GitHub Username: tomca32

Capstone_Stage1

Description

HN Capstone allows you to browse, explore, filter, and read the latest Hacker News (Y-Combinator) content on the fly.

Intended User

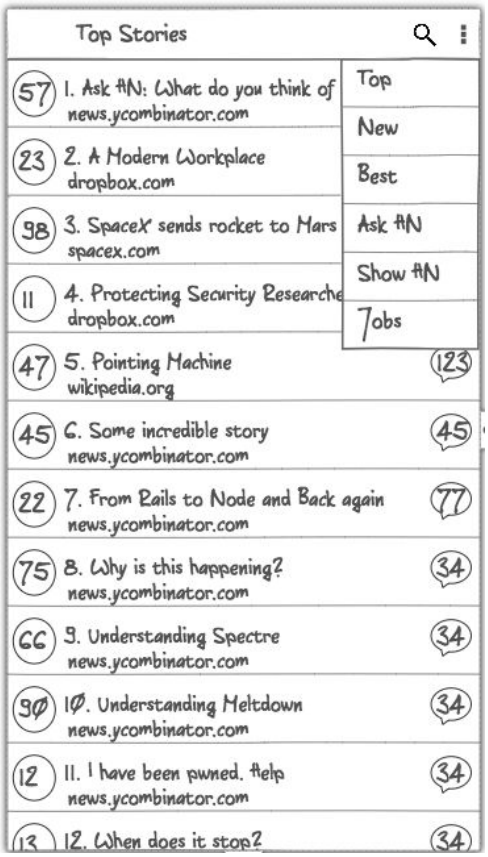
Hacker News users and readers.

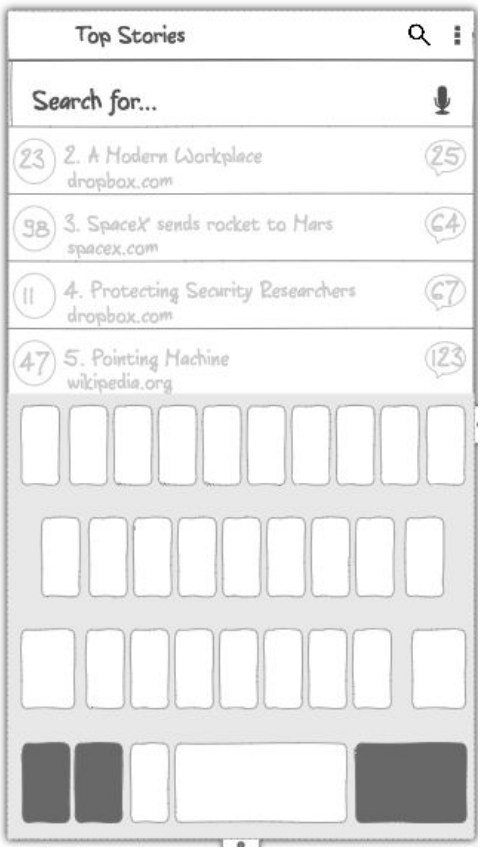
Features

- Browse and read Hacker News stories
- Sort stories by different criteria (Top/Trending, Best, New)
- Filter stories by categories (Ask HN, Show HN, Jobs - Who’s Hiring)
- Browse and read individual story comments

