

Software Engineering and Software Qualities 2018

Streetfood app: strEAT

Final Portfolio

Lewis Bails - leba@itu.dk¹, Marie Louise Gabriel - mgab@itu.dk¹, Jowita Podolak - jopo@itu.dk¹, René Haas - renha@itu.dk¹, Anders Bruzelius - abru@itu.dk¹, Ida Marie Borberg - idbo@itu.dk¹, Katrine Iversen - kati@itu.dk¹, and Silviu - me@somewhere.com²

¹Software Development (Design), IT University Copenhagen

²Supervisor*

November 29, 2018



strEAT

GLOBAL LOCAL FOOD

*Benevolent Dictator for Life

Contents

1 Introduction	8
2 Team Protocol	10
2.1 Team Guidelines	10
2.1.1 Meetings	10
2.1.2 Workload	10
2.1.3 Handling Issues	10
2.2 Tools We Use	11
2.2.1 L ^A T _E X	11
2.2.2 Github	11
2.2.3 Google Drive (DEPRECATED)	12
2.2.4 Trello	12
2.2.5 Prezi	12
2.2.6 Slack	13
3 Case Exploration	15
3.1 Case Description	15
3.2 Katrine	15
3.3 Ida	15
3.4 Jowita	15
3.5 Marie-Louise	16
3.6 René	16
3.7 Lewis	17
3.8 Anders	17
4 Kick-Off	19
4.1 Interviews	19
4.2 Søren from Fresheet	19
4.3 Oscar from Baobab	20
5 Software Process Models	23
5.1 The Waterfall Model	23
5.2 The Spiral Model	24
5.3 The STEPS Model	24
5.4 Agile Software Development	26
5.5 The Pros and Cons of Each Model	27
5.6 Our decision	28
5.6.1 What kind of model do we need?	28
5.6.2 Which model have we chosen and why?	28
6 Scrum	31
6.1 Scrum Artifacts	31
6.1.1 Product Backlog	31
6.1.2 Sprint Backlog	31

6.1.3	Iteration	31
6.2	Scrum Team	31
6.2.1	Product Owner	31
6.2.2	Scrum Master	31
6.2.3	Development Team	31
6.3	Scrum Events	32
6.3.1	Sprint	32
6.3.2	Sprint Planning	32
6.3.3	Daily Scrum	32
6.3.4	Sprint Review	32
6.3.5	Sprint Retrospective	32
6.4	Conclusion	32
7	Roles and Responsibilities	34
7.1	Sprint Cycle	34
7.2	Scrum Roles and Responsibilities	34
7.2.1	Product Owner	34
7.2.2	Scrum Master	35
7.2.3	Development Team	35
7.3	Scrum Roles and Responsibilities in our Project	35
7.3.1	Product Owner	35
7.3.2	Scrum Master	36
7.3.3	Development Team	36
8	Project Plan and Estimation	38
8.1	Planning With Scrum	38
8.2	Product Backlog	39
8.3	Estimation	39
8.4	Project Schedule	41
8.5	Sprint Backlog	42
9	Software Qualities	45
9.1	App-Specific Focus Points	45
9.1.1	Usability	45
9.1.2	Performance Efficiency	46
9.1.3	Functional Suitability	47
9.1.4	Reliability	47
9.1.5	Portability	47
10	Risk Analysis	49
10.1	Product Oriented Risks	49
10.2	Project Oriented Risk	50
10.3	Risk Prioritization	51
11	Explore the Use Context	53
11.1	Research	53
11.2	The Social Context	53
11.2.1	Potential user interviews: Two students age 25	54
11.2.2	Bar Manager Interview	54
11.3	The Technical Context	55

11.3.1 Geolocation	55
11.3.2 Rating Systems	56
11.3.3 Push Notification	57
12 Use Case Diagram and Scenarios	60
12.1 Scenarios	60
12.2 Actors	61
12.3 Use Case Diagrams	61
13 Rich Pictures	67
13.1 A World With strEAT	67
13.2 A World Without strEAT	68
14 Documenting Project Requirements	70
14.1 Introduction	70
14.1.1 Purpose	70
14.1.2 Scope	70
14.1.3 Overview	70
14.2 General Description	70
14.2.1 Product Perspective	70
14.2.2 Product Function	70
14.2.3 User Characteristics	70
14.3 Requirements Specification	71
14.3.1 Functional Requirements	71
14.3.2 Non-Functional Requirements	72
14.3.3 Product Backlog	72
14.3.4 Estimation	73
14.3.5 Sprint Backlog	73
15 Class Diagram	76
15.1 Classes, Attributes, Methods, and Interactions	77
15.1.1 View	77
15.1.2 Map	78
15.1.3 List	78
15.1.4 Account	79
15.1.5 Customer	79
15.1.6 Vendor	80
15.1.7 Order	80
15.1.8 Menu	81
15.1.9 Item	82
15.1.10 Sales	82
16 Interface Design Using Mock-Ups	84
16.1 Scenarios	84
16.2 Pen and Paper Mock-Ups	85
16.3 The Mock-Up Workshop	85
16.3.1 Revised Mock-Up.	86
16.3.2 Logo and Colours - Look and Feel	86

17 Dynamic Test Specification	90
17.1 White-box and Black-box Testing	90
17.2 Testing for the project	90
18 Quality Plan	93
18.1 Quality Assurance Strategy	93
18.2 Quality Assurance Techniques	93
18.3 Quality Goals	94
19 Software Architecture	97
19.1 Architectural Patterns	97
19.1.1 MVC & MVP	97
19.1.2 MVC vs. MVP	98
19.2 <i>strEAT</i> & MVP	98
19.3 Evaluation	98
20 Reflection	100
20.1 Outlining the Project	100
20.2 Development Process	100
20.3 Conclusion	101
A Appendix	103
A.1 Interview Guide	103
A.1.1 Customers	103
A.1.2 Street food Vendors	103
A.2 Bar manager, Tipi Bar Nørrebro:	103
A.3 Student 1:	105
A.4 Student 2:	106

Config management temptale



Configuration Management

Document title: Title of document

Version: vX.YY ^a v2.07

Github link: Direct link to the most recent version on github.

Trello card link: Direct link to the trello card for this document.

Responsible: List of responsables for the document

Status: Can be "not finished", "under review", or "finished".

Changelog:

DD/MM/YY	USERNAME	CHANGE
----------	----------	--------

Comments:

USERNAME	COMMENT
----------	---------

renha	This is the template for configuration management. This ConfigMan box should be updatef by team members after each update to the respective document
-------	--

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

Configuration Management

Document title: Introduction

Version:^a v2.01

Github link: [.../MainDocument/sections/Introduction.tex](#)

Trello card link: N/A

Responsible: mgba

Status: Under review

Changelog:

25/09/18	kati	Created
27/09/18	kati	Modified
18/11/18	renha	Added configuration management

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

1 Introduction

In this portfolio, we will go through some of the basic parts of the organisation and management involved in software engineering. Our goal is to develop a mobile application, and in order to do this, we need to find the best possible way of managing this process as a team. This portfolio will therefore cover the basic project management of our project. Chronologically, the first part will be dealing with a proper set up - dividing the roles and responsibilities in the group, finding the right project management tools, structuring the framework for our work and developing a concrete idea that the whole group can agree on. We will be exploring the management framework and thereby developing a plan, including more substantial areas like a project kick-off, a case exploration, research through interviews and, following up, a mock-up version of the app, to be tested by other students. The tests will afterwards be analysed and the product outlines adjusted accordingly. For clarification and insight, a use case diagram will be provided alongside dynamic test specification, a configuration management documentation, an exploration of software qualities, a specific quality plan and a class diagram. Finally, we will thoroughly reflect on the process and the obstacles we are imminently going to come across as a group. A conclusion will be drawn on both, our experience with the project and our collaboration as a group.

Configuration Management

Document title: Protocol

Version:  v3.06

Github link: [.../MainDocument/sections/Protocol.tex](#)

Trello card link: N/A

Responsible: Everyone

Status: Finished

Changelog:

04/09/18	kati	Created
23/10/18	renha	Updated Latex description
25/10/18	renha	Updated Trello description
18/11/18	renha	Added configuration management
23/11/18	leba	Updated formatting & text
25/11/18	leba	Moved case description into case exploration
29/11/18	abru	Updated formatting & text

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

2 Team Protocol

Our project team consists of the following members:

Marie-Louise Gabriel	mgab@itu.dk
Katrine Iversen	kati@itu.dk
René Haas	renha@itu.dk
Ida Borberg	idbo@itu.dk
Jowita Julia Podolak	jopo@itu.dk
Anders Bruzelius	abru@itu.dk
Lewis Bails	leba@itu.dk
Silviu Agafitei (supervisor)	sgeo@itu.dk

2.1 Team Guidelines

In order to facilitate teamwork, we have comprised a set of guidelines that will serve as a foundation for our cooperative work on the project.

2.1.1 Meetings

We have the following times reserved for group meetings each week:

- Tuesdays 1200-1230 Supervision with Silviu
- Tuesdays 1230-1400 Weekly SCRUM meeting

In addition to these meetings, additional meetings can occur on a week to week basis depending on necessity. It is imperative that such extra meetings are agreed upon between all group members. Preferably such an agreement is reached at one of the fixed meetings.

2.1.2 Workload

The team will be subdivided into smaller groups of varying size depending on the tasks at hand. The Tuesday meetings are used to coordinate and distribute the workload to these subgroups for the following week. On Monday evenings, every subgroup will upload their work to Google Drive or the GitHub repository, so that the other group members can review the work and provide feedback.

2.1.3 Handling Issues

- If there is a disagreement, we shall communicate with each other in a respectful manner.
- If a disagreement cannot be resolved in a way that suits all group members, then the resolution supported by the majority will be enforced.
- If a group member needs help with a given task, they shall ask the other members for help.
- Constructive feedback is encouraged.
- When reviewing the work of others, no changes shall be made without the consent of the author.
- Communication should always happen through the predetermined channel (Slack).
- If a group member encounters issues with L^AT_EX or GitHub, they shall consult the group in order to resolve the problem.

2.2 Tools We Use

During the project we will be using various tools. In this section we will briefly describe those tools and their purpose.

2.2.1 L^AT_EX

We use L^AT_EX for typesetting our portfolio. We agree with the following guidelines when typesetting L^AT_EX:

1. We can use the enumerate, itemize, figures and tables environments - nothing else, except (sub)sections.
2. All external references must be included in "references.bib" and called in L^AT_EX with the "\cite[p.123]{ID}" command.
3. Figures from external resources must be referenced with "cite" in the figure caption.
4. All sections, figures and other files that is a part of our repository must be named in a descriptive manner, so that they are easily identifiable. I.e. this section should not be name "section2.tex", but rather "Protocol.tex".

2.2.2 Github

We use Github for storing all our files in a common repository. Git is a version control system and has many advantages with regards to workflow and cooperative work. Our Github repository can be found at <https://github.itu.dk/renha/strEAT>

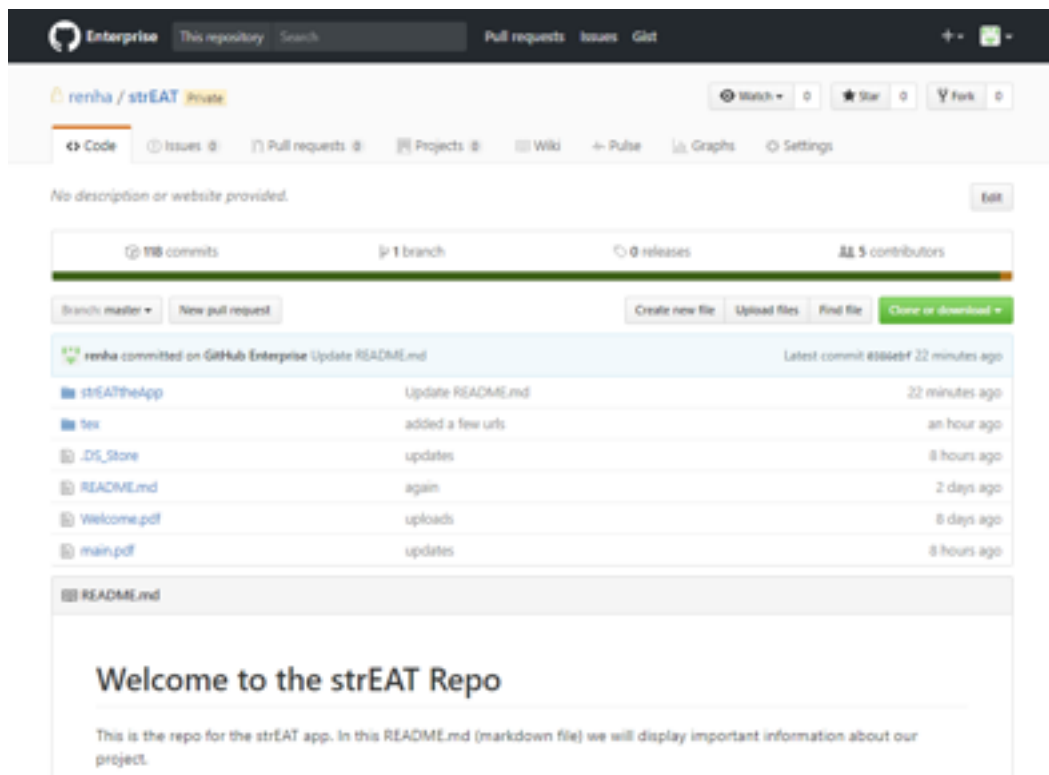


Figure 1: Screenshot of our GitHub repository as of October 19th 2018

2.2.3 Google Drive (DEPRECATED)

Initially we used Google Drive for cooperative writing of sections as well as storage of said sections. However, with the decision to use \LaTeX , and Google Drive not supporting this, all tasks previously supported by Google Docs were taken over by GitHub. As such we ceased to use Google Drive.

2.2.4 Trello

We use Trello for managing all tasks big or small throughout the project. Each task is put on the Trello board along with the group member(s) responsible for the task, and its deadline. Our Trello board can be found at <https://trello.com/b/yVBXG3Ek/strEAT>



Figure 2: Screenshot of our Trello board as of November 2nd 2018

2.2.5 Prezi

We use Prezi for making group presentations. <https://prezi.com/view/L1LnoppWfuWxw1L6RkDH/>



Figure 3: Screenshot of our Prezi presentation for our Rich Pictures and Use Context sections

2.2.6 Slack

Slack is the primary communication channel for our team. All important announcements *must* go through Slack. We also encourage team members to use Slack when communication about subtasks. Our Slack workspace can be found at <https://strEAT.group.slack.com/>

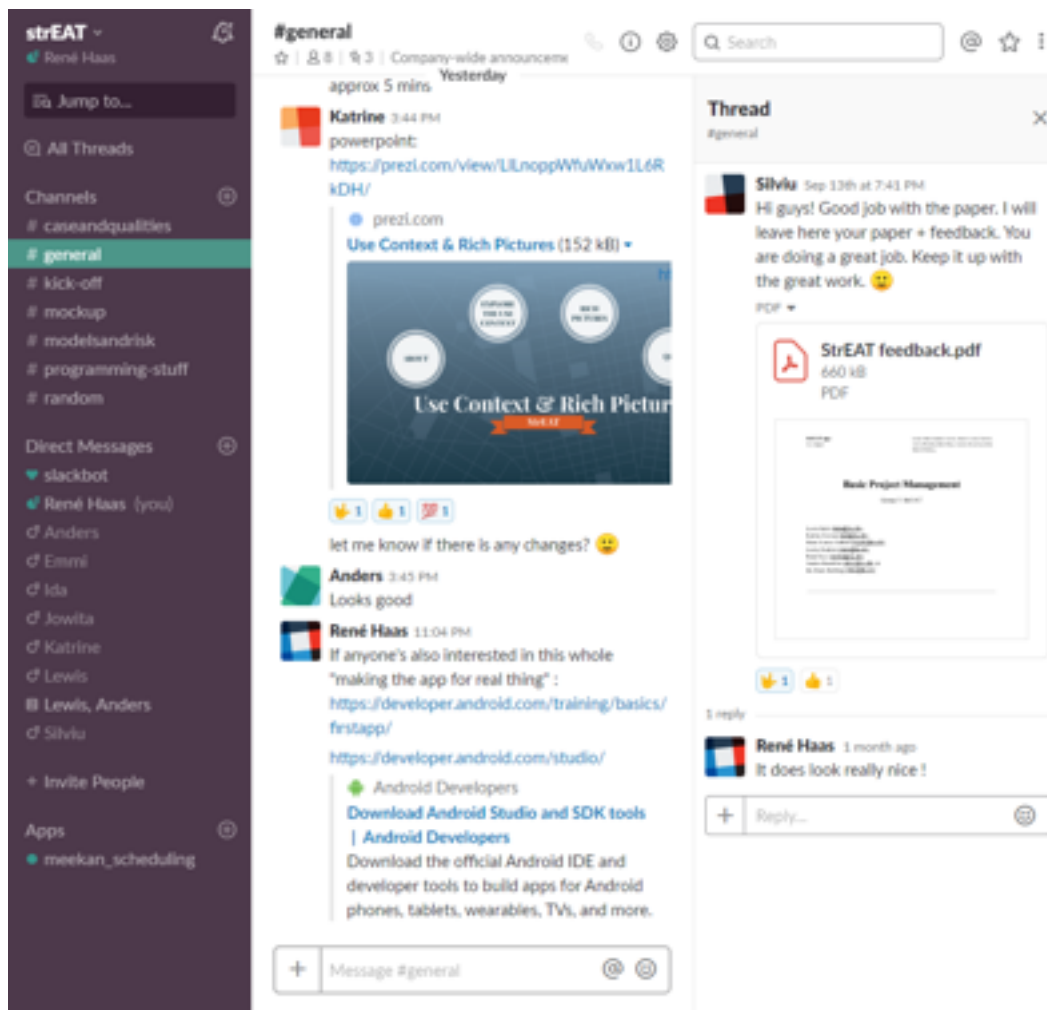


Figure 4: A screenshot of our Slack workspace as of October 19th 2018

Configuration Management

Document title: Case Explorations

Version:^a v3.14

Github link: [.../MainDocument/sections/CaseExploration.tex](#)

Trello card link: [trello.com/c/ZuWPvGHd/10-explore-your-case](#)

Responsible: Everyone

Status: Finished

Changelog:

09/09/18	mgab	Created
09/09/18	mgab	Added Marie-Louise's case exploration
09/09/18	jopo	Added Jowita's case exploration
09/09/18	idbo	Added Ida's case exploration
10/09/18	renha	Added René's case exploration
10/09/18	kati	Added Katrine's case exploration
10/09/18	leba	Added Lewis' case exploration
12/09/18	leba	Added references to Lewis' section
13/09/18	abru	Added Anders' case exploration
13/09/18	kati	Added references to Katrine's section
13/09/18	renha	Added references to René's section
14/09/18	renha	Updated René's case exploration
02/11/18	leba	Updated grammar
18/11/18	renha	Added configuration management
23/11/18	leba	Updated formatting
25/11/18	leba	Case description and formatting

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

3 Case Exploration

In this section, as we have deviated from this course's prescribed cases, we first present a case description. Later, each team member expands on the case description. Each team member has a different perspective on the case and may draw on skills from their former academic background.

3.1 Case Description

Street food is getting very popular in Copenhagen - for example, "Papirøen" has expanded and become "Reffen". Also, there are different vendors who move around in the city. At present, there is no platform for communication between street food vendors and hungry customers. This means that vendors are currently losing potential revenue and customers find themselves in a position where they cannot find the street food stall of their preference since they are moving around. Currently, the closest competitors in this market are sites like JustEat and Wolt, but they do not include smaller, non-stationary food companies at present.

3.2 Katrine

Street food has become a growing phenomenon in a range of urban centres all over the world. Compared to its roots in the developing world, this practice has consequently moved slowly into the developed parts of the world [23]. In Copenhagen, we've had a long tradition of the so-called "Pølsevogn"¹, which today is becoming a more extinct type of enterprise due to a change in Danish food culture. However, others would argue that it is a part of Danish cultural heritage. Today we are experiencing how the street food culture in Copenhagen is expanding rapidly, and also becoming a means for tourism. An additional important aspect to consider, which Irene Tinker [23] points to, is how urban street food has become:

“[...] a critical element in the survival strategies of the urban poor [...]”.

With this in mind, it seems like an interesting and relevant project to develop the street food app *strEAT*, which will hopefully be a great tool for street food vendors in this micro-enterprise business, as it will allow the customers to have better access to vendors.

3.3 Ida

In recent times, the UK has felt the popularity of street food, and its emergence is largely due to the heavy interest from investors [38]. Street food has also gained popularity over the past few years in Copenhagen. Copenhagen Street Food opened a street food market in 2014, at Papirøen (Visit Copenhagen), and moved to Refshaleøen in 2018. Street food trucks are also seen in Kødbyen Vesterbro, Copenhagen and Verdenshjørnet Nørrebro, Copenhagen. Some of the food trucks have a permanent spot, while others move around to different spots in Copenhagen. The food trucks without a permanent spot can be hard to find for the customer, and there is neither an app where the customer can find the food truck nor order food from (not even from the food trucks with a permanent spot). Since popularity is rising for street food and investors have found their way into the market, one could argue that there is a need for an app that is specific to street food trucks.

3.4 Jowita

Building a community willing to be a part of the app is one of our biggest challenges. Street food vendors are the very core of this community and their hesitation to join it must be taken into consideration.

¹Translates: sausage wagon

Besides the obvious advantages mobile ordering apps bring to the business, more and more restaurants point out problems that they brought to the operations. It even became a ‘fashion’ for the restaurants to cut ties with mobile ordering apps [24]. While restaurant owners admit that food delivery platforms like UberEATS increase the number of orders their businesses receive, a lot of them also say that these apps increase operational headaches behind the scenes and that they don’t have a significant effect on profit margins [21]. This is especially true when a food vendor juggles between multiple apps and websites, therefore it is vital that we convince street food businesses to focus on this one, tailored directly to their needs, *strEAT* app. To our advantage, street food vendors usually have lower revenue than regular restaurants and thus, unlike them, they don’t have the bandwidth to develop their own apps (restaurants’ own ordering platforms are becoming one of the biggest threats for food ordering apps). Nevertheless, in order to convince street food vendors to join us, we need to simplify operations for them as much as possible (for example, by communicating directly with kitchen staff).

3.5 Marie-Louise

During the last few decades, food has become one of the most present elements of people’s lifestyles that has reached its peak with social media enabling consumers to display the objects of consumption and the location of such a purchase. The majority of elements of people’s everyday lives have to follow aesthetic-driven marketing strategies that appeal not only to the eye but also to the Instagram feed. Hence, the demand for aesthetically pleasing food, and sustainable concepts behind it, has spread from the high street to smaller cafes, shops and markets. Street food markets are one of the most concise examples of this phenomenon. They’ve come a long way from being mainly frequented by workers in their short breaks, or offering only simple snacks, to now mostly being stylish venues with marketing strategies, investors and sophisticated food options, often catering to the needs of people with allergies or ethical food consumption guidelines. A melting pot for representative examples is the Reffen food market in Copenhagen that includes 70 food containers with innovative concepts. That offers a lot of choice, especially in combination with all the other individual food trucks and bikes in and around the Copenhagen area. As great as that is for the consumer, it is also confusing: there is no way for the (potential) customer to find information regarding not only vendor menus but also prices, locations and sometimes even opening hours. Not even Reffen, as the biggest concept, has any detailed information on their website. It therefore seems overdue for street food vendors to take a step further into modern marketing options and explore economical growth opportunities that come with a virtual network of customers and vendors alike, where information can be retrieved and the aforementioned elements are made more transparent. Delivery services in Copenhagen, like Wolt or JustEat [26], are currently covering most establishments but are neglecting food market trucks and stalls. Therefore, it seems a natural consequence for that sector of the market to be given a sufficient option to advertise and display their food, even when their potential customers are not on location, and, potentially, also to distribute it to the customer as to be fully able to compete with restaurants.

