GitHub Username: roklyt

NotADoctor

Description

The app tries to find a diagnoses with at least 90% certainty in a question answer scenario and then to find the user a suitable doctor in his immediate vicinity. The mechanism are based on the api of infermedica

Intended User

This app is made for everyone. But especially for the people who need a little push to go to a doctor because they have no excuse here anymore they don't know which one.

Features

- Save user profile with all relevant infos about the user.
- Self diagnose chat which results in an diagnose.
- Google Maps search to show the customer the next practice that meets his needs.

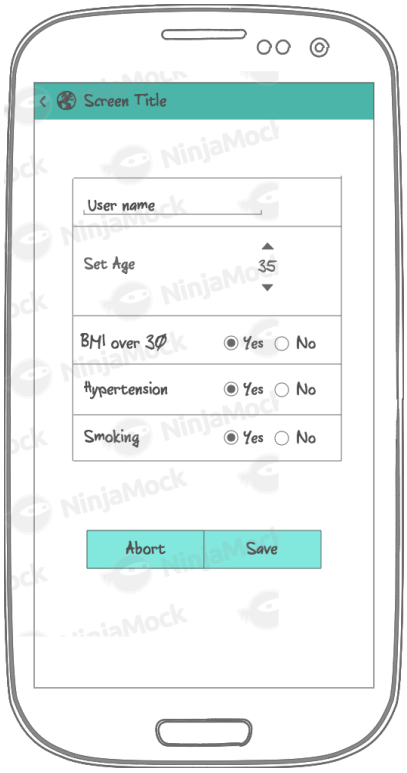
User Interface Mocks

https://www.ninjamock.com/s/HVDWTSx

Welcome screen

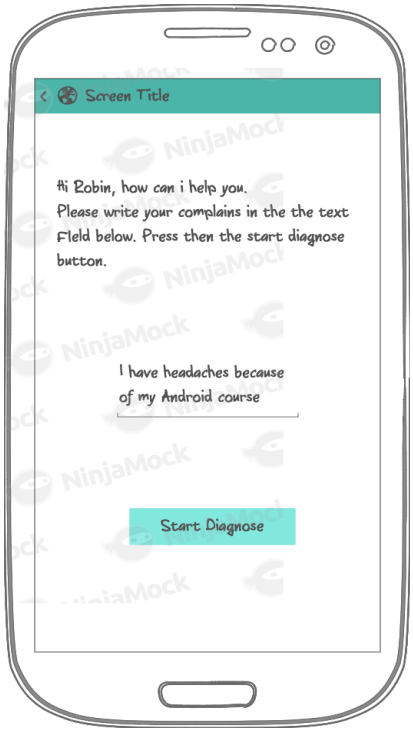
The user has a overview about the user profiles. More than one profile is important here, so that the user can for example also add his children who do not have smartphones or grandparents. By clicking a profile the diagnose process starts.

Edit and create screen



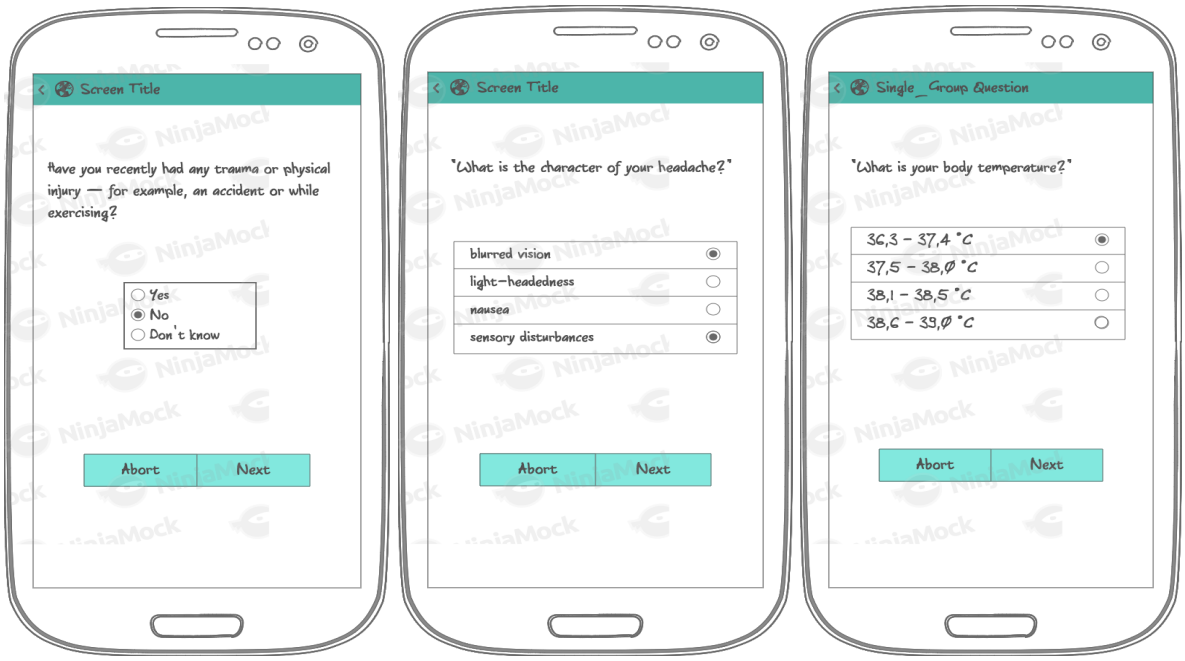
The customer can create or edit his profile and set some common risk factors.

