Capstone Project

GitHub Username: judit-juhasz

Tanits

Description

We help you to bring healthy and successful children up by building your very own personalized early childhood development assistant. We combine artificial intelligence with the latest early childhood education results.

Early childhood education has significant impact on the areas of cognition, language, socioemotional health, education and the labor market. Despite of its importance, most children cannot get proper care from their parents. Some of the reasons are the lack of time to learn parenting, the multidisciplinary nature of early childhood education and the uniqueness of every children.

We give you the right amount of information, in the right time, specifically tailored for you, your children's and your family's situation.

Your genius was born. Do you want to be a conscious parent? Tanits!

Intended User

This is aimed at first time parents with child at the age 0-1 months, who

- Want to be conscious parents
- Interested in the scientific background of the recommendations
- Looking for a supporting, positive community

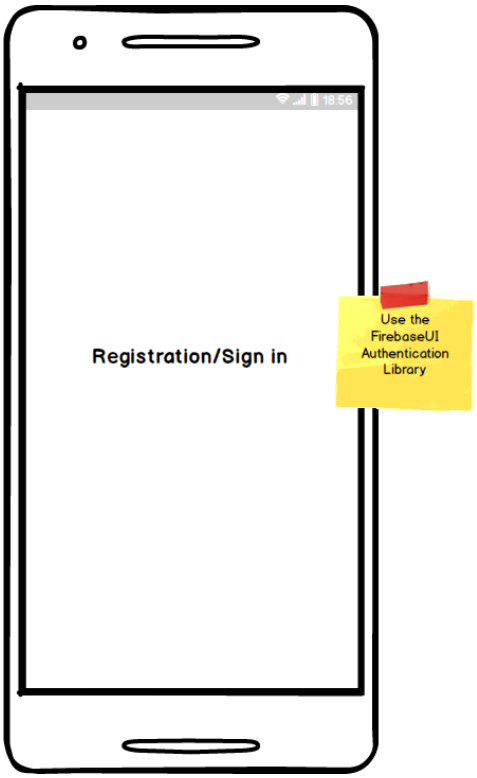
Features

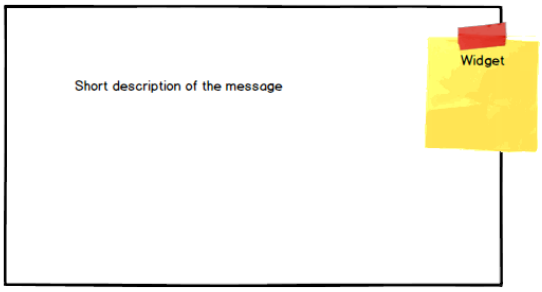
Main features of the app

- Registration
- Edits profile
- Receives messages/tasks
- Manages tasks (filtering, rejecting/finishing with feedback)
- Shows latest tasks on a widget
- Allows the user to send feedback/question through email