3.6 René

I want to use my time for the "explore my case" to investigate the current market a bit: Searching on Google, I found this app which does street food [3], which is in the direction we had in mind, running in Vancouver, Canada. There is also another variation on this idea which is available in Britain from “British Street Food Development”. They write on their website:

"It’s revolutionary, with live GPS maps showing who’s trading where and when. It details the specials of the best traders, and encourages punters to photograph – and review – their food. The idea is that it is helping to build a bigger street food community. And, for the first time this year, it allowed people to vote for their favourite traders at the British Street Food Awards." [16]

The (totally revolutionary) live GPS mapsTM is also a core feature of the app that we want to develop, and we have also discussed other modes of income for the company with features like reviews and photo service as possible extensions. We could also add an online shop which will empower our customers with the option to do catering. Street food is getting pretty cool in Copenhagen and notably we have “Reffen” as a hub for the Copenhagen based market and it will be a good place to pitch our idea for the kick-off. I’ve not been able to find any indication that others are working on this in Copenhagen.

3.7 Lewis

An estimated 2.5 billion people eat street food every day [13]. The culture, made popular by developing nations as a means of acquiring cheap food, now seemingly pervades every corner of the globe. In the US alone, the current food truck market, which is valued at USD 856m, is expected to rise to approximately USD 1b by 2020, having sustained an average annual growth of 7.9% since 2011 [12]. A figure far surpassing that of its brick-and-mortar counterpart, which stands at 2% [18]. Not one to be left behind, Europe has also begun to warm up to the idea of a burgeoning street food market. In the UK, vendors can expect to begin plating up for as little £2,000 [36], a relatively small investment which encourages competition and variety. Staying in Britain, 18-34 year olds are eating out approximately 25% of the time, well above the national average of 15% [42]. Of the £6.1b restaurant market, approximately £0.9b is attributed to street food and mobile van revenue, a 14.7% share. With nearly half of all internet traffic being attributed to mobile devices [32], 91% of smartphone users using apps [?], and street food consumption trending upward, the success of casual dining applications such as Just Eat, UberEATS, Wolt and Deliveroo are of no surprise. However, these services deal with delivery, and a gap in the market still remains for venues with no fixed location who are looking to attract new clients. This will be addressed by our new application, *strEAT*.

3.8 Anders

Food is one of very few “products” that have the amazing property of being indispensable to all people, all with their own tastes and preferences. As such, the market for food is naturally ripe with opportunities by catering to various niches of consumers. However, new opportunities usually lead to new challenges. With our app, we want to solve one of the inherent challenges related to the phenomenon of street food. Most people probably have a favourite place to eat, and while your preferred brick-and-mortar restaurant won’t be going anywhere in a physical sense; your favourite food truck is very likely to. With *strEAT* we want to address this, by enabling consumers to keep track of the street food vendors at all times using existing GPS technology. At the same time, we want to give the vendors a platform for communicating with their customers, and enable them to improve their business through this communication. Additionally, we want to give street food lovers a platform, not only for locating the vendors they know, but also to explore the market of street food and discover new vendors – and hopefully new favourites.

Configuration Management

Document title: Kick-Off

Version:^a v3.03

Github link: [.../MainDocument/sections/KickOff.tex](#)

Trello card link: [trello.com/c/aRGvKpZq/13-project-kick-off](#)

Responsible: renha

Status: Finished

Changelog:

??/??/??	?	Created
18/11/18	renha	Added configuration management
23/11/18	leba	Updated text
24/11/18	abru	Fixed broken reference

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

4 Kick-Off

For our project kick-off we decided to go to Reffen and contact our potential customers and, of course, have some very nice food, see figure 5. In general, there was a very optimistic attitude towards the project among the vendors. Several of them mentioned that the vendors at Reffen do not yet participate in any market platform as they will not tolerate a fee on transactions completed through third-party platforms.



Figure 5: A collage showing our trip to Reffen for the kick-off event.

4.1 Interviews

During the kick-off event we had a very nice time, ate some very good food, and met some interesting people. In the following section we will briefly report on two of the interviews we conducted during our kick-off event.

4.2 Søren from Fresheet

We met with Søren, of CPH Foodtech Community and founder of Fresheet.co, in his futuristic container at “Reffen GREENS”. Søren is growing “microgreens” with hydroponics and Arduino boards, which

he then sells to the vendors at Reffen. Søren really liked our idea and is available for questions if we need it. He has very good connections to the vendors at Reffen and has also started several companies himself. We got Søren's email and contact info.



Figure 6: Søren in his hydroponics container at "Reffen GREENS"

4.3 Oscar from Baobab

Baobab offers exotic African food with signature dishes including fruit from the Baobab tree which, interestingly enough, begins to hollow out itself when it reaches its first millenia, according to Oscar, the owner. Oscar mentioned that there is a strong opposition to services that put a fee on transactions (e.g. JustEat) which is why the Reffen community has not yet subscribed to these services. When we mentioned that the app would be a “flat subscription” with no fee on transactions, Oscar seemed very positive towards the project and recommended us to contact him when we have a prototype for the app.



Figure 7: Oscar working at the Baobab resturant.

Configuration Management

Document title: Software Process Models

Version:  v3.09

Github link: [.../MainDocument/sections/SoftwareProcessModels.tex](#)

Trello card link: [trello.com/c/yFINPq7o/12-choose-a-software-process-models](#)

Responsible: kati

Status: Finished

Changelog:

08/09/18	kati	Created
		Added "Introduction", "Agile...", "The pros and cons..." & "Our decision"
09/09/18	leba	Added "The Waterfall Model"
		Updated pros and cons & decision
09/09/18	abru	Added "The STEPS Model"
		Updated pros and cons & decision
09/09/18	jopo	Added "The Spiral Model"
		Updated pros and cons & decision
01/11/18	abru	Added Latex table
02/11/18	leba	Updated formatting & text
18/11/18	leba	Updated text
18/11/18	renha	Added configuration management
23/11/18	leba	Updated text
25/11/18	leba	Updated referencs and general QA

Comments:

abru Legacy section, imported from Google Docs to GitHub

abru NEED TO CHECK CITATIONS IN THIS SECTION

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

5 Software Process Models

In order to choose a software process model that fits our case and preferences, we have decided to look into some of the models introduced in this course. Therefore, we will present the basic features of each model and evaluate the pros and cons of each in order to make our decision.

5.1 The Waterfall Model

Originating out of the manufacturing industry and later adapted for software engineering, the Waterfall process model uses clearly defined phases to move projects towards completion [6]. An intuitive structure, Waterfall requires teams to complete each block of tasks before progressing any further. The simplest Waterfall models are followed in one direction, however, more advanced models may use backwards paths to earlier phases as required. Backwards paths are used to provide model flexibility [37].

There are several advantages of the Waterfall model to highlight. With even the more complicated Waterfall models able to be represented with straight forward diagrams, they are quite easily digested by those uninitiated into the world of process models [22]. This ensures that all those involved in the project are able to follow along, mitigating the risk of some members or teams losing track [43].

The structure also encourages thorough research and definition of requirements in the early phases, in part to expose potential problems that may occur in latter phases [37]. Problems exposed early in the production cycle are much cheaper to fix, by a factor of 50 to 200, than those uncovered in latter stages [30]. Furthermore, determining the project's actual requirements allows teams to keep the end goal in sight, represented as the final block in the model. This gives the teams something common to work towards, and prevents the final goal being out of focus for certain teams.

Finally, the Waterfall process emphasises the accessibility of information between phases via well-structured documentation. Should a team working in a downstream phase wish to inspect work done at an earlier point in time, its documentation should be available for reference.

In relation to the advantages, several disadvantages can also be identified. Despite the apparent advantages of having an intuitive and methodical process model, the rigidity of the simpler Waterfall structures can also prove detrimental to final output quality, client satisfaction and overall flexibility regarding unforeseen project changes.

Requirement changes from clients or end-users are commonplace in the software development industry, and their accommodation usually requires a pivot in terms of product design, development and testing [33]. Feedforward Waterfall models show all information and requirement gathering only taking place in the first phase of a project, with little room for revision (whether that be by internal or external parties) and subsequent change via backwards paths [46]. This is in contrast to the iterative approach employed by other methods, such as the spiral model [9] or agile development [5]. Furthermore, the addition of a customer representative, like the “Project Owner” in the Scrum framework for Agile development [39], is missing in the pure Waterfall model. The absence of this external line of communication means customer feedback on the existing software is not possible, which may result in an unsatisfactory product. Smaller projects or those with hard and fast objectives, where there is a narrow field of potential deviation from functional and form requirements, tend to be less sensitive to client and end-user exclusion [43].

Finally, compared to the more agile development models which promote constant testing, the Waterfall model leaves testing until late in the piece. This means difficulties or flaws in the software features can go unnoticed deep into the project's timeline [31].

In conclusion, the advantages and disadvantages of the simplest form of the Waterfall process model revolve around the fact it flows in one direction. It is easily digested, focused on the final product, appropriate for small-scale projects or those with well-defined requirements, and encourages high quality documentation. However, the minimal client-developer interaction throughout the life of the project and

no possibility of revision can result in a product that may not line up with what the former had in mind. Finally, delayed product testing can allow design flaws to exist long into a projects lifetime, potentially increasing expenditure and resulting in lengthy delays or the loss of work. In response to these flaws, several modified waterfall models have been proposed. These include, but are not limited to, Rapid Development [30], the Sashimi model by Peter De Grace, and the V-model [17].

5.2 The Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model together with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, they have an important role in the development of a product using the spiral model [9].

Handling risks is the most important feature of the spiral Model [41]. Such risk resolutions are easier done by developing a prototype. The spiral model supports dealing with risks by providing the scope to build a prototype at every phase of the software development. Each phase of the spiral model is divided into four quadrants. The functions of these four quadrants are:

1. Objectives determination and identification of alternative strategies and solutions. Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the beginning of every phase. A detailed management plan is drawn up.
2. During the second quadrant all the possible solutions are evaluated to select the best. Then the risks associated with that solution are identified and steps are taken to reduce them. At the end of this quadrant, a prototype is built for the best possible solution.
3. During the third quadrant, the identified features are developed and verified through testing. At the end of this quadrant, the new iteration of the software is available.
4. In the fourth quadrant, the customers evaluate the current version of the software. After review, the decision is made whether to continue with another loop of the spiral. If the decision is made to continue, planning for the next phase is started [9].

5.3 The STEPS Model

The Software Technology for Evolutionary Participative System Development is a methodical approach developed by C. Floyd, M. Reisin, and G. Schmidt at the Technical University of Berlin. It was first presented at the 1989 European Software Engineering Conference [15], offering a different perspective on software development, specifically user-oriented software development, as a perpetual process where production and use are not separable, but rather inter-reliant parts of the software design process as a whole. As argued by the authors:

“(...) viewing software development as production, and thereby focussing our attention primarily on the product software is misleading. Instead, I consider processes of software development as the primary area of concern, I regard product software as emerging from these processes and the use of software intertwined with its development.” [15, pp. 49].

In other words, the process of designing software should focus not only on the production of software nor only the use of it, but rather on both, and more significantly their reciprocal effects, in order to

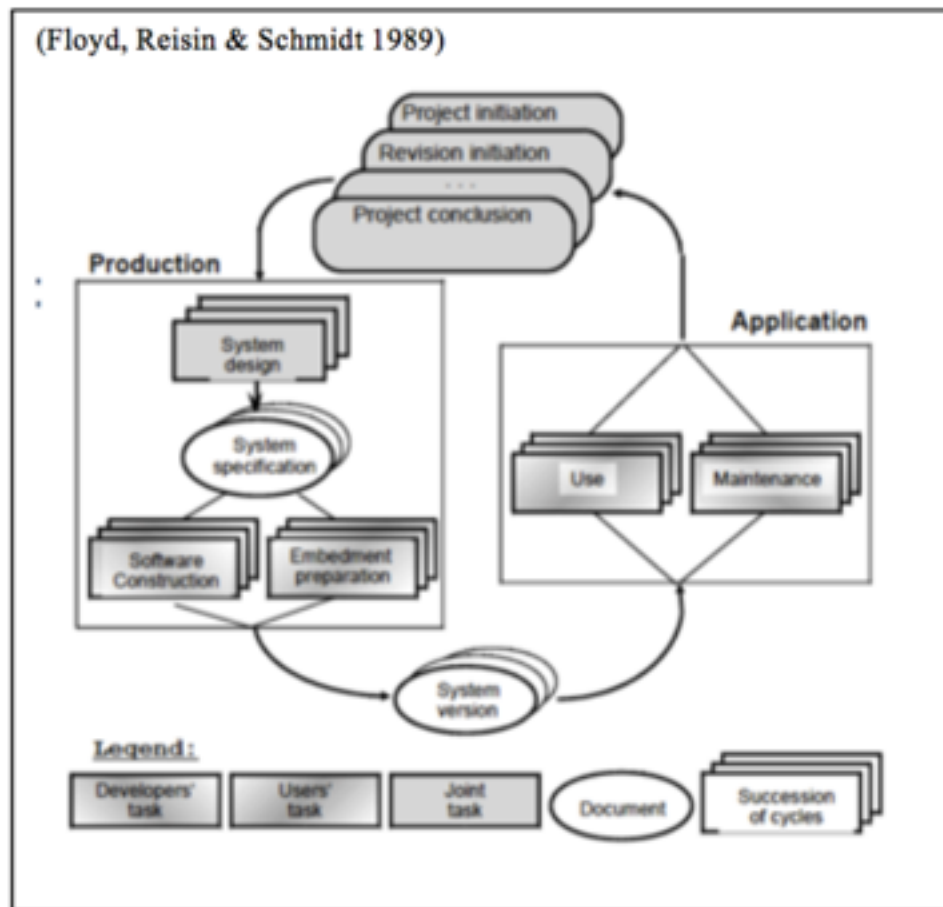


Figure 8: **NEEDS CAPTION**

ensure the quality of the software product(s). The aim of STEPS as a framework is to facilitate this cooperation between developers and users.

While the approach theoretically applies to software production in general, it is primarily concerned with what the authors describe as “the cooperative development of software to be fitted into user work processes” [15, pp. 51], and the connection between the fields of software development and work design. In such projects “(...) software development does not start from pre-defined problems, but must be considered a learning process involving the unfoldment of the problems as well as the elaboration of a solution fitting the problem.” [15, pp. 50] and “(...) technical concerns for providing high-quality products are inherently tied up with issues of communication, work, and social processes” [15, pp. 51].

As such it is imperative that neither developers nor users consider their efforts in a vacuum. While the developers’ primary task is to program the software, it is important that they understand how their work indirectly affects the users, and even more so that they understand the work processes of the users. Meanwhile, users are faced with changing work processes due to the software being developed, and have to adjust how they work in accordance. These adjustments, in turn, will affect the developers, as the design of the software now has to accommodate different processes than originally intended. As a result “(...) it is not possible to define the required functions and quality of a software system, which is to be used in working processes, completely at any fixed point in time.” [15, pp. 53] and the process of developing the software becomes iterative or cyclical, as changes to the software leads to changes in the work processes of the users which, again, leads to changes in the software, etc.

In order to facilitate this iterative cooperative process of developing software, the STEPS framework

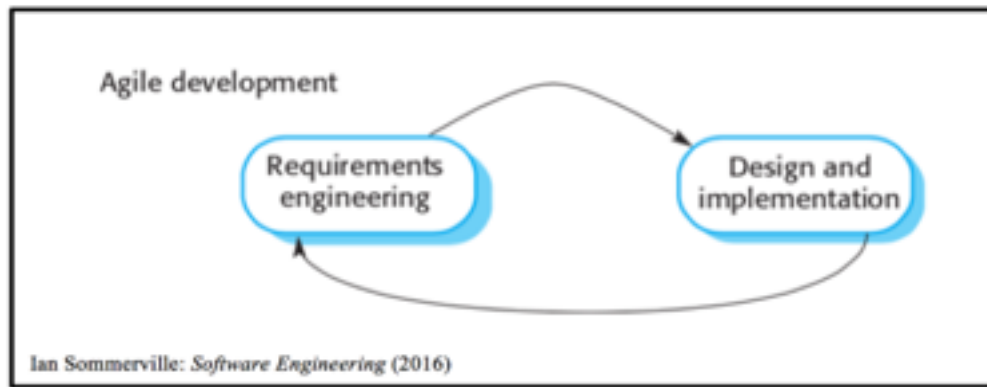


Figure 9: **NEEDS CAPTION**

presents the cyclical evolutionary project model shown below, outlining the different tasks to be performed by users, developers, and those to be performed cooperatively. As can be seen from the model, the development process never ends; there is no final product version. Instead there are new versions (prototypes) of the software addressing the changes to the work of the users as a result of the previous version.

5.4 Agile Software Development

Agile development is a software development process that emerged during the late 1990s as a response to the more plan-driven development processes. The idea behind the agile method is to develop and produce software more quickly, in order to keep up with the rapidly evolving markets, so that a software product can be delivered to the customer before the product loses its relevance [41, pp. 73].

The agile development process is incremental in the way the process is subdivided. However, the increments are rather small, and the software product itself is therefore updated frequently after its release (Ibid.: 74). For this reason, the agile development process can be considered useful when developing mobile applications, since smaller updates, in a timespan of a couple of weeks or months, are quite common for mobile applications. As [41, pp. 76] points to, the agile development process is, today, almost always used for developing small or medium-sized products for sale, or for the development of custom-made systems where the customer is committed and involved in the process. Essentially, communication between customer and developers is a central aspect when applying agile development methods. Additionally, the agile development process is more well suited to smaller teams, in contrast to the plan-driven approaches, which is useful for large teams of developers working from different companies and countries (Ibid.: 75).

The agile development process contrasts the plan-driven development processes in the way it considers design and implementation as its core activities. Other activities however, are an integrated part of the core activities. Likewise, the requirements of the software product are incorporated and considered in the design and implementation activities, rather than being a separate component (Ibid.: 74).

Figure 9 illustrates the agile development process, and the way in which it alternates between the requirements and the core activities of design and implementation. Another feature that distinguishes the agile development process is the more informal way of testing the product. In most cases, the testing will be incorporated in the design and implementation activities (Ibid.: 75 + 84).

5.5 The Pros and Cons of Each Model

As we have now a few software process model, we will try to identify which model will suit our project the best. In order to get a better overview we have lined up the main pros and cons of each model in table 1 below.

Model	Pros	Cons
Waterfall	<ul style="list-style-type: none"> + Intuitive + Final outcome oriented + Suitable for small projects + Promotes documentation 	<ul style="list-style-type: none"> - Unaccommodating to requirement changes - Excludes client from process - Delays testing
Spiral	<ul style="list-style-type: none"> + Risk analysis and risk handling at every phase makes it good for projects with many unknown risks that occur as the development proceeds + Flexibility with requirement allows elements of the product to be added in when they become available or known + Allows extensive use of prototypes 	<ul style="list-style-type: none"> - Good for large projects; expensive - Difficulty with time management as the number of stages is unknown - Requires excessive documentation
STEPS	<ul style="list-style-type: none"> + No final product + Emphasis on end-users + Specialised model for work processes 	<ul style="list-style-type: none"> - No final product - Heavily relies on user involvement in development - Limited usability due to specialisation
Agile	<ul style="list-style-type: none"> + Quick software development \Rightarrow delivers the software product to the customer faster + Communication between customer and developer in focus + As customers are involved it also ensure the best result + With this model it is easier to adjust when requirements are changed 	<ul style="list-style-type: none"> - If not well organized it can be hard to manage

Table 1: Pros and Cons of the Software Process Models

5.6 Our decision

5.6.1 What kind of model do we need?

The process used in different companies depends on the type of software being developed, the requirements of the software customer, and the skills of the people writing the software. As there is no universal process that is right for all kinds of software, most companies have developed their own development processes. For our case, we need a software process model that fits our team in relation to our size, the kind of product we are developing, and the time span we are working with. In this project, we are currently a rather small team of 7 developers working together on developing our mobile application, *strEAT*. In addition, we need to get our software product on the market rather soon, since our deadline is December 3, 2018. Likewise, our app needs to be in use by customers during spring 2019 at the latest, since the street food “season” in Copenhagen is in the summer. Furthermore, it will be necessary to develop our app together with the users/customers. We need to investigate along the way what the needs of the customers are, and how, as an example, street food vendors in Copenhagen would like to use this app. Therefore, we need a software project model which focuses on customer involvement.

5.6.2 Which model have we chosen and why?

The Waterfall model contains a major drawback in terms of our project: minimal client-developer interaction. From the beginning, we knew the role of supervisor (Silviu, acting as a client) would be significant and we would like to keep close ties with him, with constant updates and feedback from both sides. Moreover, one phase of the Waterfall process cannot start until the previous is finished. We are, therefore, reluctant to use such a linear process as we want to be able to send feedback from one phase to another.

The Spiral model, which contains the stepwise approach of a waterfall, is very focused on risk handling. While it might be considered an advantage for our app development, it is unfortunately too complex and expensive for our needs. The STEPS model facilitates cooperation between developers and users, it is primarily focused on software intended for use in existing work processes, which does not match the conditions of our case. Additionally, it is heavily reliant on the involvement of the users, of which we currently have none.

The idea behind the agile method is to develop and produce software more quickly before the product lose its relevance. The agile process is incremental and continuous as the software is being developed, therefore it is easier to change the plan to reflect changing requirements. This feature is very relevant in our case - the nature of app development, with its variety and constant updates, makes it the most liveable, fast-paced market one can imagine. In fact, incremental development is now the most common approach for the development of application systems. Moreover, iterative development and agile methods are better for informal teams in which there is a constant close communication between its members and customers. Thus, we concluded the agile methodology is best suited for the purpose of our project.

As previously mentioned in the section about the software qualities of our case, the agile software development model is well suited for the qualities we have identified. In relation to the usability and functional suitability of the app, we need strong communication with the street food vendors in Copenhagen, in order to develop an app that will actually fit their needs. Likewise, we need to know what those who are using our app to find street food need. For this, communication is the key. Since agile software development prioritises frequent communication with customers, this seems suitable for our needs. Considering both performance and reliability, we need a software process model that allows our team to frequently launch new updates and quickly fix small errors. We need to be able to do so in order to keep up with the market and demands, and in order for our app to always be the best version of itself. This illustrates that agile software development is a well suited approach for our case, since its process of smaller increments will make this possible. Lastly, in relation to portability, it can be

considered that the app will only be able to compete within the growing market of food-related apps if it is adjustable and follows people's preferences and needs in real-time, with ongoing observation of trends and target groups. In order to develop an adjustable app, we need a process model that allows adjustments when they appear.

Agile software development is typically applied using one of several widely accepted frameworks. Such frameworks, sometimes referred to as methodologies, include Extreme Programming, Scrum, and DSDM [41, pp. 73]. In the next section, we will explain a bit about the Scrum framework, as we will use this in our work.

Configuration Management

Document title: Scrum

Version:^a v3.05

Github link: [.../MainDocument/sections/Scrum.tex](#)

Trello card link: N/A

Responsible: leba

Status: Finished

Changelog:

18/11/18	leba	Created
18/11/18	renha	Added configuration management
23/11/18	leba	Added reference to figure
25/11/18	leba	Figure removed and consistency checks
25/11/18	leba	Removed introduction heading
28/11/18	kati	Last touch-up

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

6 Scrum

In this section we will briefly dive into the scrum framework, and the different components it introduces.

Scrum is one of the several frameworks that fall under the agile development umbrella. Today, it is one of the most popular frameworks, partially because of the difficulties it addresses in relation to the management of agile software development, since it was developed as a method to wisely manage and secure the time and resources when applying agile software processes [41]. As a project management model with a high level of user interaction and a "... condensed time-to-market" [14], Scrum has become highly popular among mobile companies and application developers, as it provides a good combination of functions such as project overview, to-do's and task lists [14].

Scrum consists of several artifacts, roles and events [39]. These will be discussed briefly below and in detail in the Roles and Responsibilities, and Planning and Estimation sections later in the report.

