GitHub Username: musapkahraman

Kitchen Assistant

Description

Kitchen Assistant helps you track your food, prepare a meal plan and create a shopping list. In addition to that, you can either select recipes from our database or add your own and see how much people like it.
As you will have a healthier overall diet by planning your meals in advance, you will also save time and money by having a shopping list when you go to a store. Create a recipe board with *Kitchen Assistant* detailing the meals you would like to eat for the week, including breakfasts, lunches, dinners, and snacks. Figure out what ingredients you will need to create your meals, and add these to your shopping list with the amount of each food you will need.
Don't worry if you don't have time to go shopping. *Kitchen Assistant* can also show you recipes with ingredients that you already have at home and save you from pondering about what to cook.

Intended User

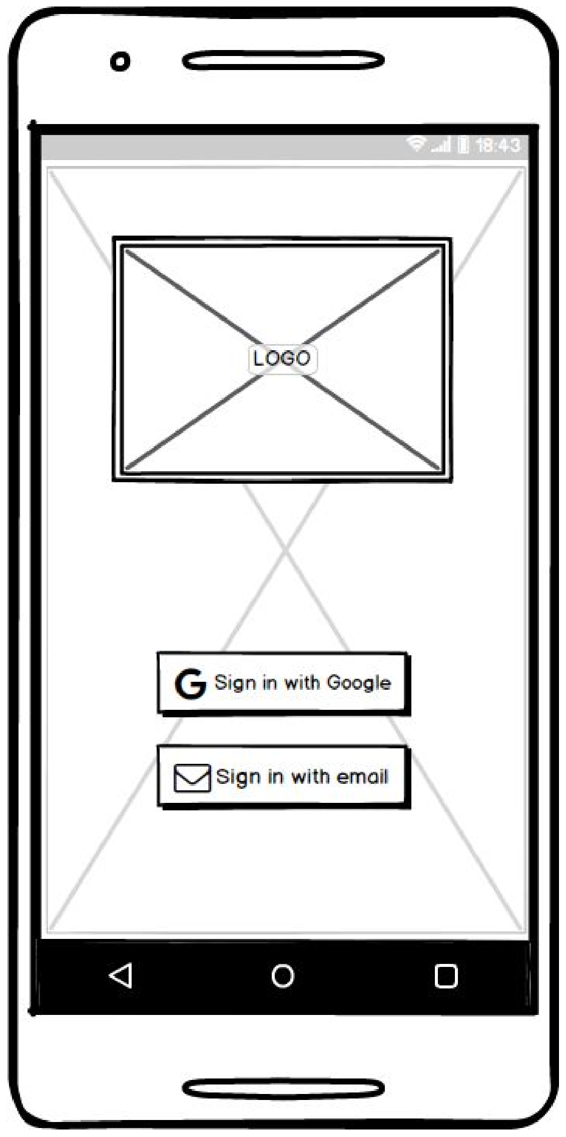
Kitchen Assistant is an app for anyone who cooks at home. User interface will be in English. This app is not targeted for children.

Features

- Sign in with either email or Google Id Provider.
- Browse recipes and rate them.
- Filter and sort recipes by ingredients, cooking time, rating, written language, cuisine, and course.
- Set a timer for a specific step in accordance with the recipe.
- Save recipes with pictures by using the device's camera.
- Build a weekly meal board to plan all kinds of meals for the week.
- Define eating routines to help complete the meal board automatically.
- Get meal suggestions for a maximum of cooking time and specific ingredients.
- Save items in the food storage to help generate the meal suggestion list or to help generate a shopping list considering the weekly meal board.
- Share the shopping list with anyone who will go shopping with their smartphone.
- Get notifications for expired food in the storage.
- Scan universal product codes, barcodes, by using the device's camera and fill food information automatically.

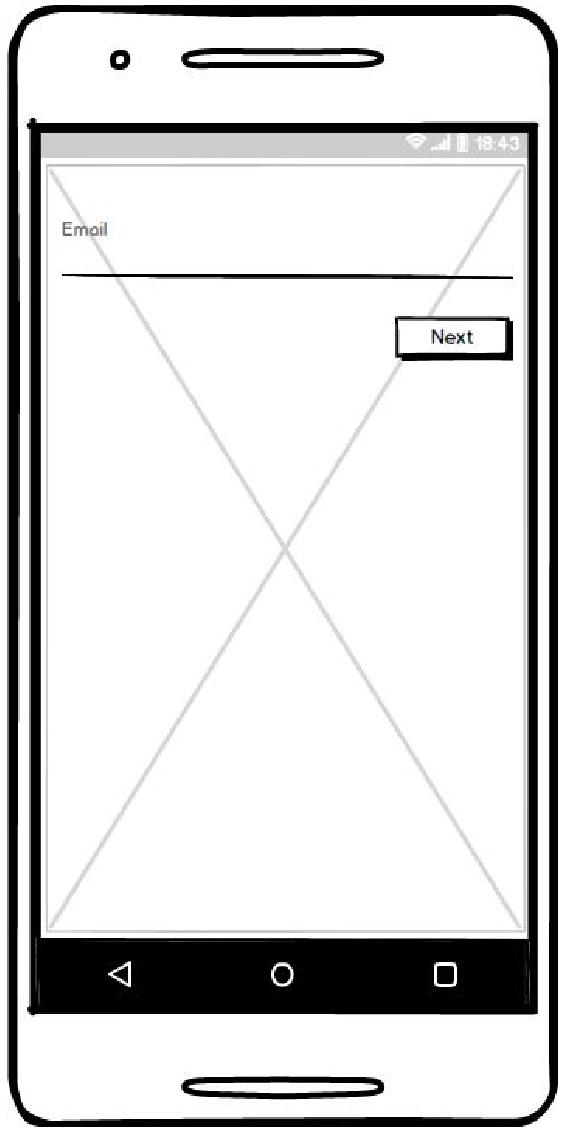
User Interface Mocks

Screen 1



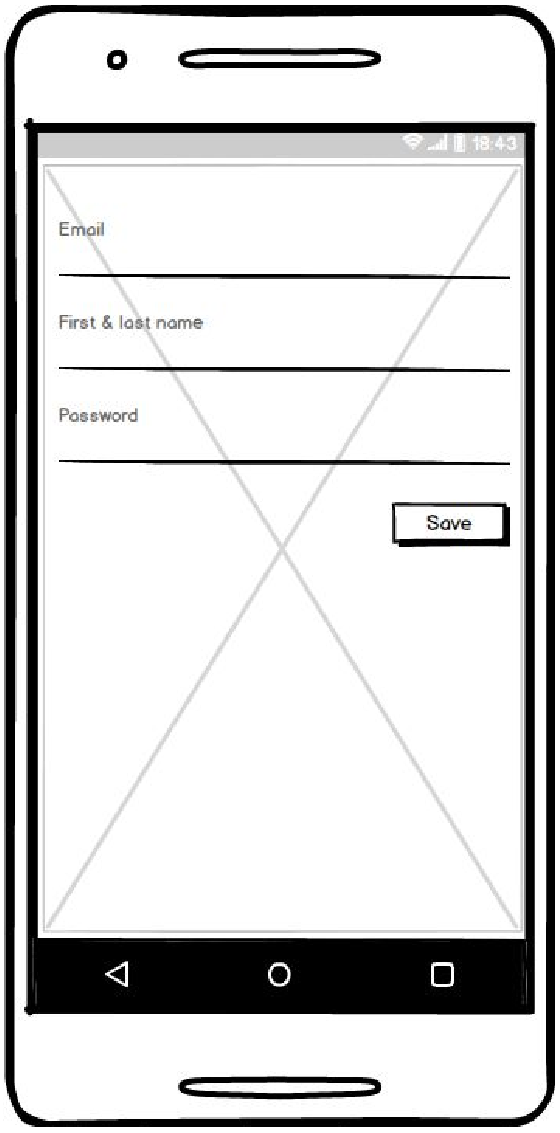
Login screen is the opening screen. If the user is already signed in, then Screen 7 (Recipes) shows up instead.

