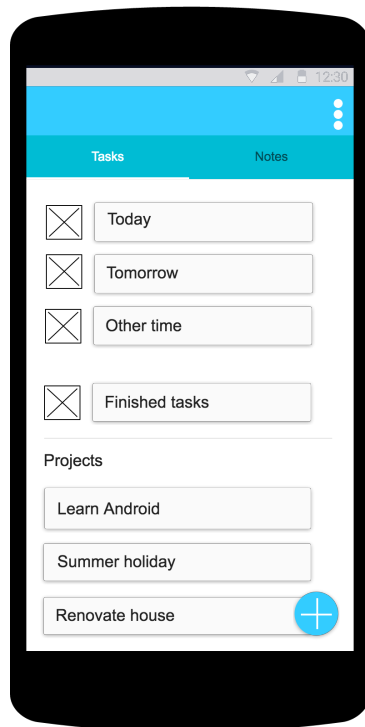
Requirements

- App is written solely in the Java Programming Language
- The app includes support for accessibility
- App keeps all strings in the strings.xml file.
- The app enables RTL layout switching on all layouts.
- The app uses AsyncTask to check for active internet connection.

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

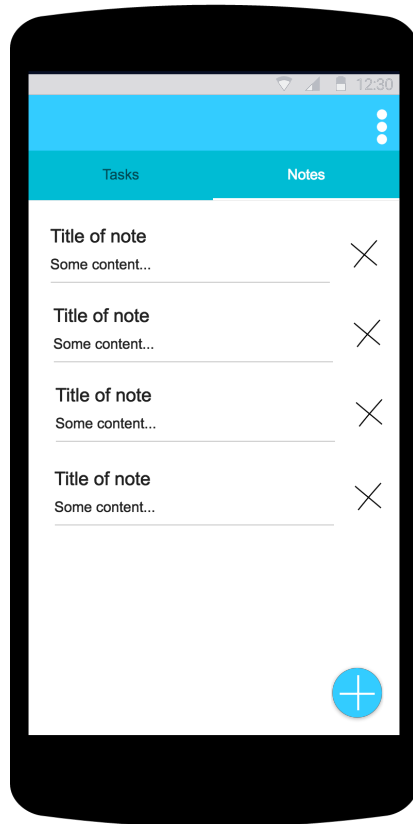
Tasks Screen



Main screen, displayed after user opens app

- Contains task categories, projects and FAB
- Task categories
 - Today: task due date = current date
 - Tomorrow: task due date = current date + 1 day
 - Other time: task due date = current date + 2 days and more OR no due date
 - Finished tasks: archive for tasks marked as “done”. User can permanently delete tasks that are here, if he wants
 - After clicking on task category, app displays a screen with contents of that category
- Projects
 - Project is a group of tasks which contribute to the same end goal
 - Once user creates a new project, he can add new or existing tasks into it
 - After clicking on project, app displays a screen with contents of that project
- FAB
 - Positioned in the bottom right corner
 - Allows user to create new tasks, projects or notes Settings
 - Positioned on the right side of action bar
 - Allows user to manually sync data
 - Allows user to logout

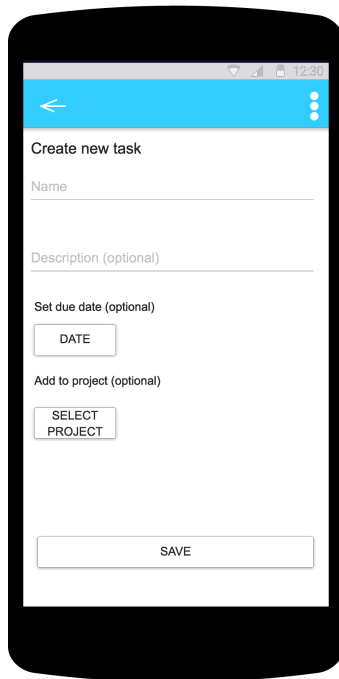
Notes Screen



Displayed after user selects Notes tab.