Start the diagnose screen



The user can enter his complains and start the diagnose.

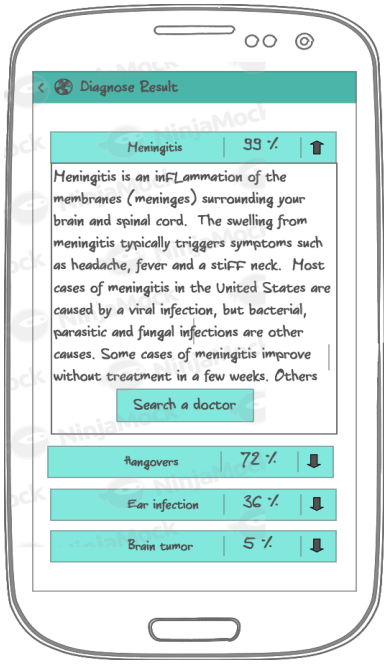
Question pages



There are three different kind of question pages.

- Single Question: Yes, No, don't' know
- Group Question: The user can choose multiple answers.
- Single Group Question: The user can choose only one answer.

Result page

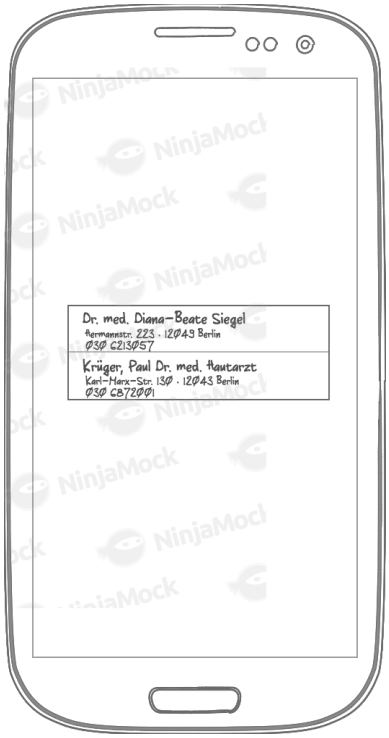


The user gets multiple results with different percentages. The result with the highest percentage is opened and summary about the diagnose is shown.

Find a doctor

On this page the user can choose any doctor which was found via google maps in his area.

Widget



The widget shows the doctors from the last successful diagnose.

Key Considerations

How will your app will be written?

The app will be written solely in the Java Programing Language.

How will your app handle data persistence?

The app is using room,live data and viewModel.

Describe any edge or corner cases in the UX.

- If the user is in the middle of a diagnose and hots the back button the welcome screen appears after a confirm dialog.
- Orientation change while api request is in progress.

How resources will be stored in the project.

- All resources will be stored in the res folder of the project.
- Colors.xml
- String.xml
- Styles.xml
- Dimen.xml
- Pictures will be stored in the drawable folder.

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

- GSON to serialize and deserialize the requests and responses against the infermedica api.
- Retrofit for api calls.
- Butterknife to get rid of findviewById.

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

- Google add service to show on the find a doctor page a add banner.
- Google maps and google places to find the right location.

Which Library versions will be used

- 'com.android.support:appcompat-v7:28.0.0'
- 'com.android.support.constraint:constraint-layout:1.1.3'
- 'com.android.support.test:runner:1.0.2'
- 'com.android.support.test.espresso:espresso-core:3.0.2'
- 'com.squareup.retrofit2:retrofit:2.5.0'
- 'com.google.code.gson:gson:2.8.5'
- 'com.jakewharton:butterknife:9.0.0-rc2'
- 'com.jakewharton:butterknife-compiler:9.0.0-rc2'
- 'com.google.android.gms:play-services-maps:16.0.0'
- 'com.android.tools.build:gradle:3.2.1'
- Android Studio 3.2 Build #AI-181.5540.7.32.5014246
- JRE: 1.8.0_152-release-1136-b06 x86_64.13.6

How will the app support accessibility

- All font sizes are in sp
- Content description for all images.

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
- Create database
- Define tables(User, symptomes, diagnoses)
- Create user adapter
- Create diagnose adapter
- Create API logic for the diagnose dialog

Task 2: Implement UI for Each Activity and Fragment

- Build UI for WelcomeActivity
- Build UI for EditCreateActivity
- Build Fragment for diagnose dialog
- Build UI for resultActivity
- Build UI for findDoctorActivity

Task 3: Implement Google apis

- Implement Google maps
- Implement Google add

Task 4: Edge cases and error handling

- Add error handling
- Handling for edge cases

Task 5: Implement Widget

- Create widget layout
- Combine the widget with the app

Task 6: Material Design

- Polish the app and add Material design
- Add translation Animations
- Animate buttons
- Add design for Tablet and landscape mode.

Task 7: Write tests

- Test adding and deleting a user profile