**How to Use this Template**

1. Make a copy [ File → Make a copy... ]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

**Submission Instructions**

1. After you've completed all the sections, download this document as a PDF [ File → Download as PDF ]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

---



**GitHub Username**: herrskytte

# Smileyface

## Description

**This application let's you check out your favorite restaurant and see the results from the latest food inspection!**

People are concerned about what they eat. Norway has a modern and efficient set of regulations that ensures that food safety is prioritised, every day of the year. Mattilsynet (Norwegian Food Safety Authority) has since the beginning of 2016 made their inspection results public for all Norwegian restaurants as a Smiley. You can see the inspection results in every restaurant.

Experience shows that errors can occur and that people can become sick from what they eat and drink. The Norwegian Food Safety Authority's reports ensures that the regulations are being observed. The consequences can be reduced food safety and an increased probability of guests becoming sick from the food.

It is unfortunate when the majority of food outlets employ a lot of resources and work hard to fulfil the requirements that have been imposed. The introduction of Smiley's in the food service industry has therefore been something that consumers have wanted for a long time. Once the result of the inspection is communicated through a Smiley symbol, there is an extra stimulus to making an all-out effort to create a consistently hygienic business.

Smileyface lets you search for a restaurant, street or city to see the latest results from food inspections. When was the restaurant last inspected? Has it previously had bad results or just a streak of happy smileys? Smileyface gives you all this information in a handy application, so you will know all you need about food safety before you even leave your house!

When you have found a place you like the application lets you see where it is in Google Maps and gives you directions if you wish.

Additional notes:
The app is based on recently released open data from Norwegian food and safety authority (http://data.norge.no/data/mattilsynet/smilefjestilsyn-p%C3%A5-serveringssteder). I have worked on this project on the inspector side so I know what the data means. The smileys and data will make a lot more sense to Norwegians as we are used to the data and smiley system, but no app like this is made yet.

## Intended User

Anyone eating at restaurants in Norway, both locals and tourists

## Features

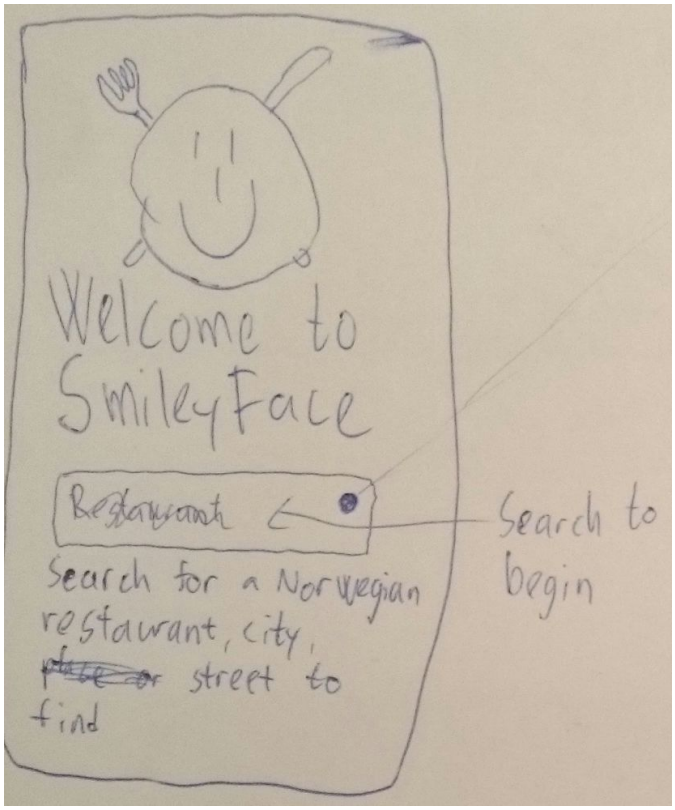
Main app features:

- Check restaurant inspection results
- Search name, place or street
- See restaurant inspection details and previous results
- Check restaurant address, see it in Google maps and navigate there

