StackDroid

Description

StackDroid is the application for most popular Q&A site for developers – http://stackoverflow.com/. Want to always be in touch, looking for new questions and exploring answers from mobile devices? Then this app is for your!

Intended User

Developers and other people bound to IT, which are used to search answers on StackOverflow.

Features

Application provides most popular and important feature of StackOverflow web-site, basically they are:

- Authorization and registration
- Users profile info
- Newest questions list
- Filtering questions by tags
- Adding tags to favourite
- Displaying question with answers in details
- User's answers list

Technical Requirements

This document doesn't fully describe all technical, UI and UX features of the app, but provides information about all application screens. Application must satisfy all requirements from this document. New features or changes for existing features can be applied with additional tasks. Every question about requirements which is not answered in this document should be managed by developer.

Requirement	Description
Devices	Android smartphones and tablets
System version	Minimum supported version is Android 4.1 (API 16)
Supported resolutions	All
Screen orientation	Portrait and Landscape
Offline	Application should store all information listed here: ● Questions List ● User personal information ● Favourite tags Caching includes all texts and images. When user opens app in offline he should see the last shown information.
Languages	English
Application design	Managed by developer. Should be implemented according to the principles of Material Design.
Syncing data	Application doesn't sync data periodically and should do network calls only when it is in use.
Error handling	Application should handle all network errors and try to fix them. If this is impossible and error does influence to the UX, application should show error using toast or dialog.
API	https://api.stackexchange.com/
Testing	Automated tests and tests by developer

Publishing	Application should be published by developer in Google Play

Application Screens

Authorization Screen

Due to the API limitations for non-authorized users application requires authorization. First screen is authorization screen which contains logo and button.

Authorization process is implemented via OAuth, so app shows StackExchange authorization page in browser and after that handles the response url.

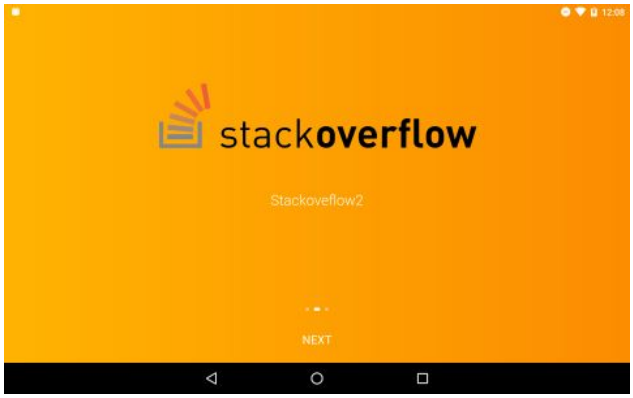
UI Element	Description	Actions
Logo	Stackoverflow logo	None
Button	Button with text “Sign in”	Clicking on button redirects user to OAuth page in browser

Walkthrough Screen

Tutorial screen which demonstrates key features of app to the user.

Phone Design

Tablet Design



Benefits are displayed in pager. Each item in pager has gradient background, image and text.

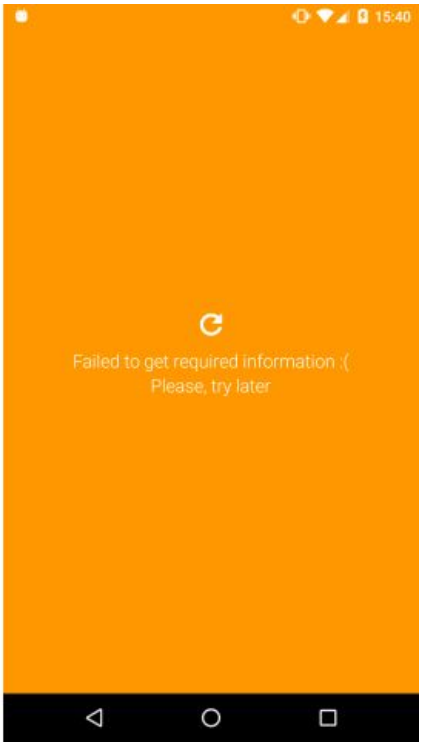
UI Element	Description	Actions
Paging indicator	Indicates current page	None explicit actions. Indicator updates correctly when current page changes.
Button at the screen bottom	Button with caps text Possible texts:	User can click this button. If current page is not last

	• FINISH – if current page is last • NEXT – if current page is not last	the next page smoothly appears. If current page is last we starting scenario described below.
Page 1	Contains icon which will be defined during development	Swipe to next page
Page 2	Contains icon which will be defined during development	Swipe to next and previous pages
Page 3	Contains icon which will be defined during development	Swipe to previous page

During user looking through the screen we load all needed information (user's info and questions). If we failed to load this data or user passed tutorial before requests we show the Splash screen or an error.

Splash Screen

Displaying an animation for loading progress and fetching information from the server.

