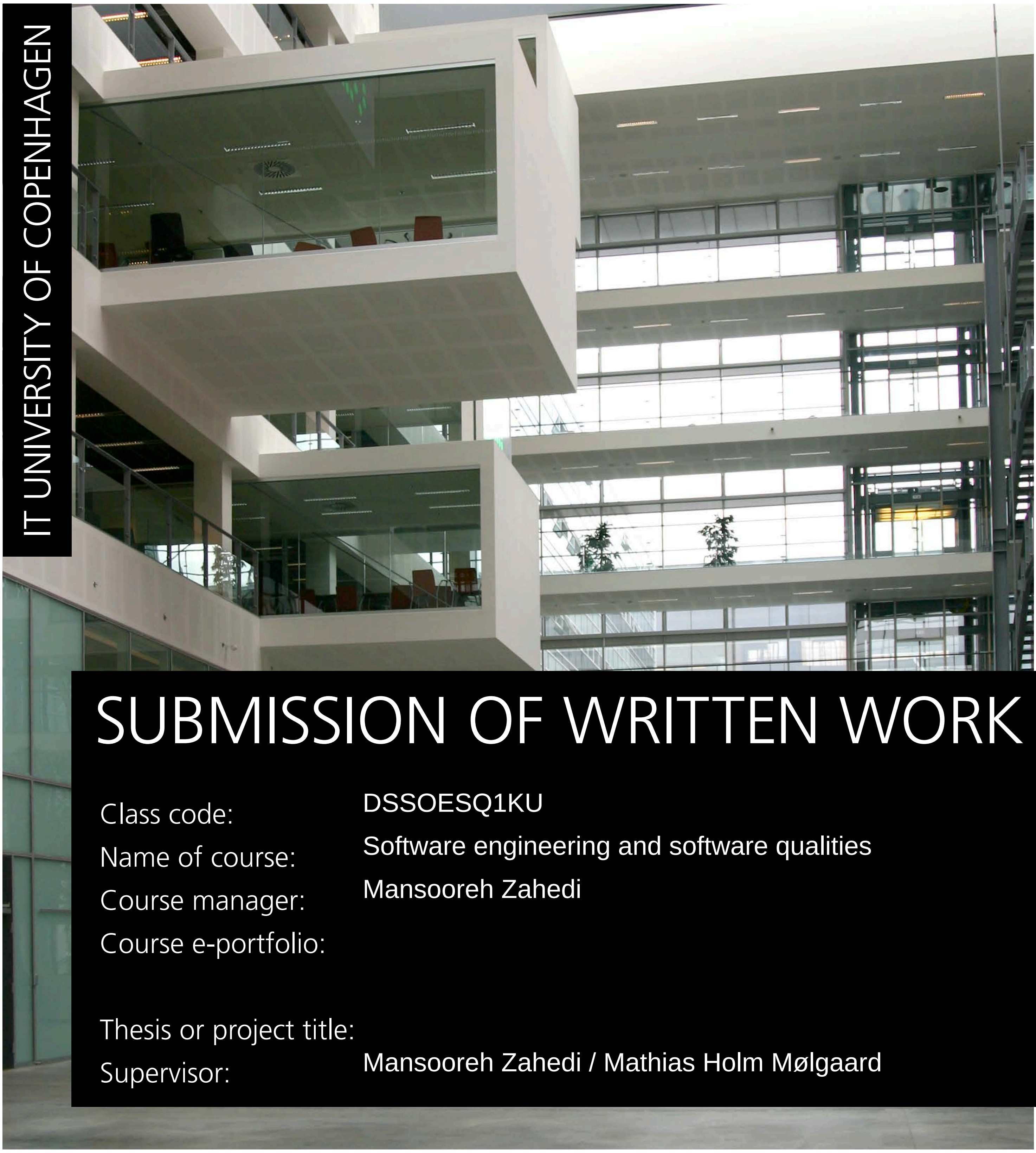


IT UNIVERSITY OF COPENHAGEN

SUBMISSION OF WRITTEN WORK

Class code:	DSSOESQ1KU
Name of course:	Software engineering and software qualities
Course manager:	Mansooreh Zahedi
Course e-portfolio:	
Thesis or project title:	
Supervisor:	Mansooreh Zahedi / Mathias Holm Mølgaard

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Silviu-George Agafitei	19/04-1995	sgeo@itu.dk
2.	Vlad-Alexandru Ilie	21/03-1995	vali@itu.dk
3.	Emil Greve Krogsgaard	29/12-1989	egrk@itu.dk
4.	Roman Novosad	25/05-1996	romn@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Software Engineering and Software Qualities

JustPark - mobile parking solution for Copenhagen

Table of Contents

Project Glossary

Term	**Alternative expression**	**Explanation**
User	**Driver, client**	User driven a vehicle in need of parking interacting with the app with intention of finding a parking place.
Vehicle	**Car**	Object to be parked at the purchased parking place.
Route calculation		Calculation of the optimal route to the user's desired parking place. (not to be confused with price calculation)
Route optimization		Optimization of the aforementioned calculation according to the live traffic info.
Price calculation		Calculation of the final price. Calculated as: [price per time unit]x[duration of parking]
Critical priority	**Critical**	App won't be functioning without this feature.
Major priority	**Major**	App will be functioning, but the feature is still essential.
Minor priority	**Minor**	The feature is nice to have, but doesn't necessarily have to be included.

Done		A task is done, when it has been quality checked and is finished according to the specification and relevant user stories defined by product owner.

Smart Parking - Case Exploration

Document title: **Smart Parking - Case Exploration**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 25/112017

Author: Roman Novosad, Silviu Agafitei

Owner: Roman Novosad

1. Purpose of Document

1.1 Overview

The purpose of this document is to is to explore the the application and the problem domain of the case.

1.2 Distribution List

This product was carried out buy the product owner and scrum master, but it's contents were communicated to the whole team to achieve complete understanding of the topic.

1.2.1 For Review

Name & Role	reason	Date sent	status

Roman - Product Owner	Product owner needs to be well oriented within the problem domain	10/12/2017	Final
Silviu - Scrum Master	The team leader has to consider technical implications within the problem domain	10/12/2017	Final

1.2.2 For Information

Name & Role / Department
Development team

Over the last decade, the number of cars in the cities has grown exponentially. However, parking estate has stayed factually the same. Obsolete infrastructure combined with limited number of parking spots is unable to cope with the influx of vehicles on the roads. Especially in larger cities, there have been various proposals on how to improve this situation. These proposals have been revolving around three main concepts:

1. Improving driver experience
2. Decreasing travel times
3. Improving monetization possibilities

Along with these three, implementation of a smart parking system would help to decrease air pollution caused by "cruising" (Shoup, 2006) and congestions in the cities. In the upcoming report we want to address all of these while focusing on the given setting of Copenhagen.

As Copenhagen is a relatively small city (compared to some other european capitals), there have been some predecessors who have already implemented smart parking on a small scale. In relation to this, there has been a lot of research into opportunities such as parking improvements, which we will take in account while planning this project in Copenhagen and draw from it's examples to make this process as effective and seamless as possible.

There are a number of distinct facets for this solution, such as: communication of parking information (availability, location) with drivers, monetization of parking services and ensuring prevention of fraudulent driver behavior. Here, we are going to refer mostly to the more challenging - latter point. The solution for preventing drivers from parking their car at a problematic location would be an effective system of vehicle detection. There is a lot of work already produced on this topic, for this case we chose Idris et. al. (2009) paper describing number of instances of solution for detecting presence of a car directly on the parking spot.

It considers different technological concepts like:

- Active infrared sensors
- Inductive loop detectors
- Magnetometer
- Piezoelectric sensor
- Pneumatic road tube
- WIM sensors
- Microwave radar
- Acoustic sensors
- Ultrasonic sensor
- RFID

There has been a number of these systems deployed for numerous places around the globe (Idris, Leng, Tamil, Noor & Razak, 2009). After a brief examination, we can assume that for the project of a big scale, these solution requiring modification of an individual parking spaces would be considerably difficult to implement. On the other side, there have been numerous proposals of solutions using GPS, thus not requiring physical alteration of parking spaces (Pullola, 2007). We will also refer closer to the choice of specific technology once we dive deeper into the case.

This outline has so far provided us with some valid options to select from. As mentioned above, we also need to establish the communication between our solution and the end-user. This is intended to be done via smartphone, namely a mobile application with location data for parking spots as well as metadata about them, for example: the possibility of making a reservation or different types of payment possibilities for them. Studies show, that this implementation would improve driver satisfaction and traffic condition in the cities significantly. (Wang and He, 2011)

Software process models

Document title: **Descriptions of the software process models**

Document Ref:

Version No: 2.0

Document Status: Final

Date: 25/11/2017

Author: Roman Novosad, Silviu Agafitei, Emil Krogsgaard, Vlad Ilie

Owner: Silviu Agafitei

1. Purpose of Document

1.1 Overview

This document provide basic understanding of the different software development projects and choose the one most suitable for the given project.

1.2 Distribution List

To ensure that this section has sufficient standard, Scrum Master collected and summarized all the details on relevant software development projects.

1.2.1 For Review

Name & Role	reason	Date sent	status

Silviu - Scrum Master	Scrum master should be the one most with best capabilities to consider correctness of the model descriptions	10/12/2017	Final

1.2.2 For Information

Name & Role / Department
Development team
Product Owner

This section will be used for discussing the process models that could be used to develop this software application. The models are complex and choosing one requires a lot of attention to details. In order to make the best choice, we utilised the online learning platform ISTQB Exam Certification (hereinafter: ISTQB, 2017).

Waterfall model:

The Waterfall Model was introduced by Winston Walker Royce in 1970 in his paper , *"Managing the Development of Large Software Systems"* (Winston W. Royce, 1970) and represents a plan-driven approach to creating and implementing software products. This methodology has also been referred to as a linear-sequential life cycle model (Winston W. Royce, 1970). It assumes, that the development process can be broken down into a list of non-overlapping steps that have to be completed in a strict, sequential order. As a result, at the end of each stage, a formal review has to be conducted in order to decide whether or not the current stage has been finalised and whether or not the project can advance to the net phase. This type of software development model is mainly used for small projects or projects that have a very well defined list of requirements. (ISTQB, 2017)

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- The rigidity of the model makes it easy to manage
- Each phase has individual deliverables
- No overlapping phases entails that steps are processed and completed one at a time.

Disadvantages of waterfall model:

- If concepts changes after development has initiated it is complicated to pivot.
- Working software is only available at a very late stage in the life cycle.
- Inadequate model for long, ongoing or iterative projects.
- The method is is generally seen as containing high volumes of risks and uncertainty

When to use the waterfall model:

- The definition of the final product is well understood in the development team and the familiarity with the technology is s high.
- There are no ambiguous requirements
- Resources and expertise is freely available
- The project is short or well-defined.

Spiral Model

The spiral model uses a risk-driven approach to the development process, and easily integrates styles from other process models such as the incremental model (Boehm, 1986). This model is divided into four consecutive phases: Planning, Risk Analysis, Engineering and Evaluation. The project will repeatedly iterated through these phases also referred to as Spirals. When going through each phase the product will be further developed and thus produce an input to the next cycle. When a project is kicked off it will start in the baseline spiral, diving into the planning phase, from here the project will develop requirements and determine any risks associated with the project. All following spirals seeks to expand on the original baseline spiral (Boehm & Hansen, 2000). (ISTQB, 2017)

Advantages of Spiral model:

- Risks are assessed in each cycle, thus should help to minimize most risks from impacting the project.
- It is possible to adhere to developing needs.
- Software is produced in every iteration, creating builds early in the life cycle.

Disadvantages of Spiral model:

- It is rather costly and complex to use.
- The analysis of risks are generally considered costly and will often require external consultants.
- Is of limited use for smaller projects as the bureaucracy quickly outweighs the pros of minimizing risks that are already limited in small projects.

When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Users are unsure of their needs
- The Requirements gathered are complex
- New product line or less known product
- Significant changes are expected to the final product.

Incremental

The incremental model the whole requirement is divided into various builds (ISTBQ, 2017), adding piece by piece but expect that each piece is fully finished and the process continues until the whole system is built and functional.

Advantages of Incremental model:

- It is flexible and generates working software in a fast pace and early during the software life cycle.
- It is easier to test and debug in increments.
- The customer is able to respond to each built.
- Easy to manage risk because risky pieces are handled during iteration.

Disadvantages of Incremental model:

- To enable its success strong project management is a necessity.
- All aspects of the system generally needs to be defined before the project can be split into the increments that will create the product.
- Generally results in higher costs.

When to use

- The Incremental model can be used when the project has a complete understanding of all the requirements prior to the building phase.

Prototyping model

First of all, prototype model is a software development model. The model aims to create an early approximation of a final system or product, based on the known requirements. By creating an early model an opportunity for end user interaction is enabled which allows for valuable feedback that can provide guidance for the remainder of the project. In general, the method is applied for large and complicated system, which do not have an existing process or system to determine the requirements (for example a technology that was never developed) (ISTQB, 2017).

There are several advantages by using the Prototype model:

- The users are actively involved in the development, enabling great transparency between end users and project teams.
- By creating prototypes we enable the detection of errors and misunderstood requirement early in the development phase.
- Missing functionality can be detected easily.

However, there are several disadvantages when using the Prototype model:

- The model may increase the complexity of the system and the scope may be expanded compared to the original plans.
- Incomplete application may cause application not to be used as the full system was designed
- The problem analysis is incomplete.

There are several situations where prototyping is a particularly useful software model:

- When a new technology is being developed.
- As there is a lot of focus on making the user interact with the unfinished systems the model is very good at supporting the development of products that has great user experiences.

Agile model

Agile development is a software development model focused on developing using a very iterative approach. This means focusing on building small chunks of the project one at a time, realising each before delving into the next part of the project.

There are various advantages of agile model, so in this part we will outline some of them. Teams usually prefer this model because customer satisfaction increases when offering rapid, continuous delivery of useful software. (ISTQB, 2017) In this method, people and interactions are emphasized rather than process and tools and members of an agile team constantly interact with each other. It’s useful to note here, that in agile, face-to-face communication is considered the most useful form of communication. However, many agile teams are due to various advantages embracing online communication tools. Main of them being able to offshore while being cohesive and dynamic. The agile manifesto strongly implies, that working software is delivered frequently (weeks rather than months) and even late changes in requirements are welcomed.