- Contains list of notes and FAB
- Each note has title on top and short part of note content under it
- On the right side of each note is delete icon
- FAB
 - Positioned in the bottom right corner
 - Allows user to create new tasks, projects or notes

Create Task Screen



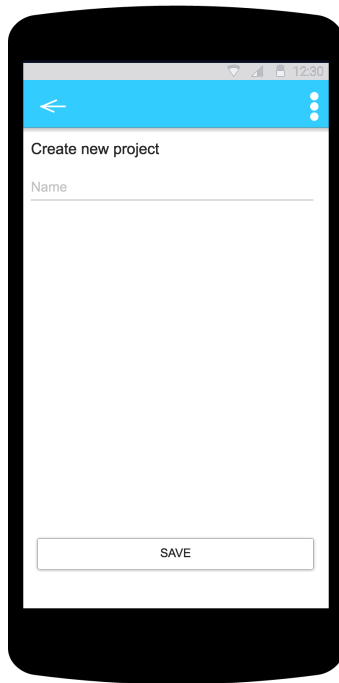
The image shows a mobile application interface for creating a new task. The screen has a white background with a blue header bar at the top containing a back arrow and a menu icon. The title 'Create new task' is displayed below the header. The form includes the following elements:

- Name:** A required text input field.
- Description (optional):** An optional text input field.
- Set due date (optional):** A label followed by a button labeled 'DATE'.
- Add to project (optional):** A label followed by a button labeled 'SELECT PROJECT'.
- SAVE:** A large button at the bottom of the form.

Displayed after user selects “Create new task” in FAB.

- Allows user to set details of new task
 - Name (required)
 - Description (optional)
 - Due date (optional)
 - Project (optional)

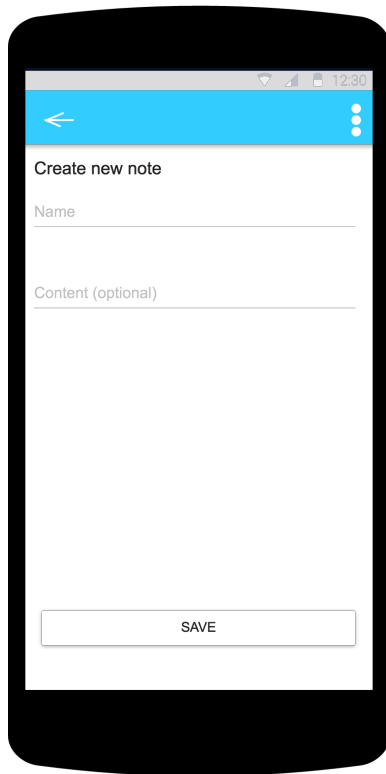
Create Project Screen

A mobile application interface for creating a new project. The screen is displayed within a black smartphone frame. At the top, there is a blue header bar with a white back arrow on the left and three white dots on the right. Below the header, the text 'Create new project' is centered. Underneath, there is a text input field with the placeholder text 'Name'. At the bottom of the screen, there is a white rectangular button with the text 'SAVE' in black capital letters.

Displayed after user selects “Create new project” in FAB.

- Allows user to set name of new project

Create Note Screen

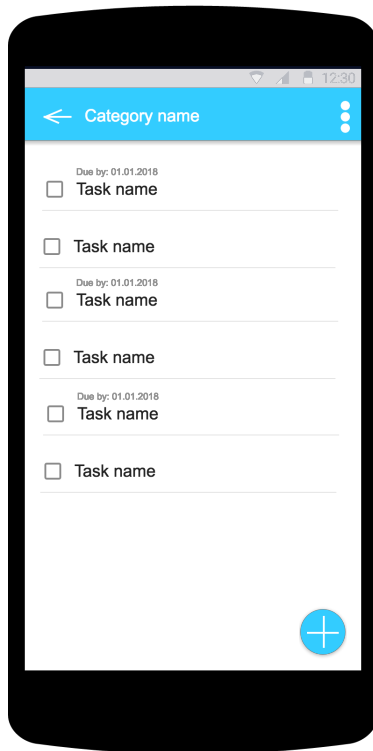


The image shows a mobile application interface for creating a new note. At the top, there is a blue header bar with a white back arrow on the left and three white dots on the right. Below the header, the title "Create new note" is displayed. Underneath the title, there are two input fields: "Name" and "Content (optional)". The "Content (optional)" field is a larger text area. At the bottom of the screen, there is a white button with the text "SAVE". The entire interface is framed by a black border representing the phone's bezel.