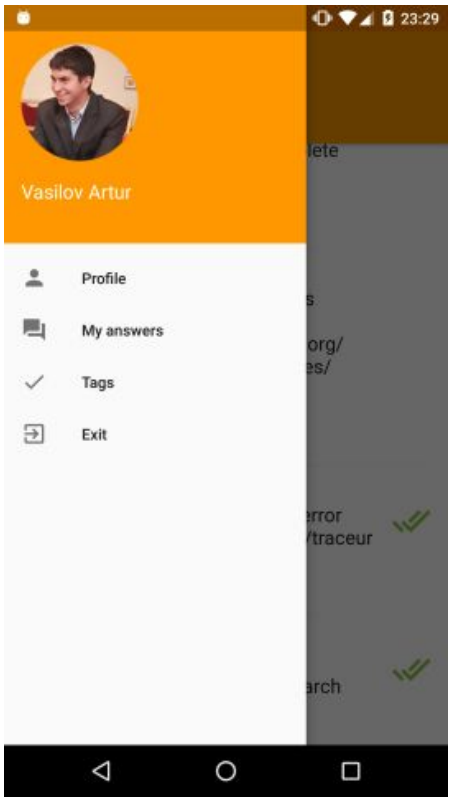


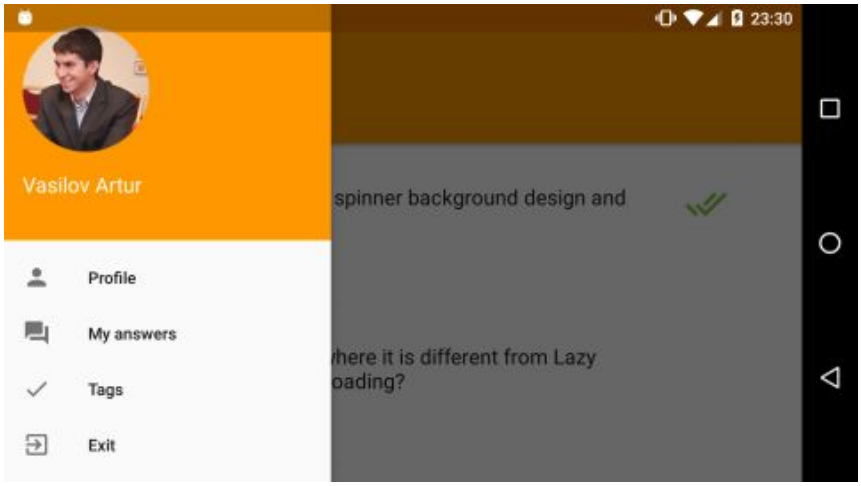
UI Element	Description	Actions
Animated Image	UI element to display progress of loading	None
Retry image	Retry icon	Click on button restarts loading process
Retry text	Text indicated error during loading information. Default text – “Failed to get required information, please, try later”.	None

Main Screen

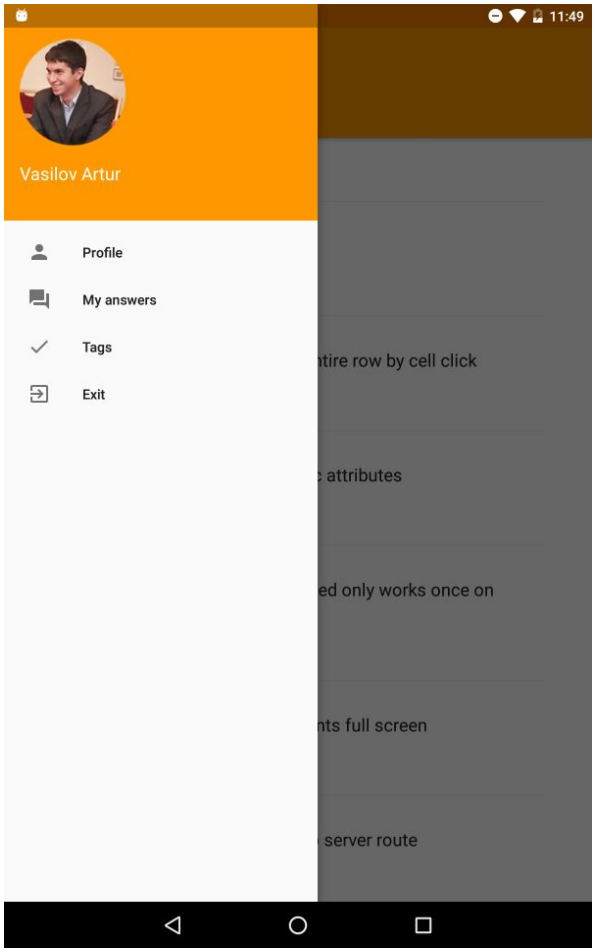
This screen goes after walkthrough. Contains toolbar with application title, Navigation Drawer and TabLayout for navigation and main content.

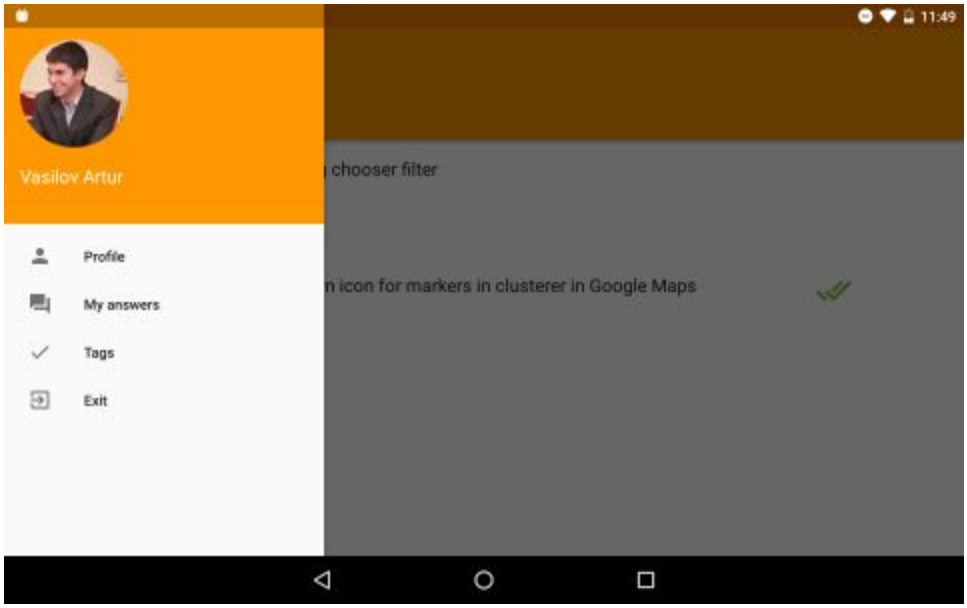
Phone design





Tablet design





Main content is displayed with pager. Each page is a list of most recent question with page's tag. First page contains all question, second – user's questions. User can add new tab by adding new favourite tag.