Screen 2



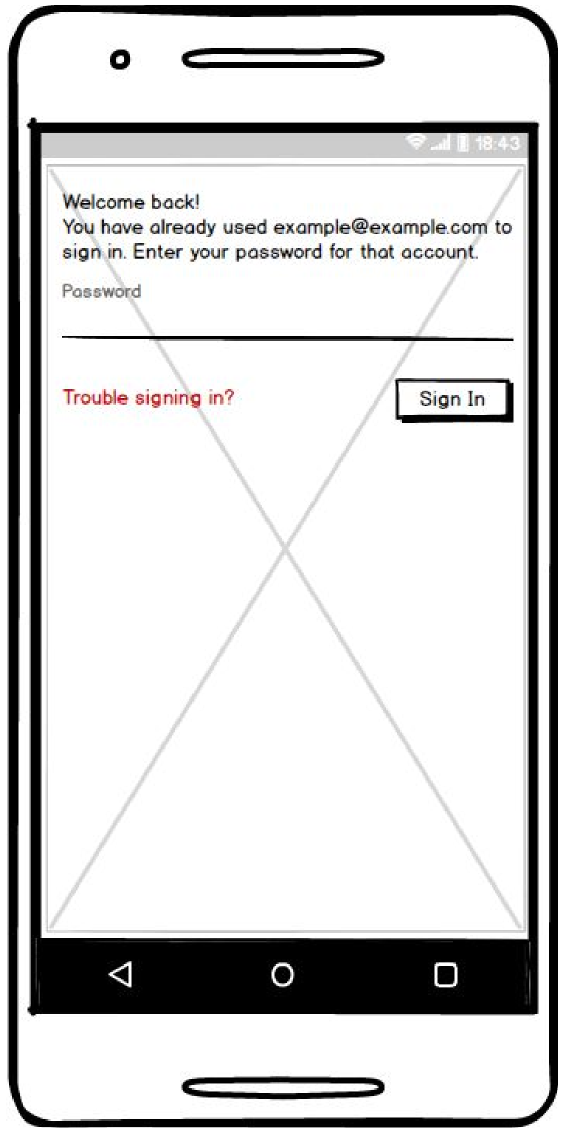
If the user selects to sign in with email, then they see this screen first to enter their email address. Input is validated and checked whether the user is registered or not. If given email address is already on the list then user sees Screen 4, else they see Screen 3 to register.

Screen 3



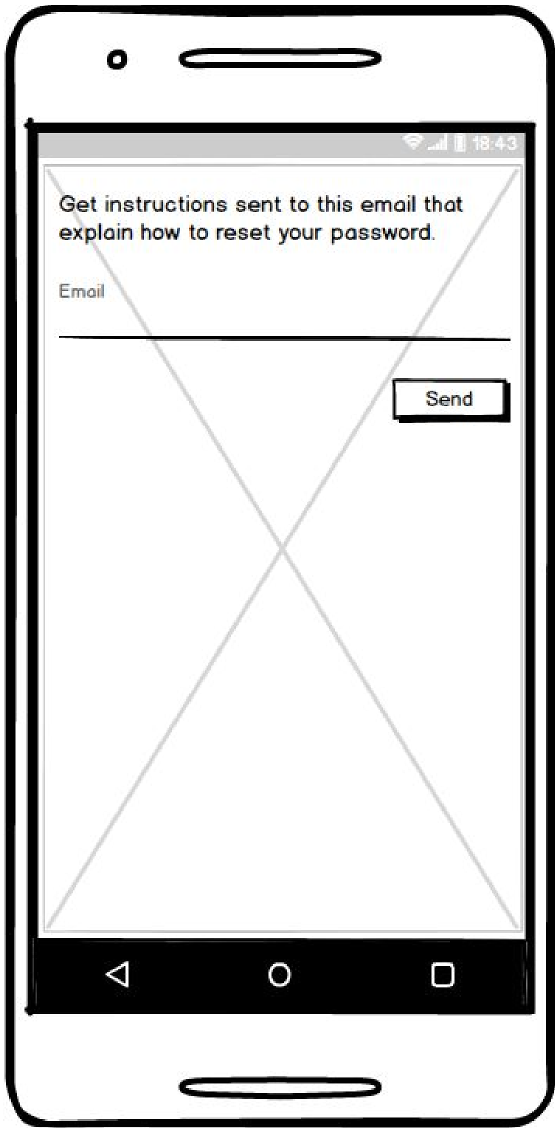
This screen is for registration. User must provide an email address, their name and a strong password.

Screen 4


18:43
Welcome back!
You have already used example@example.com to sign in. Enter your password for that account.
Password
Trouble signing in?
Sign In

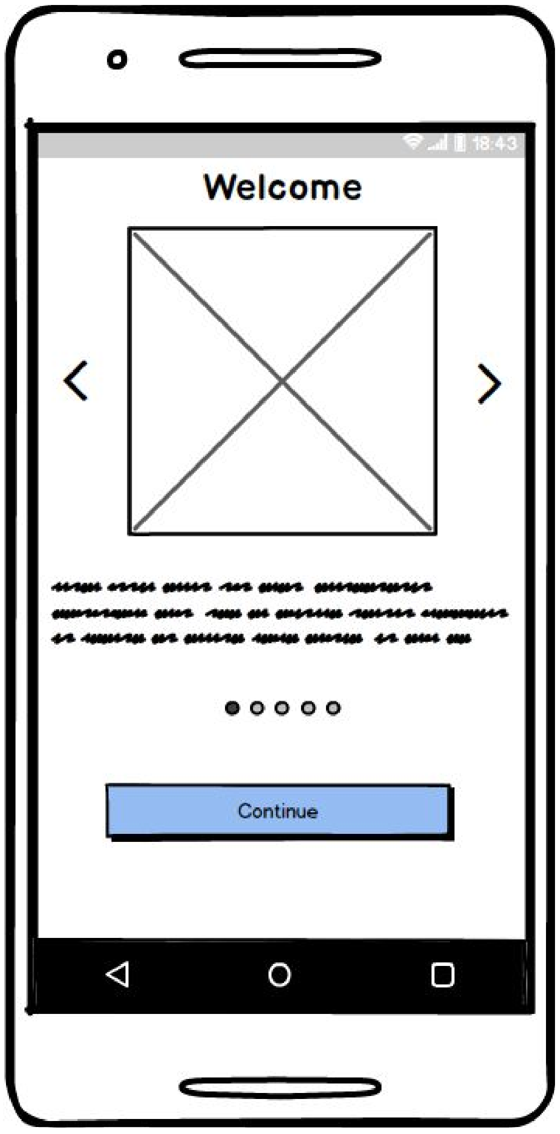
The user is already registered. They must enter their password correctly to continue. If they forget their password, then they can click on “Trouble signing in?” and see Screen 5 to reset their password.