Even though most of the teams are in favour of Agile (namely Scrum), there are disadvantages to this model. It can be difficult to assess the complete effort required to fulfil project of larger sizes. Other disadvantage of agile results from it relying on people's individual skills. There is often a great reliance on senior developers to making decisions, hence it has limited place for junior programmers, unless efficiently combined with experienced resources. (ISTQB, 2017)

Summary table for the software development models:

	Waterfall	Spiral	Incremental	Prototyping	Agile
Ease of use	High	Low	High	Low	High
Rigidity	High	Medium	Low	Low	Low
Requirements stability	Few changes	Many changes	Few changes	Many Changes	Many Changes
Risk factor	High	High	Medium	Low	Low
Complexity	Low	High	High	Medium -> High	Medium -> High
Time-span	Low	High	Low -> High	Low -> High	Low -> High

Criteria for choosing a process model for our project:

a. rapid, continuous delivery

b. taking advantage of communication between team members

c. frequent software delivery

d. attention to a good software rather than extensive documentation

e. adaptability to changes

f. speed of deployment

Project kick-off - common lunch

Document title: **Descriptions of the software process models**

Document Ref:

Version No: 2.0

Document Status: Final

Date: 25/11/2017

Author: Roman Novosad, Silviu Agafitei, Emil Krogsgaard, Vlad Ilie

Owner: Silviu Agafitei

1. Purpose of Document

1.1 Overview

This document is very short description of a team building activity carried out by our team. The purpose was to get to know each other better in order to ease collaboration in a team.

1.2 Distribution List

Each team member has participated, but original initiative was taken by a Scrum master.

1.2.1 For Review

Name & Role	**reason**	**Date sent**	**status**
Development team	Each of the team should approve the document content	29/10/2017	Final

1.2.2 For Information

Name & Role / Department
Development team

As a group we have been working together for quite a while, but for the purposes of getting closer together and getting to know each other better, we decided to have lunch in the school canteen. The whole event was mildly successful and our experience was accompanied by vivid discussions and exertion of excitement about what is awaiting us. We have spent time planning for the future endeavors of our group and how each member can contribute to the great group experience by their individual qualities. (Image 1)

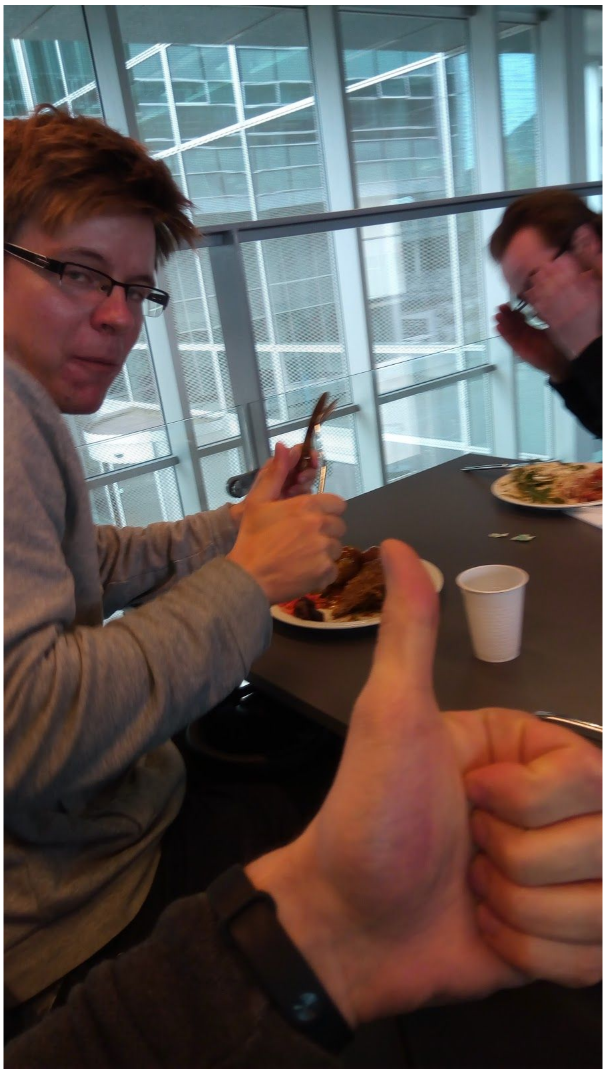


Image 1

Project Planning

Throughout the “Smart parking” project we expect a high degree of compartmentalisation, precisely larger parts of the project will need to be coordinated. Therefore, in our first group meeting we decided to use Agile methodology, namely Scrum. In scrum, the use for daily cooperation and conversations within the team are highly needed, thus this version of the agile model is the best choice from the process models available.

The project is planned to commence between 31 august - 15 december, when we should present a fully functional application. We decided to split the time in eight Sprints, which will in the end result into a functional product. Each iteration will contain: Sprint planning, sprint goal and sprint retrospective/review. The Sprint will be planned in terms of workload distribution by creating a Sprint backlog and thereafter Sprint goal will be created. The sprint review and retrospective would help us to improve the future sprint. Below, we created the representation of the future Sprint distribution. Each sprint length will be two weeks, and in the given period the sprint backlog items have to be "done". In this context, done is defined as as a shippable product/derivable.

Sprint	Period	Sprint Goal
Sprint #1	31/08 - 14/09	Structuring the core solution
Sprint #2	14/09 - 28/09	Discovering technical possibilities (pretotyping)
Sprint #3	28/09 - 12/10	Map setup and map creation (blackbox)
Sprint #4	12/10 - 26/10	Integration with Google and Septima API's
Sprint #5	26/10 - 09/11	Log In / User registration
Sprint #6	09/11 - 23/11	Route calculation
Sprint #7	23/11 - 07/12	Route optimization
Sprint #8	07/12 - 15/12	Integration with established billing systems

Table 1 - Sprint overview

Above we described the Sprint distribution throughout the project. As mentioned, we will split the project in Sprints and each Sprint will have to meet the Sprint Goal. A sprint goal is a description of what we have to achieve during the sprint. As a result of defining a sprint goal, we need to create the sprint backlog. A sprint backlog will consist of a set of requirements that have to be “done” at the end of the iteration. In Scrum a requirement is known as a user story, and all the user stories are stored in the Product Backlog (see table 2). Our Product backlog includes user stories describing all the functionalities desired in the “Smart parking” application, the amount of points allocated for the given task and the priority level. However, the product backlog is never “done”. Throughout the project, new items have to be added to the product backlog based on the customer's new requirements.

One of the most complex requirements in Scrum is to estimate the duration to implement the user stories. Usually the whole team gathers together when estimation is taking place. The most common estimation methods are “planning poker”, “planning onion” or “fibonacci planning”. We now have to decide which of the methods would fit best in our project. If planning is not applied correctly, we would have a delayed release and therefore higher costs and customer dissatisfaction.

Planning poker is a widely used method for estimation, in an Agile environment (Cohn, 2017). The planning session consists of the product owner/customer and development team, each member of the development team holding a deck cards with values from 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100. The product owner/customer reads a user story out loud and each member of the development team has to assign a card number, the higher the value the more complex the task is. The planning process is repeated until consensus is achieved or until a particular item needs to be deferred to a timer where additional information can be acquired. By taking into consideration we are creating a solution and an application within an academic course, even though the estimation is particularly difficult due to the lack of experience. Moreover, planning poker is more suitable when the backlog contains a small number of items (Berteig, 2015).

The table below displays the Product Backlog, explicitly a series of user stories with the corresponding estimation and prioritization. However, the tasks assigned with story points have been planned without having a common planning practice. The manner in which stories and tasks were assigned a “story score was ” by having a debate on the difficulty of each task.

User story	Task	Story points /Estimation	Priority
As a user I want to have a description of the tool, so I can decide if the case is relevant.	Built documentation for JustPark. Should include FAQ, quick guides, instructional video.	8	6
As a driver, I would like to be able to find an available parking spot. (epic user story)	Build an application for the parking system in Copenhagen.	89	1
As a driver, I want to be directed to an available parking spot closest to my desire destination.	Area scanner. / localization system plus availability check	34	2
As a driver, I want to have a system that updates the parking time so that I would avoid parking tickets.	Create stopwatch function. Payment update by stopwatch.	8	2
As a driver, I want to be able to pay through the app so that I am not forced to manually pay at the ticket booth.	Create a payment interface using a mixture of mobile pay and credit cards	5	4
As a user, I want to receive the billing on my email, so I	Integration with billing systems.	3	5

could keep track on my spendings.			
As a parking control officer, I want to be able to check if cars are registered for parking.	Built a number plate scanner.	8	4

Table 2 - Product Backlog

As it can be depicted from the table, the product backlog consists of that have to be done are epics, user stories that cannot be done within a single sprint. Usually, an epic user story is not broad and short on details. Moreover, the story has to be splitted into multiple smaller stories before assigning the tasks.

If in a normal Scrum environment only the development team resolves the task, in our context, academic, all of us had to work for the completion of the tasks. We decided to split the work not necessarily only between our two developers. Throughout the project, we have created Scrum planning sessions, which enabled us to create the main stories, epics, and thereafter splitting them into multiple smaller tasks, so we can ensure each sprint goal and backlog will be met accordingly without delays. However, the Scrum Product backlog should be composed out of short and understandable requirements, instead of having an epic user story.

Roles in Scrum

As we have chosen to use scrum methodology for our project, here we outline a number of roles that are specific for this methodology. Below is a table showing assignment of the roles in our team.

Roles	Person assigned	Responsibilities
Product owner	Roman	Customer / client

		communication To provide requirements
Scrum Master	Silviu	Coordination / coaching
Developer#1	Vlad	Coding
Developer#2	Emil	Coding

Table 2 - Scrum role designation

There are three main roles in the scrum methodology: product owner, scrum master and developer. In an ideal scenario, there would be one product owner, one scrum master and multiple (preferred size of 3 - 9) developers (Schwaber & Sutherland, 2013) - as our group is consisting of only four members, we decided to have two full-time developers and keep other members flexibly available for contribution to the development when needed. In a smaller team scenario, it’s a common practice to combine roles of the team members (most often we can see a scrum master who is at the same time developing).

According to The Scrum Guide, the product owner should be in a constant dialogue with the customer, conveying the information about his/her needs to the development team. As in this project there is no explicit customer; as a result the product owner will be responsible for finding all the information needed for the project from appropriate stakeholders as well as various online sources (websites related to the project, blog posts and public project documentation). This includes analysis of the market and competition, and appropriately reacting to the changing market needs throughout the project. Product owner will also be determining what should be built and in what order. This is generally done in product backlog, from where developers pull their work. It further includes building requirement specification (definitions of completion and verification criteria), from which epics will be derived. These epics will be later decomposed by the product owner and the team. Lastly, the product owner is responsible for reviewing the work done by the developers at the end of each iteration. Work can then be accepted and released, or returned for another iteration.

As it can be seen in Table 1, Silviu as a scrum master will coach and coordinate the team. What this entails is ensuring compliance of the team with scrum methodology and its correct execution. Other than this, he will be the link between the development team and the product owner. He is shielding the development team from the outside influences and making sure they will keep the required effectivity. Regarding collaboration with the product owner, scrum master will assist with defining the tasks for developers and keeping product backlog organized.

Our development team is consisting of two members but as aforementioned, the other members will be able to contribute to the development as well. A special quality of a scrum team is the ability to self-organize. There is no hierarchy in scrum. Instead, the team is responsible for their actions as a whole and this is being reflected onto the motivation and commitment of the members. The developers are expected to be cross functional, which suits experiences of Emil and Vlad. This will allow great flexibility within the team and ensuring all other members' correct expectations of the deliverables.

Generally, doing scrum should be a great experience for each member of the team if executed correctly. At the end of the day, everyone will gain experiences from collaborating and communicating with other members closely to avoid redundancy and satisfy customer in the best manner possible.

Risk Analysis

Document title: Risk Analysis

Document Ref:

Version No: 4.0

Document Status: Final

Date: 15/ 12/ 2017

Author: Vlad Alexandru Ilie

Owner: Vlad Alexandru Ilie

1. Purpose of Document

1.1 Overview

The purpose of this document is to provide a detailed Risk Analysis. This section contains a list of initial risk that were identified through brainstorming, a ranking of these factors that could drastically influence the project at a later stage, In addition to this, we recognized the necessity for further research and we took an appropriate course of action to mitigate the most severe risks.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Silviu Scrum Master	Owner of the document, needs to know what problems might arise in the future	25 September 2017	Final
Roman Product owner	Customer needs to be made aware of what	25 September	Final

	vulnerabilities the product might have	2017	

1.2.2 For Information

Name & Role / Department
Developers

This section of the report will focus on exploring various aspects of the “JustPark” solution that we, as a team, identified to be potential sources of risk. In the following paragraphs three major types of risks are presented and ranked according to a risk level - which is calculated based on the probability of the event happening and the effects it would have if it did happen.

It is important to note that the first risk assessment was conducted before the initial data generation period. The reason for this, is that we first had to identify which areas of the project required further investigation. After a series of semi-structured interviews, we conducted a second risk analysis and changed our solution in order to mitigate the three risks that, initially, had a probability of 100%.

Project risks:

1. Copenhagen already has systems that attempt to improve traffic congestions, as a result drivers might feel confused or unwilling to change from their preferred choice.
2. The system would have a limited efficiency during peak hours. In this case, traffic is caused by an infrastructure that cannot support a large number of cars. In addition to this, roads in Copenhagen need constant attention and repairing. Because of this, many areas and routes have to be closed to allow for repairs to be made.

3. To be able to direct drivers to a specific parking spot, the app needs to know where and how many parking spots are in any given area of the city. As a result, the task of creating a system of such size can easily be under-planned.

4. The solution would be required to use the billing system created by the already established private parking organisations - since they have ownership and regulatory authority over the paid parking areas.

Product risks:

1. A piece of software, no matter how advanced, cannot ensure that traffic problems will be solved.

2. The software relies on driver's participation in order to improve the driving and parking experience

3. Vague requirements

4. Part of the spectrum of the app's functionality might have been left at the developer's interpretation.

5. Unknown size of the app's infrastructure.

Team risks:

1. Divergent set of skills

2. Misunderstandings as to what the solution should be.

3. Different ways of understanding the methodologies.

4. Having different backgrounds, we believe that different things should be the prime focus.

Table 2: Risk Analysis prior to data generation

ID	Description	Probability	Effect	Level	Action

1	User resistance to change	50%	5	2.5	Marketing campaign
2	Limited efficiency during peak hours	70%	10	7	Continuously update and optimize the software
3	Difficulty assessing the size of the system	50%	7	3.5	Create an actual solution and not work with a hypothetical solution
4	Integration with established billing systems	10%	10	1	Develop a new billing system, which would cause significant delays.
5	A smartphone app cannot solve traffic issues	50%	10	5	More research.
6	Community based needing many users	50%	8	4	Marketing campaigns.
7	Vague requirements	100%	10	10	More and better discussion with whoever initiated the project or whoever is paying.
8	Devs are responsible for defining the functionality	100%	2	2	Ethnographic studies to determine what the best UX and interfaces.