6.1 Scrum Artifacts

6.1.1 Product Backlog

The Product Backlog essentially lists the requirements of the project. The items may take the form of user stories, formal/informal requirements, required knowledge acquisition, etc. The list is subject to change throughout the life of the project [39].

6.1.2 Sprint Backlog

A subset of the Product Backlog, the Sprint Backlog consists of items to be completed during a Sprint, an event which will be introduced below. The Development Team completes the tasks of the Sprint Backlog to produce the next viable product iteration [39].

6.1.3 Iteration

The Iteration is simply a version of the deliverable, updated through the work completed during each Sprint cycle [39].

6.2 Scrum Team

6.2.1 Product Owner

The Product Owner represents the interest of the stakeholders and is responsible for overseeing the quality of output from the Development Team. They are also responsible for managing the Product Backlog [39].

6.2.2 Scrum Master

The Scrum Master supports the Development Team by ensuring all members understand the principles of Scrum and its implementation. They help manage the Product Backlog, along with the Product Owner, and liaise with the Development Team with regards to its formation. They also facilitate any Scrum event [39].

6.2.3 Development Team

The Development Team are responsible for completing the tasks defined in each Sprint Backlog. They are involved in discussions regarding the formation of the Product and Sprint Backlogs [39].

6.3 Scrum Events

6.3.1 Sprint

A series of successive Sprints make up the project timeframe. The length of a Sprint will vary between projects but typically ranges from 2-4 weeks. During each Sprint, items from the Sprint Backlog are completed, with the outcome a new iteration of the product to be delivered [39].

6.3.2 Sprint Planning

As the name suggests, this event involves planning for an upcoming Sprint. This requires populating a Sprint Backlog, which may require deliberation between team members. The Development Team then have to determine how the work set out by the Sprint Backlog will be completed and who will do each part. It may be helpful to also outline a goal for the Sprint [39].

6.3.3 Daily Scrum

The Daily Scrum, or Stand-Up, is an short meeting that takes place at the same time every day and allows the Development Team to state what will be done in the next 24-hour period [39].

6.3.4 Sprint Review

This is an informal post-Sprint meeting between the Scrum Team and stakeholders to inspect the increment and propose changes to the Product Backlog if necessary. The Product Owner states what has been done and how the Product Backlog currently stands. The Development Team states how they felt the Sprint went, including problems faced and solutions used. All parties discuss what to do in the next Sprint [39].

6.3.5 Sprint Retrospective

This meeting includes only the Sprint Team and does not concern the stakeholders. It is intended to discuss the "... people, processes, relationships, and tools" of the Sprint. The Sprint Team may discuss ways to improve their own performance in upcoming Sprints [39].

6.4 Conclusion

The sections to follow in this report will refer to, and elaborate on, the artifacts, roles, and events briefly outlined above. In particular, the section entitled Roles and Responsibilities will delve a little deeper into the aspects of the Scrum Team and assign roles to each team member, the Planning and Estimation section will construct Product and Sprint Backlogs for the course deliverables themselves, and the Requirements section will do the same with regards to *strEAT* product requirements.

Configuration Management

Document title: Roles and Responsibilities

Version:^a v3.06

Github link: [.../MainDocument/sections/Roles.tex](#)

Trello card link: [trello.com/c/ysioskYb/14-roles-and-responsibilities](#)

Responsible: jopo

Status: Finished

Changelog:

13/09/18	jopo	Created
01/11/18	abru	Added Latex tables
02/11/28	leba	Updated text
18/11/18	leba	Updated text
18/11/18	renha	Added configuration management
23/11/18	leba	Updated formatting & text
29/11/18	kati	Updated references
29/11/18	jopo	Updated references

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

7 Roles and Responsibilities

This section outlines the roles and responsibilities delegated to each team member as per the aforementioned Scrum framework. Scrum emphasizes interactions between individuals and the terminology for roles is unlike most other project management systems. The Sprint Cycle section will outline how we will conduct each Sprint and its events. The Roles section continues where the Scrum section left-off by formally stating the expectations of the Product Owner, Scrum Master and Development Team. The final section assigns roles to our team members and states their responsibilities with regards to *strEAT*.

7.1 Sprint Cycle

During the Scrum sprint, all team members share information, describe their progress since the last meeting, bring up problems that have arisen, and state what is planned for the following day. Thus, everyone on the team knows what is going on and, if problems arise, team can re-plan short-term work to cope with them. Everyone participates in this short-term planning; there is no top-down direction from the Scrum Master. Daily scrum meetings, in our case, are going to be conducted in Slack. During weekly meetings, the estimated work remaining in the Sprint will be calculated and graphed by the Scrum Master [41, p. 87]. However, we can coordinate interactions between members daily with Trello that will fulfil a role of a Scrum board. In offices, it is usually a whiteboard that includes information and post-it notes about the sprint backlog, work done and work in progress [40]. In our case, Trello would be a good substitute, as everyone can update it daily and have a constant and easy access to it. This is a shared resource for the whole team, and anyone can change or move items on the board. It means that any team member can, at a glance, see what others are doing and what work remains to be done. We may also create a sprint burndown chart to see a percentage of work finished.

Generally, the tasks should be achievable in less than one day, however, our tasks will be set to one-week capacity – so that we can discuss them on weekly meetings. Once the Sprint Backlog is decided, it is fixed. The team cannot remove items from it (but adding items is possible). If an item isn't finished, the Product Owner will need to decide what to do with it. The Product Owner is the only individual that can remove things from the Sprint Backlog if it no longer provides business value [?].

On the final day of each sprint, a review session (Sprint Review meeting) is conducted to allow the Product Owner to check if all of the committed items are complete and implemented correctly. Additionally, a Sprint Retrospective is conducted to check and improve the project execution processes: what was good during the Sprint, what should continue as it is and what should be improved [?]. The team will benefit from the Sprint Backlog as it gives direction on a day-to-day basis, keeping the group on track [?].

7.2 Scrum Roles and Responsibilities

To summarize, we have three different roles in the Scrum process: Product Owner, Scrum Master and Scrum Team. Their responsibilities are:

7.2.1 Product Owner

- Ensures that the level of detail in the specification of the backlog items is appropriate for the work to be done.
- Involved in prioritising the items on the Product Backlog to define which are the most important.
- On the first day of the new sprint, the Product Owner adds new items to the Product Backlog.

- Responsible for conveying the vision of the stakeholders to the team. [?].
- Responsible for the return on investment (ROI).
- Communicates with the stakeholders about progress and problems [?].
- Selects which of the highest priority items they believe can be completed in the upcoming sprint.
- Acts as a communication channel between the team and other involved parties (stakeholders) [?].

7.2.2 Scrum Master

- Conducts all Scrum ceremonies and processes.
- Removes all hindrances or disturbances to achieving each goal.
- Selects which of the highest priority items they believe can be completed in the upcoming spring (together with the rest of the Scrum team and Product Owner) [?].
- Commits to goals and deadlines on behalf of the team [?].
- Does not interfere with the decisions of the team, specifically regarding the development, but rather is there as an advisor for the team.

7.2.3 Development Team

- Responsible for all the activities who's completion are required to achieve the sprint goals.
- Once committed, it is a team's responsibility to fulfil the commitment and deliver the agreed upon results, on time, and with high quality [?].
- Team members have to participate in the meetings and have to ensure that all the findings of the meetings are being addressed in the project [?].
- Involved in the selection of the highest priority items they believe can be completed in the upcoming sprint (together with Scrum Master and Product Owner) [?].

7.3 Scrum Roles and Responsibilities in our Project

Our team has decided that in order to achieve fairness we will let everyone try different roles in Scrum. Also, we believe rotating roles will enhance collaboration between members and avoid monotony. Hence, below is the plan for Scrum Masters (SM) and Product Owners (PO) over the course of the semester. Week 11-14 will be decided upon later, based on team members experience with fulfilling different functions and their feelings about them.

7.3.1 Product Owner

Our Product Owner's most important responsibility is the adding of new items (with a proper level of specification) to the Trello 'To Do' list on the first day of the new sprint. They also need to prioritise those items by moving tabs with the most important tasks to be done (in the current cycle) on the top of the Trello board. Moreover, the PO is a communication channel between Silviu and the team, which includes: conveying Silviu's comments to the team members that were not present during the weekly meeting, and keeping Silviu up to date with regards to progress or problems faced by the team.

Week#	Product Owner	Scrum Master
4 (38)	Anders	Emmi
5 (39)	Ida	Jowita
6 (40)	Katrine	Lewis
7 (41)	René	Anders
8 (43)	Emmi	Ida
9 (44)	Jowita	Katrine
10 (45)	Lewis	René
11 (46)	-	-
12 (47)	-	-
13 (48)	-	-

Table 2: Our roles for each week of the project

7.3.2 Scrum Master

Scrum Master makes sure that everyone follows Scrum roles and ceremonies agreed upon at the beginning of the project and makes sure every tool (Trello as a planning instrument and Google Docs for documents sharing) is working properly and is being updated according to the work done. Moreover, the SM announces the goals and deadlines to Silviu on behalf of the team but without interfering with the decisions of the team regarding the content of the tasks. He or she is in charge of minutes from meetings (both with and without a supervisor) and inserting them on a proper Slack channel. Finally, the Scrum Master will send or upload proper documents for the evaluation to the supervisor and teacher through LearnIt.

7.3.3 Development Team

As soon as a team member commits to a task it is their responsibility to fulfil the commitment and deliver the agreed upon results, on time, and with high quality. To ensure this quality, team members are also required to review each other's work via shared Google Docs document. Updating Trello is essential and each team member moves their tasks to the proper boards (from 'To Do' to 'Pending' and later to 'Finished') remembering to keep the prioritised order while doing so. All team members have to participate in the meetings (one absence is acceptable). If unable to be at the meeting in person, Skype attendance is required. For critical situations that require a fast answer, the team uses the appropriate channel on Slack. All team members that are called on the channel are required to help and give feedback. The team must be available on Slack throughout all weekdays.

All of us take part in the discussion regarding which of the highest priority items can be completed and what is the time required to complete said items. Also, the whole team is responsible for the proper use of Slack channels and each member is an active part of them.

Configuration Management

Document title: Plan and Estimation

Version:^a v3.15

Github link: [.../MainDocument/sections/Plan.tex](#)

Trello card link: [trello.com/c/xi9AzgGN/8-project-plan-and-estimation](#)

Responsible: abru & leba

Status: Finished

Changelog:

12/09/18	abru	Created
12/09/18	leba	Added "Product Backlog" & "Sprint Backlog"
13/09/18	leba	Added "Introduction", "Planning With Scrum" & sprint schedule Updated formatting & text
13/09/18	abru	Added "Estimation"
14/09/18	abru	Added "Over of estimates" table Updated text
25/09/18	leba	Updated sprint schedule
27/09/18	leba	Added project schedule (v1)
28/09/18	abru	Added project schedule (v2)
29/10/18	leba	Updated formatting & text
01/11/18	abru	Added Latex versions of tables
02/11/18	leba	Updated formatting
18/11/18	leba	Updated text
18/11/18	renha	Added configuration management
23/11/18	leba	Updated formatting & text
25/11/18	leba	Formatting and consistency update
27/11/18	abru	Updated formatting

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

8 Project Plan and Estimation

This section introduces the planning process associated with the Scrum framework and Agile development as a whole. Techniques for estimation are also introduced as part of the planning process itself.

The flexible nature of Agile development necessitates splitting project planning into two planning phases, each dealing with different time frames. These are release planning and iteration planning [41, p.680]. Release planning deals with a longer time frame, typically of several months. It deals with all the current requirements of the final product as stipulated by the client. A high-level plan for the project's multiple sprints is populated by the project requirements according to when they are expected to be addressed. Iteration planning deals with the individual sprints and how they will produce a viable product increment. The sections below define a Scrum approach to the first stages of release planning (Product Backlog and estimation), and iteration planning (Sprint Backlog). These terms were introduced in the Scrum section.

8.1 Planning With Scrum

Project planning with the Scrum framework is conducted in distinct steps with the use of several Scrum Artifacts as outlined in “The Scrum Guide” [39].

The first is to brainstorm a list of user stories [41] that outline all the features a user may want in the final product. The list may also be populated instead (or in conjunction) with a formal requirements document, detailings of knowledge acquisition, or some other description of the deliverable. This list is dynamic and will more than likely take many shapes during the life of the project. As this section outlines a plan for producing course deliverables, user stories are not applicable and will not feature in the plan.

In the case where user stories are used, the next logical step is to rank them in order of priority, with the most important features to a user's experience being at the top of the list. This is the Product Backlog, which in our case will be populated with course deliverables in chronological order. The effort required to complete each of the items in the backlog is then estimated. This can be conducted in a variety of ways; we have chosen to use the Planning Poker technique.

Once the estimations are complete, the Product Owner and Scrum Master select a number of the highest priority items that they believe are possible to be completed in the next Sprint. A Sprint is a fixed time frame, usually between 2 and 4 weeks, during which a product increment is developed. The Development Team, is involved in this step, too; the selection of Product Backlog items is an iterative process. Based on the proposed selection, the Development Team design the required increment and produce a list of necessary tasks to complete. The agreed upon Product Backlog selections make up the Sprint Backlog. Each task the Scrum Team deem necessary to complete the increment is placed to the left side of the Scrum board in the “to-do” column. Items are moved across the board during the sprint, passing through the “in progress”, “awaiting approval” and “done” columns as necessary. The names of the columns may vary, but the idea is essentially the same. The end goal being all tasks are in the “done” column at the end of the sprint. A Sprint Review is conducted at its conclusion to address outstanding backlog items, what went well/wrong, budgeting, future Sprint planning, etc. It includes all parties, including key stakeholders. It's outcome is a revised Product Backlog. A Sprint Retrospective is also conducted. This requires the Scrum Team to reflect on it's own performance and suggest improvements with regards to “(...) people, relationships, process, and tools” [39].

8.2 Product Backlog

The Product Backlog is populated by the course deliverables, as outlined on learnIT. They are arranged in the order in which they are intended to be completed.

- a) Create the team protocol.
- b) Undertake a project kick-off.
- c) Explore our case in the form of literature reviews .
- d) Explore software qualities of our case.
- e) Choose a software process model.
- f) Define roles and responsibilities.
- g) Develop a project plan and estimation.
- h) Complete a risk analysis.
- i) Explore the use context using various research activities and present the results as raw data and an analysis.
- j) Develop 2 or 3 rich pictures that depict the context in which the software will be used.
- k) Develop a class-event table and class diagram.
- l) Develop a use-case diagram and scenarios.
- m) Create an interface design mock-up.
- n) Choose document conventions and apply to existing documentation.
- o) Outline a configuration management plan.
- p) Create an initial requirements document.
- q) Create a quality assurance plan.
- r) Construct a dynamic test specification.
- s) Explore architectural solutions and document an analysis of the chosen pattern.
- t) Reflect on the project and document the outcomes.

8.3 Estimation

A critical part of project planning is estimating the time required to finish the project, as well as the individual tasks. There exists multiple techniques for deriving these estimates, all with their own advantages and disadvantages.

One such technique for agile teams, is playing a game of planning poker which “(...) combines expert opinion, analogy, and disaggregation into an enjoyable approach to estimating that results in quick but reliable estimates.” [10, p. 56]. The game itself is simple; the developers involved in the project sit down at a table and are all given a set of playing cards. Each card representing a pre-determined estimate in a given unit. The user stories are then “played” as individual hands, where a moderator, usually the project owner, reads aloud the user story. The developers are then able to ask questions about the user story to the moderator in order to resolve any ambiguity or confusion,

and then have to play a card face down representing their estimates. Once all players have chosen their card, they are simultaneously revealed. If the players are in agreement you make note of the estimation, and move on to the next story. However, it is more than likely that not all participants have the same estimation, potentially not even close to it. In such cases the outliers are asked to explain their thought process and why they chose as they did, leading to a group discussion of the estimates, at the conclusion of which the participants once again play a card face down. This process can potentially be repeated as many times as needed, but it is important to note that the aim of the game is not necessarily to have all participants play the same card, but rather to reach an agreement, as put by Cohn: “(...) the point is not absolute precision but reasonableness” [10, p. 57].

In other words, in order to conclude a “hand” of planning poker, the participants should have proximate estimates and be able to agree on a single estimate which will be used. The technique can be applied both in the release planning phase, in order to derive an estimate for the project as a whole, as well as in the iteration planning phase, to distribute work tasks optimally, or close to it, between the developers for an upcoming or ongoing sprint.

As mentioned previously, the cards represent estimates in a given unit. This unit could be either a unit of time, such as ideal days, or a unit of size, e.g. story points. Ideal days are basically a derivation of work days, where you assume that “the story being estimated is the only thing you’ll work on, everything you need will be on hand when you start, [and] there will be no interruptions” (Ibid: 45). In other words, a perfect, or rather ideal, days worth of work on the given task, meaning that an ideal day might in reality correspond to one and a half or two work days or even more if you were to fall ill. As such, the advantage of using ideal days over elapsed days, is that ideal days are not susceptible to external factors, thus an estimate in ideal days is much more likely to be accurate.

Story points are a relative unit of estimation, where the effort required to complete each user story is ranked in relation to each other. As such the individual values assigned to a given user story is irrelevant, and “what matters are the relative values assigned to different stories” (Ibid: 40). A story, a, being estimated at 10 story points by itself says nothing about its size. However given two other stories b and c, estimated at 5 and 20 points respectively, we know that a is double the size of b and half the size of c, and are thus able to allocate project resources accordingly. In order to utilize story points as a unit for estimation, the concept of velocity is required. The term covers the project teams rate of progress per iteration, in other words the sum of story points they are able to complete in a single increment. As we are only working on this project part time, we will be using story points as our unit of measure for estimation for this project. The estimates, in story points, for our Project Backlog, as a result of our planning poker game, is presented in table 3 to the right.

Story	Estimate
a	1
b	1
c	1
d	2
e	2
f	2
g	2
h	1
i	2
j	3
k	4
l	3
m	3
n	2
o	3
p	5
q	3
r	3
s	5
t	5

Table 3: Overview of our estimates in story points for the Product Backlog, as a result of our planning poker game

8.4 Project Schedule

Based on the estimations for each of the items in the Product Backlog, a project schedule may be constructed. It is subject to change as the project progresses through each sprint cycle. The project began with the first sprint in week 35. The second sprint began on the Monday of week 37, and lasted only one week because it contained the first milestone, the tollgate 1. The rest of the sprints are all two weeks each, concluding on the Friday of week 49, accounting for the autumn break in week 42. On the next page table 4 is presented, depicting the project broken into seven sprints. Each sprint in the table consists of items from the Product Backlog and the corresponding estimates from table 3.

Task	Responsible	Estimate	Start	End
<i>Sprint 1 — 28/08-10/09</i>				
Create sprint backlog	PO	-	28/08	-
Team meeting	SM	-	28/08	-
a) Create team protocol	TEAM	1	29/08	29/08
b) Undertake project kick-off	TEAM	1	30/08	30/08
c) Explore our case	TEAM	1	31/08	31/08
Supervisor meeting	PO & SM	-	04/09	-
Team meeting	SM	-	04/09	-
d) Explore software qualities of our case	TEAM	2	05/09	06/09
e) Choose a software process model	TEAM	2	08/09	09/09
<i>Sprint 2 — 10/09-17/09</i>				
Create sprint backlog	PO	-	10/09	-
Supervisor meeting	PO & SM	-	11/09	-
Team meeting	SM	-	11/09	-
f) Define roles and responsibilities	TEAM	2	11/09	12/09
g) Develop project plan and estimation	TEAM	2	13/09	14/09
h) Complete a risk analysis	TEAM	1	14/09	14/09
Submit 1st tollgate	SM	-	14/09	-
<i>Sprint 3 — 17/09-01/10</i>				
Create sprint backlog	PO	-	17/09	-
Participate in scrum game	-	-	18/09	-
Supervisor meeting	PO & SM	-	18/09	-
Team meeting	SM	-	18/09	-
Supervisor meeting	PO & SM	-	25/09	-
Team meeting	SM	-	25/09	-
i) Explore the use context	TEAM	2	26/09	27/09
j) Develop rich pictures	TEAM	3	28/09	30/09
Resubmit 1st tollgate*	SM	-	28/09	-
<i>Sprint 4 — 01/10-15/10</i>				
Create sprint backlog	PO	-	01/10	-
Supervisor meeting	PO & SM	-	02/10	-
Team meeting	SM	-	02/10	-
k) Develop a class-event table and class diagram	TEAM	4	03/10	06/10
Supervisor meeting	PO & SM	-	09/10	-
Team meeting	SM	-	09/10	-
l) Develop a use-case diagram and scenarios	TEAM	3	10/10	12/10
m) Create an interface design mock-up	TEAM	3	13/10	15/10

Table 4 continued from previous page

Task	Responsible	Estimate	Start	End
<i>Sprint 5 — 22/10-05/11</i>				
Create sprint backlog	PO	-	22/10	-
Supervisor meeting	PO & SM	-	23/10	-
Team meeting	SM	-	23/10	-
n) Choose document conventions	TEAM	2	24/10	25/10
o) Outline configuration management plan	TEAM	3	26/10	28/10
Supervisor meeting	PO & SM	-	30/10	-
Team meeting	SM	-	30/10	-
p) Create initial requirements document	TEAM	5	31/10	04/11
Submit 2nd tollgate	SM	-	02/11	-
<i>Sprint 6 — 05/11-19/11</i>				
Create sprint backlog	PO	-	05/11	-
Supervisor meeting	PO & SM	-	06/11	-
Team meeting	SM	-	06/11	-
q) Create quality assurance plan	TEAM	3	07/11	09/11
r) Construct dynamic test specification	TEAM	3	10/11	12/11
Supervisor meeting	PO & SM	-	13/11	-
Team meeting	SM	-	13/11	-
Peer reviews	-	-	13/11	-
Resubmit 2nd tollgate*	SM	-	16/11	-
<i>Sprint 7 — 19/11-03/12</i>				
Create sprint backlog	PO	-	19/11	-
Supervisor meeting	PO & SM	-	20/11	-
Team meeting	SM	-	20/11	-
s) Explore architectural solutions and document analysis	TEAM	5	21/11	25/11
Supervisor meeting	PO & SM	-	27/11	-
Team meeting	SM	-	27/11	-
t) Reflect on the project and document outcomes	TEAM	5	28/11	02/12
Final submission	SM	-	03/12	-

Table 4: Our project schedule

8.5 Sprint Backlog

Each Sprint Backlog is crafted by the Scrum Master and Product Owner, and refined with the input of the Development Team. A few Product Backlog tasks remaining at the beginning of each sprint are selected based on their level of priority. The number of tasks chosen depends on the amount of work the Scrum Team believes can be achieved in the upcoming sprint. The Sprint Backlog is converted into tasks by the Development Team, the completion of which are expected to result in a working product increment. The Sprint Backlog and tasks presented below are for the 3rd sprint.

Sprint 3 Backlog:

- Explore the Use Context
- Rich Pictures

Sprint Goals:

- A documented decision on which research activity to implement
- Documentations of the results of the research activities (raw material and analysis).
- 2 or 3 Rich Pictures together with their explanation.

Sprint Tasks:

- Brainstorm on what research is necessary for your case
- Implement the research
- Document the results in an adequate manner
- Refine your documentation
- Develop two or three Rich Pictures (in subgroups)
- Present the Rich Pictures for the whole group
- Use the notes to write short explanations for the Rich Pictures.

