Project Name

EisenFlow

GitHub Username

simbelmine

Description

EisenFlow it's a great and easy way to track your day to day life. To stay focused. To accomplish your goals.

As Dwight Eisenhower once said, "*Plans are nothing; planning is everything*".

Let's start planning and forget for all these long sleepless nights and all that stress.

To decide which of our tasks is important, or urgent, or combination of bought we can simply use the Urgent-Important Matrix created by Eisenhower.

Urgent - Important:

- These tasks require our immediate attention.
- They help us in our long-term goals.
- Crises, problems, or deadlines.

Not Urgent - Important:

- These tasks don't have deadlines.
- They help us to achieve our goals and fulfill our overall mission.
- Planning for feature tasks, improving yourself, or strengthening relationships.

Urgent - Not Important:

- These tasks need our immediate attention but don't help us achieve our goals and mission.
- Usually, these tasks are interruptions from other people and help them to fulfill their own priorities.
- Phone calls, emails, or request from family/coworkers.

Not Urgent - Not Important:

- These tasks are distractions.
- They are not pressing nor do they help us achieve our long-term goals or fulfill our mission.
- TV, games, gambling, shopping sprees and etc.

Our day-to-day life has a lot of "noise", a lot of unprioritized tasks.
Your challenge today is to try and apply the Eisenhower matrix to as many aspects of your life as you can.

Just don't forget to ask yourself: "*Am I doing it because it's Important or because it's Urgent?*"

By spending more time on Not Urgent but Important tasks, you'll have more control over your life and you'll feel calmer.

Good luck!

Intended Users

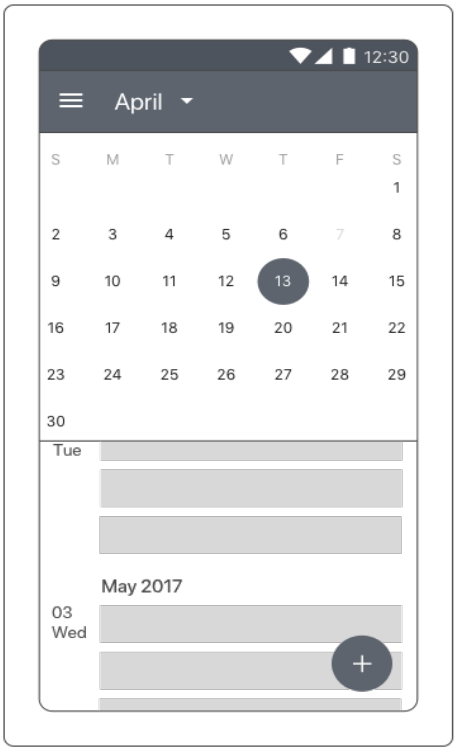
EisenFlow targets all groups of people. The app is a tool for organizing our day-to-day life.

Features

- Creates tasks.
- Organizes tasks by their priority, based on the Eisenhower's Priority Matrix.
- Smart reminders.
- List items swipe actions.

