GitHub Username: chatsgris

TaskAlert

Description

TaskAlert helps you to keep track and alert you of the errands you need to run around town at different venues. The app allows you to easily create a task, pin a location and forget about it. TaskAlert then sends out notifications to your phone when you are near any task's venue. Complete all your tasks now with TaskAlert!

Intended User

Everyone who needs reminder to complete their errands!

Features

List the main features of your app. For example:

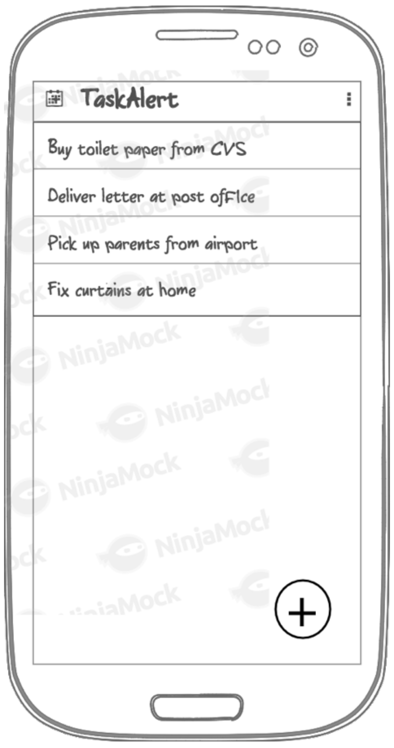
- UI to create tasks i.e. title, description and venue
- Use Places API to pin venue to tasks

- UI to display list of tasks
- Saves task information locally
- Periodically scans device location against tasks' venues
- Sends notifications when device location is near venues
- Notification opens up Maps and directs you from device location to task venues
- UI to delete tasks
- Removes task information from local database
- Widget to show list of tasks

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.

Screen 1: Main Activity



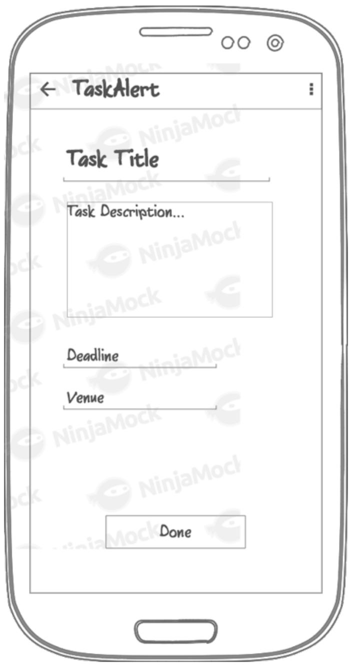
TaskAlert
Buy toilet paper from CVS
Deliver letter at post ofFIce
Pick up parents from airport
Fix curtains at home

Main activity shows a toolbar with app icon, app title and settings menu where users can choose accessibility options.

Also displays a list of tasks. User can click into each task for more details on description, deadline and venue, as well as if they want to delete tasks after completion. Directs to Screen 3.

FAB at the bottom right for user to add new task to the list. Directs to Screen 2.

Screen 2: Create New Task

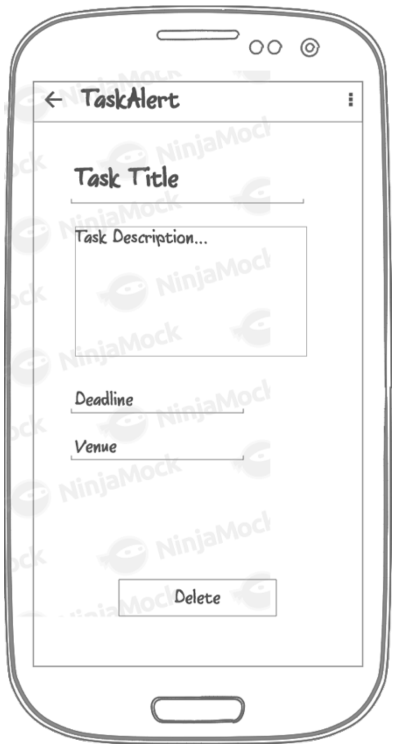


User enters information into edit-text boxes for title and description.

Choosing a venue directs you to the Place Picker API.