Screen 5


18:43
Get instructions sent to this email that explain how to reset your password.
Email
Send

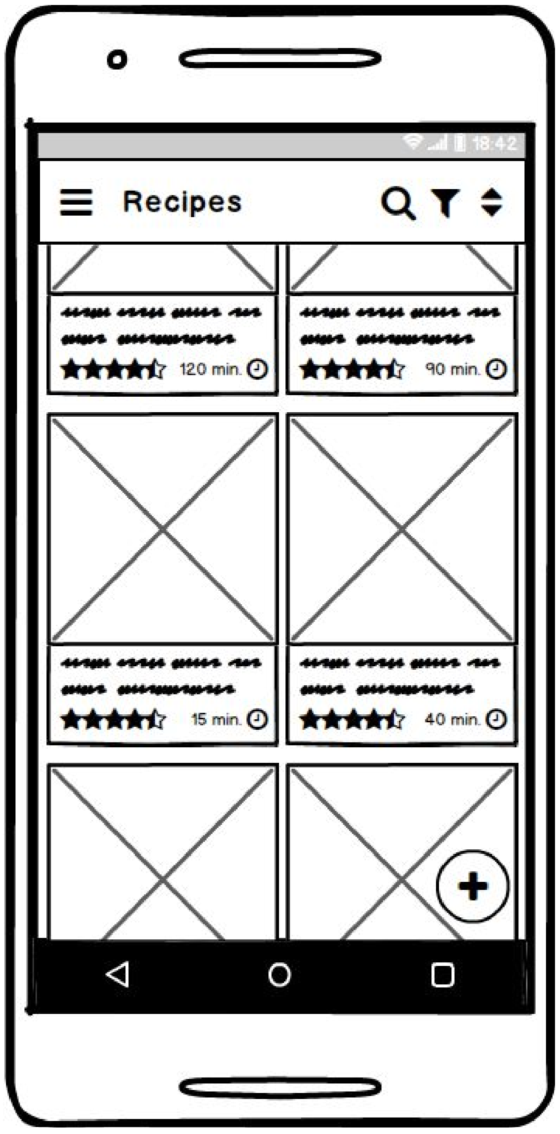
The user is provided with a reset password link in a post to their email address.

Screen 6



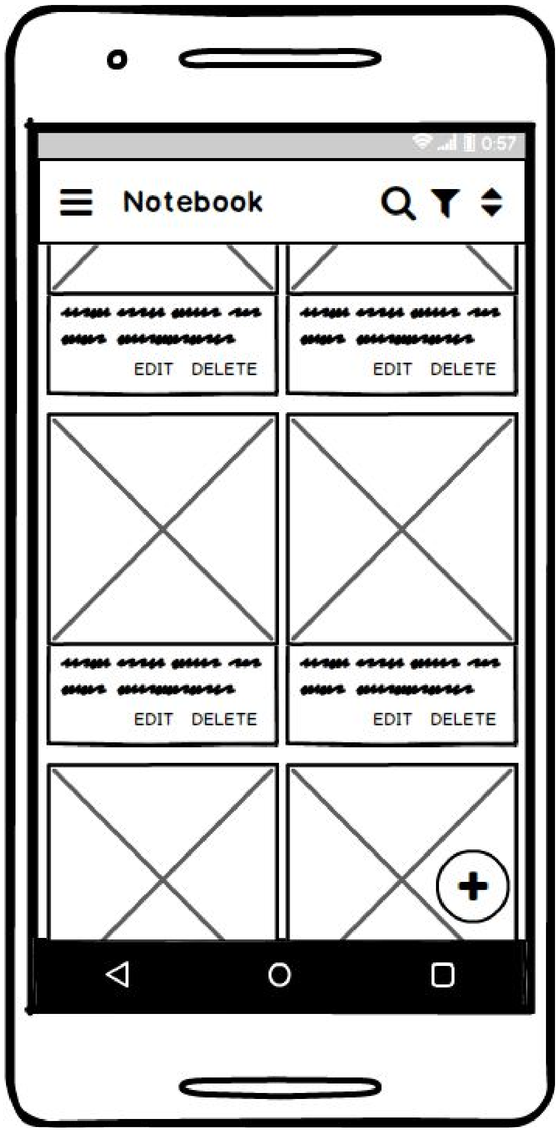
A new user sees this page after signing in. The purpose of this page is to show users the features of the *Kitchen Assistant*. Each feature is on a seperate view which are navigated with a view pager. After seeing the last feature, user can continue to the Screen 7 (Recipes).

Screen 7



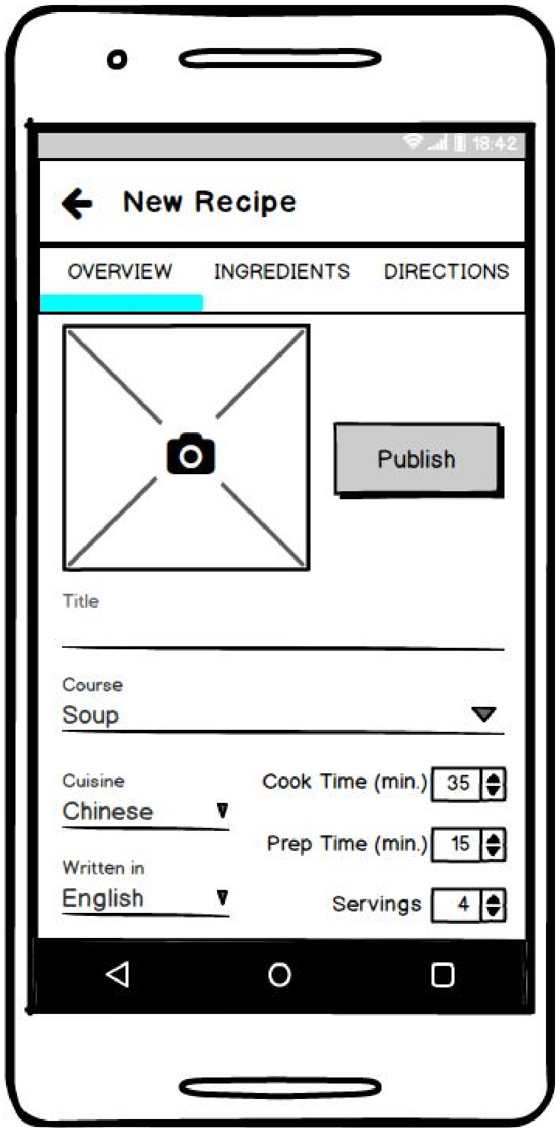
This screen shows a list of recipes in card views. Cards will have a picture, title, rating, and the total cooking time of the recipe. The purpose of the floating action button is to open Screen 9 for adding a user defined recipe. App bar action buttons consists of actions which are finding, filtering and sorting. Recipes can be filtered or sorted by ingredients, cooking time, rating, written language, cuisine, and course. If any of the listed cards is clicked, the recipe detail (Screen 12) shows up.

