MyMealPlanner

GitHub Username: mregtej

Content

Description

MyMealPlanner lets you to make easy and healthy weekly meal plans. You can choose predefined meal plans from our available catalogue or create your own custom meal plans by using our meal planner calendar tool accompanied by our list of available recipes.

Our meal plans have been classified according to the amount of calories to facilitate you to choose the specific meal plan which best fits you and your diet needs.

All recipes provide a list of ingredients, the nutritional values and recipe steps which helps you to cook them easily.

The list of ingredients for your weekly meal plan shall be automatically added to the shopping list reminder tool which reminds you to buy the ingredients some days before the beginning of the week. You can modify the list of ingredients added in the shopping list reminder manually, set the reminder time frame and display the shopping list in a screen widget.

Intended User

MyMealPlanner is the perfect app for all people who want to have healthy meal habits and may need some help to save time in planning the weekly meals and getting good advices. It is also interesting for people which wants to have an "additional help" to remind the shopping list.

Features

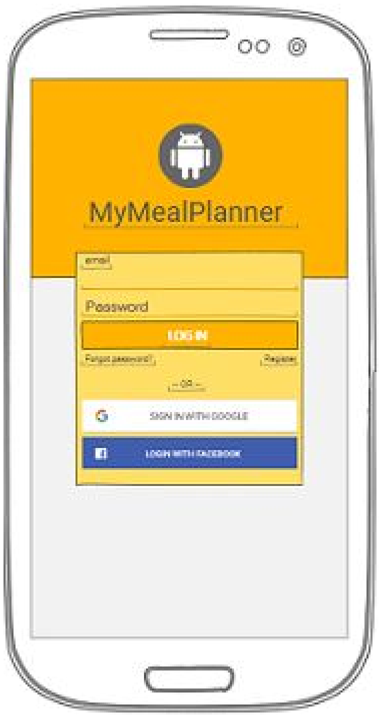
- Choose predefined weekly meal plans.
- Meal plans classified in different predefined categories.
- Filter predefined weekly meal plans by range of calories (Optional).
- Create custom meal plans by using the available list of recipes.
- Rate the available recipes (Optional).
- Display the weekly meal plans in a calendar view (monthly and weekly calendar views available).

- Available and updated weekly meal plans and recipes on cloud-based database (Firebase Realtime Database).
- Logging available via Google (Firebase).
- Share your weekly meal plans via email.
- Each recipe shall display a list of ingredients, the nutritional values and the recipe step list.
- Create automatically a weekly shopping list of ingredients when menus are set.
- Offer the possibility of remove ingredients from the shopping list manually, send the shopping list via email and to set the shopping reminder time frame.
- Offer the display of the shopping list in a Widget.

User Interface Mocks

https://ninjamock.com/s/DVJ7HTx

Screen 1: Log In

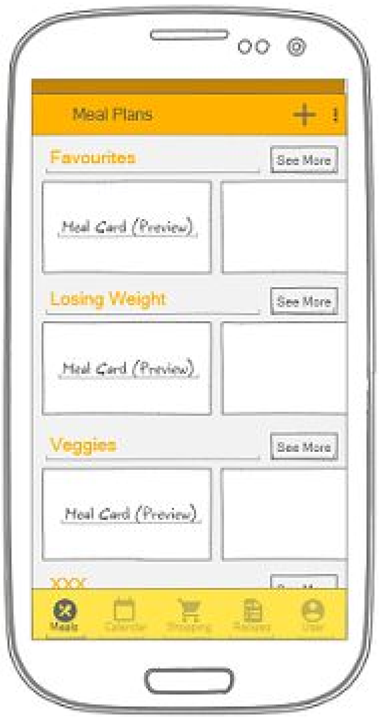


Log in screen (initial screen).

Supported log in via email, Google and Facebook (optional).

Firebase Authentication.

Screen 2: Meal Plans



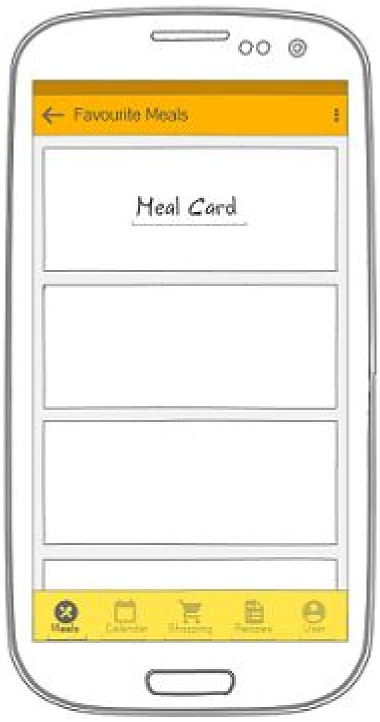
Meal Plans (initial screen after logged in).

