How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: SolviQorda

Traktive

Description

Want to get to grips with world events, on your terms? Traktive helps you filter stories from hundreds of different news sources, allowing you to track what's going on, beyond the daily news cycle. Actively pursue and engage with issues. Build cards that search for keywords that

you think are important. Bookmark any stories that you think will be important later on. Build a library of evidence and keep a diary of your thoughts.

Intended User

Researchers, community organisers, educators.

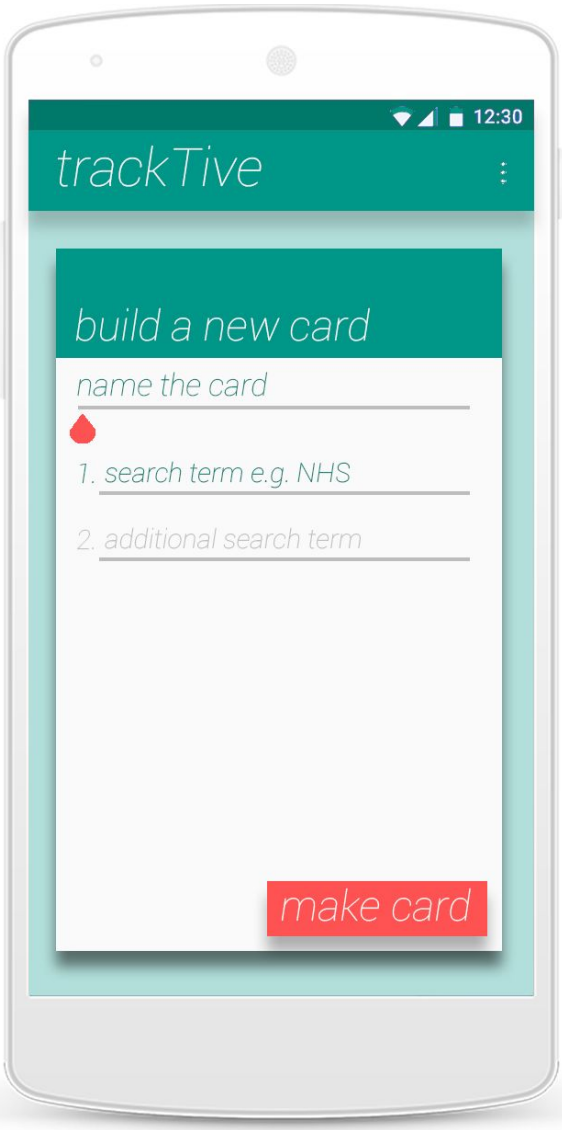
Working on some of the ideas raised in the Udacity Educational Technology course. I want to try and produce an app that encourages us to think about how everyday life is a learning process. The way that we consume news, with its emphasis on rolling coverage, often works against us taking time to think about longer trends, the 'bigger picture'.

Features

- The app allows for the custom creation of 'cards' that search for groups of words. E.g. migrants AND refugees AND mediterranean.
- The app uses the Event Registry API (eventregistry.org/documentation#searchArticles) to trawl through a large database of news sources to find articles that match the keywords, and returns them as a feed to the user.
- Users can bookmark articles that they think are relevant, which are then stored. The user can toggle between all stories, and only the stories that they have bookmarked via use of a button.
- Users can also swipe to remove anything they think is irrelevant.
- Diary feature: users can add date-stamped thoughts and notes to a card, to keep track of their understanding.

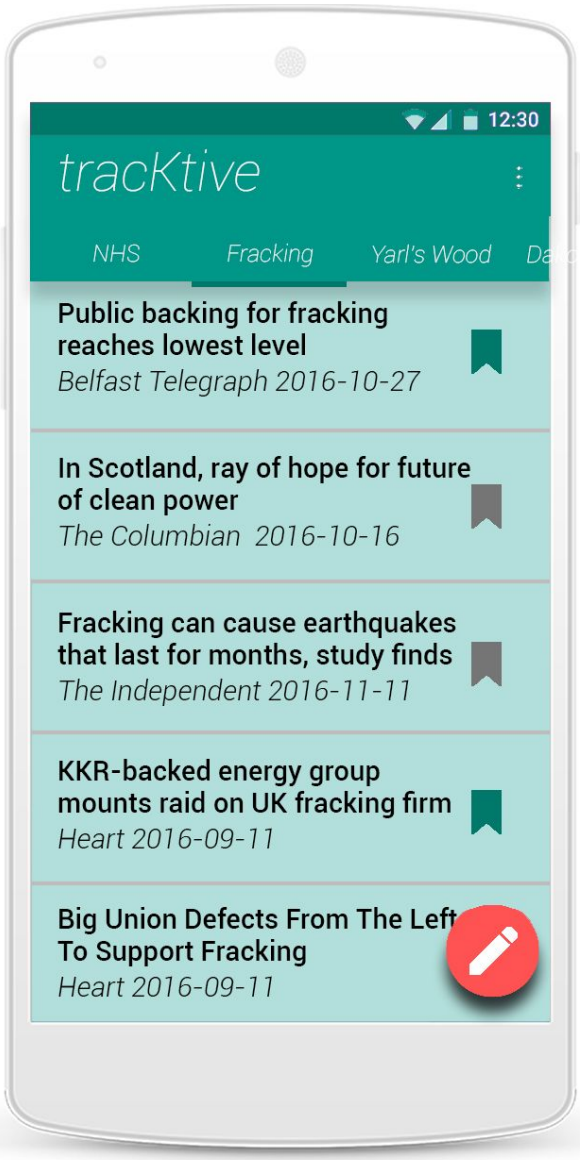
User Interface Mocks

Build a New Card Floating Dialog (opens on first App boot)