9	Unknown infrastructure dependencies	100%	10	10	Create an actual solution and not work with a hypothetical solution
10	Divergent skills	40%	3	1.2	Assign responsibilities based on what we are capable of doing.
11	Misunderstandings in regards to the solution	60%	5	3	Have more meetings where we discuss what we did, what we plan on doing and how we plan on doing it.
12	Different understanding of the methodologies	60%	3	1.8	We should study the methodologies more
13	Different focuses	40%	7	2.8	Have more meetings where we discuss what has to be done.

It is important to note that at the point of writing the risk analysis, we believed, as a group that further documentation and data generation was needed in order to fully understand the functionality of the application. As a result some of the entries presented in Table 2 have been addressed. The following table “Table 3: Risk Analysis after data generation”, presents the risks that had a probability factor of 100% and have been mitigated.

Table 3: Risk Analysis after data generation

7	Vague requirements	50%	10	5	Features that will be added after the initial release might

					require changes to the entire application
8	Devs are responsible for defining the functionality	10%	2	0.2	The GUI and UX should be created by a proficient designer.
9	Unknown infrastructure dependencies	30%	10	3	The billing system is black-boxed. To be able to integrate it with our application a partnership with the billing organisations needs to be reached.

Exploring software qualities

Document title: **Exploring software qualities**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 12/12/2017

Author: Emil Krogsgaard

Owner: Emil krogsgaard

1. Purpose of Document

1.1 Overview

The purpose of this document is to discuss the software qualities associated with the development of the JustPark app.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Roman Product owner	Customer needs to be made aware of the architecture being used.	5/10/17	Final
Silviu Scrum Master	Owner of the document, needs to cascade information to developers.	5/10/2017	Final

1.2.2 For Information

Name & Role / Department
Scrum Team

Using the ISO/IEC 25010 Product Quality model characteristics we have decided on the 5 qualities that will be of utmost importance to the success of the final product (ISO, 2017), these characteristics are listed from most important to least important:

1. Reliability

This characteristic discovers how mature the final product is. By focusing on this aspect, we will focus on how often the product is up and running, how tolerant it is against faults and how easy it is to recover in case of a breakdown.

As our product aims to replace the majority of the existing parking systems in Copenhagen, we deem it to be of utmost importance that the system is stable and has the highest possible level of availability. If this is not the case the public is likely to turn their opinion on the system and possibly reject it completely.

2. Usability

This product characteristic describes how appropriate the final product is to its intent. It is very user centric in that it accounts for how the user views the product by for example understanding of operational qualities, ease of learning and accessibility.

For a mobile app that will be used by everyone with a driver's license it is really important that it succeeds in creating a positive user experience as this will help people to accept the product faster.

3. Performance Efficiency

This aspect seeks to understand how the product performs under scrutiny of stress testing. If the functions are developed well enough then it will be capable of handling the incoming user queries. As there will be an intense load on the system in rush hour times, it is critical to product success that it is able to handle the load during peak hours.

4. Maintainability

Maintainability refers to the easiness of maintaining a given system. This can be in terms of how easy it is to modify the product, use the product for similar purposes or test it.

Even though the main requirements for our system are likely to remain stable once properly implemented, we expect that unforeseen user behaviours will spike the need for changes in the system. As different parking options emerge (for example Charging stations) it must be possible to not only add new metadata, but also make adjustments to general functions in the product.

5. Security

The focus in the security domain is on integrity and confidentiality of the user that it is serving.

As many transactions will be made through this system it is highly critical that the system is properly secured, thus minimizing the risk of stolen data. Users will also have to create accounts that work to authenticate them and their car.

This said we decided to value the security lower than our other quality aspects. This decision was made because the product will need to be fit for purpose and reach a high quality level seen from the user's perspective in the beginning of the project life cycle. Once we have gained the support of our users we can pivot our focus in quality to security.

Reflection on process model

As we have decided on using the agile method of Scrum, the first software quality characteristic that comes to our attention is the focus on maintainability. This area also focuses on creating an agile product that is not completely static after launch, but it has a design that allows for some flexibility in improving its functionality. This is likely to be needed; as future technological progress is expected to impact the parking market. However, usage of scrum might not necessarily be the best method, as it does not have a deep focus on providing proper documentation of the end product, making it harder to carry out improvements and changes in a more distant future.

In addition to this, making further progress on the tool can in many aspects be seen as contradiction to the security and the reliability of the system. Any changes made to a system will inevitably cause concerns in terms of security and reliability as it can cause disturbances in both aspects if not properly managed. In worst case this can break existing features.

However, to minimize these concerns a large emphasis should be made on conducting proper change management which is discussed in greater detail later in the paper.

Requirement Elicitation

Document title: Requirements Elicitation

Document Ref:

Version No: 4.0

Document Status: Final

Date: 15/ 12/ 2017

Author: Vlad Alexandru Ilie

Owner: Vlad Alexandru Ilie

1. Purpose of Document

1.1 Overview

The purpose of this document is to provide a complete list of requirements that have to be implemented into the application. It is important for our solution to attempt to solve the real problems that are faced by the drivers. In order to reflect reality, the functional and nonfunctional requirements have been created based on user stories and epics, which in turn were created based on semi-structured interviews.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Silviu Scrum Master	Owner of the document, needs to confirm that this is the final list of requirements	14 October 2017	Final
Roman Product owner	Customer needs to approve the final list of requirements	14 October 2017	Final

1.2.2 For Information

Name & Role / Department
Developers

The initial phases of the software development process of the JustPark application relied on a set of assumptions in regards to the social tendencies of drivers in Copenhagen. Moreover, as developers, we did not possess a good enough understanding of the social requirement that would have to be met in order for the application to be accepted and used by drivers. In addition to this, a series of risks directly related to the necessity of having more research into user behaviour, have been identified in the Risk Analysis section. In order to bridge this gap in knowledge, our group has conducted 3 semi-structured interviews with drivers of different background. The following paragraph will provide a description of the interview guidelines and our rationale for choosing this method.

The interviews have been developed as a means to better connect with our potential user base and ensure that the gap between user stories and actual user desires will be lessened. The semi-structured style encourages the use of personal intuition to navigate between set questions and prompting the interviewee for elaboration when needed. This allows for a high degree of flexibility and gives the interviewer the freedom needed to investigate topics that the interviewee brings into the conversation (Edwards & Holland, 2013)

Semi-structured interview script:

1. Can you please tell us how your general driving routines are?
    a. How often do you drive ?
    b. Where to ?
    c. Any daily commutes?
2. How would you describe the parking situation in the areas that you generally drive to?
    a. How easy is it to find a spot?
    b. How much time would you say you spent?
3. Do you have a parking story that you would consider horrible?
    a. What was the irritating elements about it?
4. How could this experience have been improved?
5. What are your general concerns in regards to parking?
6. We are in the process of designing a parking app for the city of Copenhagen that would help you find and locate available parking spots in the vicinity of your destination.
    a. What are your general feelings about creating such a service?
7. How could you see an app like this change your life as a driver in Copenhagen? (will they alleviate some of your concerns).

This interview structure has been used to conduct three interviews with drivers in Copenhagen. From these interviews we have identified the following aspects that drivers would be interested in.

Social requirements:

1. Drivers would like an app that tells them where to find a parking spot and how to get there.
2. Drivers don't want to use multiple apps just for parking:
    a. app for finding your way to the parking spot
    b. app that tells you where the parking spot is
    c. app that allows you to pay
3. A driver wants to be told whether the parking spot they are directed to is paid or free.
4. A driver wants to be able to select the type of parking (paid / free) they they will be directed to.
5. Drivers would like to specify the maximum distance between the parking lot and their destination.
6. Drivers don't want to spend time setting up and using the app.
    a. creating the account and setting up payment methods
    b. choosing a destination / parking spot has to be intuitive and fast
    c. verifying a driver's position in the parking lot has to be done automatically
7. Drivers are more interested in parking spots in the central area and not so much in the suburbs.
8. Drivers are interested in receiving parking information when they travel to less known destination.
9. The app must **not** require user input while the user is driving.

10. Drivers want to be able to terminate their parking timer without paying for the entire time.
    a. If a user occupies a parking spot and selects, for example: an interval of 4 hours but ends up spending only 1 hour. The driver will pay for 1 hour and not their initial estimation

Technical requirements:

1. Personalized user accounts (profile pictures / route tracking / parking time tracking) are not a vital necessity but could be implemented at a later time
2. Drivers want the assurance that their data is secure:
    a. nobody should have access to their driving / payment history
3. Client - Server - 3rd Parties communication channels for devices
    a. secure data transfer from a user's smartphone to our servers
    b. establish secure connection between a driver and the billing system
4. Custom made map of Copenhagen
    a. Septima.dk for geolocating all danish addresses
    b. parking lot registration:
        i. where is the parking lot
        ii. how many parking spots are there
        iii. where are all the parking spots
    c. route calculation
    d. route optimization

Based on the social and technical requirements created from interviews with drivers in Copenhagen, as well as taking into consideration our judgement of what functionality an application is supposed to have, we have created the following functional and nonfunctional requirements that our app, JustPark, has to support.

Functional Requirements:

1. A user should be able to use the JustPark app 24/7.
2. A user should be able to search for a parking spot in any part of Copenhagen including the suburbs.
3. A user should be able to view parking spots for both paid and unpaid parking areas.
4. A user is uniquely identified by their phone number / email.
5. A user confirms occupying a parking spot via any smartphone’s gps/location signal.
6. A user cannot reserve a parking spot in advance as the Smart Parking app uses a first come, first served basis.
7. The app has to be able to process a very high level of requests in real time without any latency issues.
8. A user’s activity within the app should be anonymised
    a. can be achieved by securing all communications between a user’s smartphone and our servers via SHA-256 encryption methods.

Nonfunctional Requirements:

1. The app’s GUI needs to be designed in such a way that users could be capable of intuitively using the application.
2. For users who want to be shown how the app functions, a tutorial video, no longer than 30 seconds will be available at all times in the main menu.

Understanding the social and technical context of the JustPark app:

Problem domain:

A problem domain, defined as “the area of expertise or application that needs to be examined to solve a problem”("problem domain", 2017), is represented, in our case, by location tracking methods, data processing techniques, online payment protocols and parking policies.

This application is not the first one to attempt to resolve the parking situation in Copenhagen, however the manner in which our Smart Parking app will tackle these issues is different from anything our competitors are offering. The main problem when using a parking app is the fact that so far, no application available at the moment on either Android’s Play Store or Apple’s App Store, is capable of saying how many parking spots are available at any given time in a parking lot in Copenhagen (Appendix C). The reason for this is that, already existing application use software to monitor traffic in order to make predictions about available parking spots and show users a percentage of expected available spots. For example, a driver using EasyPark would be told that in a certain area or parking lot, there should be, for example, 50% available spots. While this represents an important first step, we believe that the key to a highly reliable and accurate parking system is in developing a map of Copenhagen and its suburbs that can store information, such as: time restrictions, price, type of parking, available spaces, etc.

Our JustPark app is intended to allow users to select a destination and they will be provided with the the most optimal route as well as with the closest available parking spot to their destination. The application is not intended and will not be capable of billing users directly. The reason for this is that, the entire region of Copenhagen and its suburbs are regulated by the government with four service providers: Easypark, WayToPark, Parkman or Parkpark (Parking in Copenhagen, 2017). As a result, our solution to the parking problem in Copenhagen is not a for-profit organisation because it would not have ownership rights for any parking area in Copenhagen. Despite this, our application will take into account the owners of each parking area and will have integrated their payment mechanism.

The JustPark application represents a centralisation of car park service providers over a new and improved road map infrastructure.

Application domain:

The application has an over encompassing goal of being utilised by all drivers in Copenhagen and subsequently in Denmark.

The targeted user is any person that has a driver’s license, access to an automobile and wants to drive anywhere in Copenhagen.

Because our application is capable of providing the most optimal route to a selected destination, the app can be used as a replacement for GPS navigation devices.

Technical context:

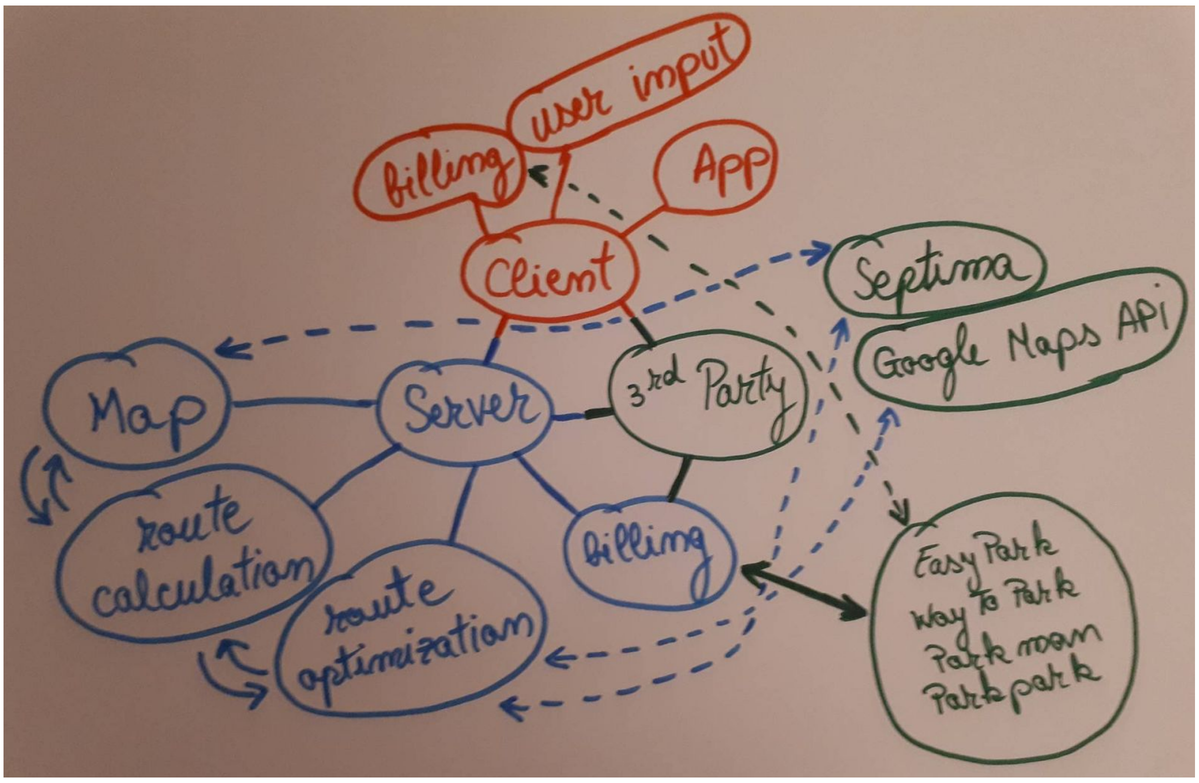
As stated in the problem domain section, our Smart Parking application mainly revolves around the use of location tracking over a custom made map of Copenhagen and its suburbs via cloud processing methods to ensure reliable and fast service.