Clicking done directs you back to Screen 1 Main Activity. All fields must be completed or an error message will show.

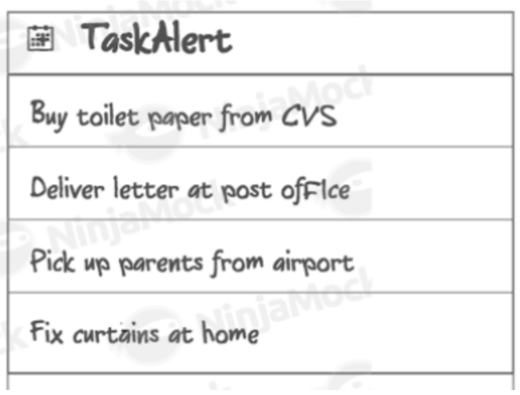
Screenshot 3: Task Detail Activity



Shows more information about each task like title, description/details and venue.

Also features a button to delete task. When clicked, user will be redirected back to main activity with deleted task gone.

Widget:


TaskAlert

Buy toilet paper from CVS

Deliver letter at post ofFIce

Pick up parents from airport

Fix curtains at home

Widget displays a list of current tasks. Clicking on any task will lead to Screen 2 Task Detail Activity.

Key Considerations

How will your app handle data persistence?

The app stores data locally using a Content Provider. The app uses a CursorLoader to load the date from the content provider on a worker thread instead of the UI thread. Each activity will use a loader by first requesting a loader through a LoaderManager then implementing the loader interface with all the loader callbackers i.e. onCreateLoader and onLoaderFinish etc. After the loader is initialized and the onCreateLoader triggered, pass the Content Provider into the method to access data. Next the data will be retrieved on a worker thread from the Content Provider. Once complete, onLoaderFinish is triggered where the retrieved data will be initialized in the UI of the app.

This process will be used on the Main Activity, Task Detail Activity and widget.

Database contains these columns: ID, title, description, created timestamp, venue longitude, venue latitude. Database will use SQLite.

Describe any edge or corner cases in the UX.

Corner case will be if the task has no deadline or venue. In this case, an error message appears and the user will be prompted to enter the most appropriate options i.e. a date very far into the future and/or a venue that’s work or home.

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

Butter Knife to inject views easily. Timber for logging if data is in wrong format so that the app does not crash. Probably a few other libraries out there depending on needs. I will be using stable versions of the above libraries.

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

Place Picker API for user to choose a venue of the task. Locations API to periodically scan task venues with device location. Maps API to direct user from device location to task venue.

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

- Configure libraries
- Set up content provider
- Set up Google Play services
- Set up dependencies
- Enables permissions
- App is written solely in the Java Programming Language
- Add suitable versions of all the libraries, Gradle and Android Studio

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Main Activity
- Build UI for Task Detail Activity
- Build UI for Create Task Activity
- Build layouts for different orientation and devices
- Build animations and transitions
- App keeps all strings in a strings.xml file
- Enables RTL layout switching on all layouts

Task 3: Implement Data Persistence

- Database contains these columns: ID, title, description, created timestamp, deadline timestamp, venue longitude, venue latitude
- Database will use SQLite
- Whenever user creates a new task from Screen 3, add this task information to our local database and display new task accordingly in Screen 1 Main Activity.
- Whenever user deletes a task from Screen 2, delete task from database.

Task 4: Creating New Tasks

- Add Place Picker API for venue selection
- Store both information respectively in the database in the appropriate format

Task 5: Pulls data from Locations API Every 60mins

- Regularly pulls data using Locations API every 60mins
- Updates data in its cache at regular intervals using a JobDispacter
- Send notifications when device location is within proximity to any task venue

Task 6: Send Notifications

- Send notifications whenever deadline is approaching or device location is near any task venue
- If deadline is approaching, clicking on notification directs to Screen 2 Task Detail Activity
- If task venue is nearby, clicking on notification directs to Maps using Maps API showing route to venue

Task 7: Add Widget

- Widget displays a list of tasks
- Clicking on any tasks will lead to Screen 2 Task Detail Activity

Task 8: Add Accessibility Support

- Add content descriptions
- Navigations using a D-pad

Task 9: Ready for Production

- Setup signing configuration