Meals classified in different predefined categories.

Meal card clickable to open the Meal Menu screen.

Horizontally scrollable CardView Lists.

Screen 3: Meal Category (e.g. Favourites)

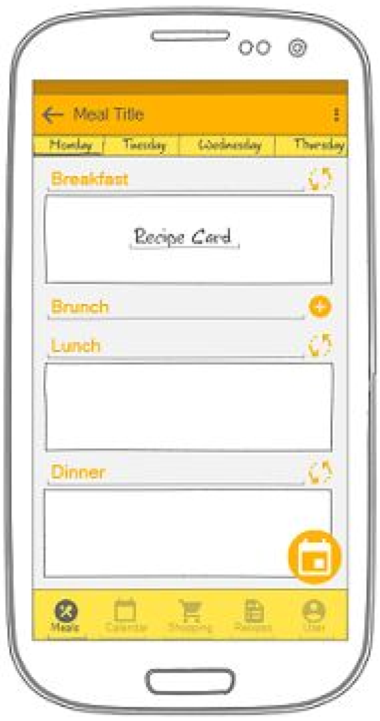


Favourite Meal Plans.

Vertically scrollable CardView Lists.

Backwards navigation to meal plans.

Screen 4: Meal Menu



Meal Menu.

Possibility to replace and/or add manually meals.

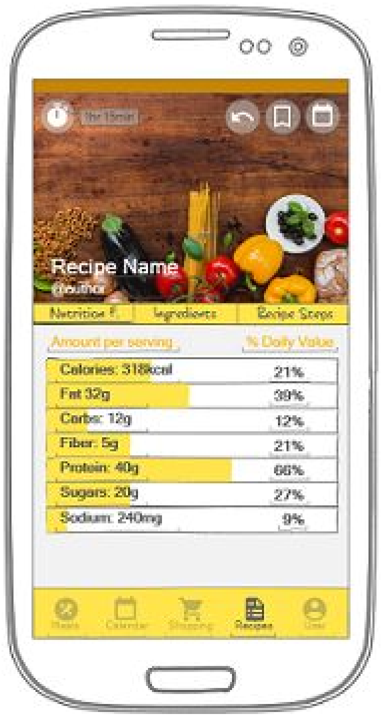
FAB button to add meal to calendar.

Recipe Card clickable to open the Recipe screen.

Vertically scrollable screen.

Backwards navigation to meal category (e.g. Favourites).

Screen 5.1: Recipe + Nutritional Facts Tab



Amount per serving	% Daily Value
Calories: 318kcal	21%
Fat 32g	39%
Carbs: 12g	12%
Fiber: 5g	21%
Protein: 40g	66%
Sugars: 20g	27%
Sodium: 240mg	9%

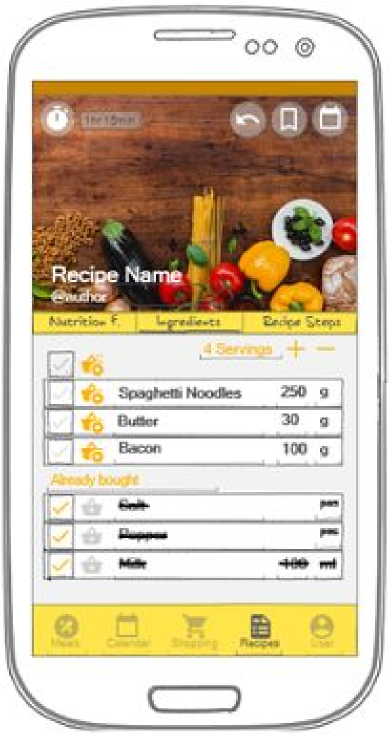
Recipe Screen. Nutritional Facts tab.

Backwards navigation to recipe list screen.

Bookmark button.

Planner button (Open calendar dialog for choosing day and afterwards, open the day planning dialog to set the meal).