Sprint Schedule:

<i>strEAT</i> — Sprint 3	Week 39							Week 40						
	M	T	W	T	F	S	S	M	T	W	T	F	S	S
Task 1 – Brainstorming														
Task 2 – Reasearch														
Task 3 – Document Research														
Task 4 – Refine Documentation														
Task 5 – Rich Pictures														
Task 6 – Present Rich Pictures														
Task 7 – Rich Picture Descriptions														

Table 5: Activity schedule for Sprint 2

Configuration Management

Document title: Software Qualities

Version:^a v3.03

Github link: [.../MainDocument/sections/SoftwareQualities.tex](#)

Trello card link: [trello.com/c/pdzN8WHs/11-explore-the-software-qualities-of-your-case](#)

Responsible: mgab

Status: Finished

Changelog:

13/09/18	renha	Created
02/11/18	leba	Updated text
18/11/18	renha	Added configuration management
23/11/18	leba	Updated text
28/11/18	leba	Updated text

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

9 Software Qualities

In this section we reflect on the software qualities pertinent to the *strEAT* app. In order to do this, it is necessary to focus on those categories that will be crucial for all stakeholders. These categories are presented in the section below, ordered by perceived importance.

9.1 App-Specific Focus Points

As said in the case exploration section, the most important factors to be considered about the food truck market are:

- Food and food markets are part of an area’s unique culture and thus constantly changing. That requires technology that is equally flexible to change in customers’ preferences.
- Food markets have to follow common business rules, as attracting investors and building brands is becoming more and more of a possibility for them.
- The app needs to maintain high standards in graphics, design and interface. When customers’ preferences for food aesthetics change, so must the way the food is presented on the app
- The chaotic nature of the street food markets, with all the moving around and change of scenery/menu options, needs to be countered by a thorough and easy-to-understand structure in the application. GPS services can, for example, help simplify finding a customer’s favourite food truck, while the menus can be changed by the vendors in real-time and thus prevent consumers from disappointment or confusion upon arriving at the venue.
- The app is filling a current gap in the market and, therefore, has to try and maintain a near-monopoly status. Keeping track of the venues is a must so that the idea of a regular customer base does not stay exclusive to non-portable venues. Instead, a new opportunity arises in which venues can switch between popular spots, possibly even within one day as to cater to the needs of multiple target groups at once.

In the following section, the necessary software qualities will be prioritised and make indirect references back to those factors mentioned above.

9.1.1 Usability

As the app is a prototype of its category, being the first app with an implemented map and real-time location service for food market stalls and trucks, it has to be attractive to the main target group of those venues, which is mainly the “millennial” generation. “A way that customers track a particular food truck and keep up on the events is participating through Facebook, Twitter, Instagram and Foursquare. [...] The more social media sites a [food] truck has, the better.” (Campbell, 2016) That gives us good prospects for an app to be successful, and also shows that the customers of outdoor food venues are mainly users of free social apps. As that concerns young people up to the age of 35, we can expect them to not be ready to spend money on an app that has not existed beforehand and can not, therefore, be estimated to be of high relevance. Instead, there will be a simple monthly subscription for vendors as to secure their spot on our list of venues. Photos will be taken by us of each venue and their food so that the app gets more approachable and the consumer can decide not only by reading the menu but also by seeing what they are potentially purchasing. The goal is to have a simple app with all needed features in excellent quality that makes it one of the consumer’s most frequently used apps:

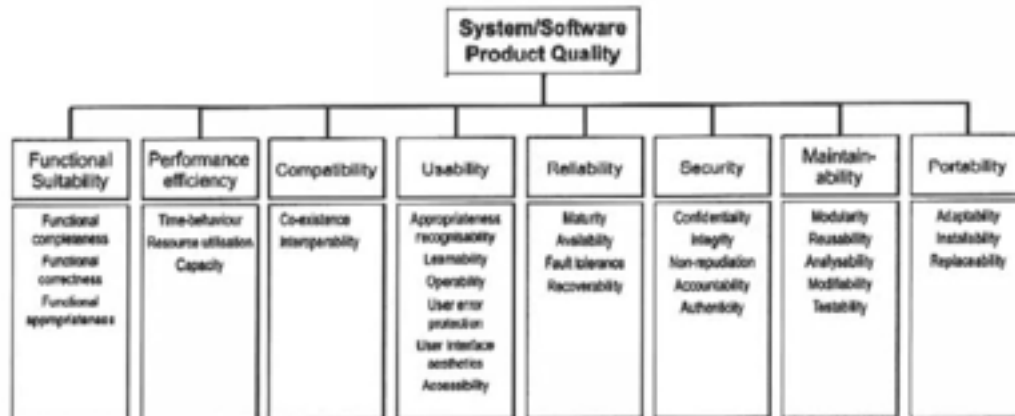


Figure 10: This diagram show different aspects important to the consideration of software qualities

- Ad-free.
- Software respects the user and is transparent about permissions.
- App is lightweight, as fast as possible, and advertising-free.
- Option for catering, pre-orders, bonus-programs, easy signUp.
- Logo & interface recognisable.
- Intuitive surface.
- Free for the end-user.
- Subscription for vendors (cancelable monthly, automatically renewed otherwise).
- Android & iOS.
- Transparency.
- Star-rating system.
- Feedback system.
- Search/filter options (vegan, vegetarian, halal, kosher, allergies...).

9.1.2 Performance Efficiency

As the goal is to design a fast and efficient product that one can download without considering e.g. the memory capacities of their iPhone, storage options can be non-permanent, as an internet connection is required for the main part of the app, the map, to work properly anyway. Therefore, cloud storage can be used to both save storage space on the users phone and enable the vendors, and us as designers, to update the app quicker and add new elements without having to request users to download a new version of the app entirely. In consequence, it becomes more user-friendly.

- NFC/GPS
- Caching instead of permanent storage

9.1.3 Functional Suitability

As the app provides no further entertainment or services that can be used without a functioning GPS and internet connection, all elementary premises for the app to be up and running at all points are of utmost importance.

- Implementation with a map needs to be accurate
- Real-time communication
- Intuitive design for the customers and vendors

9.1.4 Reliability

As formerly mentioned, the goal is that the app gets used frequently and spontaneously by the end-user. Especially in the beginning, it can be crucial if it breaks down just once, can't be updated or similar. The market is big and users might move on to arising alternatives quickly. We therefore need to ensure that, instead, we are the alternative to former (lower quality) products.

- Giving the customer and vendors the opportunity to contact us (contact information).
- Making sure the app is not crashing
- Quick fix for errors
- Java code
- Regular updates
- Regular meetings with the developing team to evaluate user feedback and errors, etc.

9.1.5 Portability

Given the originality of the product, a long-term objective should be to expand beyond Reffen food market in Copenhagen and perhaps even Denmark. For that to happen, it is important to have easily adaptable interfaces and feedback that can be transferred onto other demographics to fulfill the needs of consumers e.g. in the US or Germany.

- Adaptable for venues everywhere, e.g. Kødbyen (even other cities in Denmark).

Agile development is well suited to fulfilling these characteristics. It is the most flexible, which is an advantage for an app that has very few predecessors and needs to be as adaptable in its geographical capabilities as it is when it comes to fixing bugs or promptly acting on user feedback. The user needs to be able to see that they are being taken seriously and are an important part of our product development, while, at the same time, not being under any obligation to pay or use the product in a specific frequency.

Configuration Management

Document title: Risk Analysis

Version:  v3.05

Github link: [.../MainDocument/sections/Risk.tex](#)

Trello card link: [trello.com/c/w0awuH7n/9-risk-analysis](#)

Responsible: idbo

Status: Finished

Changelog:

12/09/18	idbo	Created
29/10/18	leba	Updated text
01/11/18	abru	Added tables
		Updated formatting
02/11/18	leba	Updated text
18/11/18	renha	Added configuration management
29/11/18	kati	Last touch-up

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

10 Risk Analysis

Risk management is very important when it comes to project management. Therefore, it is highly recommended that the project manager/project staff pin down the possible risk within a software development project. Risk, in terms of technological change and hardware [41, p.644-646], is not included in the Risk Analysis of *strEAT*, since we are not programming the actual app *strEAT*.

The following figures are divided into:

1. Risk - The possible risk is identified.
2. Affect - What the risk can affect.
3. Description - A deeper description of the risk.
4. Strategy - What can be done to prevent the risk from happening, or what can be done if the risk happens.
5. Probabilities - Categorising the risk as having a low, moderate or high probability of occurring.
6. Effects - Categorising the risk as having a tolerable, serious or catastrophic effect should it occur.

10.1 Product Oriented Risks

Risk	Affects	Description	Strategy	Probabilities	Effect
Vendors and users are not interested in the app	Product	If the needs of the vendors and users are not properly identified and researched upon, it can lead to vendors and users not using the app	Keep contact with many different vendors and users, to make sure that their needs are included in the app development	Moderate-High This can be set lower when vendors and users are investigated more thoroughly	Catastrophic
Another company is developing the same app	Product (and project)	A competing company is "secretly" developing the same app	Keep track of similar companies and try to keep track of what they are developing	Moderate-High Could JustEat or Wolt have the same idea, and if not, why?	Catastrophic

Table 6: Product oriented risks for our project

10.2 Project Oriented Risk

Risk	Affects	Description	Strategy	Probabilities	Effect
Agreed deadlines exceeded	Project (and product)	If a project member is not delivering within the agreed upon deadline, it can prevent other project members continuing or starting new tasks	Making sure that all the project members have been part of the project planning and therefore are aware of the deadlines	Moderate	Serious
Illness in the project staff	Project (and product)	If a project members is ill and cannot deliver, other project members will have to take on their tasks and possibly leading to delays in their own tasks	All project members are aware of each others' tasks, so they know what the ill member is working on, and should therefore be able to work on their tasks with relative ease	Moderate	Tolerable– Serious
Project members competencies are not used properly	Project	The project members have different competencies. If a project member feels like their competencies are not being utilized properly it can lead to declining engagement in the project. Furthermore the project might miss out on a needed set of competencies	The project members talk about or write down their set of competencies so every project member is aware of each other's competencies, and can ask for their help when needed	High	Catastrophic

Table 7: Project oriented risks for our project

10.3 Risk Prioritization

From this analysis, we can begin to prioritise the different risks according to their probability and the severity of their effects. This is a valuable tool in mitigating potential dangers in the future product development. After intense discussion, the *strEAT* team have agreed on the following prioritized lists for the product-oriented risks and project-oriented risks:

Product Oriented Risks:

1. Another company is developing the same app (product oriented) During our initial search of the market we have found no indication that other companies are developing the same app, such as JustEat and Wolt. However, we must also consider that information regarding the development of such an app would most likely not be publically available. The project team consider this as a high risk because we can not control or prevent this from happening.
2. Vendors and users are not interested in the app (product oriented) This would completely destroy the feasibility of the project and a priori we would assume a high likelihood for this risk. In order to mitigate this risk, we decided to do a small market feasibility analysis at Reffen during our project kickoff. This consisted of interviews with vendors about their opinions concerning the project. In general, there was a positive attitude to the project and the vendors showed interest in the product. In order to continually keep this risk low we should keep in touch with the vendors at Reffen and possibly conduct more interviews. Therefore, this is a risk that the project team can keep track off.

Project Oriented Risks:

1. Project members competencies are not used properly (project oriented) To make sure that competencies in the team are used properly, we will make note of the different group members competencies in line with their previous education and work experience. We will incorporate this into the planning, and also make sure that different team members try out tasks that they haven't done before.
2. Agreed deadlines exceeded (project oriented) Missing deadlines can have serious consequences for the team and we will need to avoid this happening at all costs. In order to mitigate this risk, we have all deadlines posted on our Trello board.
3. Illness in the project staff (project oriented) It is very difficult to prevent team members from getting ill, thus this risk is a difficult one to control. Due to the size of our team, the event of a single team member falling ill for a few days should not disrupt our work too much. In order to mitigate the consequences of ill team members, we agree on a protocol for when team members fall ill and therefore need help for finishing deadlines. It is each team members responsibility to tell the other team members if they cannot finish a task because of illness. We will add this to our team protocol. We consider this risk relevant but with low prioritization.

The risk analysis will be followed up by the project members. If other risks are identified throughout the project they will be added to the figure and prioritized list.

Configuration Management

Document title: Explore the Use Context

Version:^a v3.06

Github link: [.../MainDocument/sections/UseContext.tex](#)

Trello card link: [trello.com/c/O4hq8etf/7-explore-the-use-context](#)

Responsible: idbo & kati

Status: Finished

Changelog:

05/19/18	kati & idbo	Created
01/11/18	abru	Updated formatting
02/11/18	leba	Updated text
02/11/18	idbo	Updated text
14/11/18	abru	Updated table
18/11/18	renha	Added configuration management
23/11/18	leba	Updated text
25/11/18	leba	Consistency checks

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

11 Explore the Use Context

In order to understand the context of use of the *strEAT* app, we need to understand both the social and technical context of our app. This necessitates thorough market research. Research findings are distilled into the aforementioned social and technical contexts.

11.1 Research

After some brainstorming, we decided to focus on some main research topics, such as: How do we make this app reflect customers/users' preferences?

- Are the street food vendors visible in Copenhagen?
- Do people actually need an app to find the location of street food vendors?
- What do users expect from an app like *strEAT*
- Additional features.

In order to research this, we found that interviews would probably be the best way to go. Therefore, we created an interview guide that reflected an investigation of the user preferences and, furthermore, involved getting in contact with people working in the industry/or closely related to the industry. To see the interviews visit Appendix [A](#). In relation to the technical context, we found that one of the main features of our app should be the map, since our app idea revolves around locating the street food vendors for the users. In addition, we found that the interviews reflected an interest and need for a rating system. This also correlates with some of the main issues raised in the section about the software qualities. Therefore, we decided to investigate the technical aspect of Geolocation.

Social Context	Technical Context
Necessity of the app	Geolocation
Expectations for the app	Data collection and analysis
Usage	Data protection - GDPR
Potential users	Rating system
Targeted group	Google integration
Problematic issues/critique	Facebook integration
Cost/benefit	Push notifications
Advertisement	Cross-platform OS
Subscriptions	Programming language

Table 8: Overview of our brainstorm

11.2 The Social Context

This part will cover the use context, where we have investigated the potential users of the app. In order for the *strEAT* team to get a sense of the possible users of the app, we have interviewed two potential users: two 25 year old female students, and one bar manager from Tipi Bar at Verdenshjørnet, Nørrebro, Copenhagen. In order to get the interview data, we have used the semi structured interview form. The two students are referred as Student1 and Student2, the interview person from Tipi bar is referred as Bar Manager.

11.2.1 Potential user interviews: Two students age 25

In order to understand if apps were used to order food, we asked the two students if they used apps to order food, and what kind of apps they used. Both students used JustEat and one had additionally used the apps Wolt and Too Good To Go. Student1 was asked what a food app should contain:

”First of all, provide me with a menu card: what’s on the menu; what can I get; the pricings; the opportunity to remove things from a dish if you have allergies or if you don’t like spicy stuff; and payment methods.” (Student1, page 1).

Futhermore, Student1 also thought that food apps should contain a search engine and a map (ibid.). Student2 was asked which features would be nice to have on the *strEAT* app, which was not necessarily on JustEat.

“That it is available for every area. If there are different restaurants, the restaurants should specify which area or district in Copenhagen they can go to, before you go to pay. Because often it happens when you pay and you put in the address, and then they give you feedback that “Oh we don’t deliver to these places”. So, something about that. (..) (Student2, page 1)

Furthermore, Student2 found it important that she could see ratings from other users on the *strEAT* app (Student2, page 1).

From the student’s answers, we can conclude that the *strEAT* app should contain: menu cards, pricings, the opportunity to remove food items from the dishes, area availability, ratings from other users, a search engine, a map, and payment methods. At the end of the interview with Student1, she wanted to make a point about street food and its charm:

”I kind off think the charm with street food is that you go there and eat there, not that you order it and go home and eat, so I think actually, maybe, just create some sort of overview maybe a point system, and sometimes street food can be quite expensive... maybe some sort of clip card system, where you get the tenth dish for free, I think that would be cool.” (Student1, page 1)

The team should take into consideration that a part of eating street food is actually to eat the street food close to the street food truck. Therefore, the ordering feature in the *strEAT* app might not be necessary. Rather, menu cards, ratings, pricings, and a map are more useful for the users of the app. If these findings are considered an accurate representation of potential users’ attitudes, then this should be investigated more by the project team.

11.2.2 Bar Manager Interview

The two interviewers from the project team both live nearby Verdenshjørnet, where the Tipi Bar is located. A couple of years ago there was ten street food trucks or more at Verdenshjørnet, but now there is only one street food truck left. In order to understand what happened with the area and why the street food trucks moved, we interviewed the bar manager of Tipi Bar.

“When we started the bar, or the owner started the bar, it was supposed to be a bar in the middle of a lot of street food and it worked for a while, but because they [the food trucks] opened and closed whenever they wanted to... they didn’t have opening hours like stores have, it made it difficult when guests came here, because they maybe came here for a specific kind of food truck and that was maybe closed, and there was maybe two [food

trucks] opened out of the ten [food trucks] we had here. So people came here and were more disappointed than happy. So then we slowly built out Tipi Bar and started getting more guests here and then the food trucks moved out and now we have the only food truck that is actually open, which is ‘Chop stick’ [laugh] (..) (Bar Manager, page 1).

The original plan for Verdenshjørnet did not work out, since the street food trucks did not have common opening hours, and the project for Verdenshjørnet were not able to create a collaboration between the bar and the street food trucks because of that issue. Furthermore, the bar manager pointed out that the area was pretty “dead” during the winter, and that was also an issue.

From the interview of the bar manager, we can conclude that the *strEAT* app should contain the street food trucks opening hours. The bar manager did not know where the street food trucks moved to, but it is the project teams plan to find street food vendors and interview them about whether they would have an interest in a street food app, what an app like *strEAT* should contain, and if they are still open during the winter.

11.3 The Technical Context

In this section we will go some of the technical aspects that we have found interesting to investigate further. Here we include the main topics of geolocation, rating systems and push notifications.

11.3.1 Geolocation

Our interviews highly suggested that our assumptions regarding using a map to get an overview of the street food vendors in Copenhagen, is a good way to go. Therefore, we will explore further the technical context of using geolocation in our app.

Today, geolocation is increasingly being used for mobile apps, to identify the current geographical location of the user/mobile device. Geolocation uses both Wi-Fi, geofencing, Cell ID, and GPS to gather data for detecting the location. Contrary to GPS (Global Positioning System), geolocation also uses cell site triangulation for identifying the current location of a mobile device. Cell site triangulation, is a way of detecting the location of a device by combining the measurements from three different cell towers (if possible). The cell tower antenna arrays sends signals to each other, in order to detect the most precise positioning of the mobile device. This also means that it will work better in urban areas where the cell towers are located closer to each other, than in rural areas. In picture X it is illustrated how three cell towers communicate in order to locate the mobile device (red circle). [\[44\]](#)

We will therefore consider using geolocation, since the main aim of *strEAT* is to enable users to find street food vendors close to the current location at the exact moment when the user checks the app. As the bar manager from Tipi Bar mentions, she doesn’t know where the street food trucks went once they stopped showing up at Verdenshjørnet. Likewise, she explains that many customers were disappointed when they showed up and the street food trucks were gone or closed. Therefore, it seems relevant to be able to use geolocation to track the street food trucks, for the user to detect their location. Additionally, another feature could be that it should be visible on the map whether the truck is open or closed, for example, by changing the colour of its marker on the map. Lastly, geolocation seems to be ideal when used in urban areas, and, therefore, it is appropriate for our app, since we will track the food trucks located in Copenhagen.

In relation to geolocation, one thing we need to be aware of is the EU’s General Data Protection Regulation (GDPR). This regulation is a set of data protection rules for European companies to abide by [\[11\]](#). Because geolocation collects location data, which is personal data, the GDPR also applies here. One of the main requirements in our case is to inform the user of about how their data is collected, stored and what it is used for.

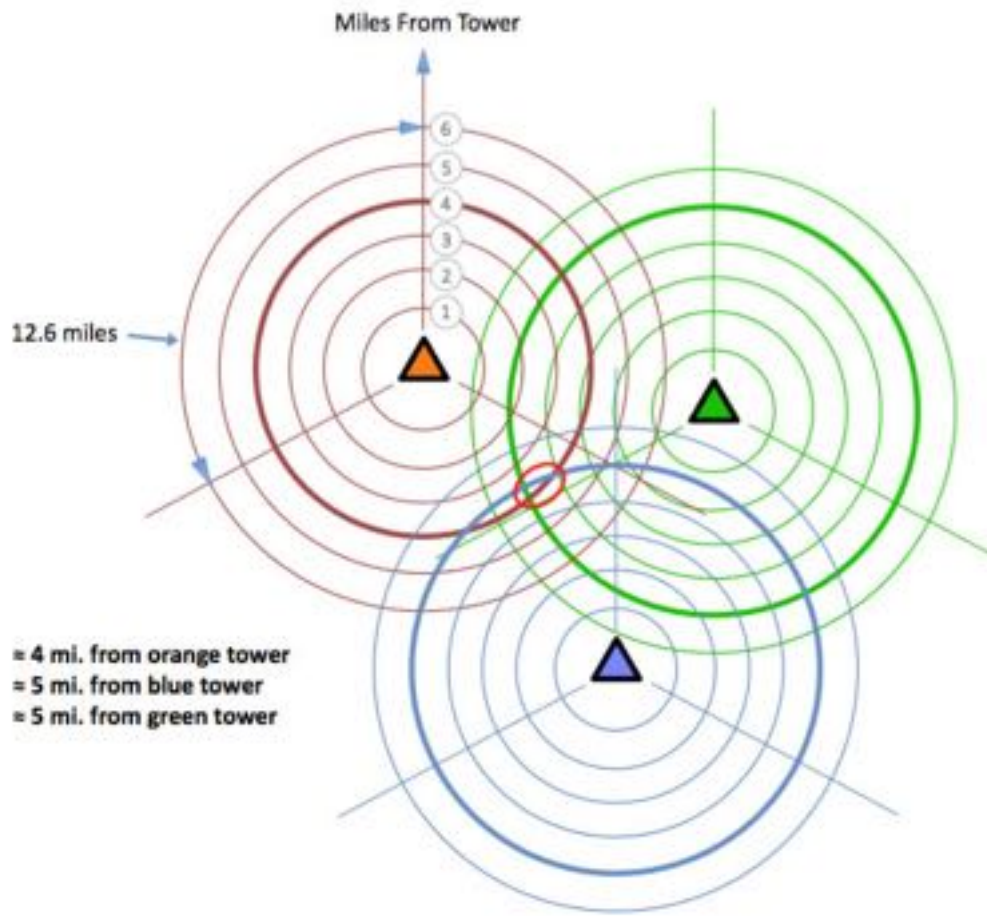


Figure 11: Picture of cell site triangulation. The figure is taken from [27]

11.3.2 Rating Systems

Another feature that the two interviewees mentioned was a rating system for the street food vendors. Therefore, this part will not focus on a rating system for the app *strEAT*, but ratings for the street food vendors inside the app. Furthermore, this part will not explain the back-end implementation of a rating system, but the potential impact from the customers ratings on the street food vendors. In Wharton University of Pennsylvanias online business journal “Wharton@Knowledge” Chen Jin, a post-doctoral researcher answers questions about his research from his paper on ‘The Impact of the Rating System on Online Marketplaces’ (2017). [25]

There are two types of rating systems: unilateral rating systems, where the customers are the only ones who can submit their ratings, and bilateral rating systems, where both customer and service provider can rate each other. Furthermore, Jins research showed that the rating system had a different impact on the service providers, the customers and the platform. In relation to *strEAT*, it would make more sense to implement a unilateral rating system, since there is no reason for the vendors to rate the customers.

Jin argues that it is always in the platforms favour to have a rating system, because it pushes the service providers to give extra effort, so that they will get good ratings from the customers, and therefore more customers. If the platforms are charging the service providers with a fee, they can increase and decrease the pricing in proportion to how many customers are using the platform.

In relation to the *strEAT* app, the project team has been concerned about the effects of a rating system on the street food vendors, especially if they received a bad review. Jin also thought that the

service providers circumstances would get worse after the implementation of the unilateral rating system, because they were more 'observable' and if the platform became popular, could be charged a bigger fee from the platform. In fact, that was not the case, because the service providers made a better effort, they got better customer reviews, and therefore received more customers. Jins article is not focusing on the effect for the service providers if they receive a bad review. The project team finds it important that the rating system in *strEAT* is not only a star rating system, but the customer has to leave a comment if they want to rate the street food vendor, since a star rating without a comment does not say much about the place. Mean and cruel reviews should of course be deleted.