Navigation menu contains header with user information and action items which are listed below.

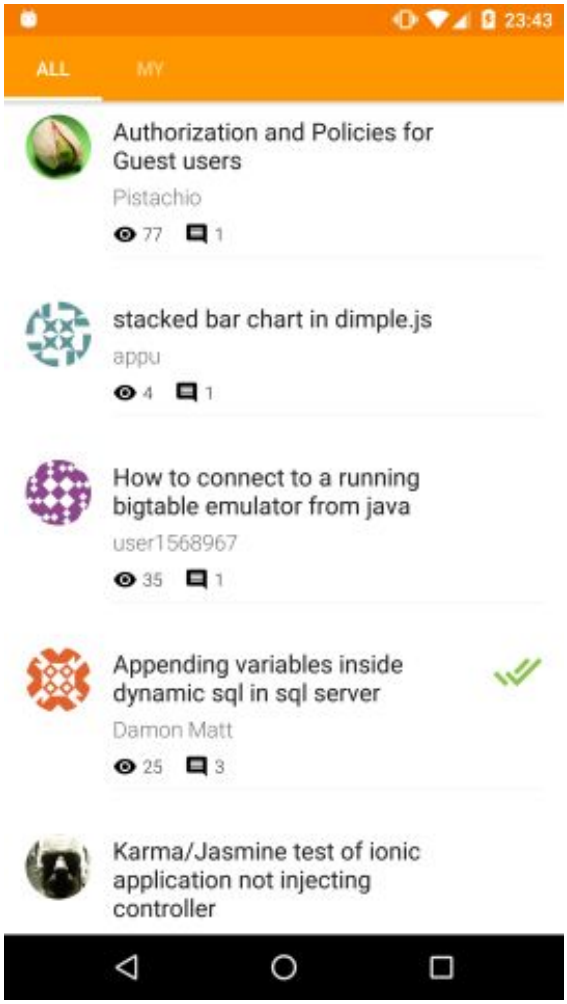
UI Element	Description	Actions
Toolbar title	Standard toolbar title with text "StackDroid"	Toolbar hides when user scrolling content up
Tab	Each tab in pager contains text which is tag title	Clicking on tab selects referred item in pager
Avatar icon	Displays user's profile image from site	None
Name text	Displays user's name from site	None
Profile navigation item	Navigation item with text "Profile"	Clicking on item leads to profile screen
My answers item	Navigation item with text "My answers"	Clicking on item leads to the answers screen

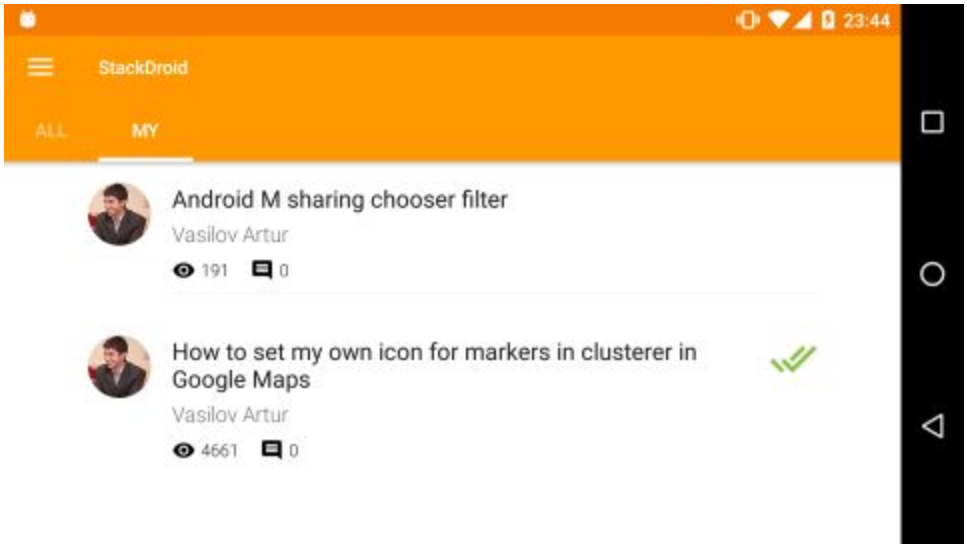
Tags	Navigation item with text “Tags”	Clicking on item leads to the tags screen
Exit	Navigation item with text “Exit”	Clicking on item clears user data and shows an authorization screen

Questions Screen

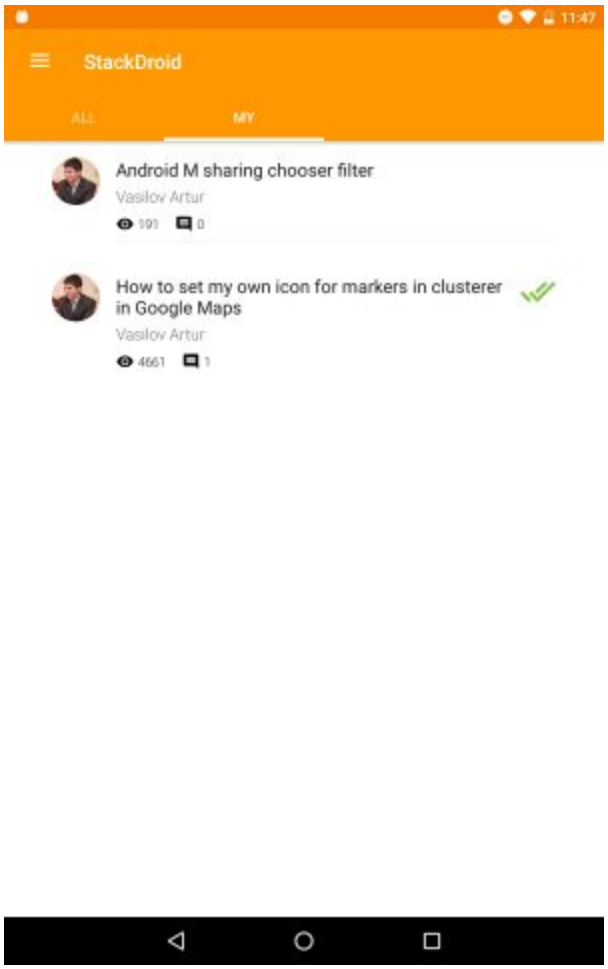
Contains a list of questions. List may be scrolled endless and new questions should be loaded. During the loading progress application shows progress indicator.