## User Interface Mocks

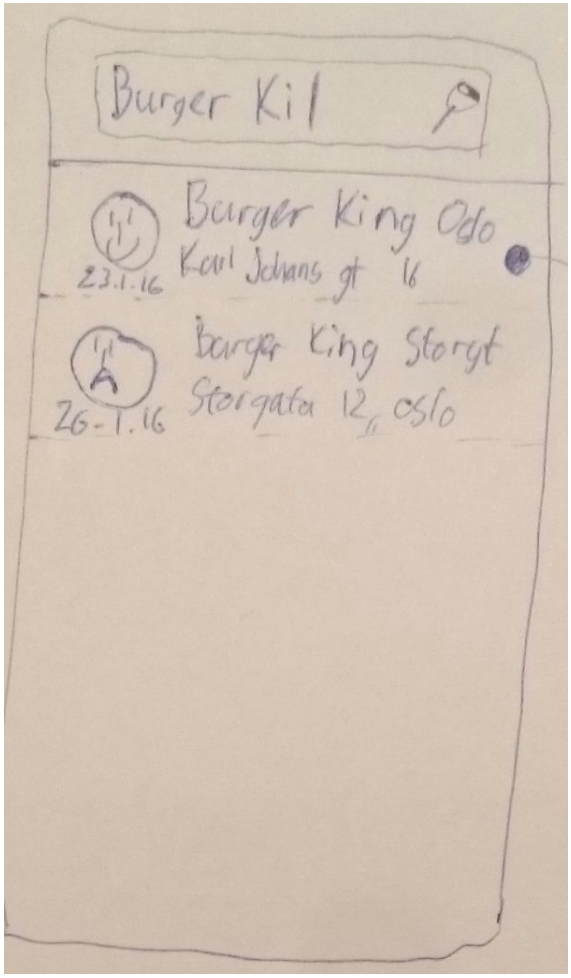
Created by hand, sorry for the poor drawing quality. Hopefully the flow of the app will be clear.

### Screen 1




A nice welcome screen with the app name, logo and explanation for the main app feature (search)
When the user inputs texts in the search-box the app will automatically transition to the next screen with the cursor and text in place

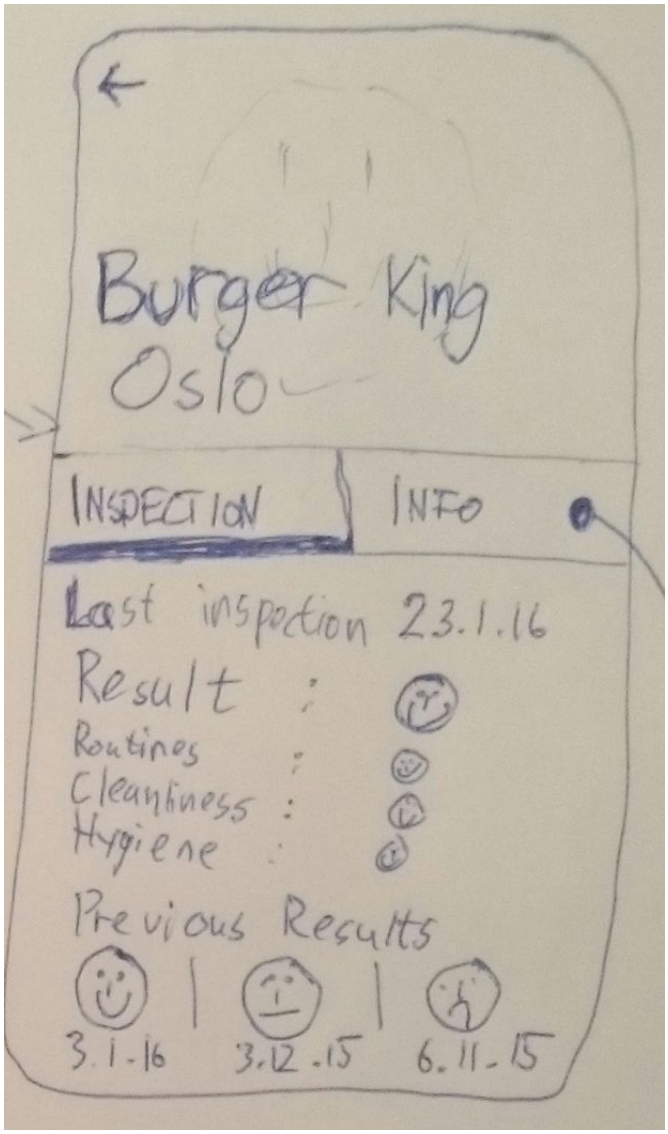
## Screen 2




The main screen of the app is a list of restaurants, filtered by the text input in the search box. The filter applies to restaurant name, and any part of the address