Screen 5.2: Recipe + Ingredients Tab



Spaghetti Noodles	250	g
Butter	30	g
Bacon	100	g
~~Salt~~		pac
~~Pepper~~		pac
~~Milk~~	~~100~~	ml

Recipe Screen. Ingredients tab.

Settable number of servings (scalable amount of ingredients).

Add all ingredients to shopping cart button (and add single ingredient to shopping cart buttons).

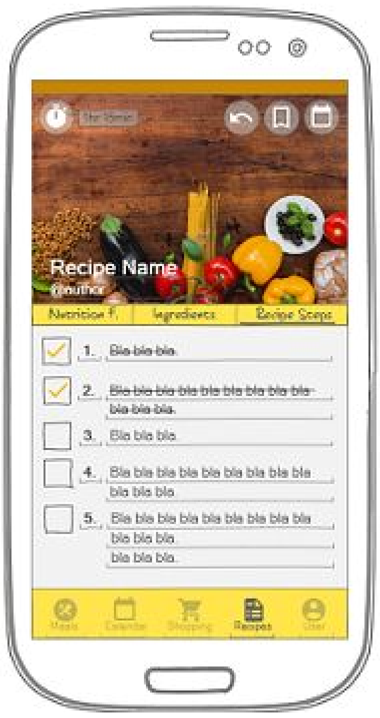
Select all ingredients button (don't need to buy them) or select each ingredient buttons.

Backwards navigation to recipe list screen.

Bookmark button.

Planner button (Open calendar dialog for choosing day and afterwards, open the day planning dialog to set the meal).

Screen 5.3: Recipe + Steps



Recipe Name
@author
Nutrition F.
Ingredients
Recipe Steps
1. ~~Bla bla bla~~
2. ~~Bla bla bla bla bla bla bla bla bla bla bla bla bla~~
3. Bla bla bla
4. Bla bla bla bla bla bla bla bla bla bla bla bla
5. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
Meals
Calendar
Shopping
Recipes
User

Recipe Screen. Recipe steps tab.

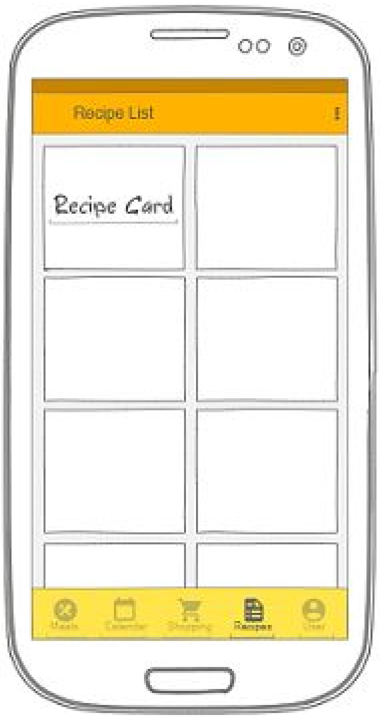
Done step buttons (mark step as completed).

Backwards navigation to recipe list screen.

Bookmark button.

Planner button (Open calendar dialog for choosing day and afterwards, open the day planning dialog to set the meal).

Screen 6: Recipe List

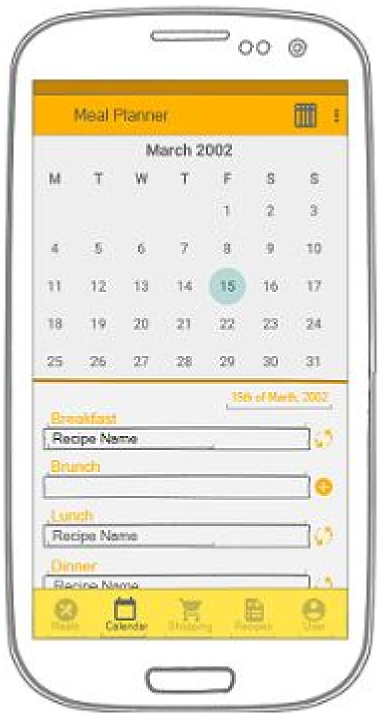


Recipe List Screen.

Recipe Card clickable to open the recipe screen.

Vertically scrollable CardView List.

Screen 7: Meal Planner



Meal Planner Screen.