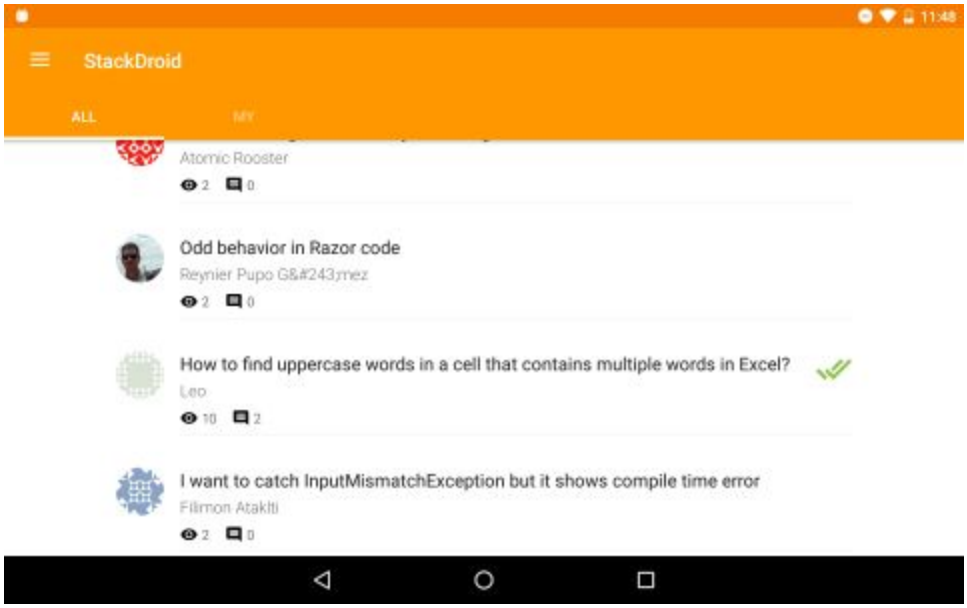
Phone design





Tablet design



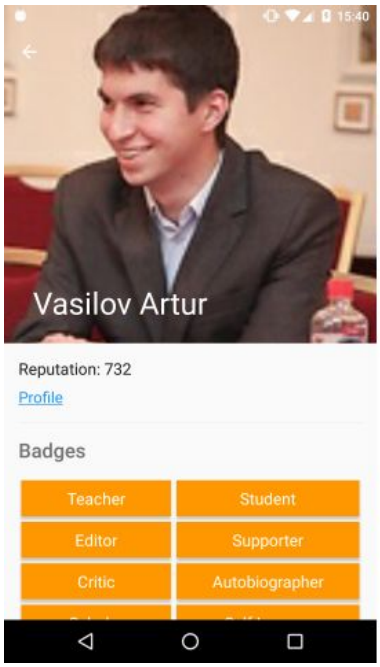


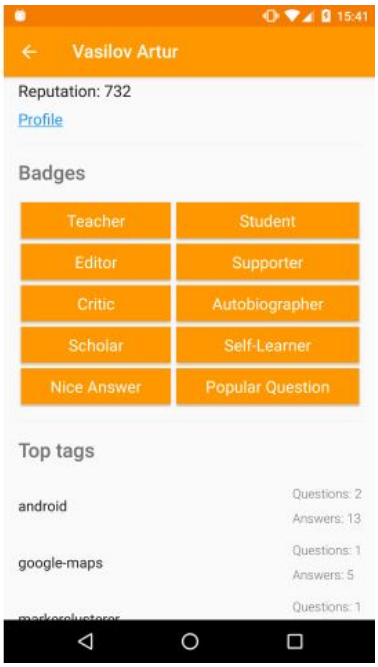
UI Element	Description	Actions
List of question	List of question with related tag	Scroll up and down. List should be endless when scrolling bottom. List also implements swipe-to-refresh pattern. Swipe leads to next or previous tab.
List item	Contains question title, author icon and name, views count, answers count and whatever this question is answered or not	Clicking on item opens details question screen

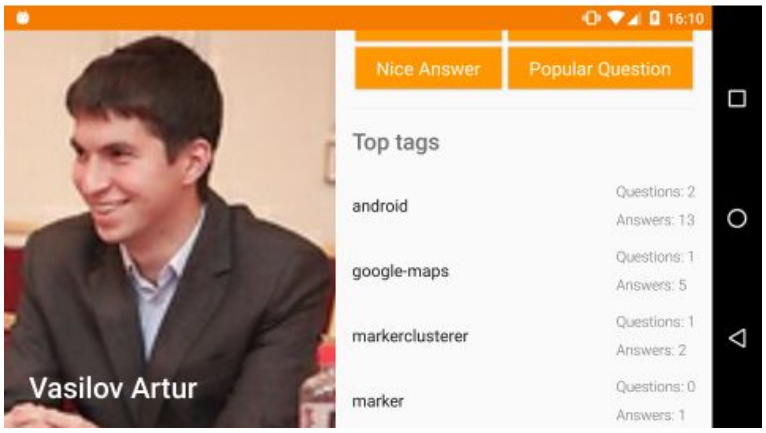
Profile Screen

Screen with full information of user. It could be any user from the site. This screen usually opens from clicking user's image in questions or answers screens. Information loading progress is displayed via progress bar.