So far, the toughest challenge is the creation of the custom map of Copenhagen and its suburbs in order to be able to track parking lots occupancy rates and level of traffic. The first step would be to buy an already existing map of Copenhagen that contains all Danish addresses (P/S, 2017). This map would have to be modified in such a way that it would be able to store information about each parking lot. Each parking lot would be registered on the map as an address containing multiple parking spots, which contain different types of information and details about the parking conditions.

A user would start using the app by first either typing in the address of the destination or selecting one on the map. Once the user finished picking the destination, they would be asked whether or not they would like to park at the destination. If the user declines the offer, the app will simply provide the user with the best route. If the user accepts the offer, not only will the app provide the user with the best route to the destination, but once the driver is close to the destination (for example less than 300 meters to the destination) the route will be automatically updated so that it would take the driver to an available parking spot that is the closest to the desired destination. Once the driver arrives, the app will automatically detect via the driver’s smartphone GPS location and cross-referencing it with the custom made map to accurately tell the position of the parked car. Once the driver confirms the position of the car on the map, the he/she will be directed to the billing system that corresponds to the service provider of the car park.

After a user pays, the map is updated to reflect that the parking spot has been occupied.

Technical figures portraying the system structure

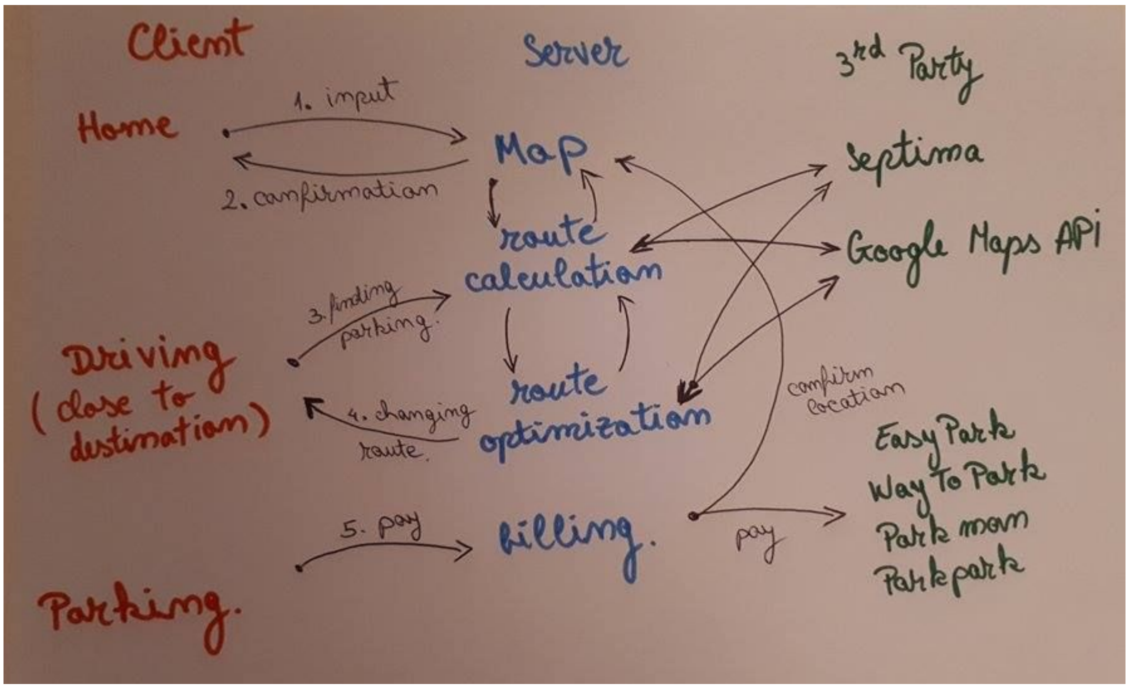


The first figure represents an abstraction of all components that form our solution to the parking problem in Copenhagen. It can be seen that the client, driver only has access to the smartphone application, which represents a direct connection to our servers. In addition to this, the user is required to provide the app with a valid destination and depending on parking policies, the driver will have to pay.

The server represents everything that our application has to be capable of doing reliably. As previously stated, the core component is the custom made Map of copenhagen and its suburbs. As it represents the backbone of the entire solution, it is of the utmost importance that the map reflects reality, in terms of where each parking lot and parking spots are.

In addition to this, the server contains the algorithm by which a driver is provided with an optimal route to a selected destination. The route calculation and route optimization functions will be applied for two distinct purposes. The first time, these will be used to provide the driver with a reliable route to the selected destination and after that, the two functions will be used, in a dynamic manner, to alter the route and change the end stop with a parking spot that is or will be available by the time the driver arrives. The map and the methods for providing the route rely on infrastructures and software that have already been created by Google Maps as well as, by the Danish organisation, Septima.

The last function that our application is meant to achieve is to have an integrated billing process. This could be achieved easily, because all that the application is required to do is connect the driver with the billing system of the owner of the respective car park. On the other hand, as stated in "Table 2: Risk Analysis after data generation", this step represents a potential risk because there is no way to know the manner in which those systems function in the back-end. In order to fully mitigate this risk, our application needs to have access to the programming interfaces of the established billing systems. This can be achieved by creating a partnership or an agreement with those companies.



The second figure illustrates the process that a user is required to undertake in order to successfully utilise the app. One of the main requirements for the application is to be as black-boxed as possible. Users do not need to know about the internal working of the app. As a result, the user will have a number of interactions with the app that is as small as possible.

In such a way, the design of the app will influence the manner in which drivers utilise it. the only aspects that are needed from the driver is the final address, type of parking and payment method.

The first step for a driver is to select a destination and, before they start driving, to choose whether they would like free or paid parking or perhaps, no parking at all. Next the driver will be provided by the app with a route and possible parking spots at the end. At this point, they have to either confirm the route and start driving or make changes to the route and repeat step 2. The next step will be achieved by the application without the need for the user's involvement. While a car is getting close to the destination, depending on the user input and on their preferences, the application will be capable of altering the route such as the endpoint will be an available parking spot. The last step, once the driver has reached the destination and has parked the car, the user is required to register the car as parked on the map. This step will be achieved automatically through the user's smartphone location tracking sensors and the position will be cross referenced with the one one the Map in order to guarantee that a specific parking spot has been taken.

Illustration of the problem domain using Rich pictures

Document title: **Illustration of the problem domain using Rich pictures**

Document Ref:

Version No: 3.0

Document Status: Final

Date: 09/12/2017

Author: Roman Novosad

Owner: Roman Novosad

1. Purpose of Document

1.1 Overview

THis document depicts the problem domain within the area of parking in copenhagen. They will aid in developing a common understanding of important aspects of a problem and application domain.

1.2 Distribution List

This document is applicable for the whole development team.

1.2.1 For Review

Name & Role	reason	Date sent	status

Roman Product owner	Problem domain will be reviewed by the product owner	5/12/17	Final

1.2.2 For Information

Name & Role / Department
Scrum Team

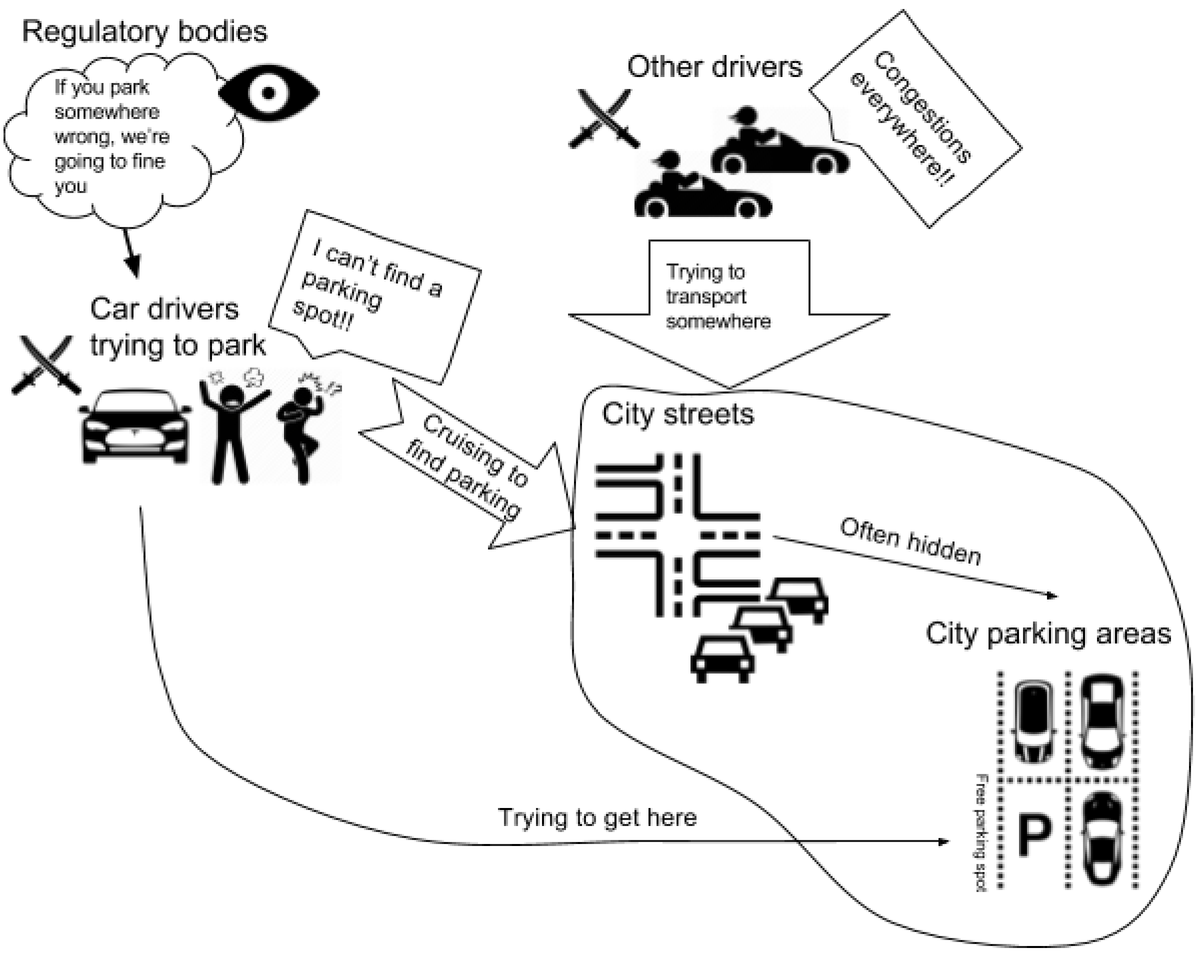
Already in 1981, Peter Checkland carried out a research with his colleagues at Lancaster University, where he discovered the high value that rich pictures are adding to the proper explanation of the problem domain. In his later works, he outlined, that *"[a] picture is a good way to show relationships; in fact it is a much better medium for that purpose than linear prose" (Checkland & Poulter, 2006).*

The Rich picture is very useful after conducting initial interviews involving the problem description by the actors from the examined setting, since this way we can display their understanding of the situation graphically. These vivid expressions will then be used to create a holistic understanding on how the actors see the problem and to get feedback on the picture to improve it*(Checkland & Poulter, 2006).* In the following rich pictures, we will focus on the problematic aspects (Iversen, Mathiassen & Nielsen, 2004) of finding an appropriate parking spot in Copenhagen.

Rich picture legend

⇨	Action
↘	Relationship
🗨 💭	Expression bubbles
🚗	Drivers in need of a parking spot (angry)
⚔	Conflict
👁	Regulatory bodies overlooking proper parking behaviour
🚗🚗🚗	Congestions
╬	Streets
P	Parking area with an available parking spot
📱	Mobile app
🚗	Drivers not in need of a parking spot

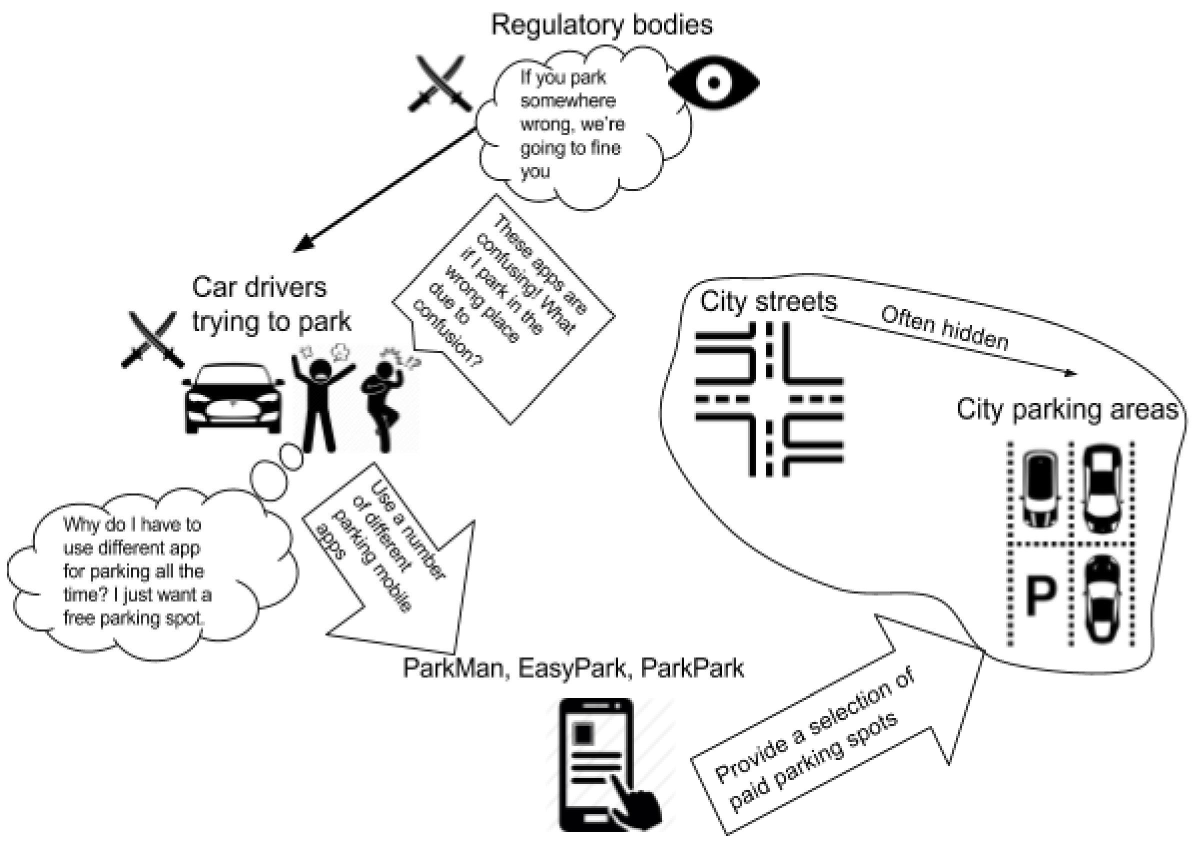
Rich picture 1 - Cruising



Our first Rich picture explains the following situation : There are some drivers wanting to get to their destination with need to find parking there. For an arbitrary reason, they have chosen to find free parking spots manually (not using any of the mobile solutions). They can be looking for the parking for various reasons - i.e. going to school, visiting a friend or shopping. (Interview A) As we depicted above, the parking spots are often "hidden" in the city streets, and seeking an available parking spot manually is often challenging. This significantly contributes to the congestions in the city, further escalating to the bad mood of the drivers in need of parking spot and other drivers present in the streets in question (often emerging into conflict between these two). Situation gets worse during peak hours, and reoccurs mainly on weekdays.