In terms of the rating system and the benefits for the platform, there is no argument for not having it if it make sense to the service that it provides. The pricing strategy though has to be carefully planned together with marketing research [25].

11.3.3 Push Notification

An additional feature of the app, that we have discussed is push notifications. In relation to this, we have considered to allow the customers to “follow” their favourite street food vendor in the *strEAT* app. When a customer follows a street food vendor, they will be asked to approve push notifications from the app. When a street food vendor changes location or have any news about their menu, discounts or opening hours, the customer will receive a push notification.

“In its essence, a push notification is a brief message or alert that is sent through an installed app to everyone who has installed the app and who has enabled the receipt of these messages. It does not matter whether you have an iPhone, an Android or any other brand of phone; [...] To provide more accessibility, the app does not have to be open at the time of the notification in order for the message to be visible. This allows you to reach a wide range of people by “pushing” your message to an entire group at the same time.” [7].

Having push notifications are beneficial for both the app owners, street food vendors and customers. Push notifications can result in increased user engagement, as customers will keep using the app [8]. When customers receive push notifications about their favourite street food vendor, they are more likely to purchase food from that truck. Likewise, they will benefit from possible discount notifications. In addition, push notifications also “help businesses avoid the dreaded act of download-and-ignore” [7]. However, in relation to all of this, it is highly important to keep in mind not to push too many notifications to the customers. As this might result in deletion of the app.

In order to deliver a push notification, three actors are involved: The operating system push notification service (OSPNS), App publisher and Client app. Each mobile operating service has its own OSPNS service, that allows a push notification to include the required text, app badges and sounds (REF twilio). The app publisher, which is responsible for enabling the OSPNS, and the Client app, which is the app from which the customers will receive the push notification. [7]

As mentioned, each mobile operating service has its own OSPNS service. In relation to this, it is important to mention that we have planned to allow the *strEAT* app to be cross-platform. “Cross-platform refers to the ability of software to operate on more than one platform with identical (or nearly identical) functionality.” [34]. However, this can refer to numerous things depending on the specific context:

- The type of operating system
- The type of processor
- The type of hardware system [34]

In relation to the *strEAT* app it is important that the app can run on two operating systems: Android (Linux based operating system) and iOS (Apple based operating system).

Configuration Management

Document title: Use Case Diagram and Scenarios

Version:^a v3.11

Github link: [.../MainDocument/sections/Usecase.tex](#)

Trello card link: N/A

Responsible: kati

Status: Finished

Changelog:

18/10/18	kati	Created
27/10/18	renha	Updated formatting
30/10/18	kati	Updated "Actors"
31/10/18	kati	Updated formatting & text
01/11/18	abru	Added use case diagrams
02/11/18	leba	Updated formatting & text
02/11/18	abru	Updated diagrams, formatting & text
06/11/18	kati	Updated diagrams
09/11/18	leba	Updated text
18/11/18	abru	Added use case diagram legend
		Updated diagrams, formatting & text
18/11/18	renha	Added configuration management
23/11/18	leba	Updated text

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

12 Use Case Diagram and Scenarios

This section discusses various scenarios users of the app may find themselves in and presents the findings in the form of use case diagrams. In order to create use case diagrams, we brainstormed the tasks/activities the software is supposed to support.

- The app needs to be able to locate street food vendors close to the user.
- The app needs to make it possible for the user to find specific food.
- The app needs to support a review system.
- The app needs to provide information about the specific street food vendors: opening hours, menu, smiley report, etc.
- The app need to support the street food vendors by visualizing sales.
- The app needs to support a payment/ordering system.
- The app needs to support a way for users to follow their favourite street food vendors, so that the user will be notified on location changes, opening hours, etc.

12.1 Scenarios

From these bullet points we decided on three specific scenarios that we thought would be interesting to work with in our Mock-Up.

Scenario 1: A user (customer) is hungry and wants to find some food nearby. The user has heard about an app called *strEAT* that locates food trucks in Copenhagen. The user decides to use the *strEAT* app to find the location of a street food vendor, instead of looking for a place to eat on foot. After the user downloads the app, the user creates a new profile using their email. They locate a specific vendor on the map and look up their information. On the map, they can see that the food truck is not that far away from their current location. In the information view, the user can see that the food truck is currently open, and that they have a Mexican menu that looks tasty. They check the reviews and choose to order a taco and a sparkling water. Afterwards, the user walks directly to the vendor, picks up their food, and pays for it.

Scenario 2: The user (customer) has a favourite food truck that they want to locate. Unfortunately, the food truck has a new location every week. They sign in to their *strEAT* app and use it to search for the food truck. As they know the name of the food truck, they search the food truck by its name. The user finds it and gets its current location on the map. Now the user knows the exact location of the food truck today. They walk to the food truck, to order their food of choice.

Scenario 3: A user (street food vendor) would like to find out where to locate his truck today. Luckily they recently signed up and registered their food truck in the app called *strEAT*. Therefore, they sign in as a street food vendor. On the *strEAT* vendor page, the user can access their sales figures, which recommends the top 5 best locations for their food truck, based on previous collected data. Now they know where to go today in order to maximize profits.

12.2 Actors

Afterwards, we identified the most relevant actors involved in our scenarios. In order to create use case diagrams, it is necessary to identify the specific actors involved [41]. The actors are the so-called users of the app. In our case, we have identified two kinds of human actors. These were the customers and vendors, who would both be users of the *strEAT* app. However, actors could also be other systems that our app will interact with. The two types of actors involved in our scenarios are described below.

Actor: Customer

Purpose: A person looking for street food.

Characteristic: Someone hungry.

Examples:

Actor A is out in town and wants to find an area with a nice selection of street food. They want to decide on what kind of food they would like when they locate the food truck. They would therefore use the map to locate an area with several food trucks.

Actor B is craving something specific, and would therefore only use the app to find this. Therefore, they would use the search function to locate a specific food truck.

Actor: Vendor

Purpose: A vendor looking to sell their food.

Characteristic: A person that owns a food truck and wants to broaden their customer base.

Examples:

Actor A is moving around a lot, and wants to keep customers updated on their current location. Actor A will therefore use the app to provide customers with their current location.

Actor B has flexible opening hours. They, therefore, use the app to advertise for the food truck's opening hours. Thereby, customers that follow the food truck on *strEAT* will get a notification when they update the opening hours.

12.3 Use Case Diagrams

Finally, we developed three use-case diagrams based on the actors in the scenarios. These diagrams are presented on the following pages in figures [12], [13], and [14], along with some elaborations on their content. At the end of the section, in table [9], the legend for the diagrams can be found.

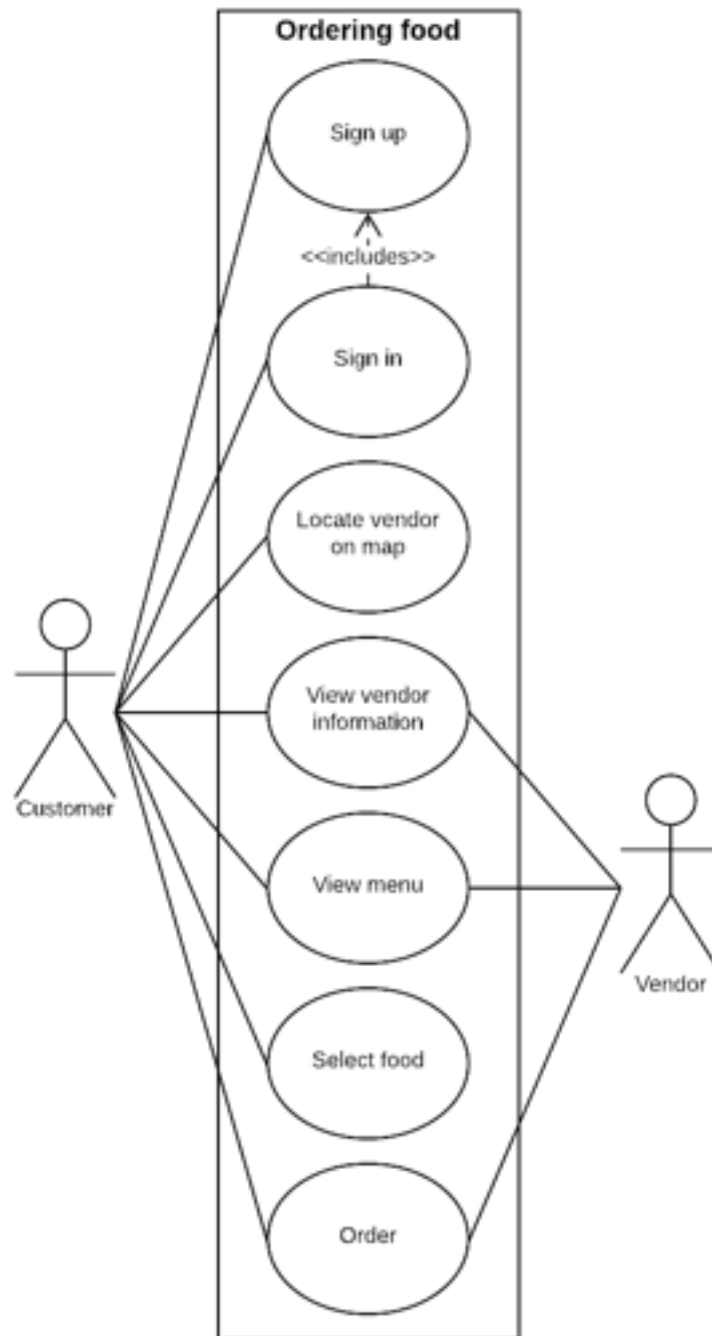


Figure 12: Use Case Diagram for Scenario 1

In figure [12](#), the first scenario is illustrated. Here the customer is the primary actor, and the vendor is the secondary actor. The user can sign in, which includes (necessitates) that they have signed up. They now have the possibility to locate a vendor on the map to find a street food vendor nearby. From here the customer gets the possibility of viewing specific vendor information, which has been submitted by the vendor. When a customer accesses the vendor information, they can now choose to view the menu, again submitted by the vendor. While looking at the menu, the customer can select food that they wish to order. When they place the order, the order will automatically be sent to and processed by the vendor.

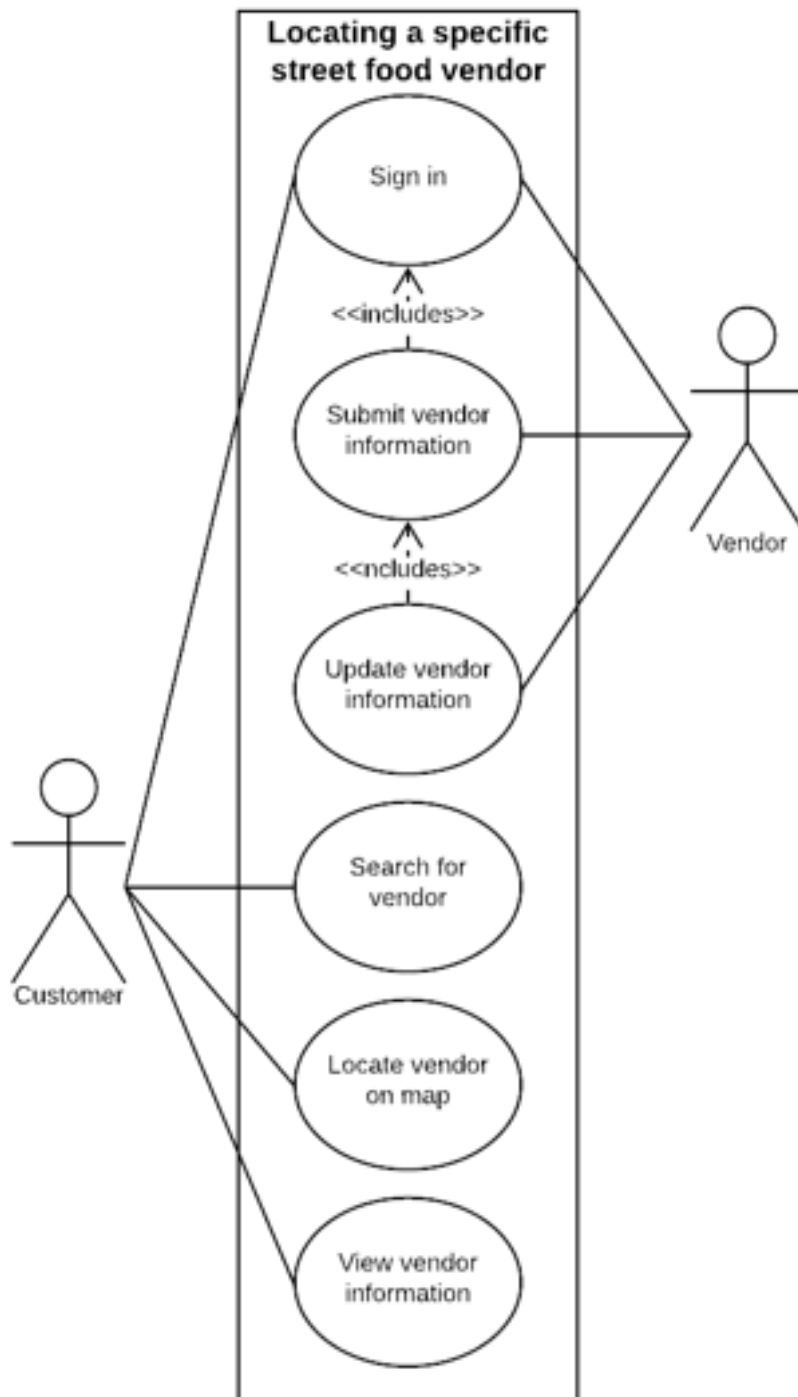


Figure 13: Use Case Diagram for Scenario 2

In figure [13](#), the customer is still the primary actor. In this diagram it is shown how a customer is searching for a specific street food vendor, instead of looking up nearby street food vendors on the map. In this scenario, the customer already knows what vendor they are looking for. Likewise, it is illustrated how a vendor can change their vendor information. If the vendor changes location or opening hours, the vendor information will automatically be updated. When a customer searches for a vendor the customer will immediately be able to view these changes, either in the information view or on the vendor map (since the map using geolocation will track the street food truck).

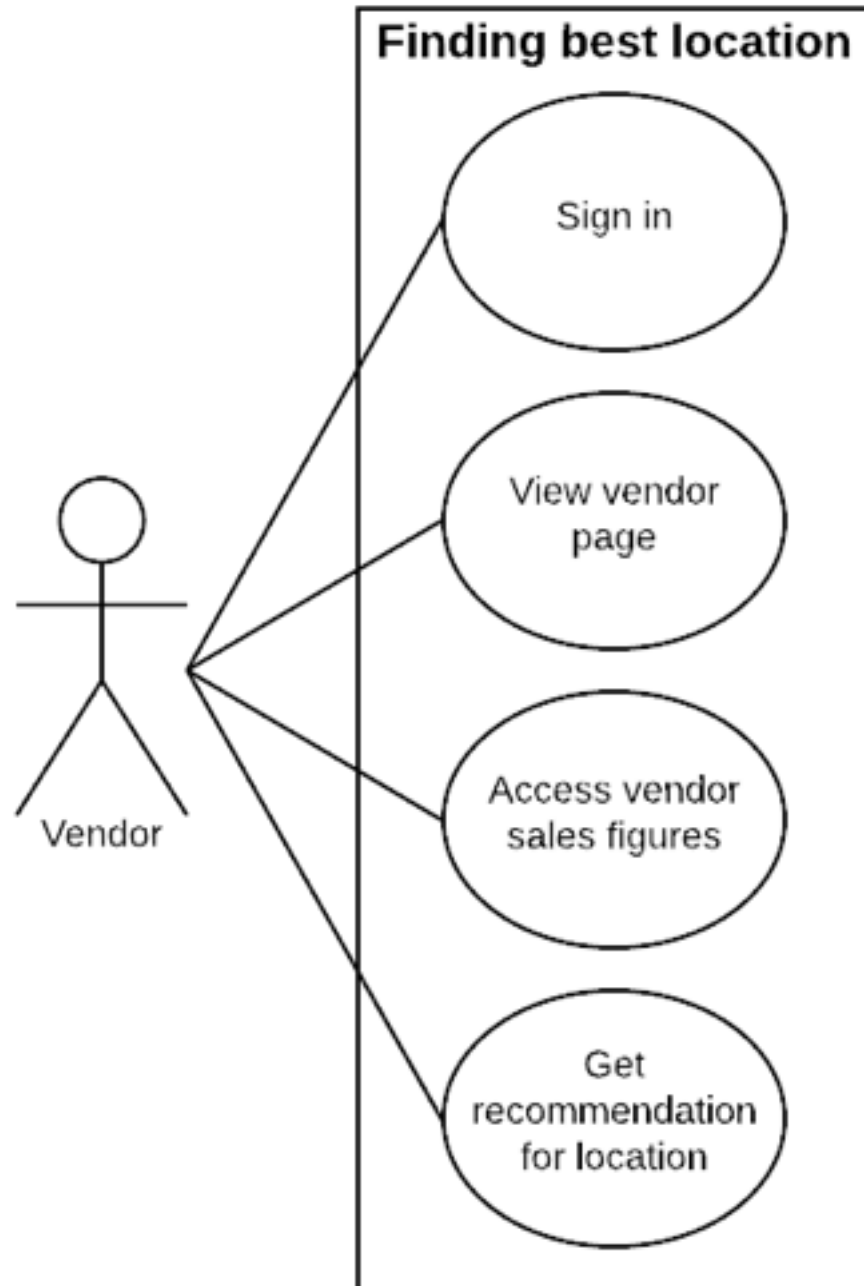


Figure 14: Use Case Diagram for Scenario 3

In figure 14, the street food vendor is the primary actor. When a customer orders food, the sales data will be stored according to the time the order was placed and the location of the food truck. This data is stored in the app, such that the street food vendor can access it in their "sales figures". When accessing the sales figures, the vendor will get the current recommendations for where to locate his street food truck.





	System boundary box: The square box defines a scenario or system, and contains the use cases of the given scenario. A single use case diagram can contain multiple different systems, that may or may not interact with each other. Each scenario can consist of any number of use cases. [29]
	Use Case: The elliptical containers represent individual use cases. They can be present both in- and outside of the system boundary boxes. Use cases are associated with the actors of the scenario, and sometimes each other. [29]
	Actor: The stick figure represents an actor. Actors are the ones who interact with the scenario, and by extension the use cases. They must be people (or other systems) outside of the given system, who contribute to (or take something from) the system through their interactions with the use cases. [29]
	Association: The straight line represents an association between an actor and a use case. They do not further specify the nature of this relationship, only that it exists. An actor can be associated with any number of use cases, and a single use case can relate to more than one actor. [29]
	Include/exclude: The dotted line with open arrowhead is another type of association. It is used to map the relationships between the individual use cases. There are two different kinds of such associations; <i>extend</i> and <i>include</i> , the type of a given association is specified along the dotted line. When a use case (A) includes another use case (B), it means that when doing A, you must also do B – in other words A is dependant on B. If instead A extends B, then you have the option to do A when doing B, however it is not required – A is an optional extension of B. [29]

Table 9: Use case diagram legend

Configuration Management

Document title: Rich Pictures

Version:^a v3.07

Github link: [.../MainDocument/sections/RichPictures.tex](#)

Trello card link: [trello.com/c/ldfwhhOo/4-rich-pictures](#)

Responsible: jopo & mgab

Status: Finished

Changelog:

??/??/??	?	Created
27/10/18	jopo	Added rich picture 2
01/11/18	jopo	Added new rich pictures
02/11/18	jopp	Added descriptions
02/11/18	leba	Updated formatting & text
02/11/18	renha	Updated formatting
09/11/18	leba	Updated text
18/11/18	renha	Added configuration management

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

13 Rich Pictures

In the following rich pictures, we are displaying the most important stakeholders for the *strEAT* app and their connections to each other. Potential issues that need to be taken care of in the future are displayed next to the arrows to transport the subjects from the pre-analytic stage of the rich picture format to the analysis stage that follows. The graphic is a tool developers may return to throughout the life of the project should they require a quick overview of the original ideas. It should be referred to throughout the entire process to guarantee an authentic and consistent outcome.

13.1 A World With *strEAT*

Figure 15 depicts the process of looking for street food vendor with the *strEAT* app. The design is centered around the user and an app. The hungry user, willing to eat street food (either at the vendor's location or ordered online for pick up), opens the app. If already registered, the app immediately provides him with a map of food trucks and the possibility to filter choices. For each food truck, they have information about the current location, a menu with pictures of food, and the possibility to pay online and hence skip the line when picking up the food. The relationship between vendor and customer is simple - the vendor provides the customer with food, and the customer pays the vendor. Arrows from the vendor to the app indicate that the vendor must update the menu, opening hours and take care of *SkipTheLine*TM orders (stipulating the time needed to prepare the food for each order placed). The app, in return, brings the vendor more customers and useful statistics (e.g. which locations were the most profitable). They pay monthly subscription fee to the developers and the developers provide app maintenance, taking into consideration customer feedback. Feedback may include discrepancies between the app and reality, e.g. outdated menus, incorrect opening hours, etc.

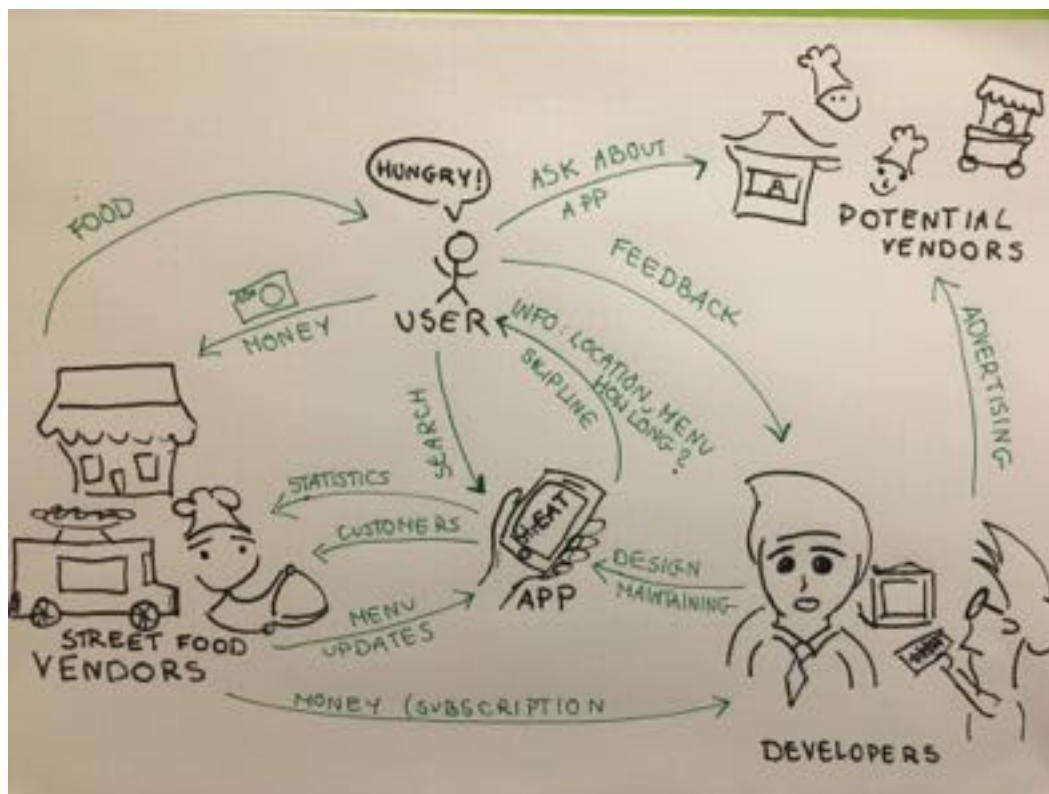


Figure 15: The world with the app

13.2 A World Without strEAT

Figure 16 is also centred around the user but here no app is taken into account while searching for food. We have a hungry person (purposefully not called a "user" anymore) willing to eat, not in an ordinary restaurant, but at a street food establishment. This may be because either they prefer the atmosphere that comes with street food, or that the prices are usually lower. The user has a couple of options. They might, of course, wander through the city streets and stumble upon a food truck by chance, but that may not be a very satisfying solution. They might ask friends or locals (if being from outside of town) for recommendations. The recommendation obtained might be useful, but the location of the food truck remains to be known, as the owners may be partial to relocation now and again. The hungry person might then use an ordinary Google search, but that usually points to restaurants; food truck vendors owning and maintaining a website for their business is not particularly common. However, it still might point to some social media accounts (e.g. Facebook and Instagram) that might disclose the vendor's current location.