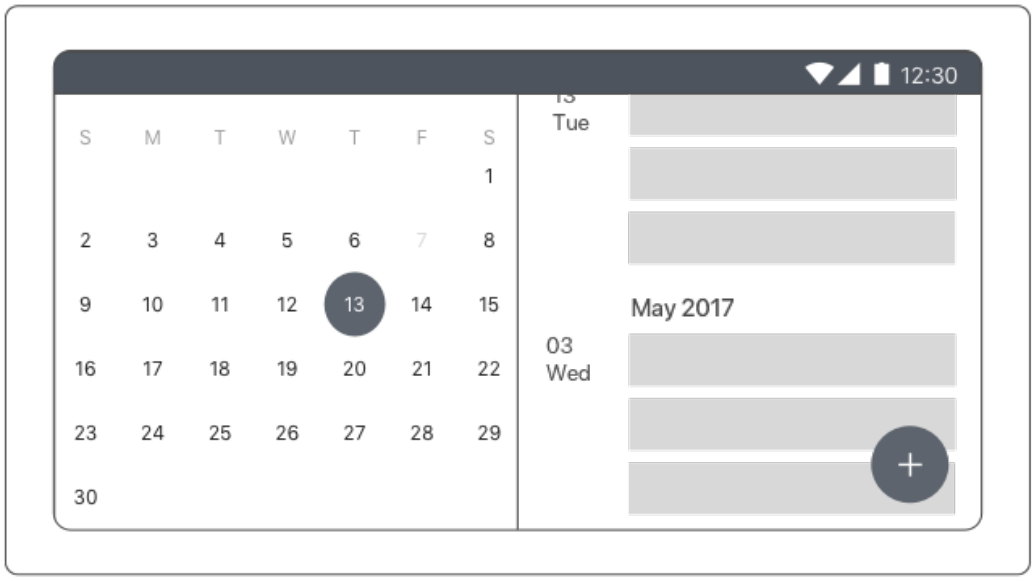
User Interface Mocks

Main Activity, Portrait

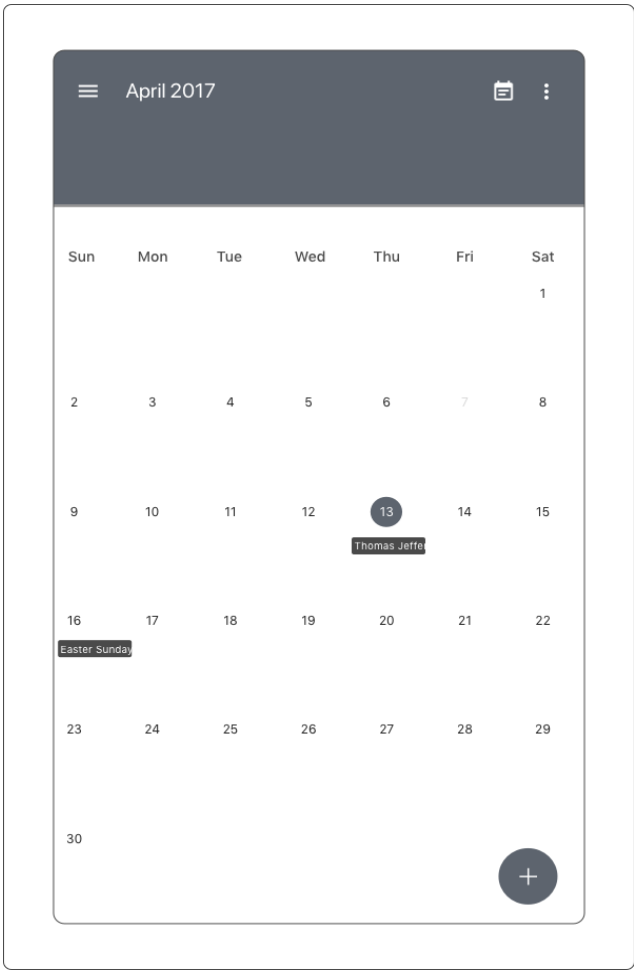


The main screen holds all the tasks for the specific days, and from the action bar a calendar view can be shown/hidden; each day of calendar view shows if it has tasks or not.