Apart of this, there are two other concerns of the drivers trying to find parking. First of them being price - the demand of the majority of the drivers is to be able to find as cheap parking as possible, sometimes for the expense of the parking spot being further from the destination. (Interview B) Second problem is the concern of parking in the wrong place, while fines are being high, there is an understandable worry about finding a parking spot, and despite paying for the parking being fined by the appropriate operators of the paid parking areas. (Interview A)

Rich picture 2 - Using a mobile app



Regulatory bodies
If you park somewhere wrong, we're going to fine you
These apps are confusing! What if I park in the wrong place due to confusion?
Car drivers trying to park
Why do I have to use different app for parking all the time? I just want a free parking spot.
Use a number of different parking mobile apps
ParkMan, EasyPark, ParkPark
Provide a selection of paid parking spots
City streets
Often hidden
City parking areas
P

In our second rich picture, the drivers wanting to get to the available parking spot are choosing one of already available parking applications for smartphones. As expected, this is not contributing to the bad driving situation as much as in the above mentioned scenario, but there are other issues impacting the drivers. Their dissatisfaction is fueled mainly by larger selection of the apps for parking and their limited usability (Interview A). Problematic situations arises when a driver is not able to find a parking spot at the place of arrival. Currently, there is no system that would have ability to monitor availability of parking spaces in Copenhagen and navigate drivers to an available parking spot and therefore it is possible, that these mobile solutions are not going to help (Interview A).

Similarly to the first scenario, the user is risking being fined for an inappropriate parking (as a result of mobile app not showing the exact available parking spot) (Interview A). Apart of this, other concern of drivers using this solution is being price - the demand of the majority of the drivers is to be able to find as cheap parking as possible, sometimes for the expense of the parking spot being further from the destination (Interview B).

Scenarios

Document title: **Scenarios**

Document Ref:

Version No: 3.0

Document Status: Final

Date: 09/12/2017

Author: Emil Krogsgaard

Owner: Roman Novosad

1. Purpose of Document

1.1 Overview

This document seeks to explore different usage scenarios as seen from the users point of view.

1.2 Distribution List

This document is applicable for the whole development team.

1.2.1 For Review

Name & Role	reason	Date sent	status
Roman Product owner	Usage scenarios will be reviewed by the product owner.	5/12/17	Final

1.2.2 For Information

Name & Role / Department
Scrum Team

A usage scenario describes a real-world example of how one or more people, or organizations interact with a system or a software piece. It is used to describe the steps, events, and actions which occur during the interaction. We used two types of scenarios: the traffic jam section and reservation / pay as you go system. The first one consists of a very detailed version, indicating exactly how someone works with the user interface; a second type consists of a high-level description of the actions and events within the apps interface but not indicating how they are performed.

The structure for our scenarios are based on the following model:

§ **The system under discussion:** an easier and faster parking application;

§ **Primary Actor:** application user/driver;

§ **Goal:** faster parking process for the user;

§ **Conditions/assumption:** every user has JustPark application;

§ **Outcome:** finding the parking spot easier

Therefore, we have created two scenarios based on the model: a detailed version of the process and the interaction between the user and the application process.

Traffic jam

Jesper is living 30 km away from his current office. Every morning he has to commute and due to traffic jam and the lack of parking spots the searching process becomes stressful. Moreover, his company does not provide parking facilities. By using JustPark the whole scenario changes.

First of all, the application will require to access the localization and afterwards the user can type the destination he will be travelling to. Thereafter, the app functionality would be as a GPS. When Jesper reaches a 500 meters' distance from the destination, JustPark will provide the nearest available parking spot at the destination. Another option is if Jesper finds a parking spot and parks his car without before starting the app. When using the application, he is informed whether the spot is free or not. The same situation would be applied if Jesper doesn't want to use the GPS function of the app. In this case, he could turn on the app when he reaches the destination and choose the "find near parking spot" and he would be redirected to the nearest available spot. The last step of the parking is to press "start" when the car is parked and "stop" when he will be leaving the spot.

Reservation process / pay as you go system

Jette is always receiving fines for not extending the time spent in the parking lots. By using JustPark the extra steps are avoided and highly simplified:

I. Switch on the app.

II. Choose destination. Jette is guided through GPS. A parking spot is displayed when the user is close to destination. Press accept spot. Park the car.

III. Press “START parking”. Timer will start.

IV. Amount of time passed. Jette returns to her car.

V. Open the app. Press “STOP parking”. A total time and price will be displayed and the corresponding amount of money will be withdrawn from his credit card.

VI. A receipt is being sent via email/SMS.

Actor descriptions

Actors are represented by the future users of the system. Actors model the user's perspective of the system and they are located outside the system; therefore, in order to depict actors, it is important to define the boundaries between actors and the system(Papajorgji, 2016). In our project we could identify only one actor that is not being part of the system. This may be defined as the first type of actors, which is the most common type, an application user. However, we created a technical use case diagram where we displayed the interaction between actors that are not part of the system. The second category includes IT systems interacting with the application system, and human actors that are being part of JustPark processes.

In this chapter, we decided to present only the most relevant type of actor, the application user, and further describe their characteristics and a sample example of their interaction with JustPark app.

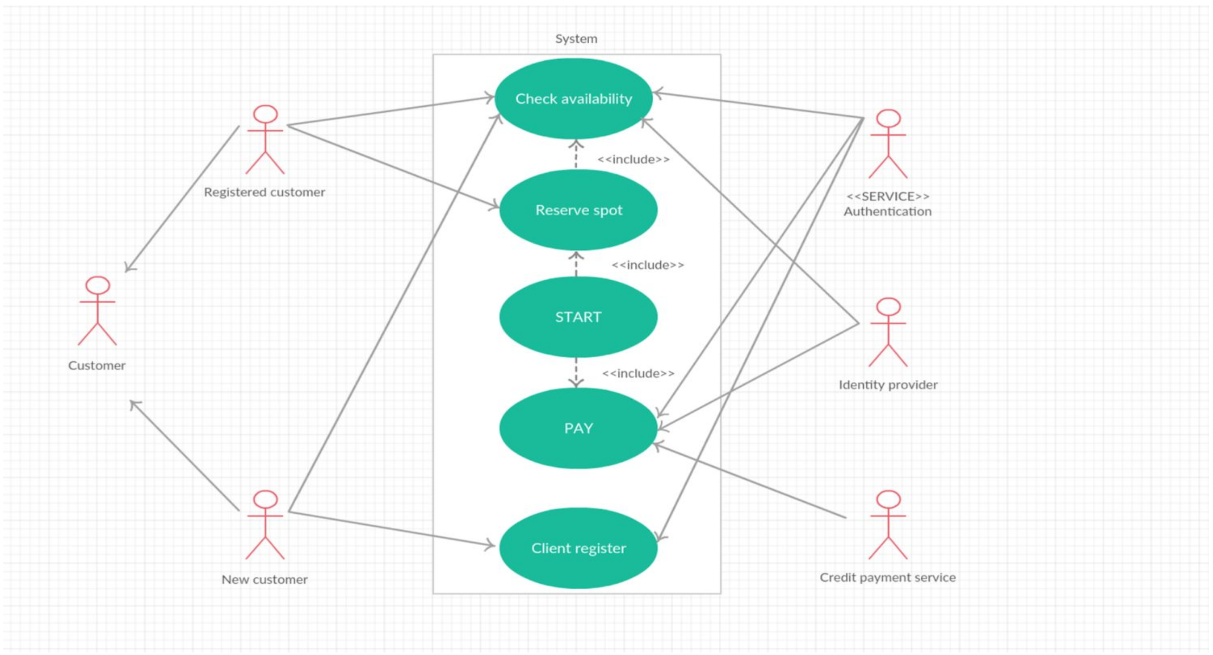
Application user / drivers

Purpose: A person that owns a car. The user wants to find in the quickest way possible a parking spot.

Characteristic: The system’s users include same type of accounts.

Examples: Account owner A travels to Copenhagen Centrum. The user is concerned how much time it would take to find a parking spot. He switches on JustPark, selects destination 500m before destination is reached all available parking spots will be displayed.

Use case diagram



In definition, an actor is always outside the system and anything that is part of the system is a secondary actor (Papajorgji, 2016). However, we decided to be more precise when it comes to user interaction and the 'backend' that makes the system work. First use case diagram describes all the actors involved in the JustPark process.

1. A customer may have an account or not, therefore we divided a customer into two categories: registered user and new customer. If the customer doesn't have an account, a first

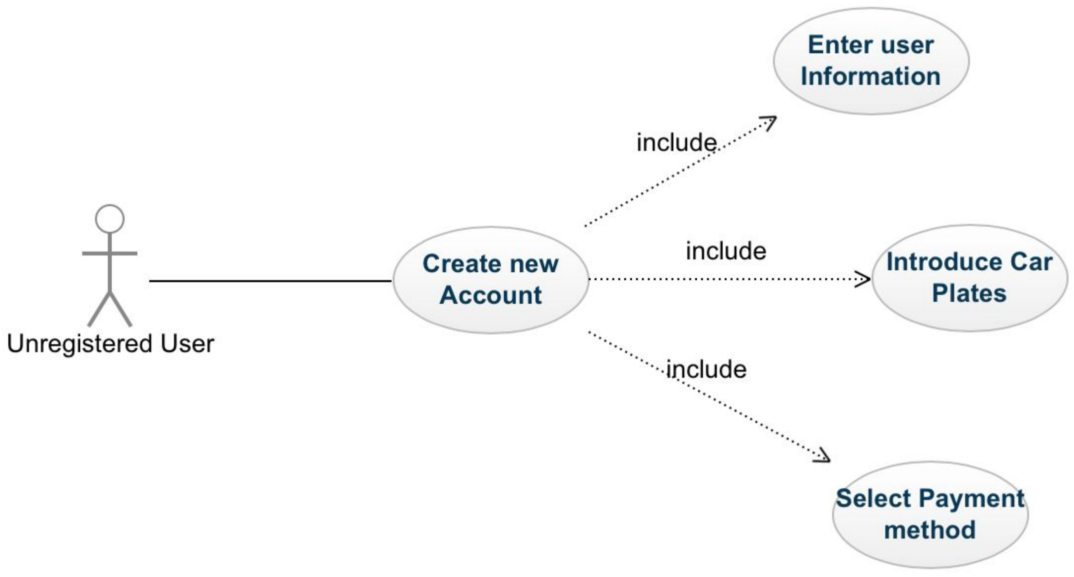
step is the registration process. This step has to be verified and an internal actor may be involved. First of all, the Service authentication actor represents the person in charge of verifying customer's authentication data and approving the application. In general, this is an automated process, but some scenarios may require human interaction.

2. After registration, the user has to select a destination. A GPS tracker would be activated and when the user reaches a 500m distance to destination he/she would be asked to check and select from the available parking spots. The reservation starts by pressing a start button and finishes when the stop button is being triggered.

3. The third step is the payment process, where the payment and receipt are automatically generated. Here we list three actors getting involved: The authentication actor, responsible for the user's authenticity, the identity provider is in charge of the issues regarding parking spots. If there were any spots marked as free and they weren't or if the user got a fine for parking in that area etc. The third actor is the payment service that has to approve and make sure the payments were done correctly.

Compared to the first technical use case, the second diagram follows the user step by step. We use it to display the complexity of the app and as it can be noticed, the app has a lower complexity than most of the existing parking apps.