User Interface Mocks

Screen 1

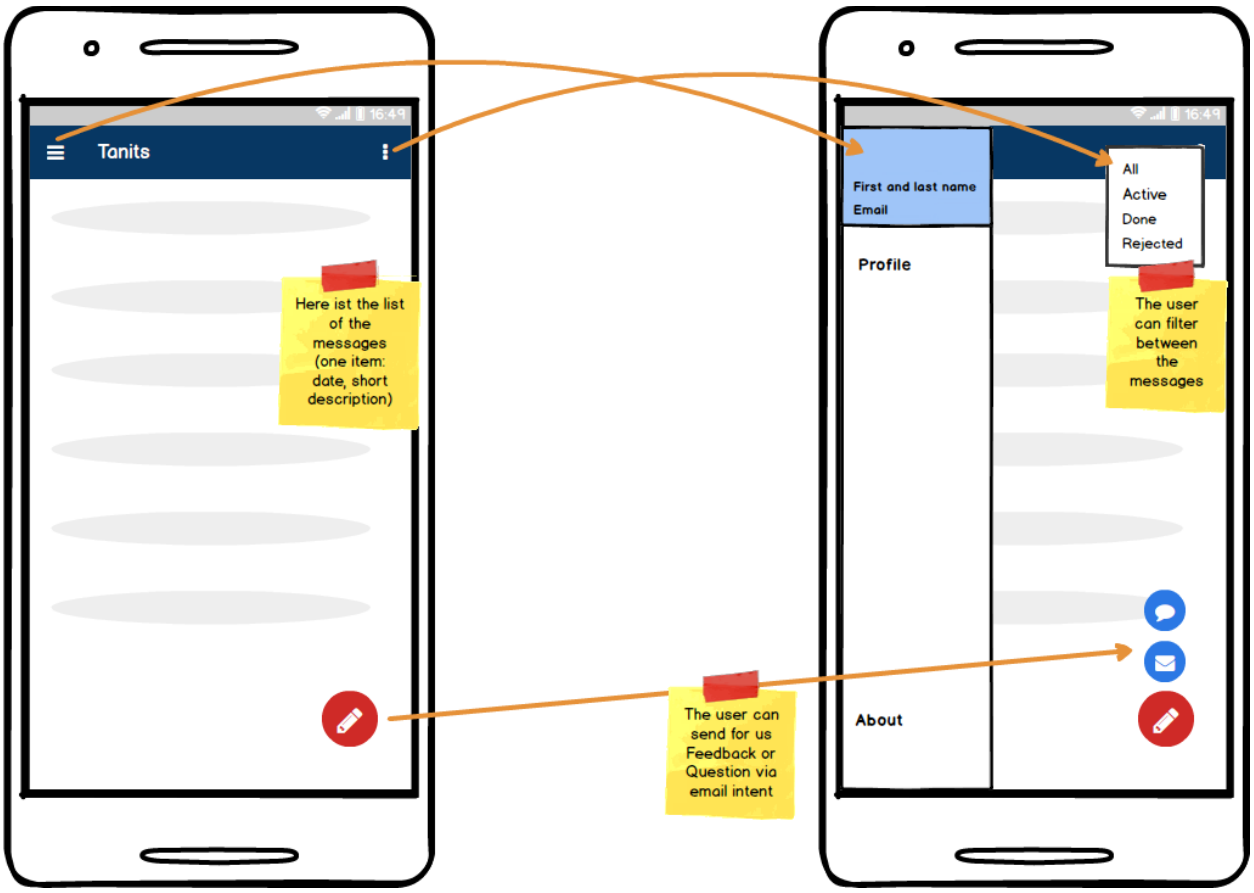




At application start we use Firebase Auth library for Registration/Sign in.

The application also installs a widget where we display a short, motivational description of the latest messages.

Screen 2



After login/registration a list displays the available messages. The user can switch between group of messages:

- All messages: All received messages
- Active: Tasks those has not been finished/rejected.
- Done: Tasks those have been finished
- Rejected: Tasks those has been rejected by the user.