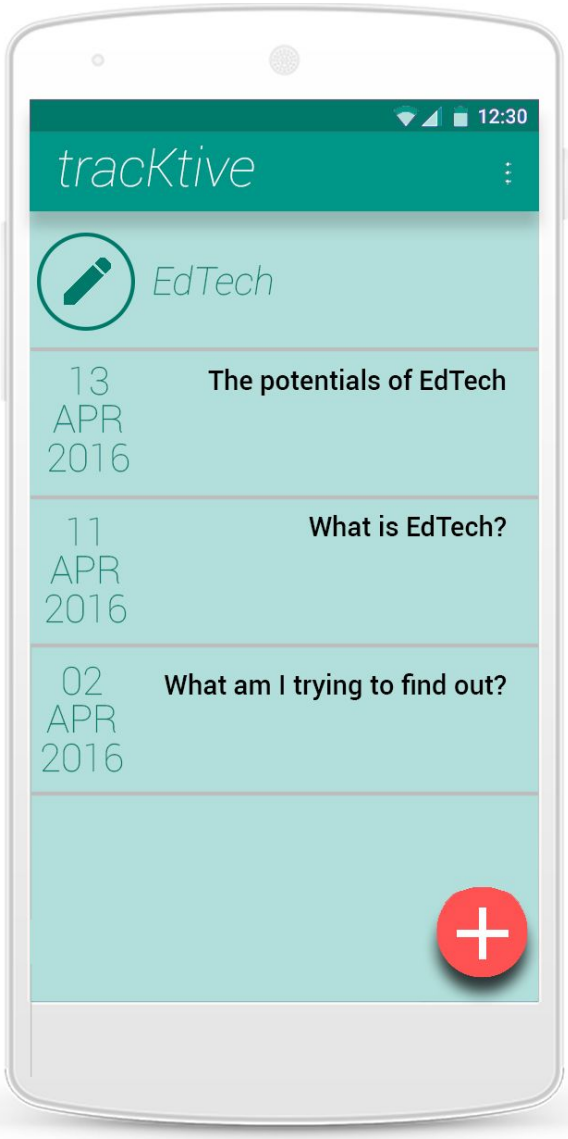
Dialog allows for the user to build a card around a particular topic, guiding the user to think of which keywords might be useful, with a clear accented button that guides the user to the final operation, making the card.

Main activity - Card Feed



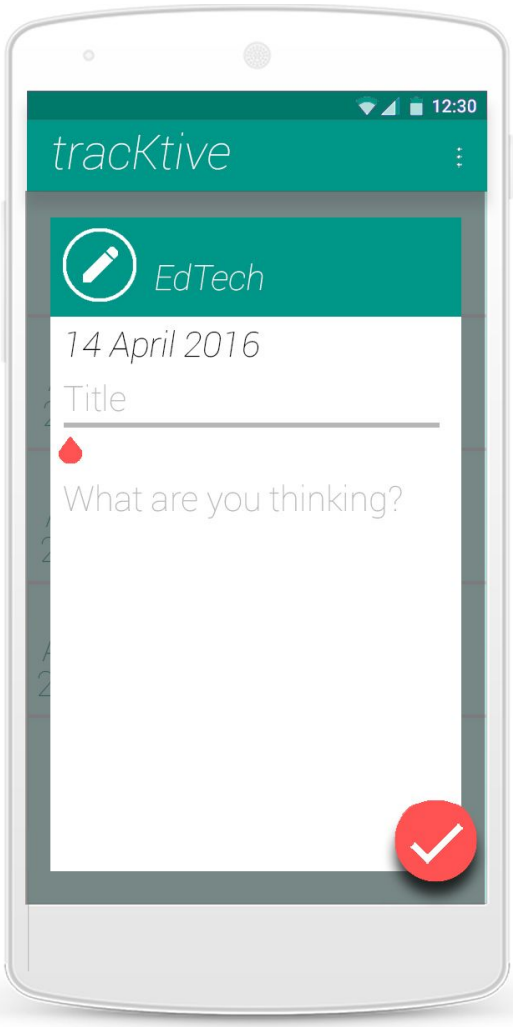
Tabbed displays split the user's different cards. Accented FAB points to the diary activity, which opens a list of text entries specific to the card. ScrollView allows for the user to browse through the chronologically ordered articles, whilst clicking on one of them opens a detail activity. Users can bookmark any article they think is relevant, and use the three-dots button on the action bar to change the order of the list to include only bookmarked articles.