Screen 8



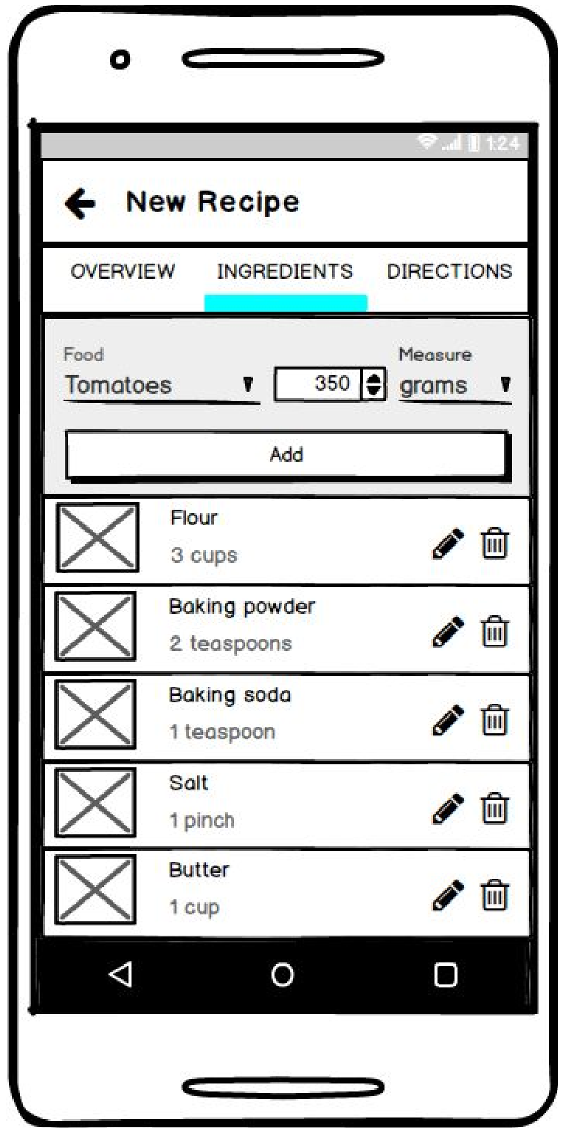
This screen is almost the same as Screen 7, but one thing: this screen shows local recipes that the user either created or bookmarked. They can be edited or removed.

Screen 9



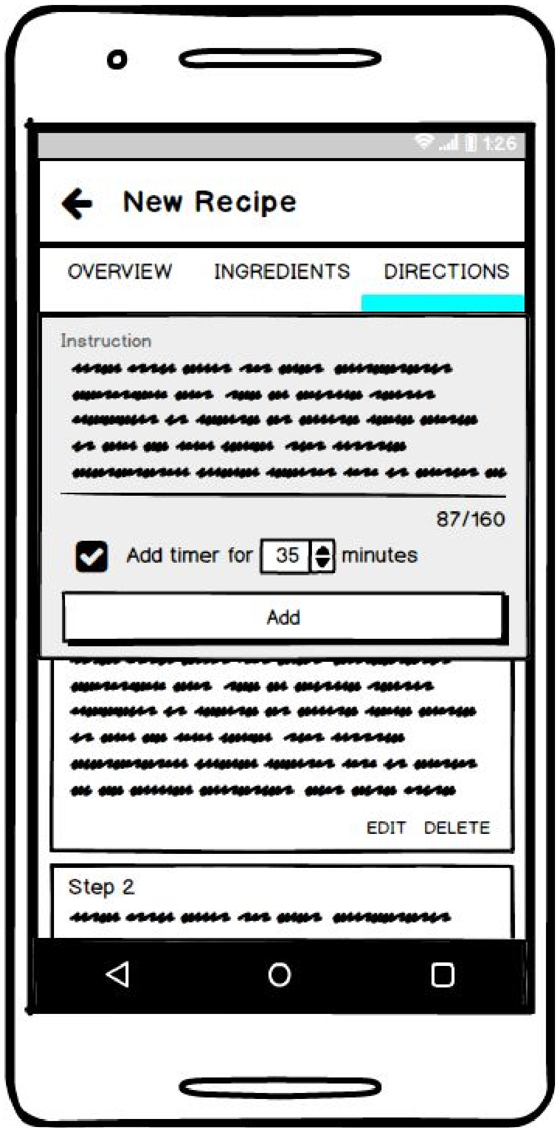
This screen shows an editable view for overview part of the recipe. The user can set or change the image by taking a photograph with their device's camera. Publish button is only available if every field of the three parts of a recipe is filled; until then, the recipe is only saved in local storage.

Screen 10



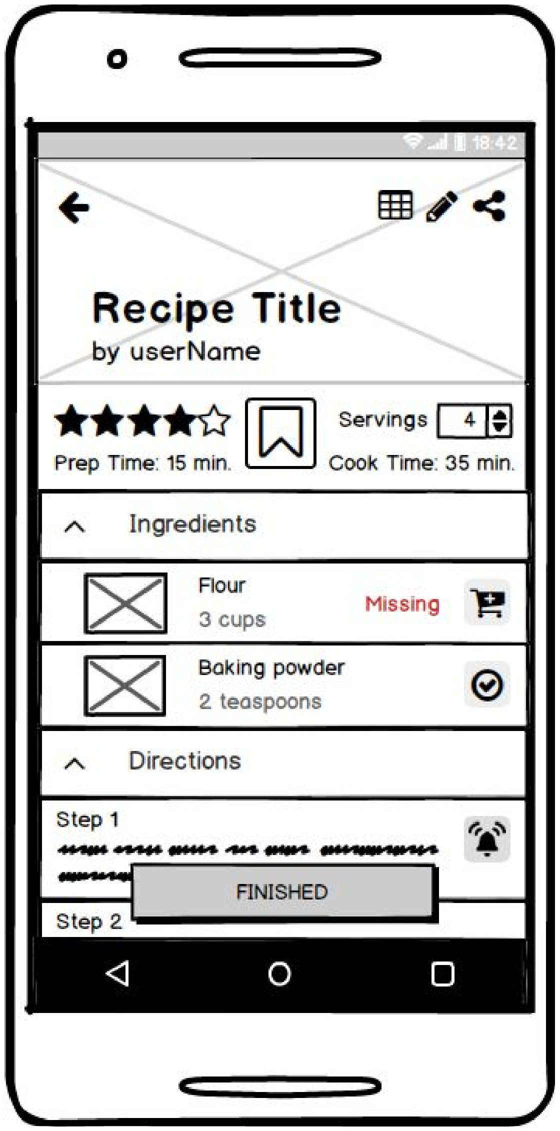
This screen shows an editable view for ingredients part of the recipe. The user fills the editable fields and clicks on Add button to add an ingredient. Each item is editable and removable.

Screen 11



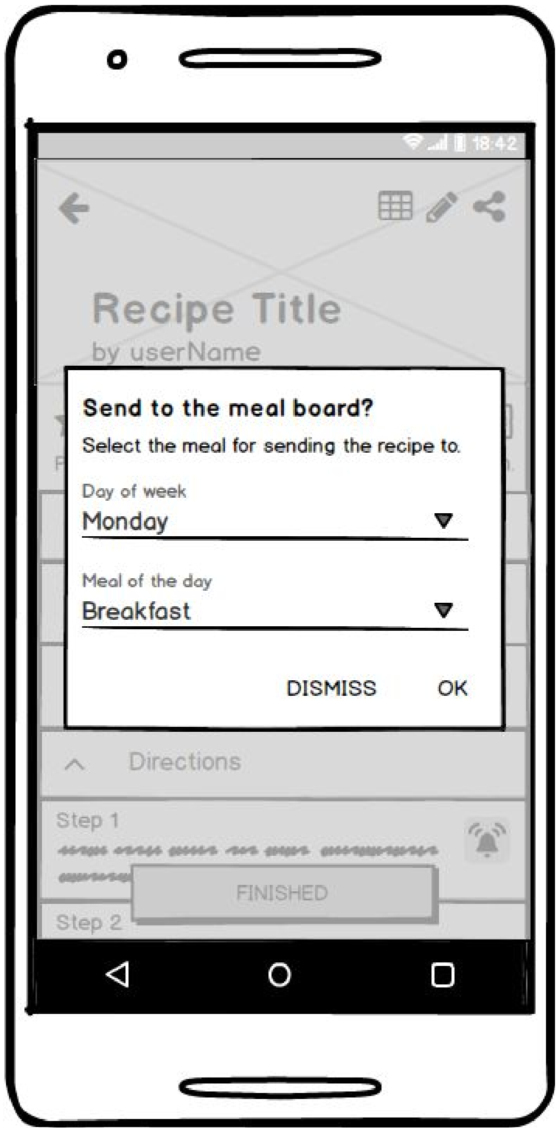
This screen shows an editable view for directions part of the recipe. Because these steps will take some time to finish, users can add a timer for each step if they want. The user fills the editable fields and clicks on Add button to add a step. Each card is editable and removable.