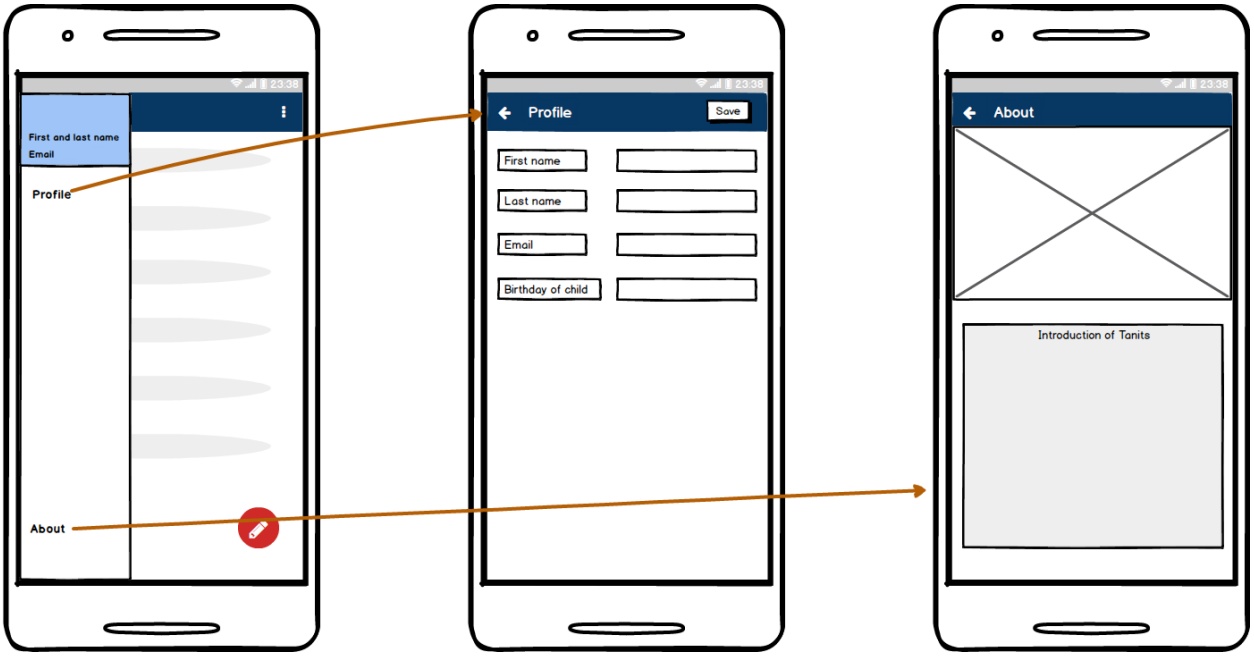
Via the burger button, a sidebar will be shown. On this sidebar the user can have access to their profile and basic information about the app. It will be useful later to add more features.

The user can send 2 type of messages to the developers via the floating action button of the activity:

- Feedback
- Question

Both of the messages will open a mail app, but they provide different default subject.

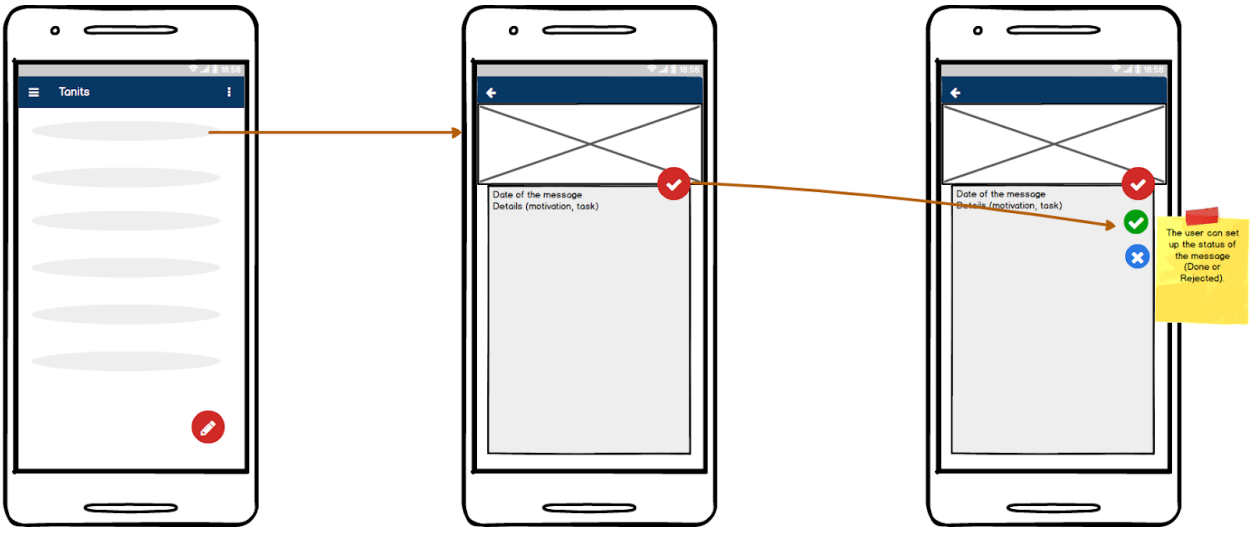
Screen 3



The profile menu in the sidebar opens an activity where the users must provide they first name, last name, email, and the birthday of their child. The changes can be saved via the button in the header.