Screen 3



Diary screen for one particular card, again a ScrollView. Pressing the FAB leads to a floating dialog where the user can write timestamped notes.

Screen 4

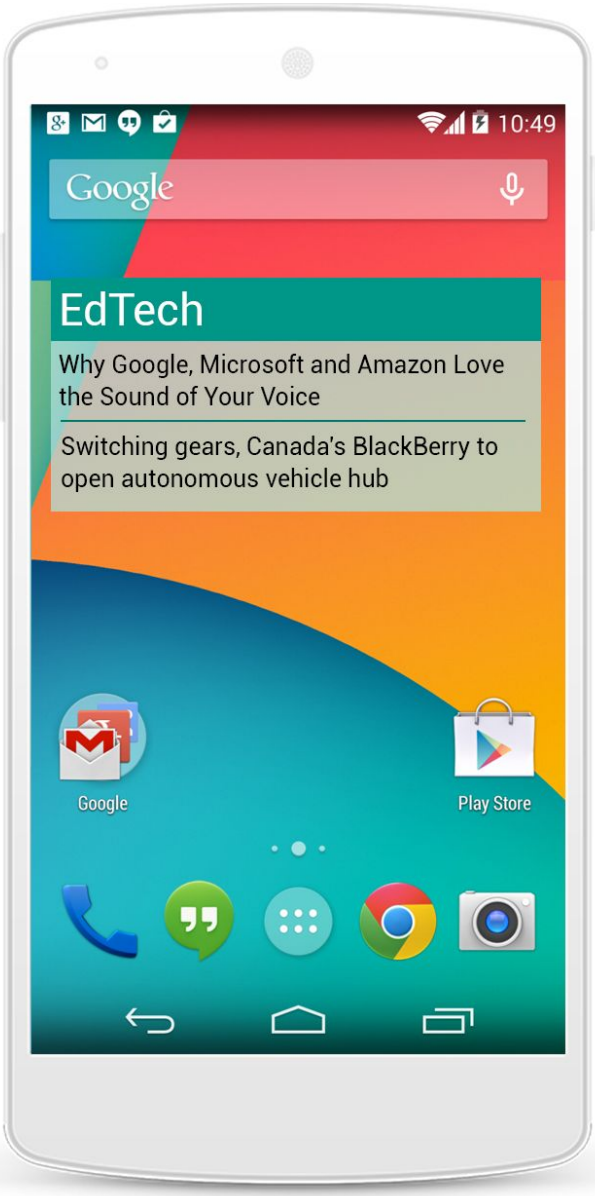


Floating Dialog for the diary. Large font encourages user to make an entry - box will fill up sooner, but can be scrolled if necessary. Compare this to the feeling we get writing a sentence that takes up a tiny part of a large white space. FAB directs the user to complete. If the user presses a pre-existing diary entry they should be able to access this page with the content filled from the pre-existing entry, ready to edit.

Screen 4

Detail view for a news story - browser icon indicates that the whole article can be read online, will use an implicit intent to handle the URL. Bookmark icon is by default in grey.

Screen 5



Widget - uses a ScrollView with adjustable height for the user to decide how much space they want the widget to take on their homescreen. User can determine which card they want to show in the widget in the Settings Activity.

Key Considerations

How will your app handle data persistence?

The app will use a SQLite DB and associated Helper class. To ensure effective handling of data the app will implement a content provider that deals with matching the URI to the user input.

In addition, a sync adapter will help to ensure that the app uses battery and radio efficiently, as well as keeping the user's cards updated.

Describe any corner cases in the UX.

There should be a way of handling accidental back button presses during text entry. If a user presses back they should be prompted to ask if they are sure - another potential would be that their entry gets saved even if they do press back.

If the user accesses an article through a detail view, when they press back from the browser, it would be optimal for them to return to that detail view.

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

I'm going to use OkHTTP to optimise the API calls.

Describe how you will implement Google Play Services.

Use analytics to think about how it could be a better learning experience - to see how often people are interacting, how often they are writing diary entries, how long are they spending on the app, and in which activity. If they use it in another country then it would be worth adding localisation options, rather than just the current English default.

App will use test ads, implementing AdMob.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Configure Analytics, AdMob and Picasso.
- Configure GitHub repo, with a basic readme explaining the purpose of the app.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity (the tabs of cards)
- Build UI for Detail Activity
- Build UI for add a card Floating Dialog
- Build UI for Diary activity
- Build UI for add/edit diary entry Floating Dialog
- Build UI for Settings Activity.
- Build alternative UIs for tablet which have floating dialogs as master/view layout - i.e. side by side.
- Generate drawable for icons.

Task 3: Build database and content provider classes

- Build DB helper class with SQLite. DB needs to have inner join with a DB for diary entries.
- Build content provider with URI matcher
- Build Contract class to determine relationship between data and view.
- Build Sync Adapter to make regular calls to API.

Task 4: Build view classes

- Build cursor adapter for the Card ScrollViews.
- Build cursor adapter for the diary entries.
- Build widget adapter class.

Task 5: Testing and finalising

- Build test classes.

- Configure gradle builds for release.
- Generate signed APK.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"