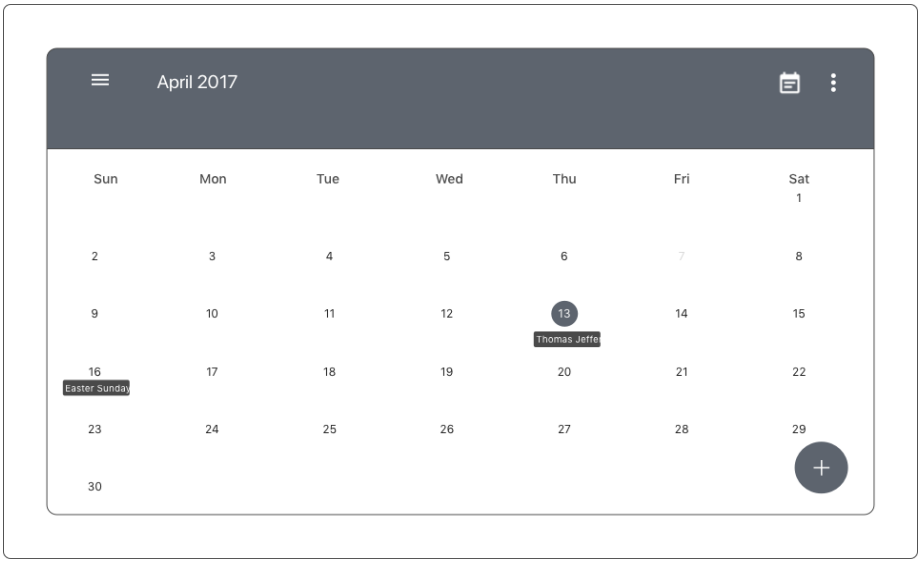
Main Activity, Landscape



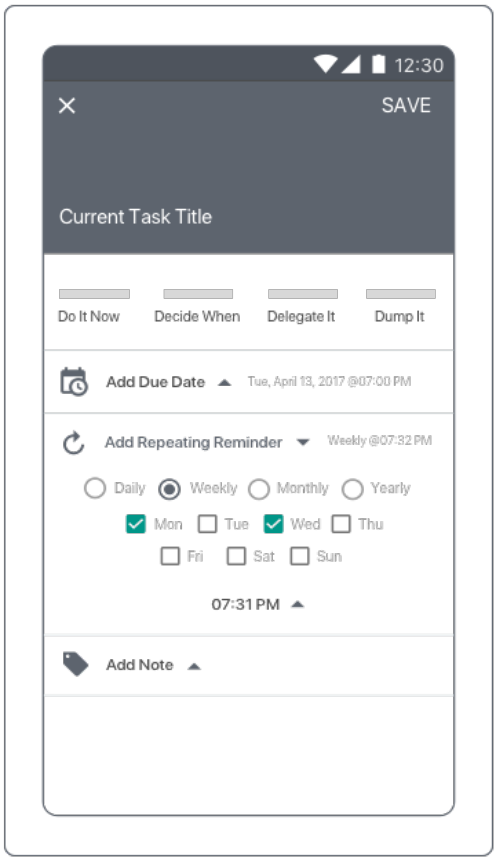
Main Activity, Portrait - Tablet



Main Activity, Landscape - Tablet

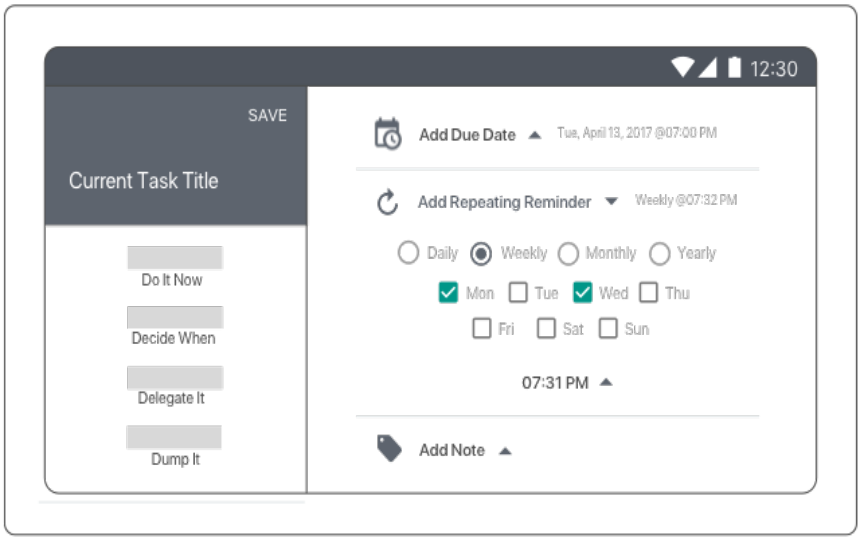


Details Activity, Portrait

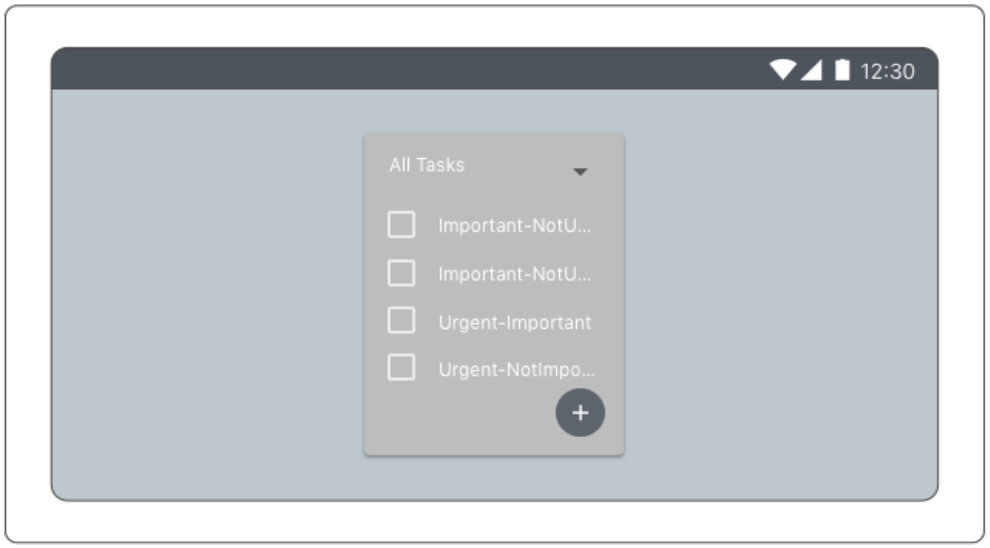


The details screen allows to create a single task by giving it a priority, due date, if it repeats or not, and notes; what can be set up changes according to the chosen priority.

Details Activity, Landscape



Widget



The widget will allow the user to have an easy access to the tasks; filter them or add new ones to list of tasks.