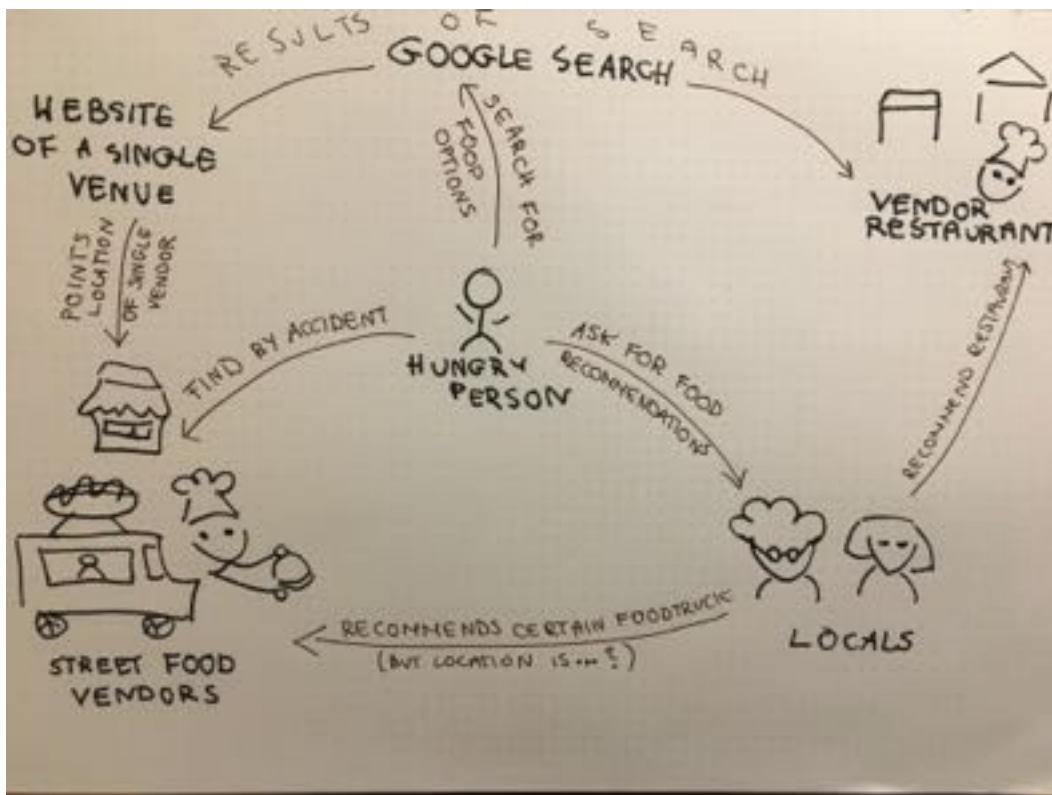


Figure 16: The world without the app

Configuration Management

Document title: Documenting Project Requirements

Version:^a v3.05

Github link: [.../MainDocument/sections/Requirements.tex](#)

Trello card link: [trello.com/c/Pli19fXu/25-document-project-requirements](#)

Responsible: leba

Status: Finished

Changelog:

01/11/18	leba	Created
01/11/18	leba	Updated formatting & text
01/11/18	abru	Added text for "Estimate"
09/11/18	leba	Updated formatting & text
13/11/18	leba	Added non-functional requirements
18/11/18	renha	Added configuration management
25/11/18	leba	Added pre-ample

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

14 Documenting Project Requirements

This section is presented in such a way as to portray an abridged version of a typical requirements document found in the industry. As per the layout suggested in the course notes, the document shall consist of an introduction, general description, and requirements specification.

14.1 Introduction

14.1.1 Purpose

The following document is intended to formally state the requirements of the current system as stipulated by the current list of user requirements, presented in the form of user stories. Due to the iterative nature of the agile development methodology, the user stories that make up the Product Backlog, and subsequently define the system requirements, are not fixed and may change over the course of the project at the discretion of the Product Owner.

14.1.2 Scope

The product required to be delivered is a cross-platform mobile application that focuses on connecting street food vendors with potential customers.

14.1.3 Overview

The remainder of this requirements document consists of a general description of the product (perspective, function, and users) and requirements specification (user stories, Product Backlog, and Sprint Backlog).

14.2 General Description

14.2.1 Product Perspective

Existing food-based mobile applications fail to address the needs of vendors who lack fixed locations. *strEAT* aims to cater to those needs by providing vendors with a medium through which they can inform potential customers of their whereabouts within Copenhagen.

14.2.2 Product Function

Essentially, the app will provide customers with the means to locate street food vendors throughout Copenhagen, view menu's of these vendors, and place orders. Vendors will be given the opportunity to update their current location via the app, view orders, and view various sales statistics.

14.2.3 User Characteristics

Users of the application are customers and vendors. The customer is one who is searching for food, specifically street food, perhaps of a specific vendor or simply browsing the available options. The customer is expected to be an active smartphone user of <50 years old. The vendor is an entity who is providing the food, ideally of the non-formal street variety, to customers.

14.3 Requirements Specification

Agile development necessitates requirements reevaluation after each product iteration, i.e. after each sprint cycle. As such, the functional user requirements presented below, in the form of user stories, are the result of the first brainstorm sessions and subject to change at the discretion of the Product Owner. Likewise, the initial non-functional requirements are also presented below and are subject to change.

14.3.1 Functional Requirements

14.3.1.1 User Stories

Customer

1. As a customer, I want to be able to choose list or map view.
2. As a customer, I want to be able to sort the list by specific criteria (e.g. price range, distance from me, and rating).
3. As a customer, I want to be able to search for a vendor by name.
4. As a customer, I want to be able to see the menus of the vendors.
5. As a customer, I want to see current vendor locations on a map.
6. As a customer, I want to see my location on a map.
7. As a customer, I want to be able to click on a vendor on a map to get their information (e.g. opening hours, price range, cuisine, distance from me, rating, and estimated waiting time).
8. As a customer, I don't want to exit the map to see vendor information.
9. As a customer, I want to filter the map or list by specific criteria (e.g. price range, distance from me, rating, cuisine, followed vendors, and estimated waiting time).
10. As a customer, I want to be able to "follow" vendors and receive notifications.
11. As a customer, I want to be able to specify a pickup time.
12. As a customer, I want to be able to suggest a vendor to a friend.
13. As a customer, I want to be able to pay through the app.

Vendor

1. As a vendor, I want to be able to change my location.
2. As a vendor, I want to be able to change my general information.
3. As a vendor, I want to be able to change my menu(s).
4. As a vendor, I want to see my sales figures over multiple time periods.
5. As a vendor, I want to see my customer demographics.
6. As a vendor, I want to see my sales figures per location.
7. As a vendor, I want to know how often we get clicked on in the app.

14.3.2 Non-Functional Requirements

1. Within 2 minutes of login, the vendor shall be able to update their location (via any of the possible methods).
2. The customer shall be able to find a suitable vendor's menu via the search function (with criteria filtering) within 1 minute of opening the application.
3. The customer shall be able to go from a vendor's menu to completed order payment (with order confirmation) within 1 minute.
4. The user shall be able to navigate to and change personal information within 1 minute of opening the application.

14.3.3 Product Backlog

Based on the functional requirements, presented above as user stories, we arrive at the following Product Backlog ordered based on perceived importance:

1. As a customer, I want to see vendor locations on a map.
2. As a customer, I want to see my location on a map.
3. As a vendor, I want to be able to change my location.
4. As a customer, I want to be able to click on a vendor on a map to get their information.
5. As a vendor, I want to be able to change my menu(s).
6. As a customer, I want to be able to choose list or map view.
7. As a customer, I want to be able to search for a vendor by name.
8. As a customer, I want to filter the map or list by specific criteria.
9. As a customer, I want to be able to sort the list by specific criteria.
10. As a customer, I want to be able to see the menus of the vendors.
11. As a customer, I want to be able to pay through the app.
12. As a customer, I want to be able to specify a pickup time.
13. As a vendor, I want to be able to change my general information on the app.
14. As a customer, I want to be able to "follow" vendors and receive notifications.
15. As a customer, I don't want to exit the map to see vendor information.
16. As a customer, I want to be able to suggest a vendor to a friend.
17. As a vendor, I want to see my sales figures over multiple time periods.
18. As a vendor, I want to see my customer demographics.
19. As a vendor, I want to see my sales figures per location.
20. As a vendor, I want to know how often we get clicked on in the app.

14.3.4 Estimation

All estimates for the project will be approximated through planning poker involving both the project team and the project owner, in order to ensure they are satisfactory and as accurate as possible. The agreed upon estimates for the different tasks in the backlog were presented in table 3 in the Project Plan and Estimation section of this portfolio.

14.3.5 Sprint Backlog

Each Sprint Backlog is crafted by the Scrum Master and Project Owner, and refined with the input of the Development Team. A few remaining Product Backlog items remaining at the beginning of each Sprint are selected based on their level of priority. The number of items chosen depends on the amount of work the Scrum Team believes can be achieved in the upcoming Sprint. The Sprint Backlog is converted into workable tasks by the development team, the completion of which are expected to result in a working product increment.

14.3.5.1 Sprint User Stories

1. As a customer, I want to see vendor locations on a map.
2. As a customer, I want to see my location on a map.
3. As a vendor, I want to be able to change my location.
4. As a customer, I want to be able to click on a vendor on a map to get their information.

14.3.5.2 Sprint Goal

The outcome of this sprint is to have a basic working app with: a front page prompting for customer or vendor login; a customer home page showing Google maps centred on the customer's location and populated by vendor location pins; and a vendor home page with buttons for "Update location", "Update profile" and "Update menu". When prompted by the customer clicking on a vendor's pin, the application will request vendor information from a database and present it as a text box next to the pin.

14.3.5.3 System Requirements

Below are the system requirements derived from the scenarios described by the Sprint Backlog items.

"...I want to see vendor locations and my location on a map."

- Customer is presented with a login page and proceeds to login via the chosen method.
- Customer is presented with the Google Maps API centered on their current location and populated with pins representing currently operating vendors.
- The vendors shown are retrieved from the application's relational database based on the current time and the opening hours stated by the vendors themselves.
- The customer's current location is updated in the Customer object.

"...I want to be able to change my location."

- Vendor is presented with a login page and proceeds to login via the chosen method.
- Vendor is presented with the vendor homepage which consists of options for changing location, profile and menus.
- Vendor selects the location option and is presented with another page prompting the vendor to select either "Use my current location" or "Enter location manually".
- The former allows the application to automatically detect the vendor's location.
- Selection of the latter presents the vendor with a textbox with an autofill dropdown menu of registered addresses matching the current text.
- The vendor must select an option from the autofill populated by registered addresses.
- The new location is stored in the Vendor object.

"...I want to be able to click on a vendor on a map to get their information."

- Presented with vendor pins on the Google Maps API, the customer selects a pin.
- The application retrieves the selected vendor information (name, price range, cuisine, opening times, estimated waiting time) from the relational database.
- The information is presented in a textbox beside the pin.

Configuration Management

Document title: Class Diagram

Version:^a v3.05

Github link: [.../MainDocument/sections/ClassDiagram.tex](#)

Trello card link: [trello.com/c/taKWSdfK/6-class-diagram](#)

Responsible: abru

Status: Finished

Changelog:

??/??/??	?	Created
27/10/18	abru	Added class diagram (v1) & description
01/11/18	abru	Updated class diagram (v2), description, introduction & formatting
17/11/18	abru	Added class diagram legend
		Updated class diagram (v3), description & formatting
18/11/18	renha	Added configuration management
24/11/18	abru	Updated formatting & introduction
25/11/18	leba	Formatting

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

15 Class Diagram

UML Class Diagrams, which show the different object classes and their interactions, are used to make an easy to read visualization of the organization of a system. The diagram includes all the classes in the system as well as their respective attributes, methods and associations to one another. The Class Diagram for the *strEAT* app is presented below in figure 17, the legend for the diagram can be found in table 10 on the next page, and it is followed by descriptions of the classes, and their respective fields, methods, and associations.

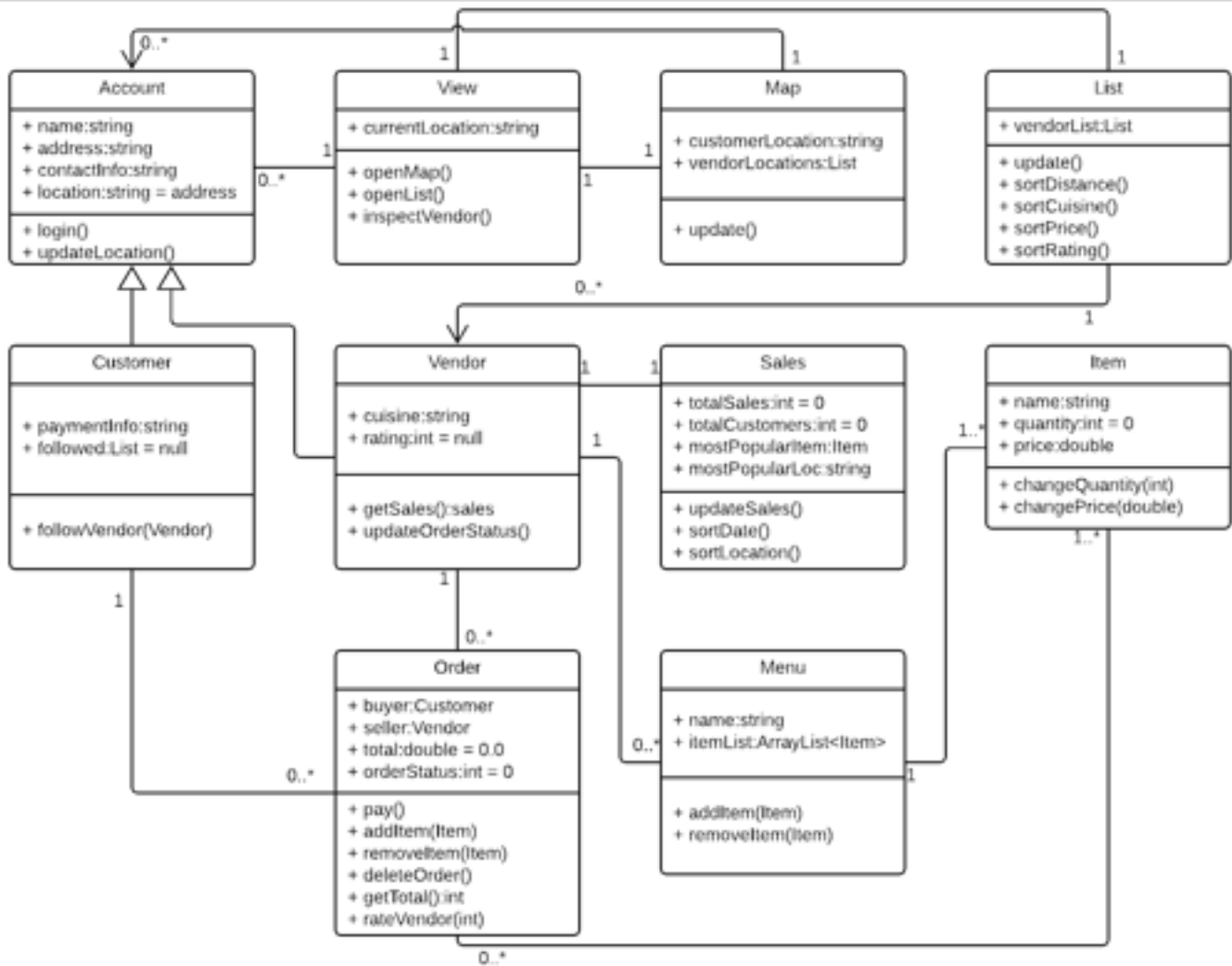


Figure 17: UML Class Diagram for the *strEAT* app






	<p>Class: A rounded box represents a class object in the diagram. In the upper section of the box is the class name, below, in the middle section, are the attributes (fields) of the class, their types and an optional default value. In the bottom section, the operations (methods) of the class are presented, along with their parameters and an optional return type. The + sign is the access modifier, in this case <i>public</i>. Other modifiers are <i>private</i> (-), <i>protected</i> (#) and <i>package private</i> (~). [28]</p>
	<p>Bidirectional association: A straight line represents a bidirectional association between two classes in the diagram, and is the default relationship between two classes in a class diagram. [28] It means that both classes are aware of and interacts with each other. [28]</p>
	<p>Unidirectional association: A straight line with an open arrowhead represent a unidirectional association between two classes. It is similar to the bidirectional association, however in this case only one of the classes is aware of and interacts with the other. The arrow points from the knowing class to the known class. [28]</p>
	<p>Inheritance: A straight line with a closed arrowhead represents inheritance. The arrows point from the subclass to the superclass. The subclass extends the superclass, meaning that it takes on all of attributes and operations of the superclass in addition to its own. [28]</p>
	<p>Multiplicity: The numbers, and the occasional *, written at both ends of the associations are called multiplicities. They represent the amount of instances of each class involved in the given relation. For example, a class with an association that has a 0..* (zero to many) multiplicity value are associated with zero or more instances of the class it associates with. Similarly a 1 denotes that an instance of the class associates with exactly one instance of the other class. [28]</p>

Table 10: Class diagram legend

15.1 Classes, Attributes, Methods, and Interactions

Below are descriptions of the classes, attributes and methods presented in the class diagram, as well as the interactions between classes.

15.1.1 View

The front end view, that contains the *Account*, *Map* and *List* objects.

Attributes:

currentLocation is the current location of the *Account* in use stored as a string.

Methods:

openMap() creates a new *Map* object.

openList() creates a new *List* object.

inspectVendor() selection of a *Vendor* object from a view (map or list) shows the information related to the *Vendor* object.

Interactions:

View has a bidirectional association with the *Account* class, each instance of View is associated with 0 or more instances of *Account*.

View has a bidirectional association with the *Map* class, each instance of View is associated with 1 instance of *Map*.

View has a bidirectional association with the *List* class, each instance of View is associated with 1 instance of *List*.

15.1.2 Map

The Google Maps API which is populated with pins representing the different *Vendor* objects

Attributes:

customerLocation records location of the *Account* using the app as a string and shows it as a pin on the *Map*.

vendorLocations stores the current locations of all *Vendor* objects as a list and shows them as pins on the *Map*.

Methods:

update() updates the *map* object to match the current locations of *Vendor* and *Customer* objects.

Interactions:

Map has a bidirectional association with the *View* class, each instance of Map is associated with 1 instance of *View*.

Map has a unidirectional association with the *Account* class, each instance of Map is associated with 0 or more instances of *Account* or either of it's sub-classes *Customer* and *Vendor*.

15.1.3 List

List view of the *Vendor* objects that match the search criteria of the *Customer*.

Attributes:

vendorList contains all the *Vendor* objects that match the search criteria in a list.

Methods:

update() updates the *vendorList* if the search criteria has been changed by the *Customer*.

sortDistance() sorts the *vendorList* by distance from *Customer* to *Vendor*.

sortCuisine() sorts the *vendorList* by the cuisine fields of the *Vendor* objects.

sortPrice() sorts the *vendorList* by the price fields of the *Vendor* objects.

sortRating() sorts the *vendorList* by the rating fields of the *Vendor* objects.

Interactions:

List has a bidirectional association with the *View* class, each instance of List is associated with 1 instance of *View*.

List has a unidirectional association with the *Vendor* class, each instance of List is associated with 0 or more instances of *Vendor*.

15.1.4 Account

The *Account* class represents a user of the application and stores their information.

Attributes:

name name of person or vendor depending on the type of Account stored as a string.

address delivery address in case of *Customer* account or business address in case of *Vendor* stored as a string.

contactInfo contact phone number, e-mail etc. stored as a string.

location current location of user to whom the account belongs stored as a string; default value is address in case geolocation is unavailable.

Methods:

login() logs the user into the application.

updateLocation() allows for manual or automatic geolocation of the user and updates the *location* field.

Interactions:

Account has a bidirectional association with the *View* class; all instances of Account are associated with 1 instance of *View*.

Account is the parent class of the *Customer* and *Vendor* classes.

15.1.5 Customer

Represents a customer; someone with the intention of finding a vendor.

Attributes:

paymentInfo stores payment information to allow for in-app purchases as a string.

followedList stores the references to the *Vendor* accounts the *Customer* has requested to follow as a List<Vendor>; default value is *null* for a new *Customer*.

Methods:

followVendor(Vendor) adds a reference to the *Vendor* account to the *followedList*; takes a *Vendor* as a parameter.

Interactions:

Customer has a bidirectional association with the *Order* class; each instance of Customer is associated with 0 or more instances of *Order*.

Customer is a sub-class of *Account* and inherits all of its attributes and methods.

15.1.6 Vendor

Represents a vendor; a business looking to sell their goods.

Attributes:

cuisine represents the type of food the *Vendor* sells as a string.

rating stores the current approval rating of the *Vendor* as a int; default is null if no ratings have been received.

Methods:

getSales returns the *Sales* object of the *Vendor*.

updateOrderStatus increments the status *orderStatus* field of the *Order* instance by one.

Interactions:

Vendor has a bidirectional association with the *Order* class; each instance of Vendor is associated with 0 or more instances of *Order*.

Vendor has a bidirectional association with the *Menu* class; each instance of Vendor is associated with 0 or more instances of *Menu*.

Vendor has a bidirectional association with the *Sales* class; each instance of Vendor is associated with 1 instance of *Sales*.

Vendor is a sub-class of *Account* and inherits all of its attributes and methods.

15.1.7 Order

Represents a selection of food items from a particular Vendor.

Attributes:

buyer stores the *Customer* who is making the order.

seller stores the *Vendor* who is preparing the order.

total stores the total price (in kr.) of the order as a double; default value is 0.0 for a new *Order*.

orderStatus indicates the status of the order represented as an int; 0 = not accepted, 1 = accepted, 2 = in progress, 3 = finished; default value is 0 for a new *Order*.

Methods:

pay() accesses the *paymentInfo* of a *Customer* to complete payment for the *Order*.

addItem(Item) adds an *Item* object from a *Menu* to the current *Order*; takes an *Item* as parameter.

removeItem(Item) removes an *Item* object from the current *Order*; takes an *Item* as parameter.

deleteOrder removes all instances of *Item* from the current *Order*.

getTotal() returns the summed priced for all instances of *Item* in the current *Order*.

rateVendor(int) takes an int as parameter and adds it to the rating of the *Vendor* fulfilling the *Order*.

Interactions:

Order has a bidirectional association with the *Customer* class; each instance of Order is associated with 1 instance of *Customer*.

Order has a bidirectional association with the *Vendor* class; each instance of Order is associated with 1 instance of *Vendor*.

Order has a bidirectional association with the *Item* class; each instance of Order is associated with 1 or more instances of *Item*

15.1.8 Menu

The *Menu* of a *Vendor* containing *Item* that *Customer* can add to *Order*.

Attributes:

name stores the name of the *Menu* as a string.

itemList stores all *Item* objects associated with the *Menu*.

Methods:

addItem(Item) adds a new *Item* object to the *itemList*; takes an *Item* as parameter.

removeItem(Item) removes an *Item* object from the *itemList*; takes an *Item* as parameter.