Screen 12



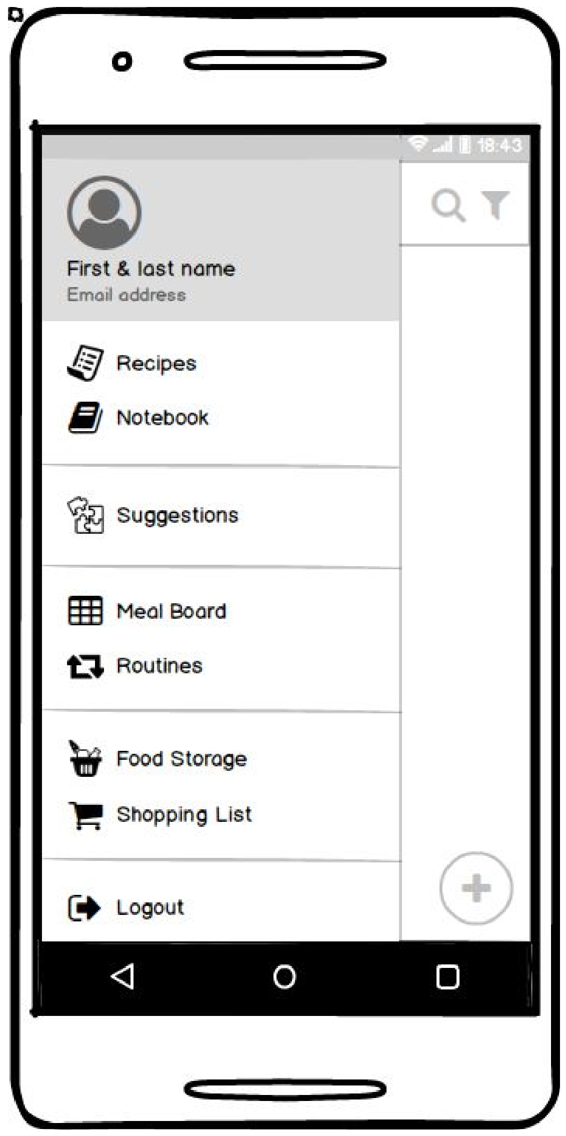
This screen is shown when a recipe is clicked. This is the detail view of a recipe. A collapsing toolbar is at the top with action buttons, which are: “add to weekly meal board”, “edit”, and “share”. Below them is a rating bar for the user to rate the recipe, bookmark button for the user to save this recipe, and servings editable text for the user to change the amount of ingredients needed to cook this meal. Ingredients are listed with icons on their right side indicating if the ingredient is missing in the storage or not; if there is a missing ingredient, the user can add it on the shopping list. Steps will have a bell icon if there is a timer set for them; the user can start the timer by clicking on the icon. Pressing the “Finished” button will reduce the amounts of ingredients from the food storage, so that the user can track their food.

Screen 13



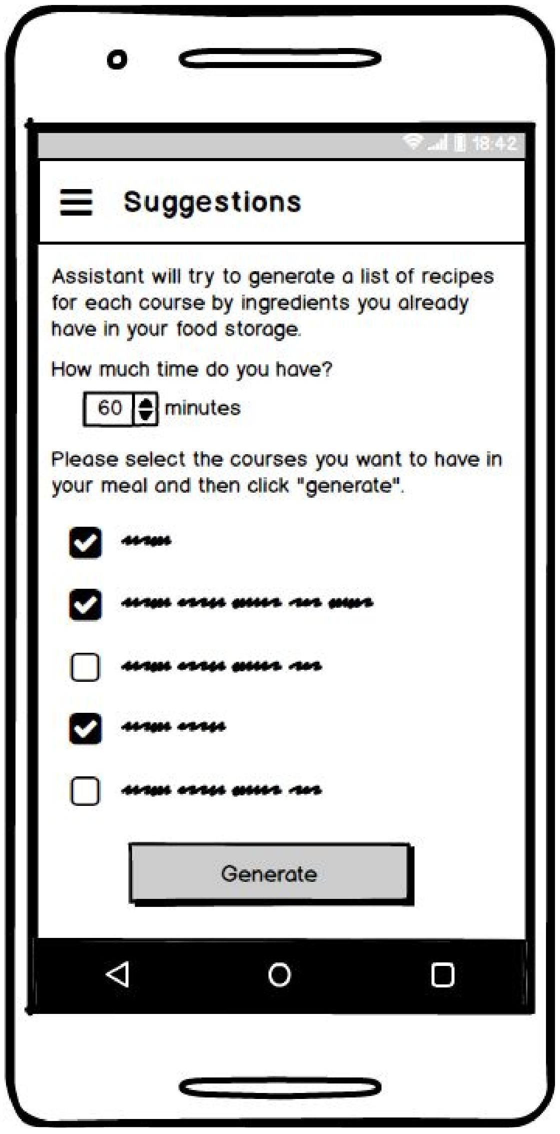
Clicking on the “add to weekly meal board” action button on toolbar, which has a table icon, will lead the user to this screen. This is a dialog box for user to decide which day and meal they want to add this recipe into.

Screen 14



This is the navigation drawer that user will use to navigate through screens.

Screen 15

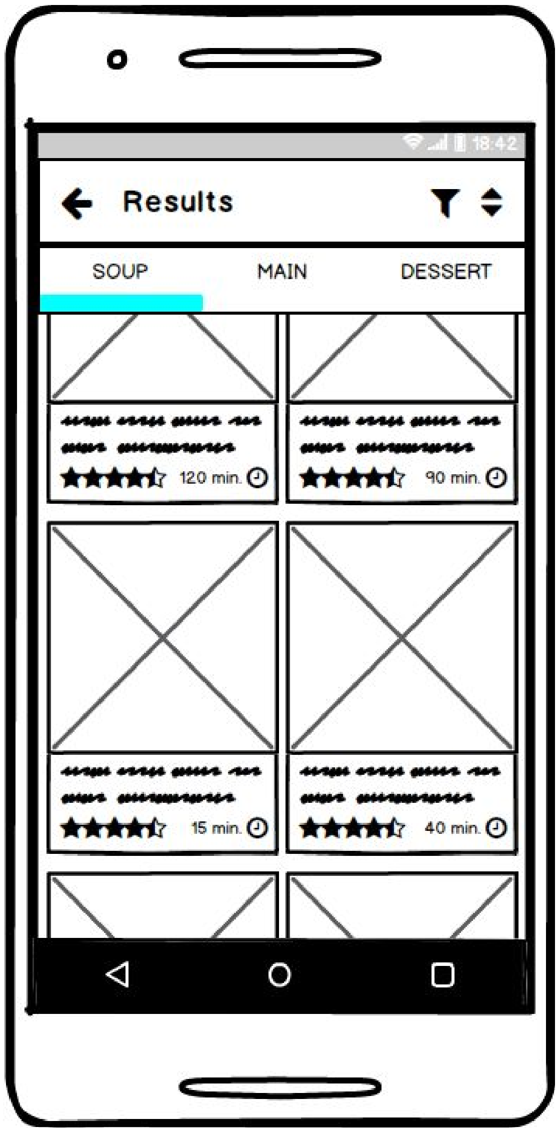


Assistant will try to generate a list of recipes for each course by ingredients you already have in your food storage.