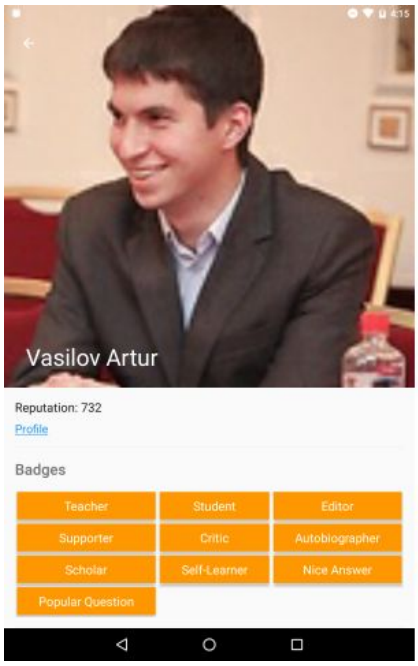
Phone design

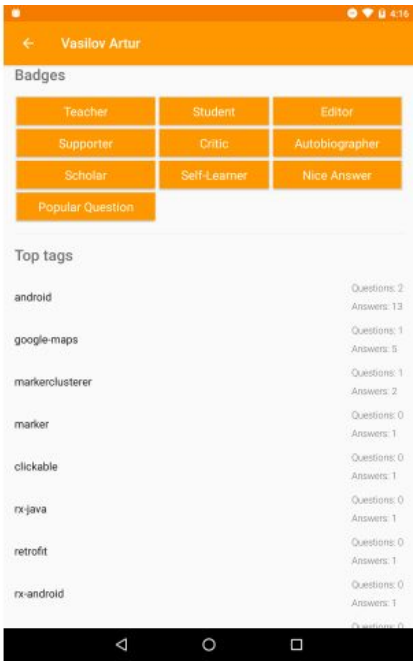


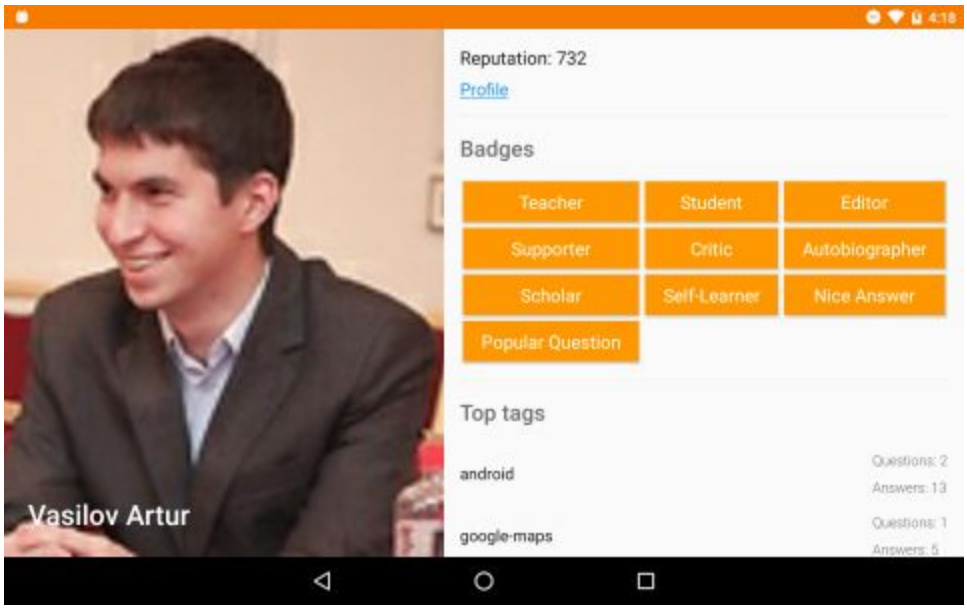




Tablet design





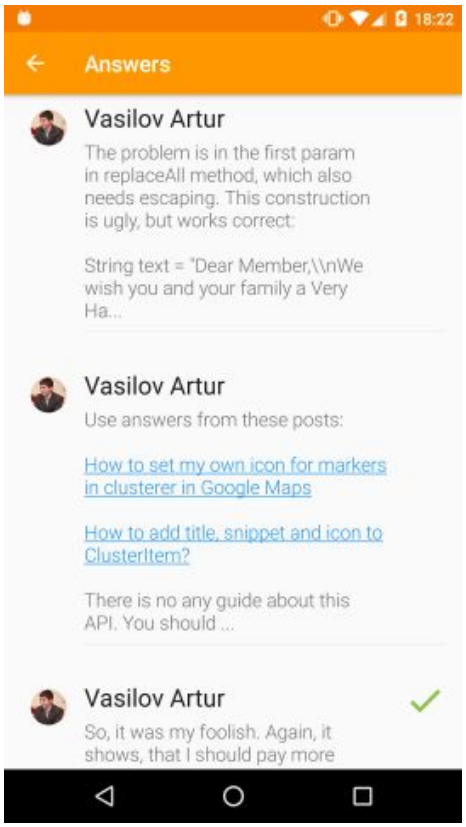


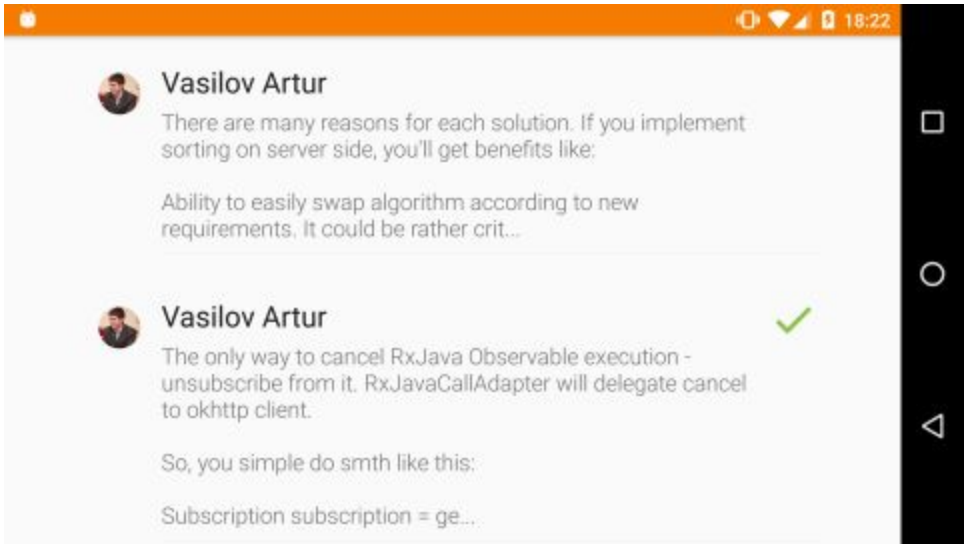
UI Element	Description	Actions
Profile image	High-resolution profile image. Size of the image must be computed and	None

	exactly this size must be downloaded from network.	
User name	User name from the site	None
Profile link	Link to the profile on StackOverflow site	Clicking on link should open user profile in browser
Reputation	Current user reputation	None
List of badges	List of all user's badges from the site	Clicking on badge opens page info in browser