Displayed after user selects “Create new note” in FAB.

- Allows user to set name and content of new note

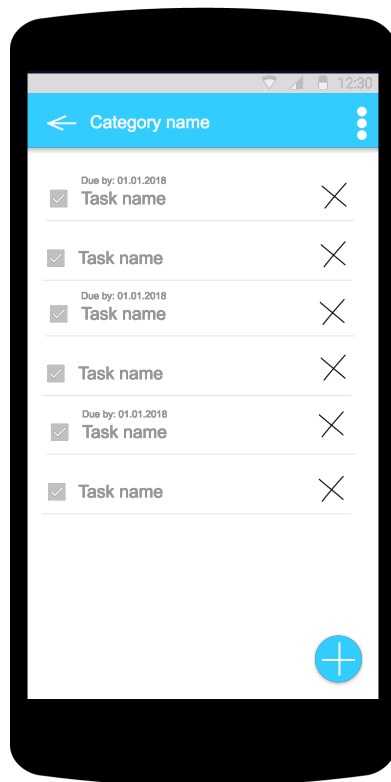
Normal Category Screen



Displayed after user opens any normal task category – Today, Tomorrow or Other time

- Each task has checkbox and name
- If task has due date, it will be displayed on top
- if user marks task as done, it will be hidden from this list and moved to Finished tasks category
- FAB
 - Positioned in the bottom right corner
 - Allows user to create new tasks, projects or notes

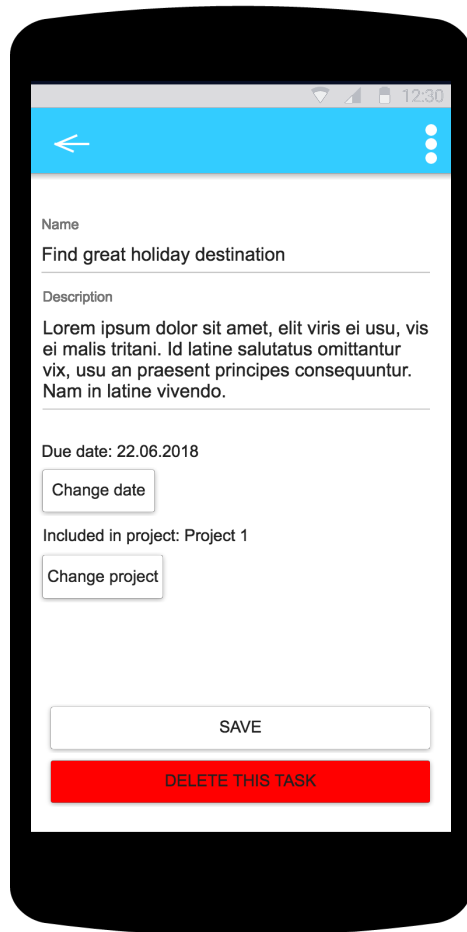
Finished Tasks Category Screen



Displayed after user opens Finished tasks category

- Allows user to see all completed tasks
- By clicking on checkbox, user has option to mark task as not done
- Each task has delete button, which permanently deletes it from DB
- FAB
 - Positioned in the bottom right corner
 - Allows user to create new tasks, projects or notes

Task Details Screen



The image shows a mobile app screen for task details. At the top is a blue header bar with a white back arrow on the left and three white dots on the right. Below the header, the screen is white. It contains several sections: a 'Name' section with the text 'Find great holiday destination'; a 'Description' section with a paragraph of Lorem Ipsum text; a 'Due date' section showing '22.06.2018' and a 'Change date' button; an 'Included in project' section showing 'Project 1' and a 'Change project' button; a 'SAVE' button; and a red 'DELETE THIS TASK' button at the bottom.

12:30

<

...

Name

Find great holiday destination

Description

Lorem ipsum dolor sit amet, elit viris ei usu, vis ei malis tritani. Id latine salutatus omittantur vix, usu an praesent principes consequuntur. Nam in latine vivendo.