1. Sign In / Log In
2. Select destination
3. Check available spots
4. parking area displayed / Park
5. Start parking timer
6. Stop parking timer
7. Review availability and service
8. Payment generated automatically

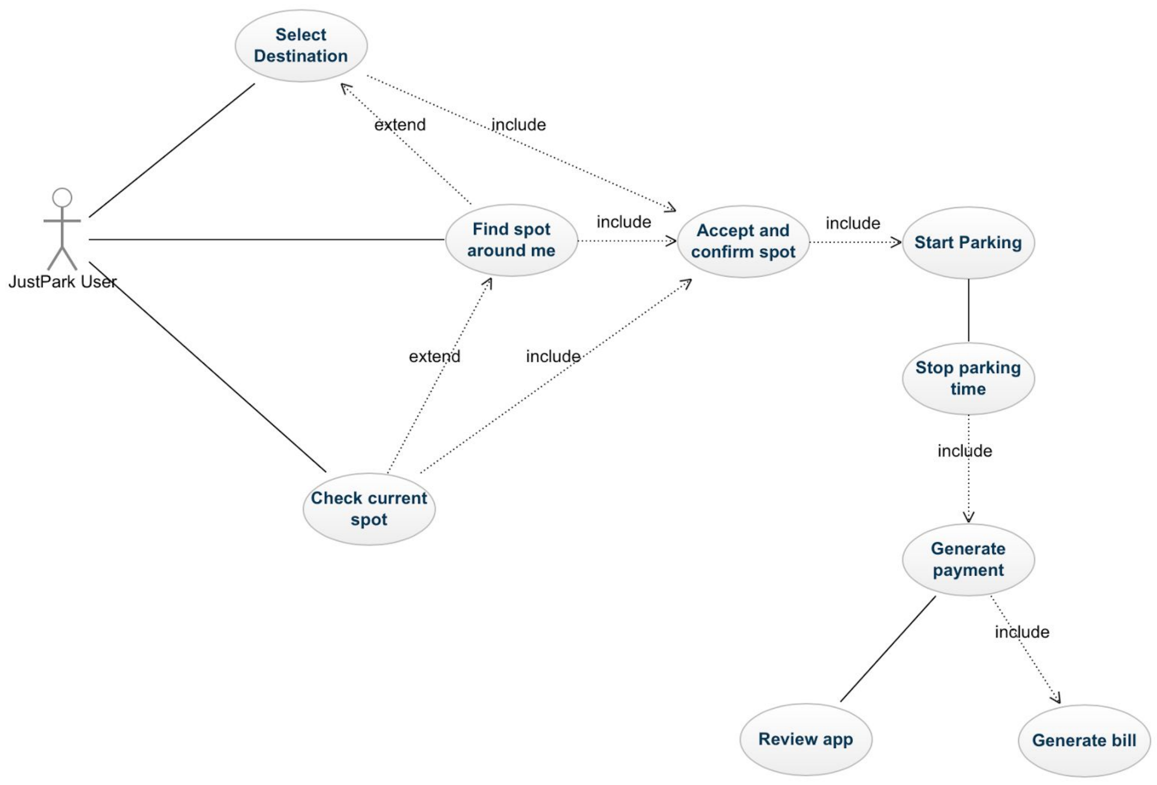


Use case description

USE CASE NAME	Create account
USE CASE ID	1
BRIEF DESCRIPTION	Register as new User
ACTORS	Unregistered User
PRECONDITIONS	The app is already installed

MAIN FLOWS	1. The use case starts when the user select Registration. 2. The system requires user details 2.1 Car plates 2.2 Payment method 2.3 Personal Information (Name, Address, phone number)
POST CONDITION	The account is now created
ALTERNATIVE	none

General Use case

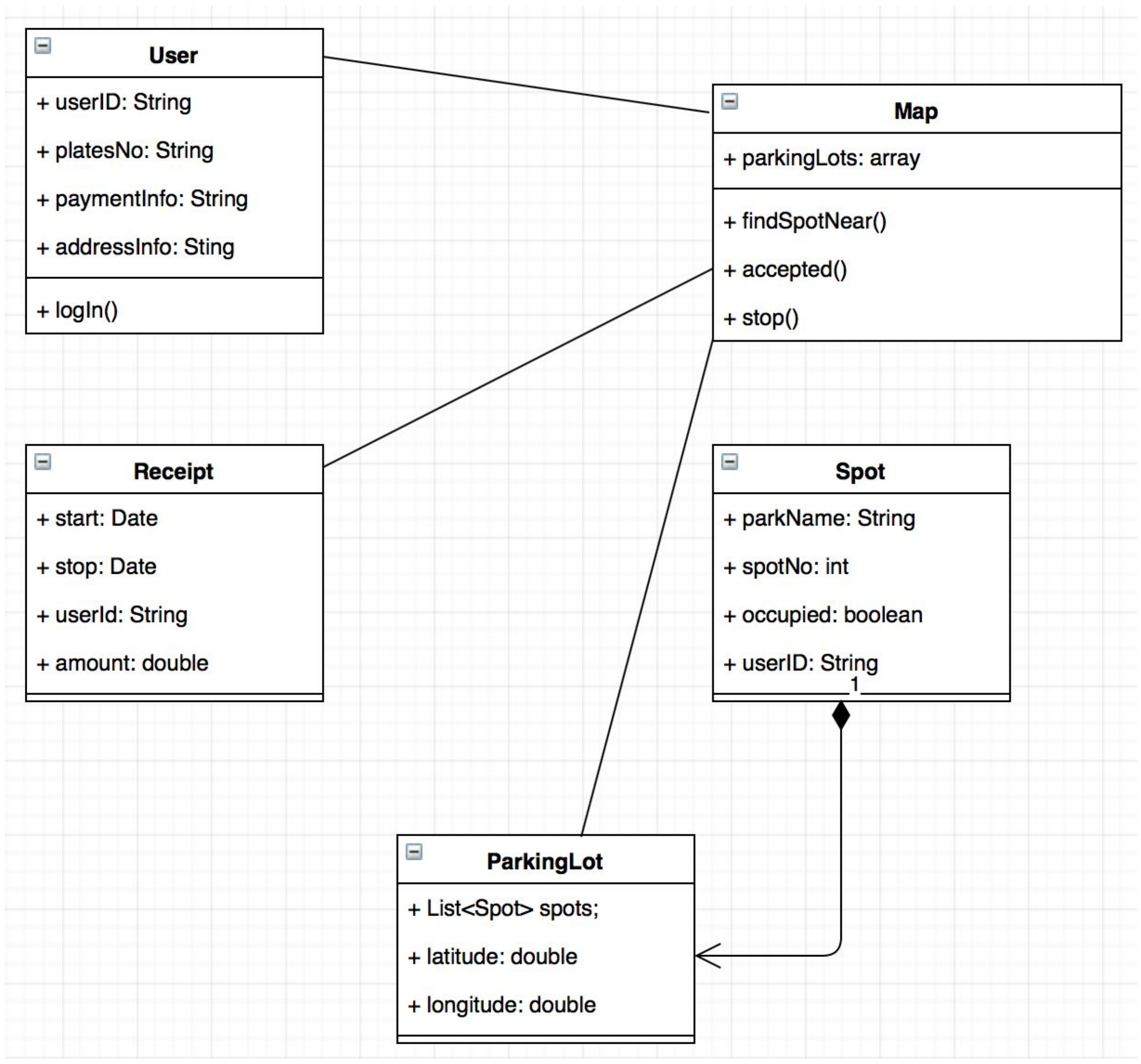


Use case description #2

USE CASE NAME	General Use Case / Search for parking spot
USE CASE ID	2
BRIEF DESCRIPTION	The user search for an address / for a spot
ACTORS	JustPark User
PRECONDITIONS	USER ALREADY INSTALLED THE APP

MAIN FLOWS	1. The use case starts when the user selects a destination or asks for parking spot. 2. The system provides the user GPS guidance until the parking spot is reached. 3. The user presses the “Start” button to start the parking. 4. Before the leaving point, the user triggers the “Stop” button. 5. A review box is being displayed. 6. The user receives the bill via SMS/mail/app.
POST CONDITION	None.
ALTERNATIVE	The user receives a fine for not paying the parking.

Class Diagram



UML class diagram

Disclaimer: The above mentioned UML diagram represents only our system and not any third party actors. (established billing systems)

The UML class diagram above describes classes, their attributes and operations as well as relations between them. The more detailed version of the class diagram can be found in the appendix, where each method contains the corresponding parameters. (Appendix D)

The User class contains four attributes and one method. By using logIn() method the Map class is being instantiated. However, this is not necessary a composition, considering the Map can be accessed even without having to create a user.

The Map class is constituted by three methods and one attribute. The three methods are based on the user input. FindSpotNear() is displaying the nearest free parking spot. The stop() method is used to create the receipt, using the timer for calculating the price.

The ParkingLot class contains four attributes and no methods. ParkingLot class instantiated by the Spot class, considering ParkingLot contains all the spots. Therefore, the ParkingLot class is in composition with Spot, by being part of it. (a parking lot cannot exist without the spots) The last class, Receipt has four attributes and no methods. This class is meant to contain the information needed to generate receipts.

Interface Design using Mock-ups

Document title: Interface Design of the JustPark parking solution

Document Ref:

Version No: 5.0

Document Status: Final

Date: 13/12/2017

Author: Roman Novosad

Owner: Roman Novosad

1. Purpose of Document

1.1 Overview

The purpose of this document is to depict design requirements for the JustPark solution.

1.2 Distribution List

Reviewed by product owner.

1.2.1 For Review

Name & Role	reason	Date sent	status
Product Owner	Specify the requirements		

1.2.2 For Information

Name & Role / Department
Development team

According to the aforementioned scenarios, we have constructed a design mockup depicting a user journey when making a parking place reservation. (Image 1) Interactive design was created using Invision app (full link to the interactive interface is available in the footnote)[1]. We have rapidly prototyped the mockups in order to collect user feedback needed for future development. During the mockup workshop, we found a number of aspects that we wanted to further incorporate or modify in the design. The initial design is available in the Appendix E.

In the final design mockups displayed below, the user is led through a process of booking a parking spot. Unlike in the design we originally developed, user is not booking a parking spot in advance, but rather traveling to a closer proximity of the destination and only then is

[1] https://invis.io/QRENFTT7C#/265980188_Image_005

guided to the nearest available parking spot. This enables more dynamic designation of the parking spots and mitigates an erroneous behaviour from the user side.

Registered user getting an available parking spot (Image 1)

After the user is presented with the login screen, he/she can input login details and log into their app. (this goes for already registered user) After that, the user will have to select a "Map" option on the welcome screen. From here, user is able to navigate in the map and choose parking destination. As aforementioned, there is no particular parking spot that a user would be navigated to. Instead, there is an area, and closest parking spot is selected upon arrival. In this stage, the app works like the GPS in for instance Google Maps. When a user parks in the desired parking spot, the next screen shows up and user is prompted to register for parking. (Start button) The parking time is counted until user's departure when the user has to check out (Stop button).

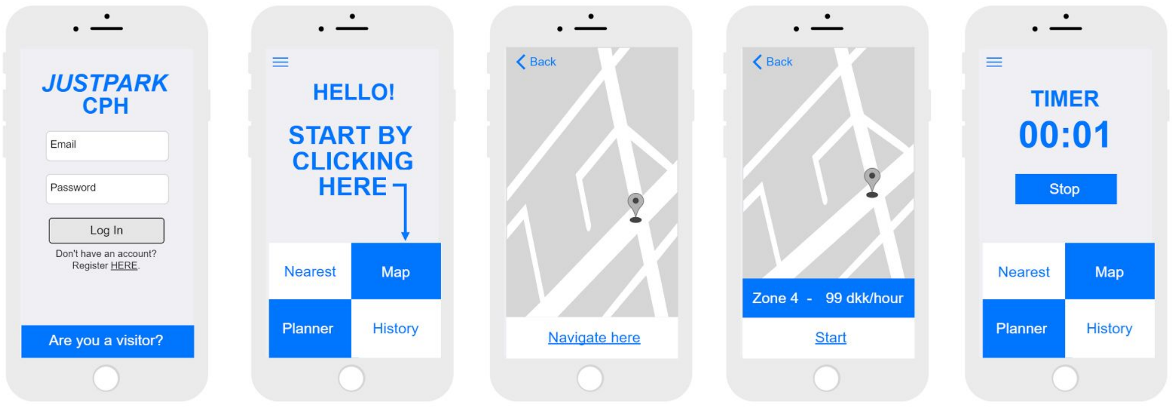


Image 1

Registered user ending a parking session (Image 2)

When a user is already checked in the parking spot, at some point he/she will have to leave the spot. In this case, the app will display a "Stop" button below the timer for elapsed parking time.

Immediately after launching the app, user will be presented with this timer. If the user wishes to leave the parking place, he/she can simply end parking by pressing "Stop" button and will be presented with a final calculation of the price. Below the calculation, user will be able to choose a preferred payment method (pre-set earlier by the user) or choose new payment method in case none of the available is desired. As the user presses “Pay” button, they the transaction will be made via integrated payment gate and upon a successful transaction user will be shown a simple thank-you message.

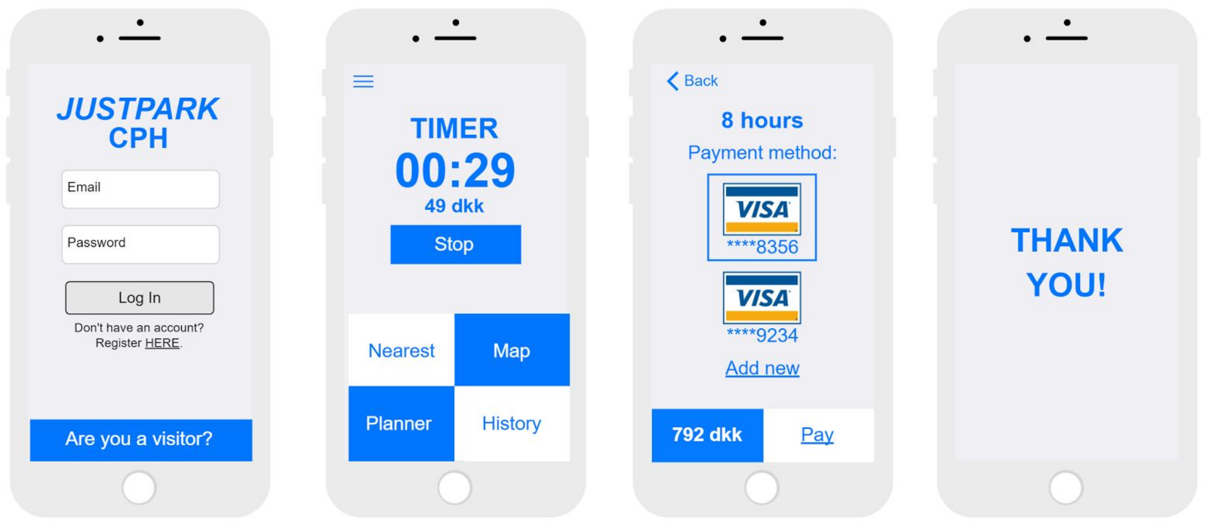


Image 2

When all the necessary mock-ups are constructed and refined, we can create development tasks based on them. These tasks have to be as exact as possible, however with the development framework we have chosen, we cannot exclude further iterations on them.

Documenting Project Requirements

Goals

- Simplification of parking for drivers and residents in Copenhagen
- Improvement of waiting time for availability of the parking spot
- Decreasing number of illegally parked vehicles in the city
- Adding ease of use to the process of finding and placing user’s vehicle in a Copenhagen parking place

Background and strategic fit

- Mitigation of the congestion creation in the city, improving driver user experience, improving overall image of the city

Requirements

#	Title	User story	Priority	Notes
1	User login	As a user, I want a page where I can use my login information and log in to the app.	Critical	
2	User new account registration	As a user, I want to have an easy and convenient way of creating a new profile so I can start using the parking immediately.	Critical	
3	Guest parking reservation	As a visitor of the city, I want to be able to pay for parking no matter whether I have created an account or not.	Major	
4	Main timer	As a user, I want to have a convenient way of checking how much I have left from my paid parking time.	Critical	
5	Map screen	As a user, I want to be able to browse free parking locations on the map, so that I can easily choose where I want to park.	Major	Google maps API integration
6	Booking in advance - time	As a user, I want to have an option to book whatever time and duration for	Major	

	picker	parking I want.		
7	Payment - price calculation	As a user, I want to have a clear information about the amount of money I will be charged.	Minor	
8	Payment method selection	As a user, I want to be able to add and easily pick from multiple payment methods.	Minor	Forwarding of the payments to the relevant parking providers

Document convention

Below, we have created a document convention for all documents that can be classified as a configuration item. This convention is to be added in the beginning of these documents to establish some necessary metadata for the document.