User Interface Mocks

Story List Screen



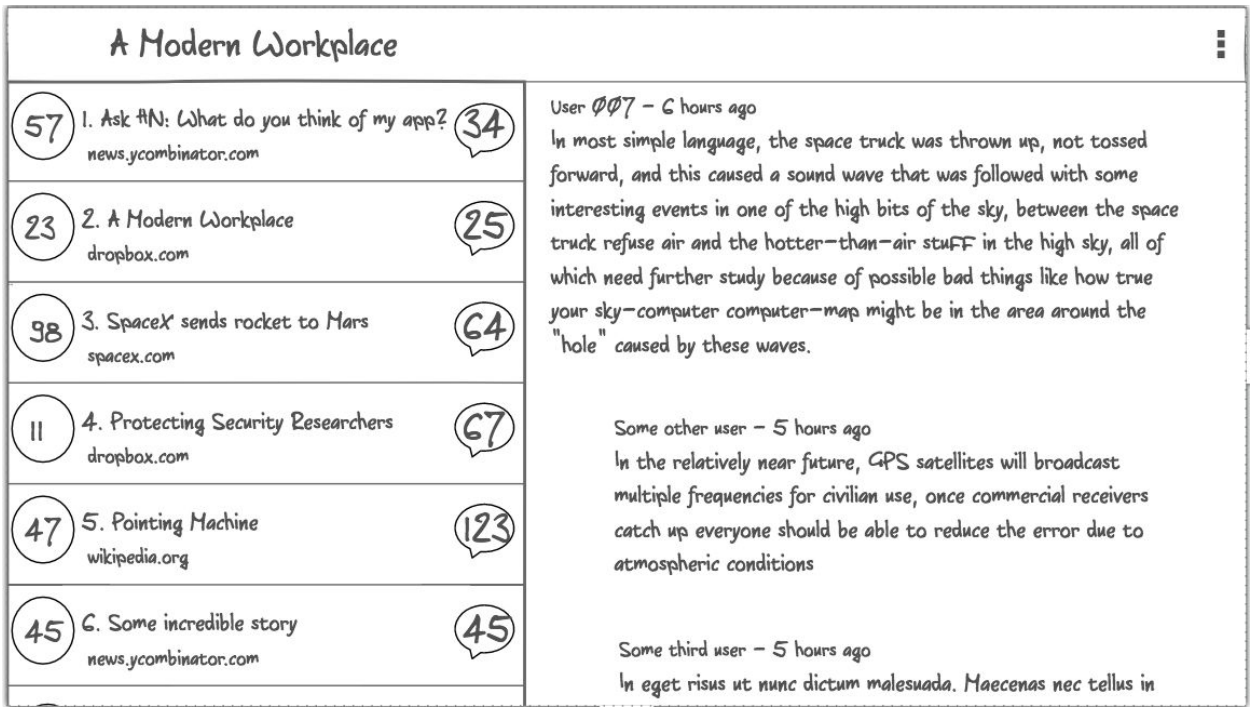


Story List Screen display a list of stories sorted by the selected sort/filter order, which can be changed through the menu. Stories have a title, number of points on the left hand side, number of comments on the right hand side, and a domain where the story is published.
Clicking the search icon brings the search input and opens the keyboard. Clicking on the story will open it in the browser (or whatever app can handle the URL), while clicking the comment count will open the comments screen. Stories that do not link to an external site (such as Ask HN) will be opened in the comments screen.

Comment Screen

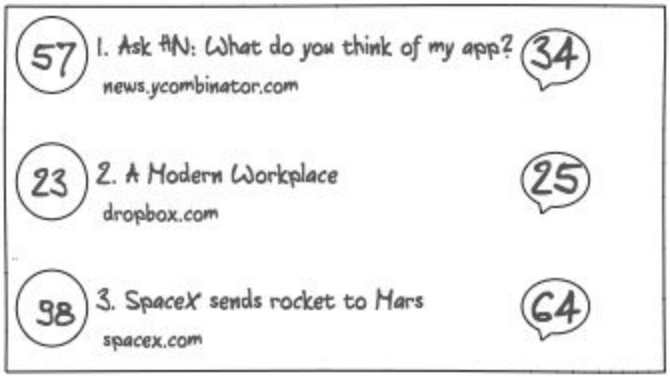
Comment Screen displays all the comments in a particular story. Child comments are visually nested under the parent, with all the siblings on the same level of indentation. Provided up button returns the user to the Story List Screen.

Story List and Comment Screen - Tablet



Tablet story list integrates the list and comments into a master-detail flow. Similar to the phone view, clicking the story list item opens up the link in a browser, and at the same time, update the comments in the detail fragment. Clicking only the comments count will update the comments and not open the link in the browser.
Menu contains the same sorts and filters as the phone version.