List of top tags	List of user top tags from the site	None

Answers Screen

This screen displays current user's answers in list. Application shows loading progress during the information loading.

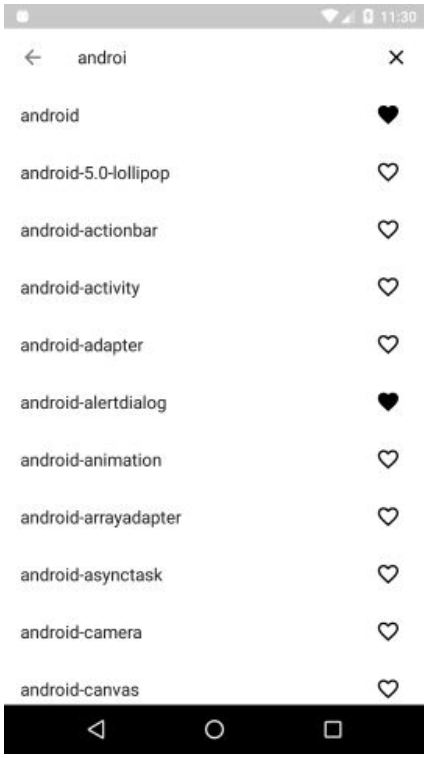


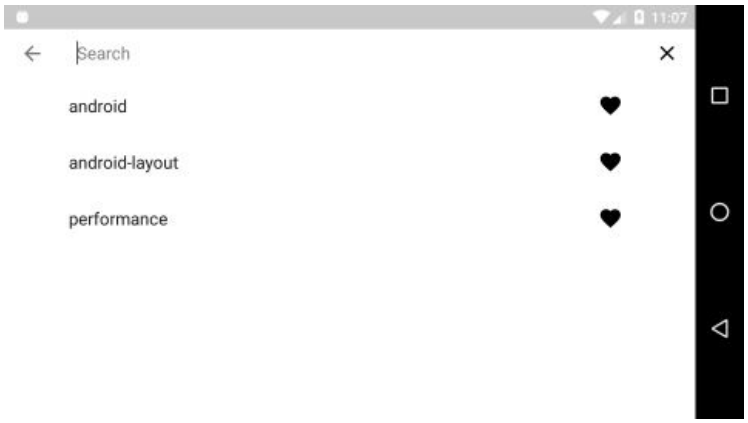


UI Element	Description	Actions
Toolbar	Toolbar with title “My answers”	Toolbar hides when user scrolls up and shows when scrolls down
List of answers	List of user’s answer	Scroll up and down
List item	Answer item has author icon, name, body preview and whatever answer was accepted or not.	Clicking on item opens this question and answer in browser

Tags Screen

Screen for searching for tags and adding it as favourite.