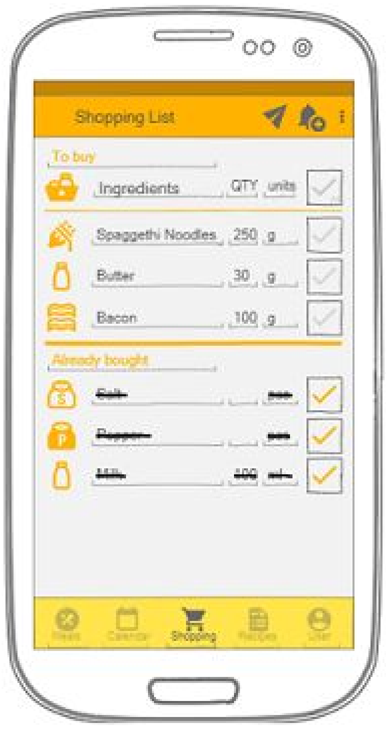
Calendar view. Choosable day.

Weekly/Monthly calendar view button on ActionBar.

Daily meals displayed. Possibility to replace/add meals (recipe choosable from recipe list).

Vertically scrollable daily meals view.

Screen 8: Shopping List



Shopping List Screen.

Weekly buyable ingredients displayed.

Add reminder button (notification alert programmable).

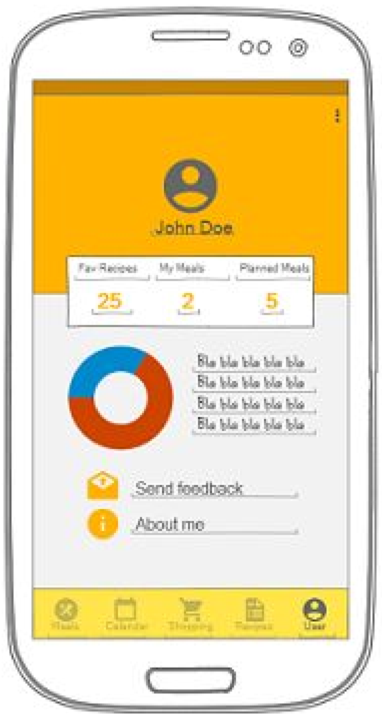
Send shopping list by email button.

Vertically scrollable view.

Mark all ingredients as bought button (and mark each ingredient as bought buttons).

Ingredient image displayed according to the ingredient category.

Screen 9: My Account



My Account Screen.

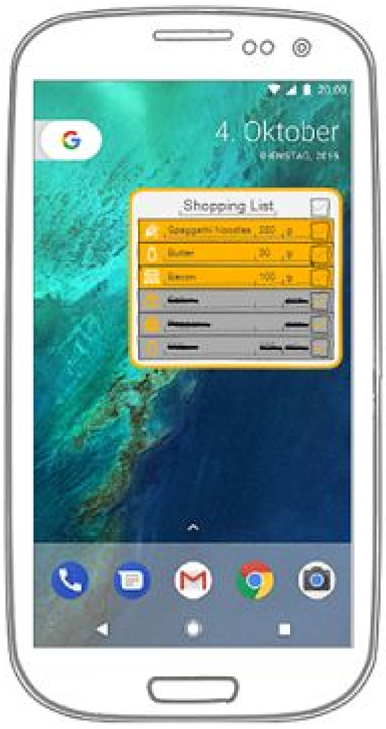
Displayed number off planned meals, number of fav recipes and number of created meals.

Displayed some statistics (tbd).

Send feedback button.

About me button.

App Widget



App resizable Widget.

Weekly buyable ingredients displayed.

Vertically scrollable view.

Mark all ingredients as bought button (and mark each ingredient as bought buttons).

Ingredient image displayed according to the ingredient category.

Key Considerations

Design and implementation constraints.

- App is written solely in the Java Programming Language.
- All displayed static texts shall be stored into strings.xml file (for also supporting future language translations).
- The app also includes RTL Layout support for all Activities (considering future language inclusions, such as Arabic, Hebrew, or Persian languages).
- The app includes support for accessibility (Image content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.

How will your app handle data persistence?

The cloud-based database shall be available, synced across all clients in real time, and remains available when the app goes offline thanks to (Firebase Realtime Database). The app will also be able to store data offline thanks to Room + LiveData.

Describe any edge or corner cases in the UX.