Document title:

Document Ref:

Version No: x.x

Document Status: Draft / Final

Date: dd/mm/yyyy

Author: The person responsible for its creation

Owner: The person accountable for its content

1. Purpose of Document

1.1 Overview

The purpose of this document is to...

Note: This document is subject to change and will be revised and approved per the document change control process.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

This section specifies who should review and approve the document in questions.

Name & Role	reason	Date sent	status

1.2.2 For Information

Name & Role / Department

Configuration management plan[2]

Document title: **Configuration management plan**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 12/11/2017

Author: Emil Krogsgaard

Owner: Silviu (Scrum Master)

1. Purpose of Document

1.1 Overview

The purpose of this document is to establish a configuration management plan to ensure that the development of the smart parking app for Copenhagen is streamlined.

1.2 Distribution List

[2] For inspiration the configuration management plan found on Nasa's website has been used. https://www.nasa.gov/sites/default/files/t2401_-_rev_b.doc

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	Status
Silviu Scrum Master	Owner of the document, needs to cascade information to developers.	13 December 2017	Final
Roman Product owner	Customer needs to be made aware of the management of configuration items.	12 December 2017	Final

1.2.2 For Information

Name & Role / Department
Developers

Introduction

To secure that the configurations related to the product developed are released and managed in a way that makes sense for our approach to development, we have created a configuration management plan that will guide us through releases, changes and documentation.

As previously stated, one of our core software quality aspects is the idea that the final product should be easily maintainable and reliable which means that we must provide a system that is easy to support. A second aspect of maintenance includes the possibility of introducing new features after go-live. Both activities are likely to be performed by support teams or developers not involved with the development aspect of the original product. Thus, when designing our configurations we need to consider how we enable our future colleagues to service the product in the best way possible.

Configuration Management tool

We have decided on using the visual studio Team foundation Server as it is a fully fledged software development tool that can provide aid in many aspects of development. This includes version control, support of agile processes and continuous integration.

Configuration Management Activities

As we are using an agile approach to develop the product we need to adhere to this approach on a documentation level. *"The agile strategy is to defer the creation of all documents as late as possible, creating them just before you need them via a practice called "document late"* ("Best Practices for Agile/Lean Documentation", 2017). As also discussed in this web based article this does not mean that we are "paper free" in the initial phases of the project, but rather that we focus on documenting higher level concepts and requirements.

Configuration Control

When the development of the tool has reached a point where it can be tested by potential stakeholders we will start to adhere to a specific change management process. This process aims to ensure that we have proper control of the configurations and releases that we present to customers. From this point and onwards all changes needed must follow the change management process. However, to eliminate some bureaucracy we will distinguish between standard changes and significant changes. The changes that can be handled as standard changes will be stated in the standard change catalogue.

Procedures for implementing standard changes

All standard changes must be logged in the Visual studio team foundation Server. The request will then be processed by our helpdesk agents, who will help to fulfill the basic amendments that are requested.

Upon creation of the standard change catalogue, a link will be available in this section.

Procedures for processing change requests

All changes not logged in the standard change catalogue must follow the change management process. This process aims to ensure that changes are thoroughly tested and that stakeholders affected by the change are properly informed. Hence, the goal is to minimize risks and adverse effects of implementing changes.

Change Control Board

As we are working with an agile approach changes will be common, hence the change control board should be summoned on a weekly basis. Its members should consist of the lead developer(s) on the change, representatives from the customer side and the scrum master.

The meeting will be held to ensure that all aspects of the change management process is being followed, thus limiting risks and potential impacts of the change in question.

Release process

When releasing a new version of the product there should be some basic information associated with the release. This is to ensure that we keep a proper history of what goes into the releases and to keep stakeholders informed on what progress is being made to the product.

Thus all releases should produce the following documentation:

A Basic description of what is in the release, if only specific parts of the product is impacted it should state what parts of the product it relates to. If the new release fixes any bugs or known errors this should be listed.

Software testing

Document title: **Software quality assurance plan**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 25/11/2017

Author: Emil Krogsgaard

Owner: Emil Krogsgaard

1. Purpose of Document

1.1 Overview

The purpose of this document is to give its audience an introduction into the design of the quality assurance plan for the development of the JustPark app.

Note: This document is subject to change and will be revised and approved per the document change control process.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Roman Product owner	Customer needs to be made aware of the management of configuration items.		
Silviu Scrum Master	Owner of the document, needs to cascade information to developers.		

1.2.2 For Information

Name & Role / Department
Scrum Team

Software quality

This section is focused on discussing how we ensure that quality is incorporated into the software delivered to the end users. As such, it is focused on how our practices will ensure that quality is built into the final product. These practices are created and conducted in respect to the scrum methodology and the general software qualities that have been stated in earlier sections.

In Ian Sommerville's book Software Engineering he describes Quality management in agile development as "*informal rather than document-based. It relies on establishing a quality culture, where all team members feel responsible for software quality and take actions to ensure that quality is maintained."* (Sommerville, 2016). We are mostly incorporating this informal approach into our development cycle, although in later sections we do have specific test cases prepared. This said, the main focus on our quality approach is to incorporate it into the daily work culture as is the general style in scrum. This means that testing is the responsibility of the developer writing the specific code. To ensure that the code is generally well written it will be discussed in the daily stand up meetings when the developers have the chance to catch up on each other's work and create alignment for ongoing developments.

In the Scrum Manifesto emphasis is put on reflection on quality during the Sprint retrospectives:

"*During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of "Done", if appropriate and not in conflict with product or organizational standards."* (Schwaber & Sutherland, 2014)

Thus, improvements in work practices that will aid in the quality assurance process can be reflected upon after each sprint helping the work and team to mature. Ultimately this also means that a complete plan of how quality will be assured can not be shared as it is an ever evolving effort in which perfection will never be reached.

Not all our processes adhere to the fundamental principles in Scrum. As stated in the Configuration management plan, we do have a formalized version of the practice *"Check before check-in"* (Sommerville, 2016). This is in essence a change management process, which introduces a specific way of implementing changes to verify that the release will not have negative impacts on the current build. Though this adds some bureaucracy to the development process, this will help us achieve our top prioritised software quality: Reliability.

Except from the change management process we want our developers to feel free from referring to and updating page after page of formal quality management documentation. Instead we want to encourage a culture of ownership work. Thus, the time that would otherwise be spent on developing documents, should be used for be verification of own work. Additionally team members can assist each other with problems and use the daily standup meetings for discussing granter matters.

Dynamic Test Specification

Document title: **Dynamic Test Specification**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 26/11/2017

Author: Roman Novosad

Owner: Roman Novosad

1. Purpose of Document

1.1 Overview

Purpose of this document is to outline information about the dynamic testing incorporated into the software development methodology chosen to carry out the JustPark project. It was chosen to do unit testing in the earlier stages of the project, system tests, regression testing and practices associated with them to ensure maximum possible quality. Below, we will describe what needs to be done in order to successfully carry out this dynamic testing and ensure a quality of produced software.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Roman Product owner	Customer needs to be made aware of the management of configuration items.	19 October 2017	Final
Silviu Scrum Master	Owner of the document, needs to cascade information to developers.	20 October 2017	Final

1.2.2 For Information

Name & Role / Department
Scrum Team

Software testing

By having a look at the glossary at the beginning of the paper, one might notice that we have defined "Done" followingly: *"A task is done, when it has been **quality checked** and is finished according to the specification and relevant user stories defined by product owner."* (Schwaber & Sutherland, 2013)

We used a term "quality checked", which means that appropriate testing practices were executed on the task in question. In the next section, we will explain relevant processes invoked by this definition.

To start with, as we have chosen to use the Scrum framework for our project we can turn to The Scrum Guide written by Ken Schwaber and Jeff Sutherland (2013, p. 18). That explicitly states: *"Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together."*

There is a number of ways how to segment the methods of testing in the software development project. Firstly, there are three distinct stages of testing on a chronological scale: development testing, release testing and user testing. (Sommerville, 2016, p. 232) Secondly, there are two main approaches of testing the developed software: black box and white box testing. Latter of these is done during the development testing and especially in Scrum, it's generally automated (Collins et. al., 2012).

When using Scrum methodology, the team should deliver a potentially shippable iteration at the end of every sprint. This means that the software should be usable, therefore it should be tested by the (customarily cross-functional) team. (Schwaber and Sutherland, 2013) On the other side, what we might consider a black box testing is a review of the iteration by product owner after the sprint is finished. He or she could accept the iteration, or return it back to development.

In our case case, we intend to employ the approach suggested by Sommerville (2016, 232-247), where developers will be responsible for white box tests - as they are the ones knowing the code, they should be able to perform these tests most effectively. In our case, developers will perform the black box testing as well, but this test should by definition not performed/written by the ones developing tested feature. In the next part, we have picked a number of examples of scenarios for white and black box testing for our project.

Test cases

Black box testing

Scenario testing

Test case #1: Login screen

1. User launches the app.

2. At the login screen, user enters login information.

Expected: If the information is valid, user will be sent to a welcome screen If the information is invalid, user will be prompted with a pop-up: “Username or password is invalid.”.

Test case #2: Payment gate

1. User opens the app.
2. User goes through the process of getting to the parking spot.
3. After finishing parking, user is prompted to pay for the parking.
4. User will pick a payment method from the list on the screen. (always containing at least the one method set-up upon the account creation)

Expected: If the payment information is invalid, user will be prompted with a message: “You have entered incorrect payment information.” If the payment information is valid, but there are not sufficient funds under selected payment method, user will be prompted with a message: “Insufficient funds. Please select another payment option.” If all the aforementioned requirements are fulfilled, required amount of money is deducted from the selected payment method and use is shown the next screen.

Test case #3: Total price calculation

1. User opens the app.
2. User goes through the process of getting to the parking spot.
3. After getting to the parking spot, user is able to see in which zone the parking spot is and corresponding hourly rate for this zone - 99 dkk/hour.
4. User checks in and stays at the parking spot for 8 hours from the current time.

Expected: The total price will be calculated with a minute precision. If user wants to park for 8 hours, the price will be [8 hours]*[99 dkk/hour] = 798 dkk.

White box testing

Branch coverage

Two cases below are designed to cover all possible scenarios of user login and registration. (Flowchart 1 and 2) Every option of the user input has to be tested. This also complies with the software qualities we outlined earlier. Furthermore, developers will be required to ensure each scenario outlined in cases below is secured. Even though white box testing is remarkably demanding in regard to the individual capabilities of a tester, it’s important to incorporate it in the project. White box testing, when executed correctly, will yield deeper understanding of the written code, opportunity for optimization and others. (Somerville, 2015, p. 236)

Document title: **Analyzing Architectural Patterns to Support Quality Attributes**

Document Ref:

Version No: 1.0

Document Status: Final

Date: 12/12/2017

Author: Emil Krogsgaard

Owner: Emil krogsgaard

1. Purpose of Document

1.1 Overview

The purpose of this document is to discuss how the architectural pattern of MVC can support some of the software qualities associated with our final product.

1.2 Distribution List

This document is to be communicated for the following stakeholders for review and for information.

1.2.1 For Review

Name & Role	reason	Date sent	status
Roman Product owner	Customer needs to be made aware of the architecture being used.	13 December 2017	
Silviu Scrum Master	Owner of the document, needs to cascade information to developers.	13 December 2017	Final

1.2.2 For Information

Name & Role / Department
Scrum Team

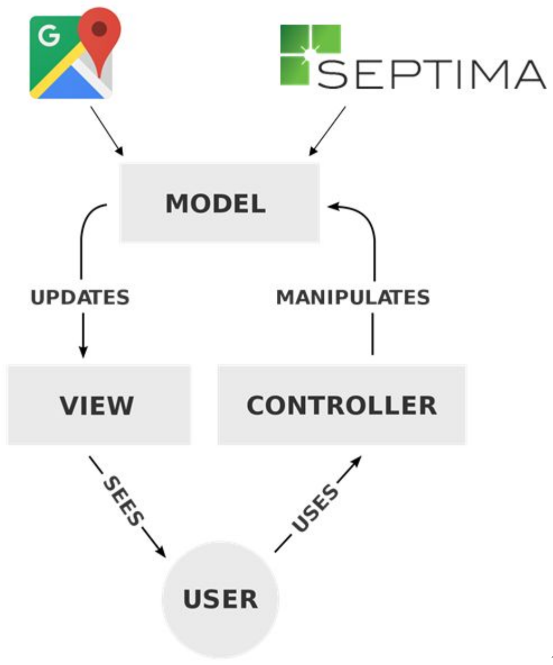
Analyzing Architectural Patterns to Support Quality Attributes

In this part of the report we will discuss how the architectural pattern of model view controller can support the software qualities that we selected in a previous section.

Model view controller

The model view controller, also referred to as the MVC, is a specific way of organizing the code that will make up the final system. This helps creating structure and make it easier for multiple developers to work in sync as it makes task separation easier.

The name of model comes from the "blocks" that the code is organized into: model, view and controller. Each block should be used for carrying out a specific purpose. The model can be described as the brain of the code consisting of the logic and behaviour, thus it should contain all algorithms responsible for the manipulation of data that is used in the background of the programme. The view on the other hand is concerned with representing the information important to the user. At the end the controller is where all the user's input method should be stored. On figure 3 you can see a visualized version of the model. In this model Google maps and Septima has been included to illustrate how external applications can be used to provide data to our system. Note that Google maps and Septima do not correspond to all the external entities that our application will communicate with.



[3]

[3] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#/media/File:MVC-Process.svg

Figure 3

Architectural fit for achieving software quality

One of our core software qualities is maintainability. This quality focuses on creating a product that is simultaneously easy to support and easy to expand upon. As MVC has a strong focus on separating different functions of the code into specific classes it should, when done correctly, enable the developer to create code that is easy to understand and support. Furthermore, as the code is divided it becomes increasingly simple for multiple developers to work on it simultaneously. However, the separation of the controller and the view often complicates matters when updates needs to be made, as one block can seldom be updated without also updating the other block.

Usability is regarded as our second most important software quality. This is supported quite nicely using MVC as the logic of the software is completely separated from the front end that the user sees. This means that it is easy to iterate over the design continuously without causing any problems to the more fragile backend architecture.