Please select the courses you want to have in your meal and then click "generate".

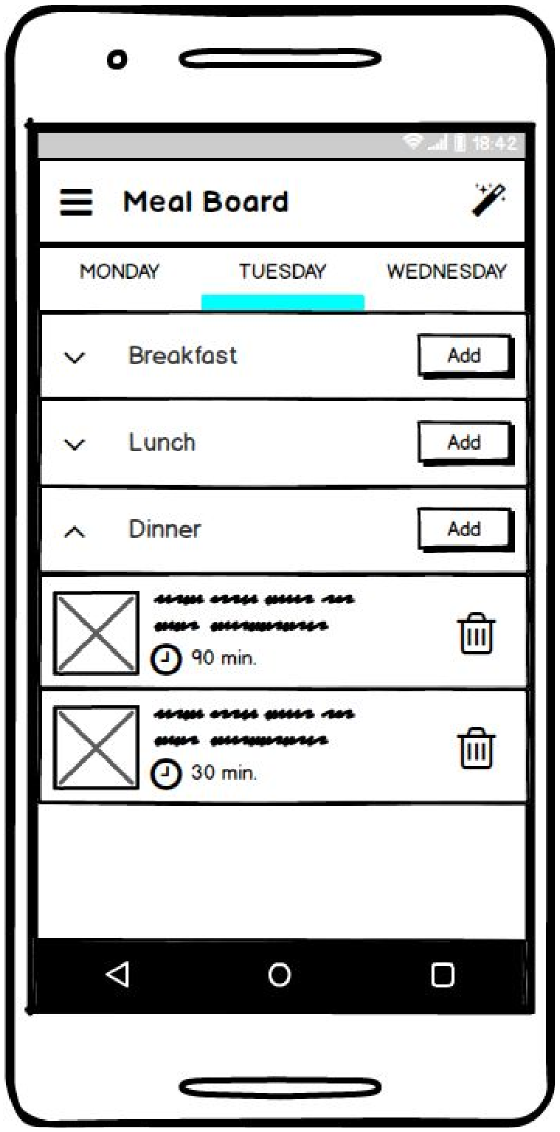
Imagine that the user comes at home from a hard day at work and has only one hour to prepare a meal for their friends. There is no time for shopping; that means they need to cook with whatever they have at home. This screen asks them how much time they have and which courses they want to present. Clicking the “Generate” button will make a search on the database by using the ingredients as the parameters. Those parameters are defined as soon as the food came into the house. The search will lead them to the Screen 16 (Results).

Screen 16



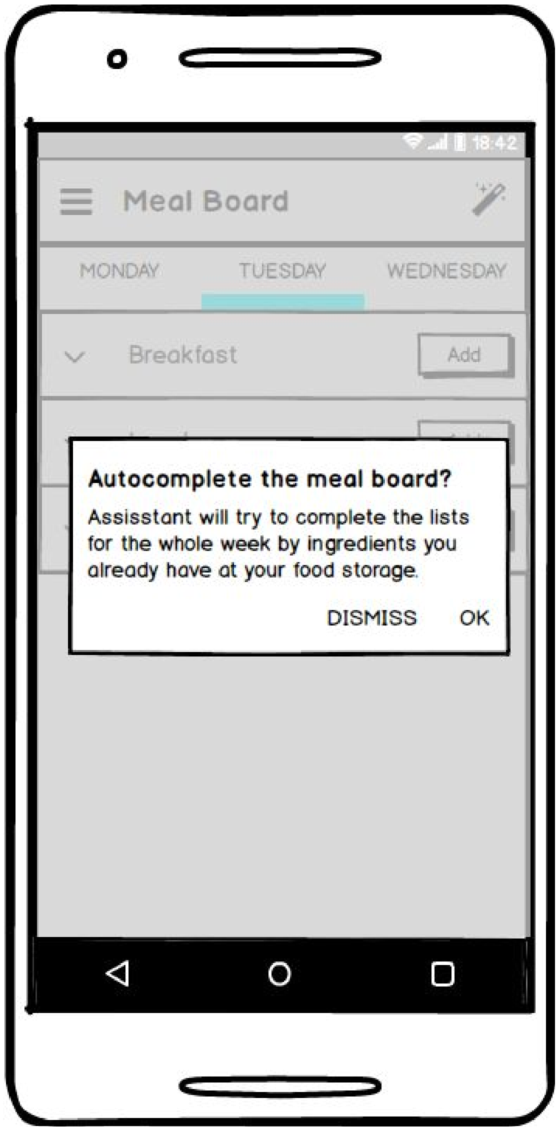
This screen shows each course as a tab and from there the user can select and cook whatever they want. Filtering and sorting the results are possible.

Screen 17



This is the weekly meal board. Each tab is a day of the week. Each day consists of meals and each meal consists of courses. One recipe is chosen for each course. Meal board is filled by three ways. Clicking on the “Add” button on meals is the first way. Second is clicking on the magic wand on the app toolbar, this will open a dialog as in Screen 18. Third way is explained in Screen 13.

Screen 18

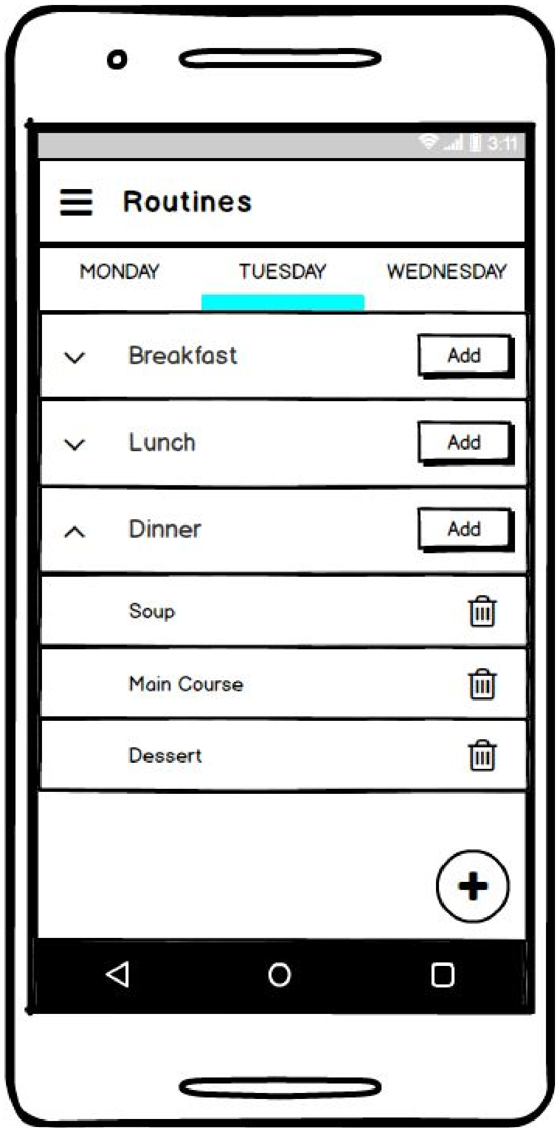


Autocomplete the meal board?