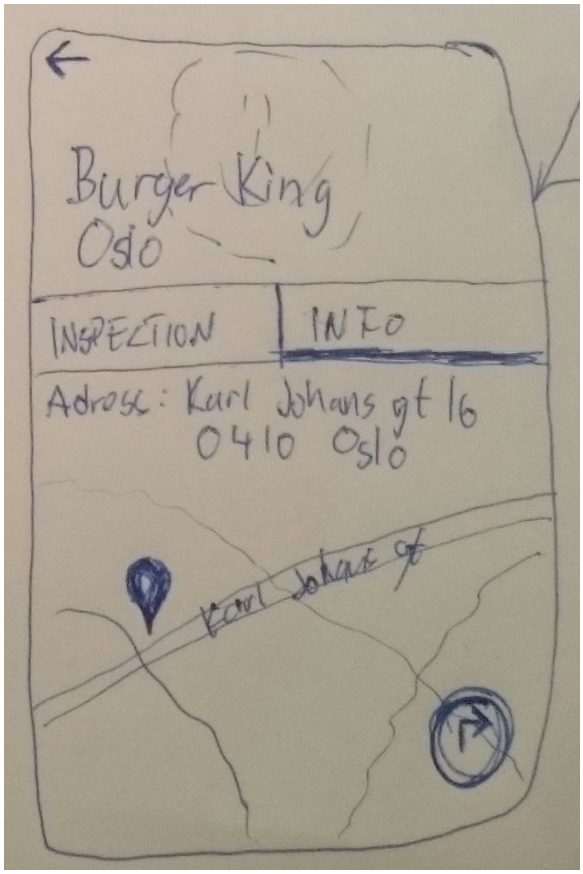
For each restaurant you can see the latest smiley-result (result of last inspection), the date for this result, name of restaurant in large font, and address in smaller secondary font