Key Considerations

Description of data persistence in the application.

In the initial stage of the application, SQLite database will be used for data persistence. Shared Preferences will be used in case of small data need to be saved for the life of the application.
In a later stage of the application, when is more complex, or/and switch to a remote database will be used ContentProvider.

Description of edge or corner cases in the UX.

- Press on the Back button on Main Screen should exit the app.
- Press on the Back button on Create New Task Screen should prompt a dialog "*Do you want to cancel? Yes/No*".
- Choosing date from calendar should position the Main Screen list of task to the correct date and corresponding tasks.
- Choosing a task from the Main Screen should open a preview of the task with the possibility of an edition.
- By choosing a filter the Main Screen list should show only the relevant tasks.
- List Actions should be able to perform the needed actions (Delete, Share, Start Timer, Undo).
- Timer Activity should have a corresponding action in the notification area of the Status bar and to be able to pause/resume timer.

Description of all libraries that will be used and the reason for including them.

- Material Calendar View (or similar): provides more functionality than a normal calendar view (material look, able to show tasks as small dots on the calendar, etc.).
- Joda-Time: provides more functionality than Java's Date and Time.
- Stetho: the Facebook library is a very convenient visual tool for the application's local SQLite database.

Description of how will be implemented the Google Play Services or other external services.

- Google Sign In: adding Sign Into Account in the non-MPV version of the app.
- Google Maps: adding a location to certain task in the non-MVP version of the app.
- Google Mobile Ads: adding ads to the application for the free version.

Next Steps: Required Tasks

This section is to describe the main features of the app (declared above) and break them down into tangible technical tasks that can be completed one at a time until the app is finished.

Task 1: Project Setup

- Create New Project by giving it a distinctive package name.
- Choose a target SDK version.
- Choose Basic Activity.
- Choose a name for the main activity and main layout file.
- After the project is created, add support libraries to the build.gradle file (appcompat-v7, cardview-v7, recyclerview-v7, design).

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for CardViews that are part of MainActivity list of tasks.
- Build UI for Create New Task.
- Build UI for Edit Task.
- Build UI for Start Timer for a certain task.

Task 3: Implement List of Tasks

- Add RecyclerView to the layout of MainActivity.
- Create RecyclerView Adapter
- Add Adapter to RecyclerView

Task 4: Add Calendar to MainActivity

- Add CalendarView from third-party library to the layout of MainActivity.
- Implement show/hide functionality.
- Show color-coded dots for the different tasks under each date (if any tasks).
- Implement click on a certain date to show all tasks for that date into the list of tasks.

Task 5: Implement Create New Task

- Show/hide views depending on the color code.
- Expand/collapse views on touch.

Task 6: Implement Edit Task

- Implement functionality to open task for editing.
- Return back to edit screen to show the updates.

Task 7: Implement Start Timer

- Implement Timer countdown.
- Pause/Resume or Stop Timer.
- Add interactive notification in Status bar to control timer if the application is in the background.

Task 8: Implement Widget

- Add scrollable ListView to widget that contains user's tasks.
- Create filter to change between the different priorities.
- Add “Add new task” ImageButton.

Additional Technical Tasks

Task 1: Create Widget

- Create AppWidgetProviderInfo to describe the metadata for the widget.
- Create AppWidgetProvider to define the basic methods that allow to programmatically interface with the widget, based on broadcast events.
- Add the widget to the AndroidManifest file.
- SetUp a preview image for the widget.

Task 2: Use Loaders and AsyncTasks

- For long background running tasks that need to keep the current progress of the task, is convenient to use AsyncTaskLoader.
- For short background running tasks that don't need too keep progress, AsyncTask is a good choice.

Task 3: Use ContentProvider

- Add an extra level of abstraction with ContentProvider as a good practice.
- Give the app the possibility to expose/hide the data easily.
- Design Content URIs
- Implement the ContentProvider.
- Implement the ContentProvider MIME Types.
- Implement the Contract.
- Implement the ContentProvider Permissions.
- Add the ContentProvider to the AndroidManifest file.

Task 4: Schedule repeating alarms with AlarmManager

- Set Up a RTC alarm.
- Define AlarmReceiver to handle the AlarmManager's broadcast about about sending local notifications at a given time.
- Add the receiver to the AndroidManifest file.
- Persist the alarms across device boots.

Task 5: Receive alarms

- Create BroadcastReceiver to handle incoming alarms.
- Create Wake IntentService to perform WakeLock to keep the device awake while doing it's job.
- Create Reminder Service to send a Notification to the user by extending the Wake IntentService.
- Call the Reminder Service from BroadcastReceiver's onReceive method. That way the service will keep running even if the BroadcastReceiver get killed.