Assisstant will try to complete the lists for the whole week by ingredients you already have at your food storage.

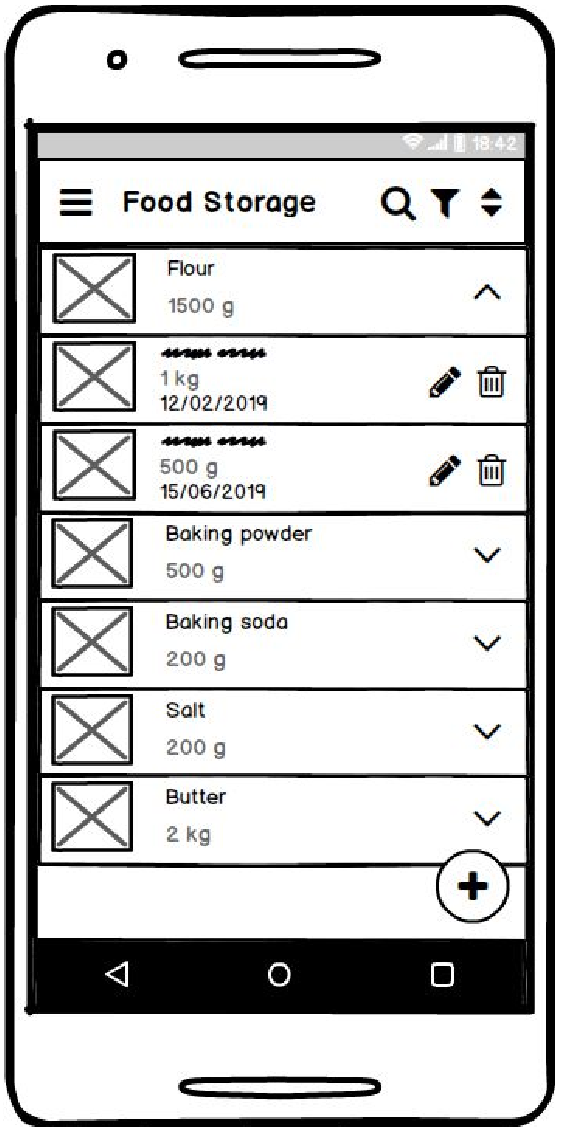
By clicking on the magic wand, a dialog pops up asking for permission to autocomplete the meal board. Autocompletion uses a template which is defined in Screen 19 (Routines). Using that template, the app will fill each course with a recipe considering the ingredients in food storage starting with the first meal of the week until there is not enough food to make a full recipe or all meals of the week are filled.

Screen 19



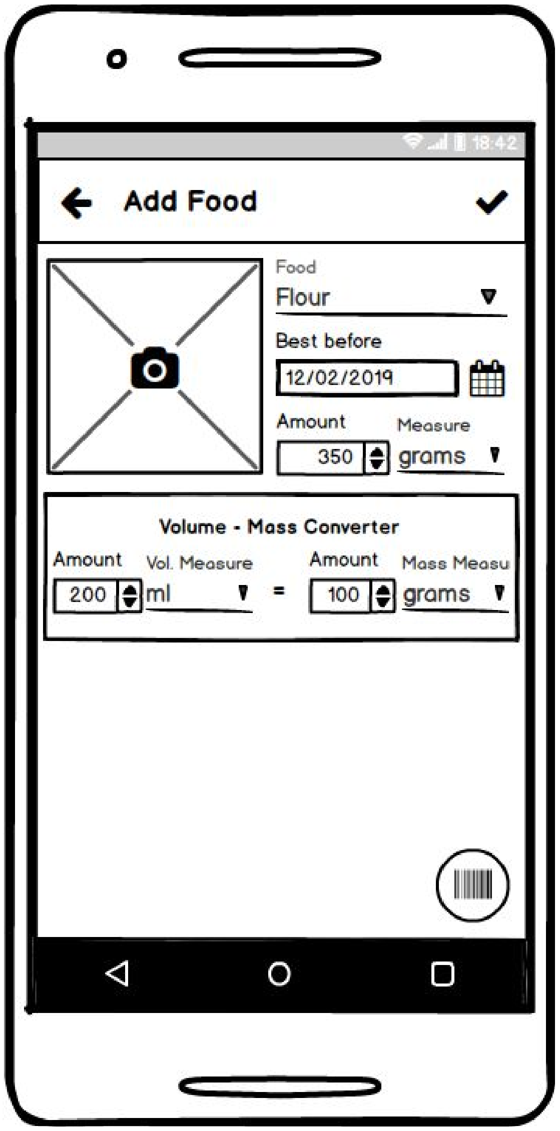
Purpose of this screen is to build a template for autocompletion of the weekly meal board. Floating action button adds a new meal for the day via a dialog which lets the user select the meal. “Add” button on each meal helps add a course into the meal in the same way the FAB does.

Screen 20



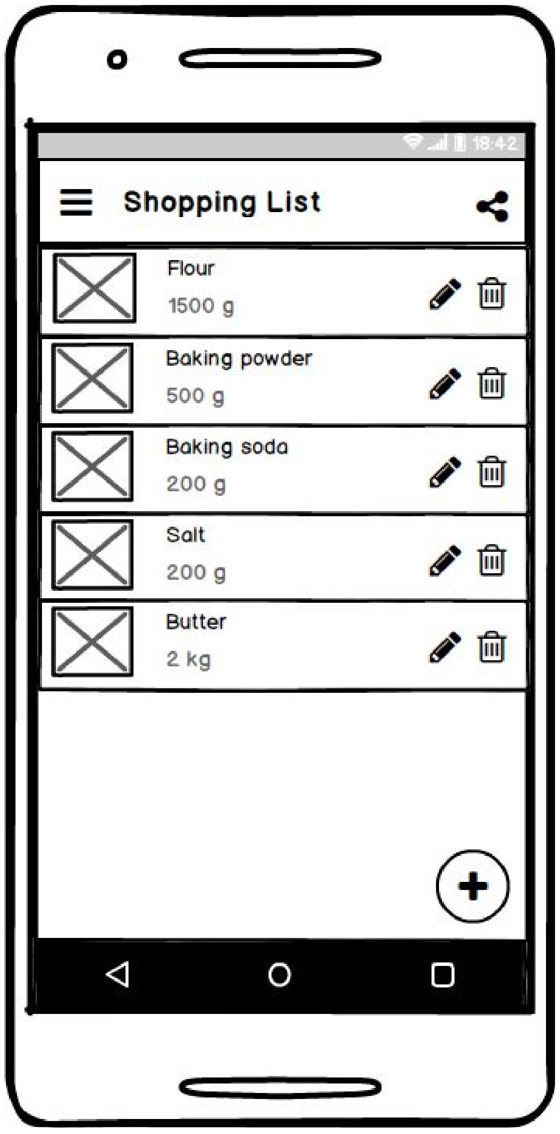
Using this screen, the user tracks the food in the storage - at home. FAB leads the user to Screen 21 (Add Food). Clicking the edit button, which has a pencil item, leads to the same screen. Searching, filtering, sorting, and deleting the items are also possible.