## Screen 3



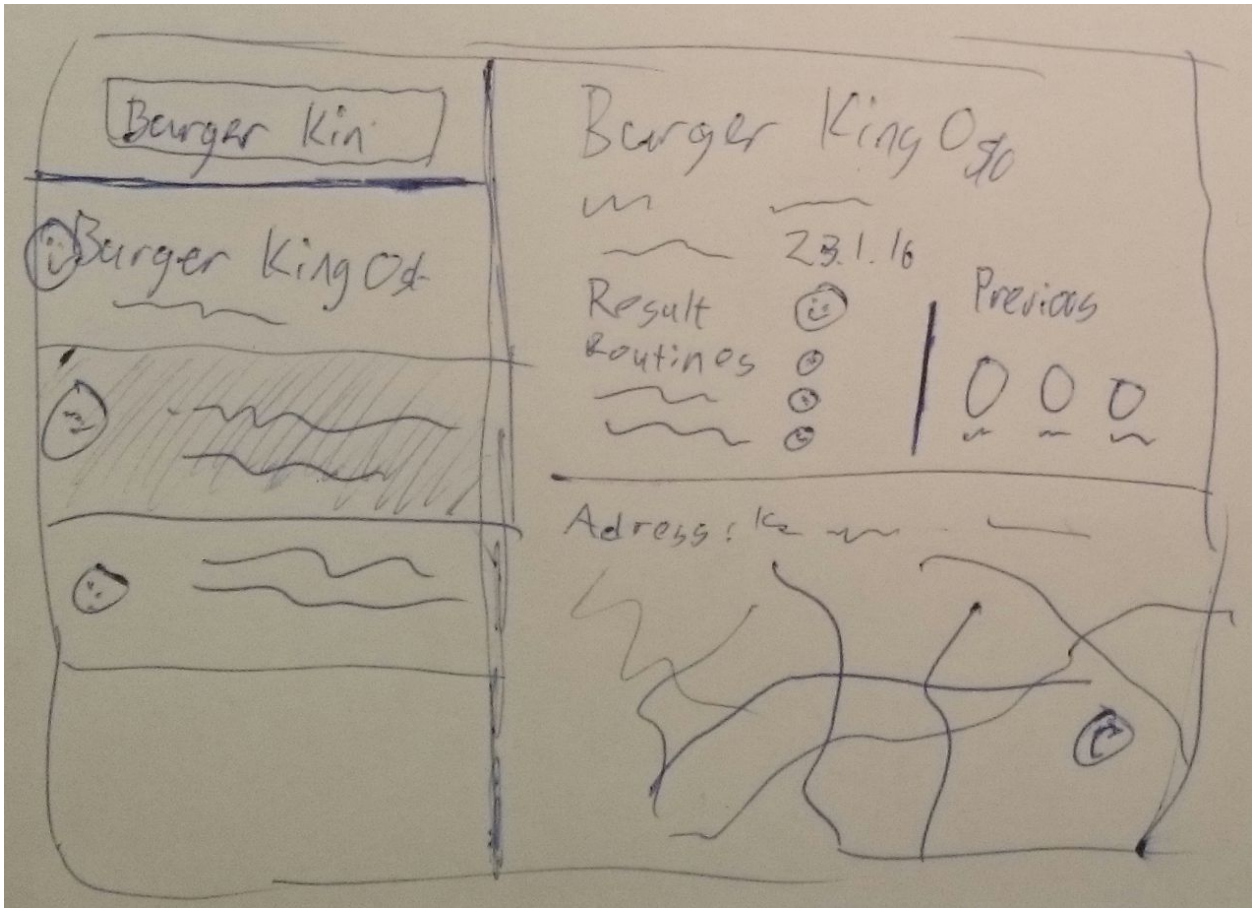

A details screen for a restaurant. This screen is divided in two parts with tabs. The first part shows the inspection details, with the result from the last inspection as well as individual results for the different categories (4 categories total). The screen also shows the previous results if any (0-3 results). A large version of the smiley-result will also be displayed faded as a background for the toolbar on top, with nice parallax scrolling.

## Screen 4




The second part of the app will use google maps to display the address on the map. Since the data does not include coordinates, I will need to do a reverse lookup first to get the coordinates. A floating action button will launch a navigation intent to allow the user to navigate there using for instance google maps.

### Screen 5




The tablet version of the app will merge all three parts of the app in one screen, showing the search/list on the left side, the inspection details on top and a map at the bottom.

## Key Considerations

### How will your app handle data persistence?

The app will implement a content provider and sync all data with open data server to make data available offline (the total dataset is about 1.4 MB). On refreshes I will sort the data by the unique id, so I can insert only new inspections. The app will refresh data on launch (using sync adapter to constantly sync data will be to much as the app will probably see casual use).

### Describe any corner cases in the UX.

No internet connection will need to be handled by providing the user with feedback, specially the first time, as the app is useless without data. Attempting to launch an intent to navigate to a restaurant without having google maps or with location turned off can also be a poor experience, but probably an expected one. Maybe consider hiding the navigate-FAB if the intent is unresolved (I think this can be checked).

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

Butterknife: easy gui-wiring
Retrofit: to connect to api to download inspections
Gson: parse the downloaded results
Google-services-location: to enable maps in the application
Google-services-analytics: to see usage statistics, most viewed restaurants, how often people use the app etc.

# Next Steps: Required Tasks

## Task 1: Project Setup

- Create project and deploy to Git. Use master/detail flow as a template to get good support for phone/tablet differences.
- Create google developer project to enable play-services (maps and analytics)

## Task 2: Setup data sync and storage

- Create content provider with tables for inspections and a separate table to store geolocation for where this has been looked up, so it only has to be done once for each restaurant
- Setup an intentservice that launces on startup, connects to the open data server and downloads data.
- Store data in content provider

## Task 3: Implement UI for list

- Build UI for welcome screen including logo graphic
- Build UI for MainActivity and listfragment (Use searchview in toolbar and create a recycleView to show the list of results

- Use a loader to fetch data from contentprovider and refresh the loader on changes in the search query
- Implement nice listitem view for each restaurant
- Create graphics for 4 different smileys (small to display in list and large to display in details later)

## Task 4: Implement UI for details

- Create layout for activity with toolbar and a fragment viewpager (with swipe naviagtion left/right)
- Create fragments for inspection details and a map to be displayed in the viewpager
- Create loader in inspection detail to load data and previous results
- Setup map service to lookup address, display location on map with correct zoom level
- Setup FAB in map activity with intent to navigate to address

## Task 5: Implement tablet UI

- Use a two-pane layout to display search-list on the left and details on the right
- Create a details view that combines the two fragments used in phone details. Maybe the inspection details will need to be a custom layout for tablet

## Task 6: Implement analytics

- Connect the app to analytics and log screen-hits and events for selecting a restaurant or starting navigation

## Task 7: Production ready

- Test app for different configurations and screen sizes
- Check that RTL works, content descriptions are filled in etc
- Test corner cases
- Create production keystore and make a release-build
- DONE :)