There is a bottom navigation bar which allows the navigation between screens 2, 6, 7, 8 and 9 by just tapping on the icons.

The app navigation implementation does not override the normal Android back button behaviour. “Backwards” navigation between screens is only available for these navigation flows:

- Screen 4 → Screen 3 → Screen 2.
- Screen 5 → Screen 6.

Additional dialogs created to facilitate the setting of a recipe to a specific dated meal and the setting of a weekly meal plan to a specific week.

When the app is offline (there is no active internet connection) a Snackbar is displayed at the bottom on the screen to warn user about it.

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

MyMealPlanner uses the following libraries and APIs:

- **Picasso (v2.71828)** library for handling the loading and caching of photos and photo previews.
- **Retrofit (v1.6.0)** as REST client for accessing to the cloud-based database.
- **ButterKnife (v8.8.1)** for UI binding.
- **Google Calendar v3 API**.
- Additional **Android Architectural Components (v1.0.0-alpha07)**.
    - **Room** for supporting local data storage.
    - **LiveData** for avoiding unnecessary calls to the database (the data are updated only when they change).
- **Palette (v27.1.1)** for extracting prominent colors from an image.
- **JUnit, Mockito or Espresso** for unit and integration testing.
    - Espresso 3.0.1, Runner 1.0.1, Rules 1.0.1, AndroidTestOrchestrator 1.0.1.
    - JUnit4 v4.13.
    - Mockito v2.10.0.
- **(+ Material Design)**

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

MyMealPlanner integrates the following Google Play API Services:

- **Firebase Realtime Database (v16.0.1)** to grant access to the meal plans and recipes stored into the cloud-based database.
- **Firebase Authentication (v16.0.3).**
- Optional: **Firebase Cloud Messaging (v17.3.0)** for sending messages to users about new updates and so on.

Next Steps: Required Tasks

Task 1: Project Setup

- Download latest stable version of Android Studio (v3.1).
- Setup SDK:
    - SDK Tools, Revision 26.1.1 (September 2017)
    - Android Emulator 27.3.8 (July 2018)

    - SDK Build Tools, Revision 28.0.2 (August 2018)
    - Android plugin for Gradle (v3.1.0)
    - SDK Platform: Android 9 (API level 28)
- Download latest stable Gradle version (v4.9) and set-up .
- Set-Up Google Play Services:
    - Firebase Authentication v16.0.3.
    - Firebase Realtime Database v16.0.1.
    - Cloud Messaging (Optional) v17.3.0.
- Configure libraries.
- Setup Gradle dependencies.

Task 2: Implement UI for Each Activity and Fragment

- Setup Main Activity and Navigation Drawer.
- Setup common UI elements for all activities and fragments (e.g. ActionBar + Settings).
- Setup main Activity + Fragment (UI modularity) frameworks (and transitions between them).
- Setup Tablet Support from UI mobile modular design.
- Build UI elements per each Activity and Fragment.

Task 3: Implement App Architecture Elements

- Implement the Model - View - Presenter pattern by using new Android Architectural Components.
    - Implement Repository class for switching between the offline (access to the local database) and online (sync from the cloud-based database) modes.
    - Implement ViewModels to store and manage UI-related data (and to allow data to survive configuration changes such as screen rotations)
    - Implement Local Database using Room + LiveData updates only when the data change.

Task 4: Implement Google Play Services

- Add Google Play Services.
- Implement accesses to the specific Google services: Firebase Database and Firebase Auth.
- Authorizing and Using REST APIs.

Task 5: Create Cloud-Based Firebase Database

- Install Firebase SDK and Firebase Console.
- Add the Realtime Database to your app.
- Configure Firebase Database Rules.
- Create Database and Provide Write / Read functionalities.
- Implement periodically pulls of meal plans and recipes from the cloud-based database.
    - Scheduling of asynchronous tasks such as downloading information from the cloud-based database and scheduling system service calls, such as the scheduling of shopping alarm notifications, are handled via SyncAdapter and/or JobDispatcher APIs.
- Configure ProGuard.

Task 6: Widget Implementation

- Design Widget UI (Shopping List).
- Implement WidgetProvider and Services (RemoteViewService) for updating the Widget content.

Task 7: Setup Additional Elements and Features

- Accessibility functionalities.

Task: App Testing

- Implement Unit tests.
- Implement Integration tests.