Due date: 22.06.2018

Change date

Included in project: Project 1

Change project

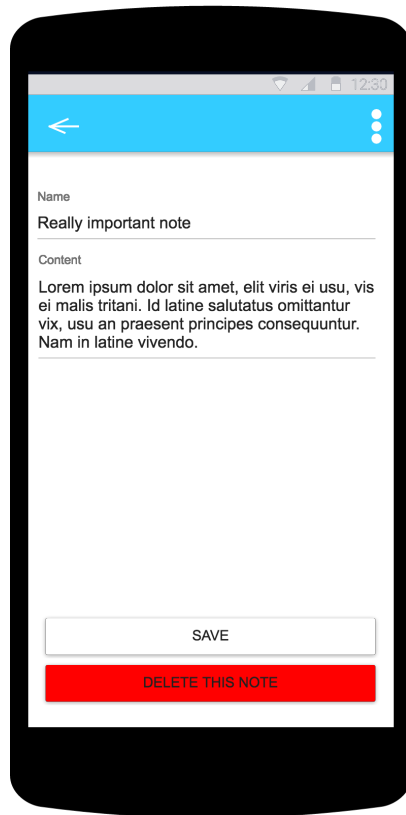
SAVE

DELETE THIS TASK

Displayed after user opens any task

- Allows user to edit all details of selected task
 - Name
 - Description
 - Due date
 - Project
- Delete this task button will permanently delete task from DB

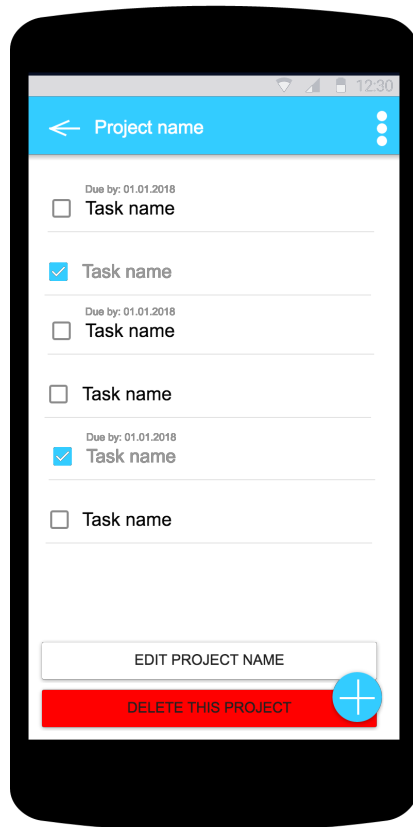
Note Details Screen



Displayed after user opens any note

- Allows user to edit all details of selected note
 - Name
 - Content
- Delete this note button will permanently delete note from DB

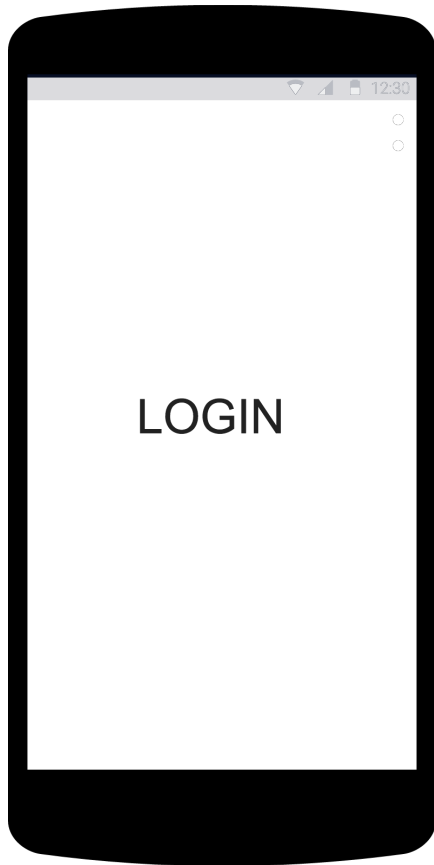
Project Screen



Displayed after user opens any project

- Each task has checkbox and name
- If task has a due date, it will be displayed on top
- All tasks in selected project are displayed, not just unfinished ones
- Edit project name button allows user to change the name of selected project
- Delete this project button will permanently delete project from DB
- FAB
 - Positioned in the bottom right corner
 - Allows user to create new tasks, projects or notes