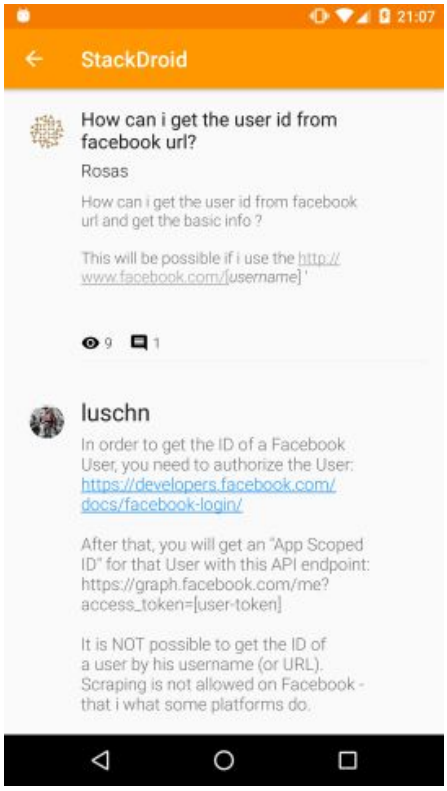


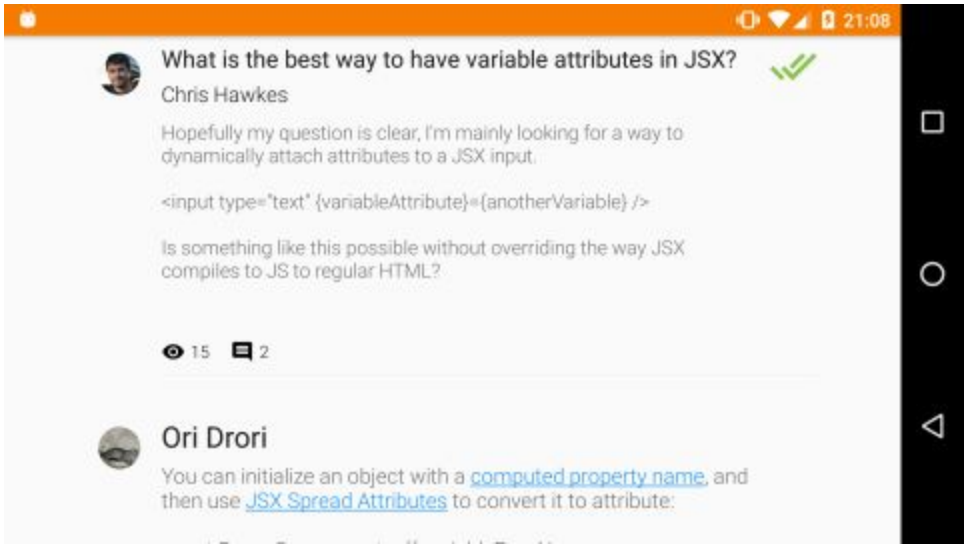
UI Element	Description	Actions
Input field for search	User input tag part into this field	Input tag text
Clear button	This button is displayed when input field is not	Clicking on button clears the input text and list of

	empty.	found tags
List of favourite tags	List of already saved favourite tags, which is displayed when input field is empty. Sorted in alphabetical order.	Scroll up and down
List of found tags	List of found by name tags from server. Sorted in alphabetical order.	Scroll up and down
Tag item	Contains text with tag name and button to add or hide tag from favourite	Clicking on button adds or removes tag from favourite tags list
Progress bar	Shows progress during loading tags from server	None

Question Screen

Details question screen with question body and with answer (also with bodies). Application shows progress bar during loading information.





UI Element	Description	Actions
Toolbar	Toolbar with title “StackDroid”	Toolbar hides when user scrolls up and shows when scrolls down
Question in details	Contains question title, author icon and name, question body, views count, answers count and whatever this question is answered or not	None
List of answers	List of answers to selected question	Scroll up and down
Answer item	Answer item has title, author icon and name, body and whatever answer was accepted or not.	None
Progress bar	Shows progress during loading all information from server	None

Application implementation

This section contains information for developer about technical process of implementing application described earlier.

Data persistence

For small data, such as tokens or user settings, application uses Hawk library.

More complex data should be stored in local database. For this purpose, developer provide a library with flexible interface for managing data and based on ContentProvider. Library is included to project as a module.

Corner cases in the UX

No special corners cases. Actions on each screen is defined in Application Screens section.

Libraries

- Support libraries:
    - v4 for support fragments (since they more flexible and often updated and less depends on current device system API).
    - v7 for AppCompat, styles, etc.
    - v13 for FragmentStatePagerAdapter for ViewPager
    - recyclerview-v7, gridlayout-v7 for backport of these widgets
    - design for NavigationView, FAB, TextInputLayout, etc.
- Network
    - OkHttp as main network library (from version 3.4 it uses own network API, which is better than standard HttpUrlConnection and HttpsUrlConnection).
    - Retrofit as flexible API for REST calls. It also uses Gson converter and RxJava adapter.
- RxJava and RxAndroid for managing concurrency and data transfer. Reactive model is extremely powerful and flexible for managing multiple requests, handling errors, data mapping and so on.

- Picasso for image loading, since it's simple and powerful enough. Also Picasso will be configured with OkHttp3 downloader.
- Hawk for simple key-value storage. It also allows data encryption.
- Analytics
    - Firebase for collecting statistics of how users uses the app
    - Crashlytics for handling crashes
- Firebase Cloud Messaging for simple notifications
- Testing libraries
    - JUnit as main testing framework
    - Mockito for mocking interfaces and Android classes for Unit-testing
    - Powermock for mocking static methods
    - JaCoCo for measuring test coverage
    - Espresso for UI and integration testing

Application Architecture

For each screen app uses MVP pattern, which is presented in the next form:

- View – interface implemented by Activity or Fragment, which contains simple actions to update user interface
- Model – POJO from remote API
- Presenter – class which handles all business logic, works with data, manipulates View, make requests to Repository (see below) and handles lifecycle.

I've also included next classes:

- Repository – main layer between network and presenter
    - Sends requests directly to the remote API
    - Handles response, saves it to persistent storage and delivers to the presenter
    - Handles server error and trying to fix them (such as session expired) or deliver to the presenter in more readable way
- Router for handling complex navigation

Typical Repository method should like this:

```
@NonNull
@Override
public Observable<User> getCurrentUser(@Site String site) {
   return mService.getCurrentUser(site)
           .compose(ErrorsHandler.handleErrors())
           .map(UserResponse::getUsers)
           .map(users -> users.get(0))
           .compose(RxSchedulers.async());
}
```

`ErrorsHandler.handleErrors()` call should handle most common errors (like session expired). `RxSchedulers.async()` executes request in background thread.

For managing lifecycle Presenter uses loader-based architecture (RxLoader - loader with RxJava wrappers).

Testing

All sensitive logic of application should be tested via Unit-tests which should run each time building release version of application. Only requirement here is that each presenter class should be covered by tests at least 80%. Developer also can write as much tests as needed.

Application architecture must provide ability to test sensitive logic. It should be done using dependency injection principles. These principles can be implemented in any way – Dagger2, static setter. During the Unit and UI tests repository is mocked to return data required by the test.

Application also may provide UI and integration tests to be sure that app works correctly for the most typical scenarios.

Google Play Services / Firebase

Most popular parts of Google Play Services are now replaced by a new platform – Firebase. Application needs analytics and cloud messaging. Before Firebase applications used Google Analytics and Google Cloud Messaging, but now it should be Firebase Analytics and Firebase Cloud Messaging.