The about page displays an image and general information about the app and the developers.

Screen 4



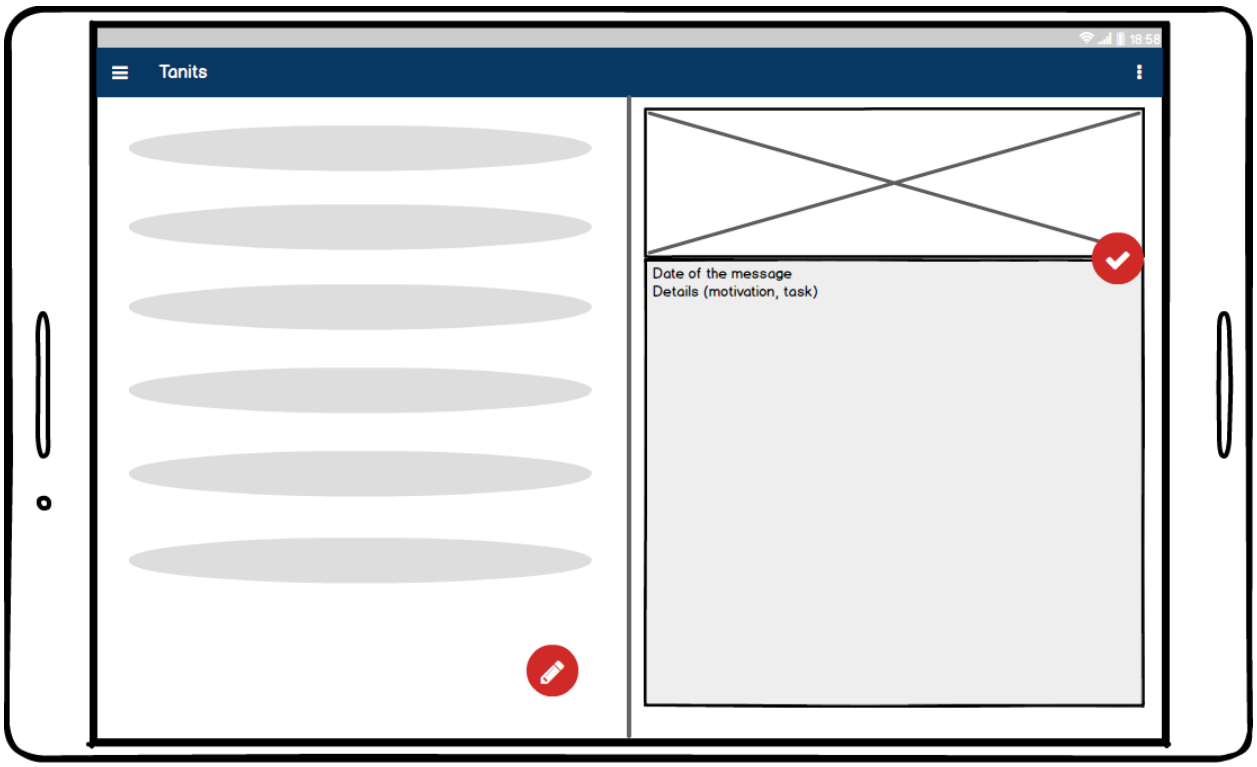
By clicking on a message, a new activity will be opened. On this activity a motivational message and a task (a daily task) will be described from the database. The user can reject or mark it as done via the floating action button. In this case the activity will be closed and the app navigates back to the task list.