Interactions:

Menu has a bidirectional association with the *Vendor* class; each instance of Menu is associated with 1 instance of *Vendor*.

Menu has a bidirectional association with the *Item* class; each instance of Menu is associated with 1 or more instances of *Item*.

15.1.9 Item

Represents a single entity on a *Menu*.

Attributes:

name stores the name of the *Item* as a string.

quantity represents the remaining pieces of the given *Item* available for sale; default is 0.

price is the price of the *Item* stored as a double.

Methods:

changeQuantity changes the quantity of the given *Item* available for sale; takes a double as parameter.

changePrice changes the price of the given *Item*; takes an int as parameter.

Interactions:

Item has a bidirection association with the *Order* class; each instance of Item is associated with 0 or more instances of *Order*.

Item has a bidirection association with the *Menu* class; each instance of Item is associated with 1 instance of *Menu*.

15.1.10 Sales

Stores information regarding previous sales.

Attributes:

totalSales gross figure of total revenue stored as a int; default is 0.

totalCustomers total number of unique *Customer* objects that have placed an *Order* stored as an int; default is 0.

mostPopularItem stores the *Item* featured in the most instances of *Order*.

mostPopularLoc records the location of *Vendor* featured in the most instances of *Order*.

Methods:

updateSales() completion of an *Order* triggers an update to the *Sales* object.

sortDate() sorts the figures by date.

sortLocation() sorts the figures by location.

Interactions:

Sales has a bidirectional association with the *Vendor* class; each instance of Sales is associated with 1 instance of *Vendor*.

Configuration Management

Document title: Interface Design Using mock-ups

Version: ^a v2.07

Github link: [.../MainDocument/sections/mock-up.tex](#)

Trello card link: [trello.com/c/bkylqqhH/5-mock-up](#)

Responsible: renha

Status: Under review

Changelog:

22/10/18	jopo	Created
25/10/18	renha	Added sub subsections & figure "Guideline..."
26/10/18	renha	Added pen and paper mock-up
		Updated formatting, & text
29/10/18	leba	Updated text
31/10/18	renha	Added "Implementation" section
		Updated formatting & text
01/11/18	renha	Added pictures from the mock-up workshop
02/11/18	renha	Added revised mock-up
		Updated formatting & text
18/11/18	renha	Added configuration management
25/11/18	leba	Consistency checks

Comments:

renha This section needs to be cleaned up a bit

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

16 Interface Design Using Mock-Ups

In this section we will:

- Identify functionality and design our software 'through the interface' together with the user.
- Gain an understanding of the functionality of the system and the necessary information to support development of mock-ups.
- Arrange a mock-up workshop with the user evaluation and usability tests.

To produce some useful mock-ups for the mock-up workshop we decided to iterate the iterate the Mock Ups in three steps:

1. **Pen and paper mock-up:** Quick sketchup on paper of the app functionality
2. **Digital mock-up:** Higher quality mock-ups using image editing software
3. **Native mock-up:** implementatino of the mock-ups on their intended target platforms²

16.1 Senarios

We choose to develop the mock-ups, and showcase our app, from the User and Vendor side respectively. We agrees to produce six images showing the following functionality:

User Side:

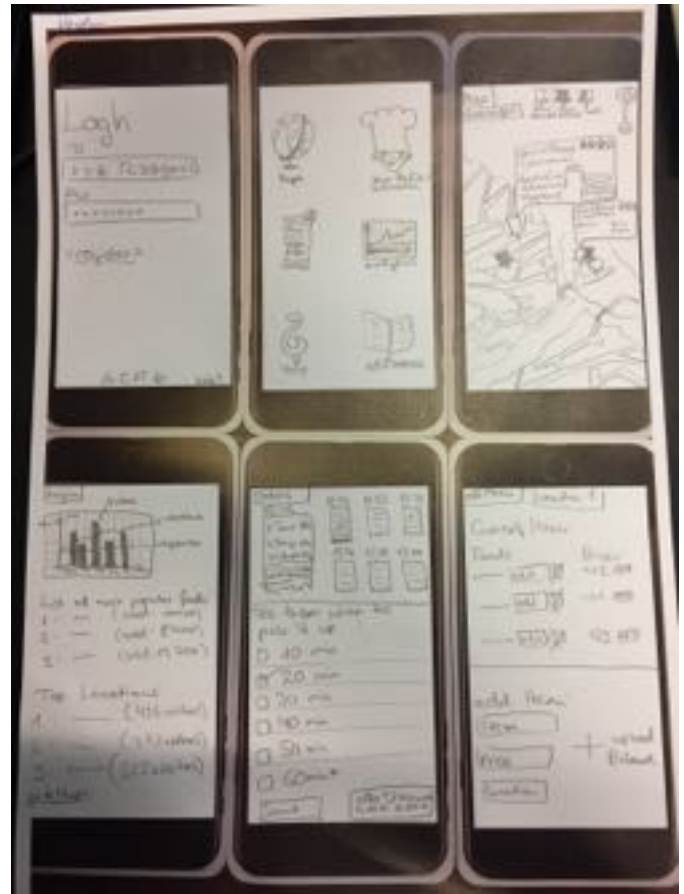
1. **Login screen:** How the login screen will look for the User
2. **Main menu:** How the main menu will look for the User.
3. **Filtering:** How the user can apply filtering on different parameters in searches
4. **Maps:** How the map and location feature will look for the User
5. **Menus:** How the User will interact the Vendor menus:
6. **Skip the line:** How the "Skip the line" feature will look for the User

Vendor Side:

1. **Login screen:** How the login screen will look for the Vendor
2. **Main menu:** How the main menu will look for the User.
3. **Maps:** How the Vendor can use the maps integration.
4. **Edit menu:** Hot the Vendor can edit his information.
5. **Orders:** How the orders from the "Skip the line" feature will look for the Vendor.
6. **Businesstat:** Overview of how the Vendor can use *strEAT* to improve her business.



(a) The pen and paper mock-ups for the Vendor side



(b) The pen and paper mock-ups for the User side

16.2 Pen and Paper Mock-Ups

16.3 The Mock-Up Workshop

In order to receive feedback on our initial Pen and paper mock-ups, we arranged a mock-up workshop. In this section we will discuss the preparation and implementation of the workshop. We interviewed two students for the mock-up workshop. We let them try out our mock-ups and got the Following feedback:

Mikkel SDT:

Mikkel had the following comments about our mock-up:

- Lacking back buttons
- Turn into swipe
- I would like to be tempted by places. So just the finding is the nice function AND spotting the menu, NOT necessarily order it.
- Vendor point of view seems nice
- Skipping the line also might try

Jonathan:

Jonathan also had some comments along with further recommendations:

²In our case Android and iOS.

- Not enough filters (let's add them)
- "I want halal"
- ("I don't want kosher")
- More cryptocurrency options (bitConnect)
- Think already exist
- You want special offers, discount when using an app.
- With the skip line you cannot have recommendations but it is nice that.
- I know what to expect from menu descriptions and photos of food.



Figure 19: Pictures from our mock-up workshop.

16.3.1 Revised Mock-Up.

With the feedback from the mock-up we were able to make a revised set of Mock Ups for the user and vendorside respectively.

16.3.2 Logo and Colours - Look and Feel

We had a session where we tried out different ideas for logos. you can see the results of this brainstorm in fig.21

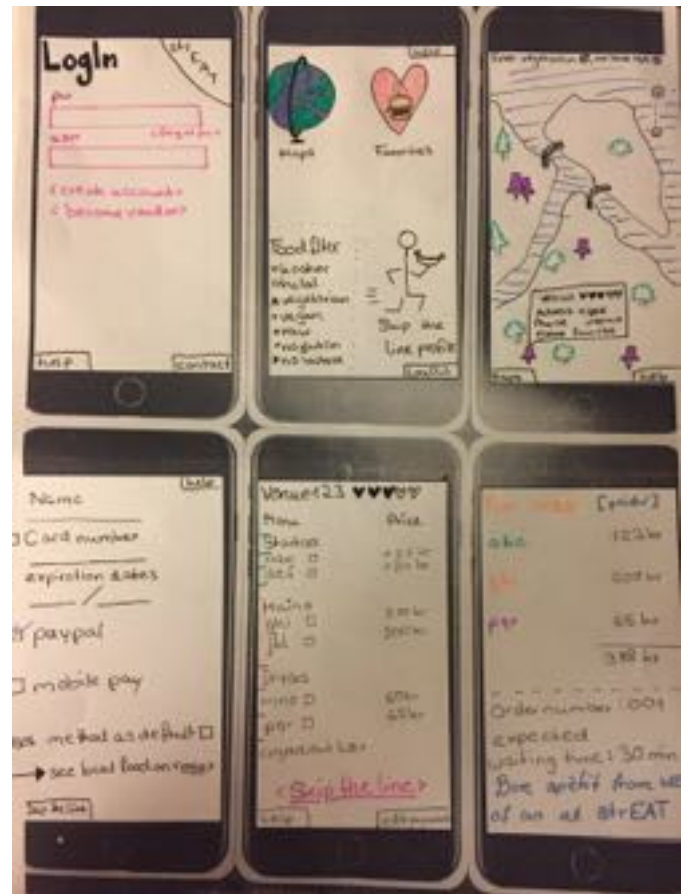


Figure 20: Our revised mock-up



(a) Logo 1



(b) Logo 2



(c) Logo 3



(d) Logo 4

Figure 21: Logo Proposals

Configuration Management

Document title: Dynamic Test Sp

Version:^a v2.02

Github link: [.../MainDocument/sections/DynamicTestSpecification.tex](#)

Trello card link: N/A

Responsible: jopo

Status: Under review

Changelog:

15/11/18	jopo	Created
16/11/18	jopo	Added sub subsections whiteBox
27/11/18	jopo	Added sub subsections BlackBox
17/11/18	jopo	Added tables

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

17 Dynamic Test Specification

Software testing is the process of exercising a program with the specific intent of finding errors prior to the end-user in order to assure quality of the software and conformance to requirements (performance, reliability, security, usability, etc.). Testing also provides valuable insights into the state of the software framework. Finding errors is especially important to the project in the early stages of development cycle, when costs of repair is the lowest. [19].

17.1 White-box and Black-box Testing

In software development, we distinct Static and Dynamic testing. The latter, which is the focus of this section, analyses how the code behaves after compilation and execution, checking parameters such as memory, usage, CPU usage, response time and overall performance of the software analyzed. Here, only the latter is taken into consideration. [2]. There are two types of dynamic testing: white-box testing and black box testing. White-box testing is a s method in which the internal structure of the program being tested is known to the tester. In Black-box testing, on the other hand, implementation of the item being tested is not known to the tester. Its name refers to the unknown: in the eyes of the tester, program is like a black box; inside which one cannot see. This method attempts to find incorrect or missing functions, interface errors, errors in data structures or external database access, behavior or performance errors, initialization errors and termination errors. [1]. Since, we do not have any tangible code, we consider black-box testing as a better approach. The test is conducted by providing inputs (information from users of the app - both customers and vendors) and verifying the outputs against the expected outcome.

17.2 Testing for the project

Testing requires systematically developed set of test cases and that the result of the test for a given input is computed before. Hence, the test process consists of following stages: test planning, testing, design of test cases, preparing test data, running program on test data, comparing results to test cases, reporting testing, and correcting errors. In this section, however, we deal with the design of test cases and comparing exemplary results of what running code could result in. Following test cases are analyzed:

- Vendor input price of the dish
- User searches for vendor
- Payment for the order
- Vendor inputs pick up time for a specific order
- Sign up
- Customizing the order

Each test case includes: a unique identification (T-number in the spreadsheet), execution preconditions (pre-cons), data and actions (inputs), expected results including post-conditions and traces.

ID	Pre-con	Input	Expected results	Trace
T0	Input price within the range to the System	Price: 1000	Error, price out of range	Range 0-200
T1	Input price within the range to the System	Price: 50	The cost of the meal is 50	Range 0-200
T2	Input price within the range to the System	Price: fifty	Error, price input should be an integer	Range 0-200

Table 11: Vendor input price of the dish

ID	Pre-con	Input	Expected results	Trace
T3	Search an existing vendor	Restaurant Momo	Error: no vendor of that name exist	Vendors from website
T4	Search an existing vendor	AsianVegan	The location of AsianVegan on Map is. . .	Vendors from website
T5	Search an existing vendor	32	Error: expects a String	Vendors from website

Table 12: User searches for vendor

ID	Pre-con	Input	Expected results	Trace
T6	Input correct payment details	Credit Card number 329003920	Credit Card does no exist	Valid payment details
T7	Input correct payment details	Pay with PayPal	Error: insufficient funds on you account	Valid payment details
T8	Input correct payment details	Credit Card: Visa : 498539857	Your purchase is ready to pick up in 20 min.	Valid payment details

Table 13: Payment

ID	Pre-con	Input	Expected results	Trace
T9	Input correct pick up time	Customer can pick up in[min]: five	Error: Input integer	Range 15-100 [min]
T10	Input correct pick up time	Customer can pick up in[min]: 5	Error: wrong delivery time	Range 15-100 [min]
T11	Input correct pick up time	Customer can pick up in[min]: 67	Error: must be rounded to 5 min	Range 15-100 [min]
T12	Input correct pick up time	Customer can pick up in[min]: 55	Delivery will be ready in 55 minutes	Range 15-100 [min]

Table 14: Vendor inputs pick up time for a specific order

ID	Pre-con	Input	Expected results	Trace
T13	Input login and password	login: tom	Error: incorrect charaters	At least 5 aplhanumerical char.
T14	Input login and password	Login: TomYates	Error: login already exists	At least 5 aplhanumerical char.
T15	Input login and password	Login: s@Kro	Error: incorret characters	At least 5 aplhanumerical char.
T16	Input login and password	Login: TomYates1994	Correct Login	At least 5 aplhanumerical char.

Table 15: Sign up

ID	Pre-con	Input	Expected results	Trace
T17	Customize your order	Customize your order: add sauce	Sauce added	Add sauce
T18	Customize your order	Customize your order: add pepper	Error: not available	Add sauce
T19	Customize your order	s@sd	Error: not available	Add sauce

Table 16: Customzing order

Configuration Management

Document title: Quality Plan

Version:^a v2.05

Github link: [.../MainDocument/sections/QualityPlan.tex](#)

Trello card link: [trello.com/c/v5H5pSzf/34-quality-plan](#)

Responsible: idbo & kati

Status: Under review

Changelog:

15/11/18	idbo & kati	Created
15/11/18	idbo	Updated text & formatting
17/11/18	abru	Added tables
		Updated formatting
18/11/18	renha	Added configuration management
18/11/18	leba	Updated text
20/11/18	kati	Updated formatting & text
25/11/18	leba	Removed introduction heading
29/11/18	abru	Updated text and formatting in tables

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

18 Quality Plan

The purpose of this quality plan is to ensure the quality of our app, and to assure that the qualities and standards prioritized in earlier stages of this project will be met.

Quality assurance is the definition of processes and standards that should lead to high-quality products and the introduction of quality processes into the manufacturing process. [41, p.709]

This quality plan will introduce the quality assurance strategy and describe the quality assurance techniques for our project.

18.1 Quality Assurance Strategy

As we are working with an agile approach, our quality assurance strategy reflects this. When working with an agile approach, the development practices and quality assurances practices are often intertwined. Therefore, the developers are also often responsible for the quality assurance. Additionally, quality assurance techniques will be applied along the way. [20] Since we are using the Scrum framework, different quality assurance techniques will be applied along the way in each Scrum Sprint Cycle. As each Sprint is planned, quality assurance techniques that match tasks from the Product Backlog will be selected. Here, techniques such as system metaphors and feedback from the Product Owner/on-site-customer, will likely be applied. As the code development is initialized so will quality assurance techniques such as Refactoring, Class-Responsibility-Collaboration Cards and Peer Programming, see [18]. As the Sprint comes to an end, it is necessary to ensure that the unit testing has been passed. In the next section the various quality assurance techniques will be explained in greater detail.

18.2 Quality Assurance Techniques

Acceptance Testing	In acceptance testing, the user gets to test the program. Only when the acceptance test is passed does the program live up to the agreed standards [41, p.249].
Continuous Integration	Continuous Integration refers to the concept of integrating the system and the production code continuously, and thereby be confronted with compatibility problems early on [20, p.3]
Class-Responsibility-Collaboration Cards	CRC is a game/activity for object-oriented languages. The game is a way for the team to discuss different class designs, responsibilities in the classes and the collaboration between the classes [4, p. 463].
On-Site Customer	On-Site Customer refers to feedback from the customers from an early stage in the development process to the project ends. Customer feedback is a characteristic for agile methods, and is therefore also an ongoing and important Quality Assurance technique [20, p.3]
Pair Programming	Pair Programming refers to two programmers working on the same code. This way of working can improve the design and decrease defects in the software [20, p.4]
Refactoring	Refactoring is restructure of existing code. Refactoring is about changing the internal structure without changing the external behaviour of the programme [20, p.4]

Table 17 continued from previous page

System Metaphor	System metaphor refers to a story that shortly describes how the system works. Often it includes the most important classes and patterns in the system. The story/metaphor is used as a tool to explain and discuss the software in a simpler way with the users. The system metaphor is also used to develop the systems architecture, again to increase communication between users and developers [20, p.4]
Unit Testing	When performing unit tests, each unit/component is tested to ensure that it works correctly and meets its specification [41, p.232]

Table 17: Overview of the Quality Assurance techniques

18.3 Quality Goals

To ensure the overall quality of our product, it is necessary to set out the quality goals. In our case, these goals are based on our product's **software qualities**. In the early stages of the project, we decided on five product qualities that should be present in the app. They are presented in table 18 below.

Qualities	Quality Assurance Techniques	Testing the Software Qualities
Usability	Using System Metaphors and Acceptance Testing we will ensure the usability of the app	Test the app on users. Both when the <i>strEAT</i> app is developed and also after it is launched, in order to improve the usability of the app. Here, it should be noted whether the problem with the usability is situated in the UI or in the data structure.
Performance Efficiency	Using Refactoring and Peer Programming we will ensure the performance efficiency of the app	Test the app while a lot of users are using the app at the same time. If the software crashes, then there is need for refactoring.
Functional Suitability	Using Continuous Integration we will ensure the functional suitability of the app	Test if the GPS signal is accurate at different positions around the city. It should be tested how the app works, when a lot of users is using the GPS function around the same three cell site towers.
Reliability	Using Refactoring we will ensure the reliability of the app	The Reliability Quality can be tested like the Software Qualities: Performance Efficiency and Usability. If users locate a problem with the product, the contact information should be readily available, and it should be easy to contact the <i>strEAT</i> team in order to report the issue.
Portability	Using CRC cards we will ensure the portability of the app	Within initial development, it should be a part of the design to ensure that the system and app can be adapted to different countries.

Table 18: Quality goals for our product

Sommerville argues that it is not possible to optimize all attributes in one system, because when some attributes is prioritized higher than others, there is naturally a sacrifice. It is important that the project team has decided on the most important attributes because then it will be easier to achieve

the qualities when developing the software [41, p. 704]. Comparing software development to regular manufacturing, software development is more complex in terms of the link in between process quality and product quality. When machines are set up to create a certain product with specific qualities, the output of the machine is the same over and over again, and therefore the quality process is easy to standalize afterwards. That is not the case when it comes to software qualities because the design of the software is not only dependent on the developers' skills and experience, but also on external factors like early software release dates. [41, p.705].

(..) the development process used has a significant influence on the quality of the software, and good processes are more likely to lead to good quality software. Process quality management and improvement can result in fewer defects in the software being developed [41, p.705]

Therefore, this quality plan is an important step for the *strEAT* project team in order to ensure high quality software with minimal defects.

Configuration Management

Document title: Software Architecture

Version:^a v3.02

Github link: [.../MainDocument/sections/SoftwareArchitecture.tex](#)

Trello card link: [trello.com/c/59sNSnZY/35-explore-architectural-solutions](#)

Responsible: kati & leba

Status: Not finished

Changelog:

26/11/18	leba	Created
----------	------	---------

26/11/18	kati	Modified
----------	------	----------

27/11/18	kati	Modified
----------	------	----------

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

19 Software Architecture

In this section we describe two common mobile application architectural patterns (the Model-View-Controller (MVC) and Model-View-Presenter (MVP)), select the most appropriate, and explain how it can be applied to our software. We then evaluate the pattern against some of the software qualities selected and outlined earlier in this report in order to make sense of why it would be appropriate to apply such pattern for our mobile application.

19.1 Architectural Patterns

Software architecture and architectural design revolves around the concept of the organizing and designing the structure of software systems. Usually, in the agile processes, this would take place in earlier stages of the overall process [41][p. 168]. An architectural pattern works as a recipe for a successful way to organize and design a system, and is usually quite similar to design patterns [41][p. 176].

19.1.1 MVC & MVP

In the world of mobile applications, many of the patterns in use are some flavour of the Model-View-Controller/Presenter pattern, whereby the UI logic is separated from the rest of the application [45]. This pattern promotes decoupling between components and results in highly modular and testable code. So long as the interfaces between components are maintained, their internals (i.e. logic, data structures, etc.) may be changed without fear of breaking the stack entirely [35].

The pattern is divided into to the tree components; Model, View and Contoller/Presenter. In the following we will briefly describe the tree componenets. **The Model** stores the system state. For example, in the case of a service application, this may refer to the storage of user data; or, perhaps, in a gaming application, this instead comprises of the game's current map configuration and player state. **The View** is the interface between the application user and the underlying business logic. It receives input from the user, passes this information to the Controller/Presenter, and is eventually told what to present to the user. **The Controller/Presenter** essentially acts as the logical functionality component. It takes in information from View, processes its request as per the business logic, and returns the information View should present to the user and how. Processing the request from View may involve modifying the system state maintained by Model and retrieving information that will eventually be passed to View [35].

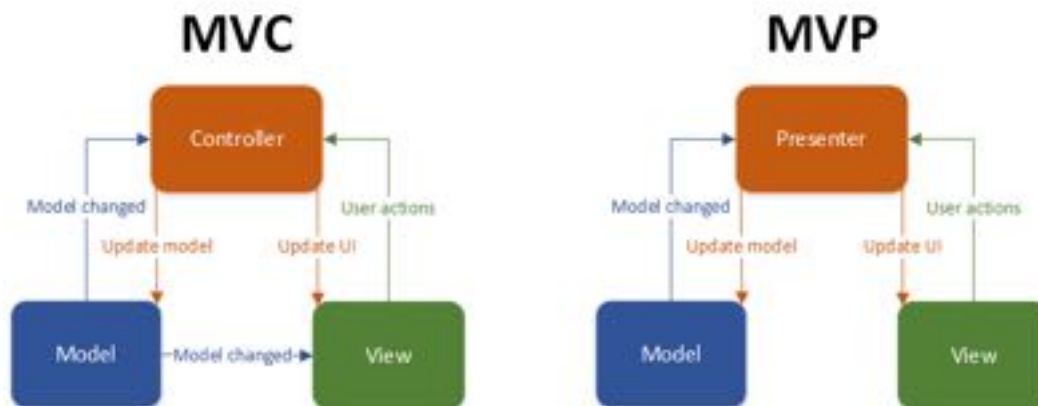


Figure 22: The MVC and MVP patterns [45]

19.1.2 MVC vs. MVP

Although the general principles remain the same, there are slight differences between the Model-View-Controller (MVC) and Model-View-Presenter (MVP) patterns. In the MVC pattern, View is aware of Model's existence and may be notified of a change to the Model directly. This necessitates more logic be placed in the View component to handle notifications from Model, which may result in a "bloated" View. The MVP pattern aims to address this by instead moving all logic to the Presenter. View now knows nothing of the Model and is solely responsible for rendering the user interface. This abstraction is in line with what many software engineers consider good development practice [41].