Screen 21



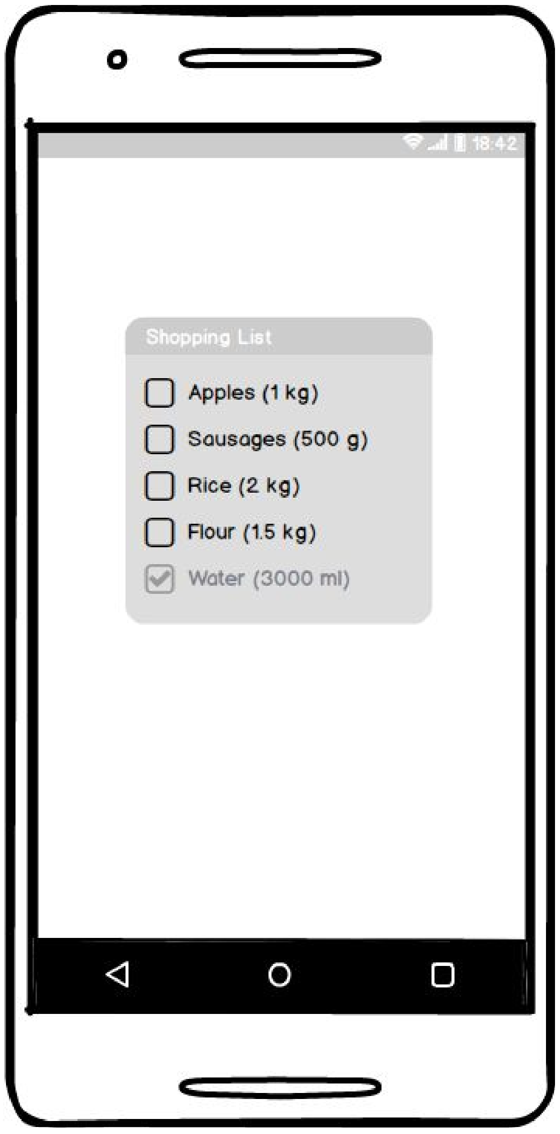
This is the screen where the user can add a new item to or edit an item from the food storage. By clicking on the picture, the user can take a picture of the item and add it to the database. Volume - Mass converter calculates a multiplier and saves it in the database for a later use to convert the measurements from the recipes into the measurements in the food storage. This is needed because recipes usually define volume measurements while the app stores food measurements in both volume and mass. FAB will lead the user to the barcode scanner view, which is only the view of the device's camera. As soon as the scanner reads the code, the app will post it to the web api and if it turns a product information, the fields will be populated automatically.

Screen 22



The shopping list is generated automatically considering the items in the food storage and the need of ingredients to cook the meals for the whole week. The user can edit and share this list with anyone who will go shopping with their smartphone.

Screen 23



Widget layout for the app. The shopping list is shown on the widget. As soon as an item is checked, it goes to the bottom of the list.

Key Considerations

How will your app handle data persistence?

App will store data locally by using Room Persistence Library. Besides, Firebase Realtime Database will be used to store common data among users.

Following data will be stored in local storage:
- Recipes which are in *the Notebook*
- Directions of the recipes which are in *the Notebook*
- Ingredients of all recipes
- Food that user has at home
- Shopping list
- Routines of the user
- Meal plan for the week

LiveData and ViewModel will be used when required and no unnecessary calls to the database will be made.

Following data will be stored in Firebase Realtime Database:
- Recipes uploaded by users
- Directions
- Ingredients
- Recipe ratings
- Food library

Describe any edge or corner cases in the UX.

App will validate all input from servers and users. If data will not exist or will be in the wrong format, the app will log this fact and will not crash.

Without internet connection, app will not load recipes and the barcode scanner will not turn a result for a product name. App will handle these situations.

App will not crash if the user will not give camera permissions.

Users might bookmark a recipe from public database and so that save them in local storage. This will give them an ability to edit those recipes and publish them again. App will prevent users from publishing the same recipe without any changes.

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

Android Support Library v27.1.1 to provide newer features on earlier versions of Android.
Constraint Layout v1.1.2 to create large and complex layouts with a flat view hierarchy.
Room Persistence Library v1.1.1 with Lifecycle v1.1.1 to access local database with lifecycle aware components.
FirebaseUI Auth v4.1.0 to handle the flow of signing in users with an email address or with Google identity provider.
FirebaseUI Storage v4.1.0 to download an image file stored in Cloud Storage.
FirebaseUI Database v4.1.0 to bind data from the Firebase Realtime Database to the app's UI.
Glide v4.7.1 to download, cache, and display images.
Retrofit v2.4.0 to make http calls.
Gson v2.8.5 to convert a JSON string to an equivalent Java object.
Butterknife v8.8.1 to bind views and ignore boilerplate code.
me.dm7.barcodescanner:zxing:1.9.8 to read barcodes.

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

Firebase Realtime Database: Users will upload and download data they want to share. These common data will only be accessed by authenticated users.

Cloud Storage for Firebase: App will store the photos of the recipes and foods. Each recipe and food will have only one photo. Users will upload these photos and app will download them when user sees a list of recipes, foods or details of a recipe.

Firebase Authentication: Users will need to authenticate in order to access Firebase data.

Google Sign-In: One of the two methods that the app will use for users to authenticate. Other method will be email - password authentication.

UPC Database: Scanned barcodes will be posted here to get product names in return.
(https://upcdatabase.org/api)

Next Steps: Required Tasks

Task 1: Project Setup

App is written solely in the Java Programming Language and all app dependencies are managed by Gradle. Each service imported in the build.gradle will be used in the app. App utilizes stable release versions of all libraries, Gradle, and Android Studio:

- Android Studio 3.1.3
- Android Plugin for Gradle 3.1.0
- The Google Services Gradle Plugin 4.0.1

Subtasks:

- Create a new project.
- Add app dependencies.

Task 2: Implement UI for Each Activity and Fragment

App theme extends AppCompat. App uses an app bar and associated toolbars, and uses standard and simple transitions between activities.

- Create first level activities.
- Implement FirebaseUI for Auth.
- Create remaining activities and fragments.

Task 3: Implement Local Data Persistence

- Create entities.
- Create Data Access Objects.
- Create TypeConverter for Date
- Create App Database
- Create repository.
- Create ViewModel.
- App will use an AsyncTask as it performs short duration, on-demand requests while inserting data to the local database.

Task 4: Implement Google Play Services integration

- Implement Firebase Storage to store pictures online.
- Implement Firebase Database.

Task 5: Implement app widget

- Add a widget layout and make use of it with shopping list data.

Task 6: Implement notifications

- Add notifications to warn the user about expired foods in the storage.

Task 7: Implement barcode scanning

- Make use of the barcode scanning library.
- Post scanned code to the web service and use the response to fill food metadata.

Task 8: Accessibility and internationalization

- All images include content descriptions.
- App provides navigation using a D-pad.
- App enables RTL layout switching on all layouts.
- App keeps all strings in a strings.xml file.

Task 9: Configure app building

- App is equipped with a signing configuration.
- The keystore and passwords are included in the repository.
- Build and deploy the app using the ‘installRelease’ Gradle task.
- Keystore is referred to by a relative path.
- App builds from a clean repository checkout with no additional configuration.