If there are no tasks to be displayed, a picture will be shown with a friendly image and a message that tells to the user to that there are no active messages.

The phone version of the app doesn't provide landscape view of the app. Rotation is disabled.

Tablet

Screen 5



In tablet view, the message list and the message will be displayed on one page according to the mock above.

Screen 6

The rest of the activities are displayed in one page with fixed width positioned to the center.

Key Considerations

How will your app handle data persistence?

In order to access a common message/task dataset, and to store user specific data, the app will use the Firebase Realtime Database.

Describe any edge or corner cases in the UX.

- No more messages (all messages of the actual messages are either rejected or finished): A friendly message and picture in the place of the message list.
- No internet access: We will use Firebase Realtime Database offline cache feature, but anyhow, we will inform the user about the missing (and eventually required) internet connection that can result missing messages in the inbox.

- Too many incoming messages when the child is older at the registration: In the first iteration we limit the age of the child that can be registered. We inform the user about this limitation in the place of the incoming messages.
- Interrupted profile editing at registration: The message list will not be shown the user until the profile is finished. Until it finished, the profile will be shown to the user, not the message list.

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

- Picasso: To make picture loading easier, and to cache them.
- Butterknife: To make the source code more clean.
- Firebase Realtime Database: Data sharing/storage between devices. Besides that it makes the async data access easier, has built in caching, etc.
- Firebase Authentication: Because the users need to be authenticated in order to store they current messages, feedbacks, etc.

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

Describe which Services you will use and how.

- Firebase Authentication: Because the app needs to store user specific data online (and only show it to the same user), authentication is necessary. The built in flow will be used (email + password)
- Firebase Realtime Database: Because we need to have access the message dataset, and we need to store user specific data. We store the dataset online (and not stored in the app) in order to make it easier to modify (e.g. extend, fix grammar errors, etc.) it without requiring the user to update the app.

Next Steps: Required Tasks

Task 1: Project Setup

First, we need to initialize the environment that we will use later. It includes the following subtasks:

- Create the github project, initialize it with an auto-generated base Android Studio project
- Add icon and other general customization to the project.
- Register and initialize the Firebase project. Follow the Firebase documentation for guidance.
- Setup test projects for testing (add libraries, examples).

- Create a more detailed design, e.g. what to put into fragments, plan how the different parts of the applications exchange information (e.g. communication between the widget and the app)

Task 2: Implement UI for Each Activity and Fragment

Create a functional UI (without too much UX work) to test the user interaction. Use dummy data for lists and other data. The purpose of this step to create a base structure that can be extended with other functionalities, and to prototype the usage of the app. This step includes:

- Build UI for the activities and fragments + add dummy data (e.g. without network queries)
- Test the functional usability of the app

At the end of this step we should have a better understanding of what to implement, and more detailed plans how to implement it.

Task 3: Add Firebase Authentication

Introduce the first Firebase library to the application. At start, the app should provide the register activity, or the main activity based on the authentication state.

Task 4: Add sample messages to the Firebase Realtime Database and display them to the authenticated user

Add sample messages via the Firebase web interface and display them in the app (i.e. replace the dummy static messages)

Task 5: Filter the sample messages

Filter the sample messages based on the current date. This task also includes the proper handling and storage of the user profile (they are related).

Task 6: Implement message management

The user can finish, decline and filter the received messages:

- Plan that how to store this information in the database
- Store this
- Add filtering menu and functionality (the user can see the finished/declined/all/active messages based on a selection)

Task 7: Add feedback buttons and content to static activities (e.g. About)

These tasks are small and doesn't require too complex work, that is why they are grouped.

Task 8: Finalize the Widget and its updater infrastructure

A Widget needs to be created and an IntentService needs to be used to periodically update its content. The IntentService that will update the widget content will use the Firebase Realtime Database to access the received messages.

Task 9: Polish the UI based on Material Design guidelines and test the application

At this point we have a fully functional application. We need to make it more user friendly from the looks point of view, and we need to provide tests for the application.