Widget



The widget is a 4x2 list of the current top stories. It will listen to the Firebase Realtime Database updates and update accordingly. The list in the widget works the same way as the list on the main stories screen. Clicking the item takes the user to the story, while clicking the comment bubble takes them to the comments screen.

The 4x2 dimensions are flexible. We should experiment with different widget sizes, such as 4x3 to see if it provides a better UX, and if so, we'll change the widget.

Key Considerations

How will your app handle data persistence?

HN Exposes and API through Firebase (https://github.com/HackerNews/API) so stories and comments will be handled with the Firebase Realtime Database.

Describe any edge or corner cases in the UX.

It might be better UX to open the stories in an integrated WebView instead of using an external application. This will be tested during development. Webview will be used if it provides a better, more seamless experience and makes more sense in the application's context. Perhaps it would make sense to provide a configuration option for this.

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

- Android Annotations, if needed, for project organization and reducing boilerplate code
- Firebase SDK for data store and retrieval
- Support and design support libraries for backwards compatibility

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

Besides the Firebase Realtime Database, the Crashlytics will also be integrated. This will allow us to detect, track, and troubleshoot the crashes that can potentially happen in the application.

Furthermore, the stories search will be done through the HN Algolia API (https://hn.algolia.com/api) - this call should be done on a background thread, probably via an AsyncTask or IntentService.

Next Steps: Required Tasks

Tasks listed here are defined from the user's point of view. No work is done unless it's absolutely necessary for the current task.

After each completed task, app will be fully functioning, just with a reduced feature set, until all the tasks are completed.

Task 1: Project Setup

This task involves creating a new project and creating a main activity.

- Create new project in Android Studio
- Include the basic libraries: Annotations and Support
- Create an empty Main Activity
- Ensure the application compiles and runs

Task 2: Story List Screen

This task involves setting up the Firebase Realtime Database, configuring it to fetch the stories, and displaying them in a list. Clicking on stories will display them in an appropriate application. No search functionality, or filtering, will be included in this task.

- Configure Firebase to fetch stories from HN API
- Build list layout according to the UI mock
- Ensure the application compiles, runs, displays a list of stories, and shows a correct story when clicked
- Ensure the list appears properly on all screens - tablets should just show an empty fragment in the comments pane, for now

Task 3: Comments Screen

This task involved building the comments screen, as well as optimizing it for larger screens. This will replace the empty fragment from the previous task on tablets with a list of comments.

- Building the comments screen according to the UI mocks
- Displaying the number of comments in the stories screen
- Fetching and displaying all the comments

Task 4: Sorting and Filtering

This task implements the sorting and filtering options displayed in the menu: Top(this is the default sort), Best, New, Ask HN, Show HN, and HN Jobs. This is implemented through the appropriate endpoints in the HN API - https://github.com/HackerNews/API

- Implement the menu with all the sort and filter options
- Retrieve and display the appropriate stories

Task 5: Search Functionality

This task includes calling the Algolia Search API, and displaying the results in a regular story list screen. This request is on demand, so it should probably be implemented with the IntentService.

- Implement calling the Search API with the inputed text
- Display stories from search results

Task 6: Widget

The main goal of this task is to implement the widget according to the UX mocks, and to make sure it updates whenever the Firebase Realtime Database updates. It might be worth it to experiment with different widget sizes to see if there's a better fit than 4x2.

- Implement a widget according to the UX mocks
- Update the widget whenever the database updates
- Experiment with different widget dimensions to see which one provides the best UX

Task 7: Polish

This task focuses on taking another look at the whole app and find ways to make it more usable. This is hard to predict as many UX issues are discovered only after the app has been built and used for a while; however, we can make some educated guesses

- Test displaying the stories in the WebView instead of using an external application
- Explore the possibility of using voice recognition in the search box
- Address UX issues that are certain to show up just from using the the application for a while
- Possibly explore additional widget sizes