19.2 *strEAT* & MVP

Taking into consideration the arguments presented in the previous section, we believe the MVP pattern is a suitable choice. The View component will be responsible for presenting the customer with the graphics of the application (whether that be a map view, list view, menu, etc.). The Presenter component will handle the business logic. This includes retrieving vendor menus, compiling Order objects and passing them to Vendor's, handling vendor and customer information change requests, etc. The Model component will essentially comprise of a database containing user information. Including, but not limited to, vendor and customer location, confirmed orders, menus, etc. The flow of information will be as such: a customer interacts with the interface constructed by the View, this request is passed to the Presented which analyses the request and decides on a course of action, the Model provides (and/or alters) the requested information to the Presenter, and the Presenter then passes on a command to View which renders a response.

19.3 Evaluation

In regards to software qualities such as Reliability, Usability and Functional Suitability, the MVP pattern seems to satisfy both. In relation to Usability and Functional Suitability, the MVP pattern allows us to easily change the view/interface of the app, without being concerned about restructuring the Presenter component. This will ensure that we can easily work around modifications of buttons, maps and lists etc. Making changes to the way the data is presented in the view, therefore does not affect the other two components. In relation to Reliability, the MVP pattern will make it easy for us as a team to collaborate on modifying the various components, since each component has its own responsibility. Therefore, it will most likely also become easier to detect errors and modify these. Likewise it can be argued that the MVP pattern works especially well with the scrum life cycle, since the strong emphasis on decoupling between components, makes it easy to adjust both small and bigger changes along the way. But overall, MVP is a tried and tested mobile application pattern popular with both iOS and Android developers.

Configuration Management

Document title: Reflection

Version:  v1.00

Github link: [.../MainDocument/sections/Reflection.tex](#)

Trello card link: [trello.com/c/EheTxxro/36-reflection-and-introduction](#)

Responsible: Emmi

Status: Not finished

Changelog:

27/11/18	kati	Created
28/11/18	leba	General QA

Comments:

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

20 Reflection

The project has given us the opportunity to learn and grow in fields that we were inexperienced in beforehand. Some of us that come from a technical background have found themselves on new territory when having to rely on group work and communication, others have just started programming and were challenged by tools like LaTeX and GitHub. It is safe to say that every member of the group contributed but at the same time learned something. In the following, we will be reflecting on the elements of the project that we tackled throughout the process and how the outcome of that matches or diverges from our initial expectations.

20.1 Outlining the Project

Going with an idea outside of the schemes provided by the course was the first impactful decision we made as a group. The goal was to create something we believed would be of actual value to the market and at the same time resembles an application that we could see ourselves using in our daily lives. That way, our motivation for building a good product was high from the start and the choice turned out to be the right one. Equally quickly made was the decision on which framework to use, as Scrum seemed like the logical choice all along, given that we were dealing with a mobile app, deadlines, and a group in which people did not know each other yet. Thus, the regular meetings, Sprints, and the changing role of the Scrum Master turned out to indeed suit us well and ensure a steady pace within the group. Additional help was offered by our supervisor Silviu who managed to keep a balance between pushing us to do our best and reassuring us that doing so will be worth the effort.

20.2 Development Process

Once we had arranged the framework and distributed responsibilities, the next logical step was to explore the software qualities. Beforehand, we divided the responsibilities into two sub groups: exploring software qualities; and choosing a process model.

1. Exploring the Software Qualities:

Out of the qualities presented in the slides from the respective lecture, we chose the five that we found most important for our case: Portability; reliability; performance efficiency; functional suitability; and usability. We still find these choices to have been adequate for our product and working with them gave us the ability to keep a clear picture of the product in mind at all times by following the definitions of the five qualities at every step. That is a fundamental part of the process because we would have definitely had much going back and forth to do had we chosen less suitable software qualities. Additionally, the qualities paved the road to our mock-up and the interview questions for our research part.

2. Choosing a Software Process Model:

For aforementioned reasons, we went with agile development, which has turned out to be the right choice for us. Had we had more experience, we might have instead chosen the STEPS model, as it would have allowed us more cooperation with potential users and thus a more flexible, cooperative structure.

Based on these decisions we were then able to proceed on to testing our ideas. We developed two different rich pictures as a solid basis for our mock-up and parallelly conducted interviews with students who we felt would be our most likely initial target group. The process of making the rich pictures was slightly confusing for us, as it was entirely new and could not quite grasp which perspective we should draw them from and if they were allowed to include text. After the next supervision that was clarified

and we created two new rich pictures and, following up, the app mock-up.

The interview part was more time-consuming but went more smoothly, and it gave us great insight in the users' priorities. We learned, for example, that our formerly highly valued delivery function seemed quite unnecessary to our interviewees, which was also confirmed by the test persons in the mock-up workshop later on. We therefore changed it into a skip-the-line function that does not mimic what delivery apps do but rather allows a more spontaneous interaction with the vendors by paying the desired food forward while already being in close proximity to the venue, and then saving time by skipping the potential queue once the food is ready. When explained to the test persons, this idea seemed more logical because it would not take away the feeling of spontaneity and flexibility they associate with food trucks and markets. Otherwise, people seemed to be widely approving of the app, even though some of them at first sight did not deem it necessary. Only after further explanation did everyone agree with the concept. Thus, in hindsight, we should have simplified our description of the application and the mock-up, potentially giving an example right away instead of expecting other students to grasp the gist of a new idea right from the start. This step should have been considered all along, as we had actually mentioned it already in the risk section.

Knowing what the app should look like, we could now make a class diagram and choose a software architecture model, along with documenting the project in the background. As confirmed by our supervisor, our class diagram was very thorough and we could have possibly made it simpler, chose to include everything we could think of though.

For the software architecture model, the MVP model seems to suit the information flow best, even though it would, of course, take more time and testing to prove that entirely.

Documenting the process was our last step before our last supervision and went without problems, given the clear instructions from the slides on learnIT.

20.3 Conclusion

The project included many new techniques, models and tasks for each member of our group and it was at times difficult to keep everyone in sync with each other and on track with the soft deadlines (e.g. handing in something for supervision). Slack and Trello helped us immensely to get everything done on time and, if necessary, solve slight confusions and time issues. Given the intense workload of the other courses in the semester, it was not always easy to find a time to meet or work on our tasks, but we made it work. The team spirit was good, we were open about issues and tried to resolve them quickly whenever they arose. Having a good basis for communication was also an advantage for dividing tasks - everyone seemed to try and contribute as much as they could in areas that appealed to them most. That, in our opinion, is the biggest reason for us keeping a positive spirit and finishing our tasks to our own satisfaction. No one was ever shamed for understanding something less than someone else, and we never excluded anyone from the group, while definitely not being too loose about deadlines at the same time. We mostly followed a policy of "whoever shows up gets to choose first" for soft deadlines and dividing the tasks, and it seems that this suited all of us and also motivated us to attend as many meetings as each team member could. It felt comfortable and purposeful to work on this project and we are grateful for the way it went.

Configuration Management

Document title: Appendix

Version:^a v3.04

Github link: [.../MainDocument/sections/Interviews.tex](#)

Trello card link: [trello.com/c/O4hq8etf/7-explore-the-use-context](#)

Responsible: idbo & kati

Status: Finished

Changelog:

02/10/18	idbo	Created
02/10/18	kati	Added Interview Guide
02/10/18	idbo	Added Interview - Bar manager & Interview - Student 1
05/10/18	kati	Added Interview - Student 2
18/11/18	renha	Added configuration management

Comments:

abru Legacy section, imported from Google Docs to GitHub

^aVersion number v.X.YY where X=1 "not finished", X=2 "under review" X=3 "finished". YY is incremented each time a change is made, and a short comment about the change is added to the changelog

A Appendix

A.1 Interview Guide

We choose to do semi-structured interviews. This means that we wrote a few basic questions for two different kind of interviews, namely potential customer interviews and potential vendor interviews. Since this kind of interview is not completely structured, as its name indicates, we often came with follow-up questions during the interviews. Below is the list of questions we wrote before conducting the interviews.

A.1.1 Customers

1. Do you use any apps to buy/find food in Copenhagen?
2. If so, what are they called?
3. Why do you use them?
4. What should a food app contain / be able to do / help you with?
5. What do you think about an app that could show you where to find street food in Copenhagen?
6. Would you ever consider using such an app?
7. Why? Why not?
8. What should/could this app be able to do if you should use it?

A.1.2 Street food Vendors

1. Do you always have your stand in the same location?
2. Do you use social media / apps to advertise your business?
3. If not, would you be interested / or what do you think of that idea?
4. What if there existed an app to show the location of street food vendors in Copenhagen, where customers could also order food. Do you think you would use that?
5. What should this app be able to do if you should use it?
6. Do you have any additional ideas of how such an app could benefit your business or your work?

A.2 Bar manager, Tipi Bar Nørrebro:

The interview person is informed about the project. - The recording starts after the first question is asked.

Bar manager:

When we started the bar or the owner started the bar, it was supposed to be a bar in the middle of a lot of street food and it worked for a while, but because [the food trucks] opened and closed whenever they wanted to... They didn't have opening hours like stores have, it made it difficult when guests came here, because they maybe came here for a specific kind of food truck and that was maybe closed, and there was maybe two [food trucks] opened out of the ten we had here, so people came here and were more disappointed than happy,

so then we slowly built the 'TIPI BAR' and started getting more guests here and then the food trucks moved out and now we have the food truck that is actually open which is 'Chop Stick' [laugh] and they're super good. So the others moved out.. to other places.. and it was hard having them around because you never knew when they were going to be opened or closed, and it was supposed to be a collaboration between the bar and them, but then it ended up being easier for us to build a big bar where people are allowed to bring food from wherever - they can bring home cooked food, have a romantic dinner here where they buy wine in the bar, or people have their birthday party where they can bring their own cakes and everything. So we did that instead of having food trucks around, but now you can bring food from wherever you want.

Interviewer:

And that works fine?

Bar manager:

Yeah, it works fine! Some people get disappointed because they remember the food truck area or they have seen commercials for it and then we kind of have to explain the whole thing again - that it worked during the summer months, but the rest of the year.. and we have a bar and the rent, so unfortunately they had to move, and I don't know where they moved to.

Interviewer:

We were just talking about it, that maybe they are all in 'Reffen', maybe they are trying to collect it together.

Bar manager:

I wish they would and that they would actually collaborate and have this 'Let's have these opening hours and get people here', because I like the idea that this place had in the beginning, that it was a food market and people would come here for lunch and they could get so many different kinds of food. So I hope that works, because it would actually be nice to go to a place with a food market, where you could be like 'what do I feel like eating today?'.

Interviewer:

We were actually thinking our idea was to have this mobile app where you could... For us we would get in contact with the street food vendors and then trying to map them all in Copenhagen, so people would be able to go in and look for what kind of food or what kind of.. price range and find them on a map or list and see where they could find them in Copenhagen and their opening hours, if they have some off course [laugh] and to see the menu or others recommendations about the place. Do you think that would be helpful?

Bar manager:

That's a very good idea! I don't know with the Asian one here, whether it's going to stay, but the other one next to it, the one with falafel, it's not going to stay here because there's too much competition here in Nørrebro when it comes to falafel. He wants his own place where people can actually sit; he wants an inside area.

Interviewer:

That's also a problem if they become too popular, I guess.

Bar manager:

Exactly! So he's doing that, and then we'll only have 'Chop Stick' here.

Interviewer:

I don't think we have any more questions, but thank you for giving us an overview of what happened with the food trucks and for your time.

A.3 Student 1:

The interview person is informed about the project. - The recording starts after the first question is asked.

Interviewer:

Do you use any apps to buy/find food in Copenhagen?

Student 1:

I use 'JustEat' and 'Wolt' and 'ToGoodToGo'.

Interviewer:

Why do you use them?

Student 1:

'JustEat' because of pizza, and 'Wolt' because of butterchicken and 'ToGoodToGo' because I like the idea of minimizing food waste.

Interviewer:

What should a food app contain / be able to do / help you with?

Student 1:

First of all provide me with a menu card - what's on the menu - what can I get, the pricings, the opportunity to remove things from a dish if you have allergies or if you don't like spicy stuff and payment methods.

Interviewer:

What do you think about an app that could show you where to find street food in Copenhagen?

Student 1:

Sure! Why not!? Yeah I like street food, I think it's funny, I think it's different!

Interviewer:

Have you ever had trouble finding a street food truck or?

Student 1:

No I think in Copenhagen, at least.. when there was ‘Papirøen’, it was in a hub and the same thing with Westmarket, everything is kinda in one place, so it’s not necessarily difficult to find, when it’s all gathered in places like that but I think if you start spreading it out and place trucks around the city, off course it will be difficult to locate them.

Interviewer:

So if there was such an app, you would rather use it for ordering food or?

Student 1:

Yeah or if I’m nearby then I would just go over there.

Interviewer:

What should/could this app be able to do if you should use it?

Student 1:

Well an overview of choices, a list, search engines, and a map, and everything I just described previously, would be an good idea, or maybe not actually.. I think kind off the charm with street food is that you go there and eat there, not that you order it and go home and eat, so I think actually maybe just create some sort of overview maybe a point system, and sometimes street food can be quite expensive.. maybe some sort of clip card system, where you get the tenth dish for free, I think that would be cool.

Interviewer:

I don’t have any more questions! Thank you for your time!

A.4 Student 2:

The interview person is informed about the project. - The recording starts after the first question is asked.

Interviewer:

Do you use any apps to buy or to find food in Copenhagen?

Student 2:

Yes.

Interviewer:

What are they called?

Student 2:

Just Eat.

Interviewer:

Ok. And why do you use that? What do you like about that?

Student 2:

Because it is the one I know. It is international known. So when I came to Denmark, since I am an international. And since I came to Denmark I didn't know any special ordering types. So I just looked up Just Eat, and that's why. And yeah.

Interviewer:

What do you think then if we should create a food app, is there any features that would be nice to have, which is not necessarily on JustEat?

Student 2:

That it is available for every area. If there is different restaurants, the restaurants should specify which area or district in Copenhagen they can go to, before you go to pay. Because often it happens when you pay and you put in the address, and then they give you feedback that "Oh we don't deliver to these places". So something about that. And also think about the environment. Like if you do it on bike, or if you use a car. Like think about the environment, and use plastic bags when you deliver the food. Like Just Eat use plastic bags which is not like good.

Interviewer:

Then what do you think about an app that could show you other places, like street food vendors in Copenhagen? If you are in the city and looking for something, at it could show you on a map where you could find a specific street food vendor and the menu. Would you use that?

Student 2:

Yes. But I would also like to see the rating for that place.

Interviewer:

So you know if... ?

Student 2:

If I'm not gonna be sick. Like so you know that it is hygienic. When it is street food a lot of places doesn't have water connected to the places, if it's just in a car. Yeah so just ratings, so you could see the standard of that place. And also then it wouldn't be a fail, if you go to a street food place that is on the app, and you come there and it is not what you expected. But if you have ratings, you can evaluate them.

Interviewer:

Ok. Perfect. Thank you.

List of Figures

1	Screenshot of our GitHub repository as of October 19 th 2018	11
2	Screenshot of our Trello board as of November 2 nd 2018	12
3	Screenshot of our Prezi presentation for our Rich Pictures and Use Context sections	12
4	A screenshot of our Slack workspace as of October 19 th 2018	13
5	A collage showing our trip to Reffen for the kick-off event.	19
6	Søren in his hydroponics container at "Reffen GREENS"	20
7	Oscar working at the Baobab restaurant.	21
8	NEEDS CAPTION	25
9	NEEDS CAPTION	26
10	This diagram show different aspects important to the consideration of software qualities	46
11	Picture of cell site triangulation. The figure is taken from [27]	56
12	Use Case Diagram for Scenario 1	62
13	Use Case Diagram for Scenario 2	63
14	Use Case Diagram for Scenario 3	64
15	The world with the app	67
16	The world without the app	68
17	UML Class Diagram for the <i>strEAT</i> app	76
19	Pictures from our mock-up workshop.	86
20	Our revised mock-up	87
21	Logo Proposals	88
22	The MVC and MVP patterns [45]	97

List of Tables

1	Pros and Cons of the Software Process Models	27
2	Our roles for each week of the project	36
3	Overview of our estimates in story points for the Product Backlog, as a result of our planning poker game	40
4	Our project schedule	42
5	Activity schedule for Sprint 2	43
6	Product oriented risks for our project	49
7	Project oriented risks for our project	50
8	Overview of our brainstorm	53
9	Use case diagram legend	65
10	Class diagram legend	77
11	Vendor input price of the dish	91
12	User searches for vendor	91
13	Payment	91
14	Vendor inputs pick up time for a specific order	91
15	Sign up	91
16	Customizing order	91
17	Overview of the Quality Assurance techniques	94
18	Quality goals for our product	94

References

- [1] Black box testing. <http://softwaretestingfundamentals.com/black-box-testing/>, 2018. Accessed 15 November. 2018.
- [2] Guru 99. Dynamic testing tutorial: Types, techniques and process. <https://www.guru99.com/dynamic-testing.html>, 2018. Accessed 10 November. 2018.
- [3] Street Food App. Streetfood app. <https://streetfoodapp.com/jacksonville>. [accessed 14th of September].
- [4] Kölling M. Barnes D. J. *Objects First with Java - A Practical Introduction Using BlueJ*. Pearson, 2016, 6th edition.
- [5] Kent et. al. Beck. Manifesto for agile software development. <http://agilemanifesto.org/>. [Accessed 11th of September].
- [6] Herbert D. Benington. Production of large computer programs. *IEEE Ann. Hist. Comput.*, 5(4):350–361, 1983.
- [7] Ian Blair. What are push notifications. <https://buildfire.com/what-are-push-notifications/>. [accessed 1st of November].
- [8] Ian Blair. What is a push notification. and why it matters. <https://buildfire.com/what-is-a-push-notification/>. [accessed 1st of November].
- [9] Barry W. Boehm and W. J. Hansen. *Spiral Development: Experience, Principles, and Refinements*. Carnegie-Mellon University Press, 2000.
- [10] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, Upper Saddle River, NJ, USA, 2005.
- [11] European Commission. Website: www.ec.europa.eu. https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/principles-gdpr/what-information-must-be-given-individuals-whose-data-collected_en. [accessed 15th of October].
- [12] The Economist. America's food-truck industry is growing rapidly despite roadblocks. <https://www.economist.com/graphic-detail/2017/05/04/americas-food-truck-industry-is-growing-rapidly-despite-roadblocks>, 2017. Accessed 11 Sep. 2018.
- [13] P. Fellows and M. Hilmi. Selling street and snack foods. rome: Food and agriculture organization of the united nations., 2011. <http://www.fao.org/docrep/015/i2474e/i2474e00.pdf> Accessed 11th of September].
- [14] Wang X. Flora H.K, Chande S.V. Adopting an agile approach for the development of mobile applications. *International Journal of Computer Applications*, 94(17):44–48, May 2014.
- [15] Christiane Floyd, Fanny-Michaela Reisin, and Gerhard Schmidt. Steps to software development with users. In C. Ghezzi and J. A. McDermid, editors, *ESEC '89*, pages 48–64, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.
- [16] British Street Food. British streetfood. <http://britishstreetfood.co.uk/hungry/>. [accessed 14th of September].

- [17] K. Fosberg and H. Mooz. The relationship of system engineering to the project cycle. *INCOSE International Symposium*, 1(1):55–65, 1991.
- [18] T. Galotro. What data can tell us about the state of the food truck industry. <https://foodtruckr.com/2017/09/what-data-can-tell-us-about-the-state-of-the-food-truck-industry>, 2018. Accessed 11 Sep. 2018.
- [19] Amir Ghahrai. Why do we test? what is the purpose of software testing? <https://www.testingexcellence.com/why-do-we-test-what-is-the-purpose-of-software-testing>, 2017. Accessed 13 November. 2018.
- [20] Zhu L. Hou M, Verner J. and Babar M. A. Software quality and agile methods. *IEEE Computer Society*, 2004. [Visited the 14th of November 2018].
- [21] Brenna Houck. Why some restaurants are cutting ties with mobile ordering apps. <https://www.eater.com/2017/8/29/16214442/restaurant-order-pay-apps-seamless-postmates-uber-eats>. [accessed 14th of September].
- [22] D. Hughey. Comparing traditional systems analysis and design with agile methodologies. <http://www.ums1.edu/~hugheyd/is6840/waterfall.html>, 2009. Accessed 11 Sep. 2018.
- [23] Tinker I. *The urban street food trade: Regional variations of women's involvement. Women, the family, and policy: A global perspective (pp. 163-187)*. State University of New York Press Albany, 1994.
- [24] Julie Jargon. Mobile-ordering apps create problems for restaurants. <https://www.wsj.com/articles/mobile-ordering-apps-create-problems-for-restaurants-1508119500>. [Accessed 14th of September].
- [25] Chen Jin. Website: www.mackinstitute.wharton.upenn.edu. <https://mackinstitute.wharton.upenn.edu/2017/how-rating-systems-impact-providers/>. [accessed 26th of October].
- [26] JustEat. Justeat. <http://justeat.dk>. [accessed 14th of September].
- [27] Phil Locke. Cell tower triangulation – how it works. <https://wrongfulconvictionsblog.org/2012/06/01/cell-tower-triangulation-how-it-works/>, 2012. [accessed 17th of September].
- [28] Lucidchart. Uml class diagram tutorial. <https://www.lucidchart.com/pages/uml-class-diagram>. [accessed 17th of November].
- [29] Lucidchart. Uml use case diagram tutorial. <https://www.lucidchart.com/pages/uml-use-case-diagram>. [accessed 17th of November].
- [30] S. McConnell. *Rapid development: taming wild software schedules*. Pearson Education, 1996.
- [31] S. McConnell. *Complete Code*. Pearson, 2004.
- [32] R. Panko. Mobile app usage statistics 2018 - the manifest. <https://themanifest.com/app-development/mobile-app-usage-statistics-2018>, 2018. Accessed 11 Sep. 2018.
- [33] David L. Parnas and Paul C. Clements. A rational design process: How and why to fake it. *IEEE transactions on software engineering*, 2:251–257, 1986.

- [34] The Linux Information Project. Cross-platform definition. <http://www.linfo.org/cross-platform.html>, [accessed 1st of November].
- [35] Martin R.C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.
- [36] Lizzie Rivera. Why 2018 is set to be the year street food gets a lot more sophisticated. <https://www.independent.co.uk/life-style/food-and-drink/street-food-2018-sophistication-why-kerb-club-mexicana-bao-pop-brixton-street-feast-a818.html>, 2018. Accessed 11 September 2018.
- [37] W. W. Royce. *Managing the development of large software systems: concepts and techniques*. IEEE Computer Society Press., 1987. Proceedings of the 9th international conference on Software Engineering.
- [38] Butler S. and Richardson A. Emerging markets: street food booms as investors get the taste. *The Guardian*, 2018. [Visited the 9th of September 2018].
- [39] K. Schwaber and J. Sutherland. The scrum guide. <https://www.scrumguides.org/>, 2017. Accessed 11 Sep. 2018.
- [40] Mountain Goat Software. Mountain goat software. <https://www.mountaingoatsoftware.com/agile>, 2018. Accessed 13 September 2018.
- [41] Ian Sommerville. *Software Engineering*. Pearson, 2016.
- [42] Simon Stenning. The uk foodservice landscape. <https://www.bordbia.ie/industry/events/SpeakerPresentations/2016/IrishFoodserviceSeminar2016/The%20UK%20Foodservice%20Landscape%20-%20Simon%20Stenning,%20MCA.pdf>, 2016. Accessed 11 September 2018.
- [43] Lucidchart Team. The pros and cons of waterfall methodology. <https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology>, 2017. Accessed 11 Sep. 2018.
- [44] Promantics Technologies. Website: www.medium.com. <https://medium.com/@promantics/geolocation-in-mobile-apps-how-brands-are-using-it-for-marketing-15adf3e5e07c>, [accessed 15th of October].
- [45] Vasily. Mvp and mvc architectures in android. <https://www.techyourchance.com/mvp-mvc-android-1/>, [Accessed 26th of November].
- [46] Conrad Weisert. Waterfall methodology: There's no such thing! <http://www.idinews.com/waterfall.html>, [Accessed 11th of September].