Required Tasks

Task 1: Project Setup

Initial configuration for project. Includes:

- Register application on http://stackapps.com/apps/oauth/register
- Configure libraries
- Setup product flavors
- Setup JaCoCo
- Setup Firebase Cloud Messaging
- Setup Firebase Analytics
- Setup Crashlytics
- Create common classes (Application Singleton, Utilities)
- Test everything

Task 2: Network Layer

Base configuration for network classes. Includes:

- Create OkHttp client singleton
    - Initialize with timeouts
    - Add api keys interceptor
- Create any Retrofit Service singleton (probably there will be some of them)
- Create Repository interface and implement it
- Add common Reactive classes
- Add ability to set repository instance
- Add errors handling
- Write tests for created classes

Task 3: Authorization

Create authorization and sessions. Includes:

- Add check if access token is saved on app startup
- Open authorization page https://stackexchange.com/oauth/dialog with parameters in broswer

- Handle registration errors (layout + logic)
- Handle registration success (save token and open walkthrough screen or main screen)
- Tablet adaptation

Task 4: Database library

Create library and use it as a module in application.

- Create new module called sqlite
- Add implementation for ContentProvider
- Add flexible interface for easy use of library
    - Table classes
    - Query / Insert / Update / Delete classes for managing requests not via ContentProvider API but via library API
    - Add RxJava features
- Test library
- Create dependency for the module sqlite in the main application module

Task 5: Walkthrough and Splash

Create walkthrough screen with pages of main features. Texts and image description may be found in Application Screens section. Task includes:

- Open this screen only if user hasn't passed it before
- Create layout for pager with different elements
- Allow navigation via swiping and button
- Save tutorial finished and not open it again
- Load required information during walkthrough
- Show splash if user finished tutorial before requests finished
- Show error if failed to load data
- Open main screen if everything is loaded and user viewed the tutorial
- Write tests both for viewing tutorial and loading data
- Orientation change handling
- Tablet adaptation

Task 6: Main Screen

Create main screen layout and implement common features and navigation. This includes:

- Create toolbar with appbar
- Add navigation ViewPager for user favourite tags, put tabs to the top of screen
    - Tab “All” and “My questions” is always shown
    - If user hasn’t favourite tags and no questions, tab layout isn’t shown
    - Each page in ViewPager contains list of questions
- Create navigation drawer menu with following items
    - Header with name and title
    - Profile item which leads to the screen with profile page
    - My answers item which leads to the screen with list of user answers
    - Tags item which leads to page with site tags, where user can search for tag and add mark tag as favourite
    - Exit item, which performs a logout
- Set user display name
- Load user avatar and transform it to circle
- Orientation change handling
- Tablet adaptation
- Write tests

Task 7: Questions Screen

Create fragment in pager with questions. This includes:

- Create pager adapter
- Create fragment layout and class
- Load question by tag from network or from local database
- Create question item layout
- Add item click listener and open question screen
- Add progress bar for loading questions
- Add endless scrolling
- Add toolbar hiding and showing with scroll
- Orientation change handling
- Tablet adaptation
- Write tests

Task 8: Profile Screen

Create new screen to display user profile. This includes:

- Create layout to display all elements
- Load badges and top tags
- Highlight link to profile
- Open profile page in browser when clicking on link
- Add progress bar for loading user information
- Load high-resolution image in scrolling toolbar
- Add avatar transition
- Orientation change handling
- Tablet adaptation
- Write tests

Task 9: Answers Screen

Create new screen to display user profile. This includes:

- Create toolbar with title
- Add toolbar hiding and showing with scroll
- Create screen layout with list of answers
- Load answers from network or from local database
- Add item click listener and open answer in browser
- Add progress bar for loading answer
- Orientation change handling
- Tablet adaptation
- Write tests

Task 10: Tags Screen

Create new screen for searching tags and add them to favourite. This includes:

- Create toolbar with search field
- Add clear button to the end of input field
- Add list of favourite tags if search field is empty
- Display an empty view if search request found nothing

- Search for input tag on a server and show results to the user
- Add ability to add tag to favourite
- Add ability to remove tag from favourite
- Add progress bar for loading tags
- Orientation change handling
- Tablet adaptation
- Write tests

Task 11: Question Screen

Clicking on question item is the questions list user got to the details question screen. Create this screen. This includes:

- Create toolbar with title
- Add toolbar hiding and showing with scroll
- Show question with body
- Load question body
- Show list of answers
- Load answers with body
- Add progress bar for loading all information
- Add transition animation for question
- Orientation change handling
- Tablet adaptation
- Write tests

Task 12: Exit

Add ability to exit from app:

- Clean up all saved data when user clicks exit
- Clear token from server
- Open auth screen

Task 13: StackDroid Widget

Add homescreen widget with user icon, name and inbox. This includes:

- Create list widget layout

- Fill widget with information of user notifications
- Clicking on item should open profile page in browser
- Update widget information periodically
- Load question body
- Show list of answers
- Load answers with body

Task 14: Notifications

Add ability to send notifications to users with Firebase Cloud Messaging. These notifications should be customizable and extendable in future releases.

- Add services for subscribing and unsubscribing for push notifications
- Show notification with passed text
- Notification should contain single text and navigate to app launch
- Application doesn't show notification if app is currently visible
- Customize push with parameters so that application can decide, whatever show notification on not

Task 15: Analytics

Add event tracking for most sensitive user actions. This includes:

- Add information about user device to user parameters
- Add information about user account to user parameters
- Track each screen
- Track sensitive events

Task 16: UI and Integration tests

Write UI and integration tests for most common scenarios of user behaviour. For example:

- Watching walkthrough and opening main screen
- Look and scroll questions on main screen
- Open all screens and verify that app behaviour is correct

Task 17: Final Preparations

Prepare app for publishing. This includes:

- Test application as user and verify that it satisfies all requirements from the document
- Review all classes and try to find bugs in code
- Use tools such as Lint or FindBugs plugin to clean up the code and fix warnings
- Use perfomance tools to test app
- Check and run Unit- and UI-tests
- Build release version of app