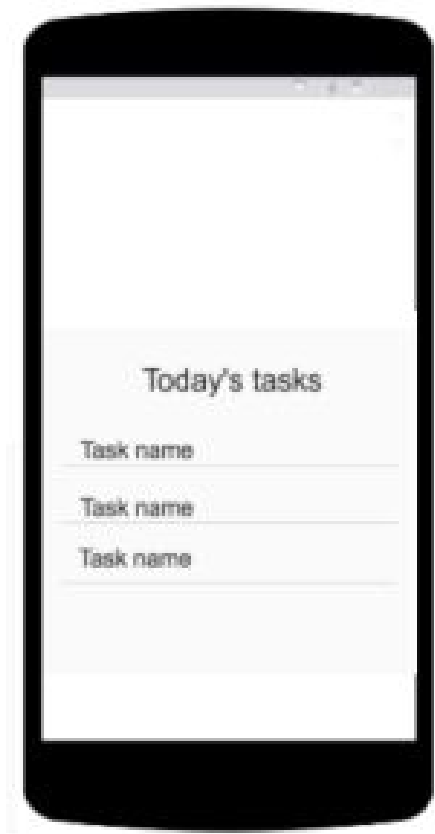
Login Screen



Displayed after user open app for the first time or logs out

- Allows user to login with Google Account
- Allows Anonymous login for code reviewer

Widget



Widget will allow user to displays tasks for today on his home screen.

Key Considerations

How will your app handle data persistence?

App will use Firebase Realtime Database

Describe any edge or corner cases in the UX.

If user deletes project that contains tasks, the tasks will stay in the app and won't be deleted. Logic behind this is that some users may simply want to get rid of the project and not lose all tasks associated with it.

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

Gradle Version 3.4.2

Android Gradle Plugin Version 3.4.2

Gradle Version 5.1.1

ButterKnife for better view binding

Affolestad Material Dialogs for various material dialogs

Firestore which allows you to quickly connect common UI elements to Firebase APIs.

RTLViewPager which allows rtl viewpager support

```
implementation 'androidx.appcompat:appcompat:1.1.0'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'androidx.test.runner:1.2.0'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
implementation 'com.google.android.material:material:1.0.0'
implementation 'com.google.code.gson:gson:2.8.5'
implementation 'com.google.firebase:firebase-core:17.2.0'
implementation 'com.google.firebase:firebase-database:19.2.0'
implementation 'com.google.firebase:firebase-analytics:17.2.0'
implementation 'com.firebaseui:firebase-ui-database:6.0.2'
implementation 'com.firebaseui:firebase-ui-auth:6.0.2'
implementation 'com.jakewharton:butterknife:10.2.0'
annotationProcessor 'com.jakewharton:butterknife-compiler:10.2.0'
implementation 'com.google.android.material:material:1.0.0'
implementation 'com.afollestad.material-dialogs:core:0.9.6.0'
```

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

Firebase Auth for user authentication

Firebase Realtime Database for saving data

Firebase Analytics which provides insight on app usage and user engagement

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Create a new project
- Set the SDK details,
- Configure libraries

Task 2: Implement UI for Each Activity and Fragment

- Build UI for main screen fragments – Tasks and Notes
- Build UI for task categories
- Build UI for new task screen
- Build UI for new note screen
- Build UI for new project screen
- Build UI for task details activity
- Build UI for note details activity
- Build UI for project details activity

Task 3: Implement Firebase Realtime Database

- Connect app to Firebase
- Add Firebase Realtime Database to app-level build.gradle
- Configure Firebase Database Rules
- Write to database
- Read from database

Task 4: Implement Firebase Auth

- Set up sign-in through google account
- Customize the sign-in UI

Task 5: Create widget that displays tasks for today

- Build UI for widget activity
- Implement Widget refresh functionality

Task 6: Implement UI for tablets

- Build tablet UI for main screen fragments – Tasks and Notes

- Build tablet UI for task categories
- Build tablet UI for new task screen
- Build tablet UI for new note screen
- Build tablet UI for new project screen
- Build tablet UI for task details activity
- Build tablet UI for note details activity
- Build tablet UI for project details activity

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
 - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"