Based on the very limited research we have done on MVC and other architectural patterns it is difficult to state whether this pattern is the most suitable pattern for achieving the software qualities that we have defined in our project. This said the positive aspects of using MVC appears to outweigh its negative forces, thus we believe that it would be a feasible pattern to use for our project.

Reflection

Reflection on using Scrum framework

When we started this project we felt slightly confused and possibly overwhelmed at the task at hand. As we did not possess previous experience with execution of software projects under scrutiny of software development frameworks, we felt that the mountain to climb seemed disturbingly steep. After learning about the various frameworks, we quickly decided on using Scrum as our approach to solving the task at hand. There was a need to start using elements of the Scrum framework already in the beginning of the project, without having enough time to research this project thoroughly. One member of our team took the position as a scrum master, but he was simply not able to gain enough information to coach the scrum team in the early stages of the project.

As we gained more understanding of the methodology, we have become more confident in the execution of the scrum practices and could focus on improving this execution to benefit the given project in the best possible way. That being said, we couldn’t avoid some hiccups along the way. Here we discuss some of them.

1. When we originally decided on using scrum as our development method, we assigned roles to everyone in the team. This included assigning the roles of product owner and scrum master to members within our team. Some of the members did not have the necessary background to be effective in their roles and as such the roles were taken too lightly by the remaining team members causing us all to have more or less generic developer roles. This resulted in a lot of uncertainty in terms of how the product should be developed, which created long passionate discussions on what technologies to use, what was in scope, how to do billing etc. - As the discussion were not facilitated in any way they often did not lead to a specific solution causing us to reenage in the argument the following week. Since the Agile development methodology relies on coordination, communication and collaboration, it can be felt on the quality of the deliverables and time management when these tasks are not performed optimally. Ultimately we feel that that even though we had a scrum master

assigned on paper, in reality we felt as if we as a team were never properly managed causing us to spend too much time on determining the path to follow.

2. Second problem we experienced was tied to a lack of effectiveness of communication inside our team. The product owner had a particular idea of how a project should be carried out, but there was a number of instances when the idea of other team members was not identical. Oftentimes, development team did not have the full overview of what the final product should look like, and when a final implementation was made, it didn't match required specification.
3. In addition to this, we felt like the practices that scrum requires were initially forced and were limiting instead of aiding us. This might have been due to lack of experience, or due to adhering to scrum practices too strictly instead of fitting them onto the team.

In a more realistic scenario we would have had a more experienced coach, a product owner closer to the customer, and larger number of developers. This would have enabled a greater certainty of the backlog items, and in general more alignment in terms of ideas than we saw when we created the project.

Software qualities

As we got further into the project and got increasingly aware of the task at hand we got increasingly uncertain to whether we had prioritised our software qualities correctly. There was an increased belief in the group that more attention should be put into ensuring that security of the final product should be our top focus when it comes to qualities, however as we originally rated it as our least prioritized quality we decided to keep it at that to reflect our assumptions in the beginning phases of the project. It was generally difficult to decide on prioritizing the software qualities. And as we went through our project we often felt the need to pivot our original assumptions.

Legal aspects

One of the problems we encountered during the project was the legal aspect of our solution. The case description "Smart Parking for Smart Cities", that originated the interest into solving the parking situation in Copenhagen, does not mention any legal requirements.

During the research period of the project, we identified that Copenhagen is divided into five distinct zones each having its own set of regulations and laws that need to be respected. In addition to this, there are only a few organisations that are allowed to operate within the parking industry. Taking into consideration that these companies are competing over proprietary rights for parking lots, the realistic chance of creating a partnership with all those companies and being given access to their API in order to integrate their billing system is slim. In a situation such as this, our solution would be restricted to suggesting parking options only for free parking lots.

Last but not least, we have to address the amount of learnings that resulted from this project. Even though we were unsure and insecure in the beginning, we have picked up on the right practices quickly. Had there been more time and other resources available for the project, we strongly believe that we could actually deliver a valuable solution for the parking situation in Copenhagen.

References

1. Berteig, M. (2015). 9 Agile Estimation Techniques. Retrieved December 14, 2017, from http://www.agileadvice.com/2015/10/13/agilemanagement/9-agile-estimation-techniques/

2. *Best Practices for Agile/Lean Documentation*. (2017). *Agilemodeling.com*. Retrieved 14 December 2017, from http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm

3. Boehm, B. (1986). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, *11*(4), 22-42. http://dx.doi.org/10.1145/12944.12948

4. Boehm, B., & Hansen, W. J. (2000). Spiral Development: Experience, Principles, and Refinements. https://www.sei.cmu.edu/reports/00sr008.pdf

5. Checkland, P., & Poulter, J. (2006). *Learning for action: a short definitive account of soft systems methodology and its use for practitioner, teachers, and students*. Hoboken, NJ: Wiley.

6. Cohn, M. (n.d.). Planning Poker: An Agile Estimating and Planning Technique. Retrieved December 14, 2017, from https://www.mountaingoatsoftware.com/agile/planning-poker

7. Conrad Weisert (2003) "There's No Such Thing as the Waterfall Approach! (and There Never Was), Information Disciplines, Inc., Chicago 8 February, 2003." *Waterfall Methodology: There's No Such Thing!*, www.idinews.com/waterfall.html.

8. Edwards, R., & Holland, J. (2013). *What is qualitative interviewing?* London: Bloomsbury.

9. Idris, M., Leng, Y., Tamil, E., Noor, N., & Razak, Z. (2009). Car Park System: A Review of Smart Parking System and its Technology. *Information Technology Journal*, *8*(2), 101-113. http://dx.doi.org/10.3923/itj.2009.101.113

10. ISO - International Organization for Standardization. (2017, August 16). Retrieved December 14, 2017, from https://www.iso.org/standard/35733.html

11. Iversen, Mathiassen, & Nielsen. (2004). Managing Risk in Software Process Improvement: An Action Research Approach. *MIS Quarterly*, *28*(3), 395. http://dx.doi.org/10.2307/25148645

12. P/S, S. (2017). *Databerigelse*. *Septima*. Retrieved 14 December 2017, from http://www.septima.dk/databerigelse

13. Papajorgji, P. (2016). Software engineering techniques applied to agricultural systems (p. 61). Springer

14. Parking in Copenhagen. (n.d.). Retrieved December 14, 2017, from https://international.kk.dk/artikel/parking-copenhagen

15. Problem domain. (2017). *Definitions*. Retrieved December 14, 2017, from http://www.definitions.net/definition/problem%20domain

16. Pullola, S., Atrey, P. K., & Saddik, A. E. (2007). Towards an Intelligent GPS-Based Vehicle Navigation System for Finding Street Parking Lots. *2007 IEEE International Conference on Signal Processing and Communications*. doi:10.1109/icspc.2007.4728553

17. Schwaber, K., & Sutherland, J. (2013). *The Scrum Guide*. Scrum.Org and ScrumInc. Retrieved from https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

18. Semi-structured Interviews. (n.d.). Retrieved December 14, 2017, from http://www.qualres.org/HomeSemi-3629.html

19. Shoup, D. (2006). Cruising for parking. *Transport Policy*, *13*(6), 479-486. http://dx.doi.org/10.1016/j.tranpol.2006.05.005

20. Sommerville, I. (2016). *Software engineering* (10th ed.).

21. The Scrum Guide. (n.d.). Retrieved December 14, 2017, from https://www.scrum.org/resources/scrum-guide

22. Wang, H., & He, W. (2011). A Reservation-based Smart Parking System. *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. doi:10.1109/infcomw.2011.5928901

23. What is Incremental model- advantages, disadvantages and when to use it? ISTBQ Exam Certification: http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

24. What is Waterfall model- advantages, disadvantages and when to use it? ISTBQ Exam Certification: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

25. Winston W. Royce (1970) Managing the development of large software systems. *August 1970 IEEE WESCON, The Institute of Electrical and Electronic Engineerins, p. 1 – 9 Available at http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf*

Appendix.

Table of content:

Appendix A: Semi-structured interview 1

I = Interviewer

D = Interviewee / Driver

I: Can you please tell us how your general driving routines are? (how often do you drive, where to, daily commutes?)

D: Basically I have to drive from home to school sometimes, home to my girlfriend, and if I have to go into the city it is a mess to park, because you have to find your own spot and that is a difficult task within central copenhagen. If it is within the outside areas maybe it is a bit easier, but still it may get tough if their are private parking spots so yeah.

I: For your general driving it is not a problem, like when you park next to your home or next to your girlfriend's?

D: Yeah because they have private parking like the buildings parking so you don't have to think about how am i going to park they have a certain ticket you put it in the dashboard, and then that is it you park.

I: Okay, cool. How often do you drive into central copenhagen?

D: Mmmm, depends a lot, two times a week approximately.

I: And how would you describe the parking situation in those areas? Liken when you drive into central copenhagen?

D: Difficult, considering you are usually not able to find a parking spot and if you do they are mostly private and then you have to pay a lot, but still you cannot check for all of them if they have or not free spots for example there are the ones at malls, shopping malls right, and they do *have some sensors checking how many spots they have available or not, but they cannot do that in the inner city where there are no private parking, i mean it is private, but it is outside so they didn’t add any sensors, so you just go by luck and if you find something that is it and if you don’t you get stressed and nervous due to the traffic jam.*

I: How much time do you think you spent when you are looking for spots in the middle of the city?

D: Mmmm, between 10 to 15 min. Maybe, it takes a lot actually.

I: Do you have a parking story that you would consider your nightmare story when you are driving around and finding a spot? Maybe it is going to be as bad as this time.

D: Actually, I do. I don't know how funny it is or not. But usually there are those streets with public parking and you cannot find a spot at least not in the evening when everyone is coming home, and then there was this guy he parked basically on two spots, more or less, and then there was half a spot and then in COpenhagen it is quite common to have it on the borders that yellow line that indicates that this is the end of the parking area or like how much you can park, how far you can park to the intersection, and the funny fact is I went maximum half a meter more than that and I had to pay a fine of more than 500 kr., so they should somehow mark them, and take an approximation how to do it, because the guys that are giving the tickets they don't care.

I: What about the actual process of finding a spot, do you have a similar story for this process too?

D: Yes, I would say to because I ended up to search for like 20 min. In Copenhagen in the central area close to kongens nytor, and it was funny because I searched for 20 min. So I was just moving around and I couldn't find anything so I had to go into those parking houses and then they are usually qpark and I had to pay 48 kr. For an hour, and that was funny because I didn't have any other opportunity and that is because on the street the spots get easily occupied if you are not like focused around, looking all the time and you end up paying a ton because you don't know which spots may be free.

I: And maybe you also dont know all the rules about the system?

D*: That is true, there are a lot of rules and a lot of companies that manage those parking areas, like five companies that keep track of them, they don't even keep track, they just want you to pay and that is it. And the public sector is even worse, you don't even have an app, for most of them you have to pay with cash or card.*

I: What are your general concerns in regards to parking?

D: Yes, but as I mentioned price, because I ended up paying a private company, because I couldn't find a spot outside. I mean there the private spots on the street or outdoor they are usually cheaper, they are not that expensive, because I wasn't able to find a proper or normal one. So it would be price and how fast I could get a spot and find out where is a free spot. Which no company has. They have some approximation, but it is not like they predict exactly where you have a spot in what area. It is actually only one company that does that the rest you just add the code, and you add it when you are already in the parking spot, but they do not mention anything about availability. Parkman has some percentages about availability. But not in the specific location.

I: So basically price and ease of use are concerns. What about proximity to location, how much of a concern is that versus price. Like are you okay with walking 500 meters.

D: Naahh, okay it depends. If I would spend less time on finding the parking, 500 meters is not that much. I could still be happy with that. Instead of moving around with the car and being stressed at least I know I am close and I have my parking and I can walk. 500 meters are just fine to walk.

I: We are in the process of designing a parking app for the city of Copenhagen that would help you find and locate available parking spots in the vicinity of your destination.

I: What is your general feelings about creating such a service?

D: Just do it. Every driver in Copenhagen needs it, if you could connect it in the public sector that would be really great. Because the public sector is the worst one and then find a way to connect it with the other apps I don't know, make an agreement with them, because they are not that good either, except for the fact that you can book it on the phone now.

I: How could you see an app like this change your life as a driver in Copenhagen? (will they alleviate some of your concerns)

D: I think change my live is…. Haha

I: Let us think about the driving aspect of your life.

D: Okay then, from what you told me it seems like I would have more free time to drink my coffee, before I get to work or school, but of course there would be some problem with the app as there always are. But I hope that it would help the sector in copenhagen.

I: Are there sometimes where you sit at home and think about going to this cafe in the city center, but then dread the parking situation.

D: Yes, and then you end up taking the metro and paying even more. Haha, probably not more. But yes I can see what you mean. You don't want to go because you are scared of being locked in a traffic jam and then the horns from everyone because you find a small spot and you are trying to put your car there and everyone is anxious around you. It is not fun.

I: So you mention this thing with the horns, do you feel that people are very aggressive when you are trying to find a spot?

D: Of course, in some concern I would be as well, but then it is understandable due to the lack of spaces and the lack of knowing where you can or what areas are more freely available. And then everyone go in a certain area because as you said maybe I would to go to a cafe and 20 other people are going at the same time as I am. And then you mentioned something about proximity, so all of us could park close to the cafe, but not in that place and therefore no jam and I suppose, no nerves or horns.

I: Do you ever feel like an accident is more likely to happen because of the parkings.

D: Accident is too big of a word, but a small collision yes, where to people crash at slow speeds, they are fairly common. And a lot of them happen due to the rush because I put it in the park right now and somebody could steal my place. Actually that happens quite often in IKEA.

I: So maybe even IKEA could use something like this?

D: Well they actually have sensors, but it is just so full of people there, mainly in the weekends, so it can be really hard to find a place to park.

I: And they mainly have sensors for the general…?

D: For each parking spot.

I: But are you guided to a specific spot?

D: No they have some lighting at the top of the wall, like on the ceiling. Where there is a car you have a red, and free spots have a green.

Appendix B: Semi-structured interview 2

I = Interviewer

D = Interviewee / Driver

I: Hello, thank you for allowing us to conduct this interview. Now I would like to ask you about your general driving routines. How often do you drive?

D: Hi, well... I don’t drive very often, but usually when I do it's to school, to university and sometimes it's for leisure such as going out in the city, or going to some sort of museum or park or whatever.

I: Okay. But you wouldn't say that you commute daily to university or to school or to work?

D: Yeah, not that often because driving in the morning is sometimes stressful because of all the traffic and the time pressure, sometimes it's just easier to just take the public transportation.

I: So would you say that during peak hours, public transport is better than driving yourself?

D: depends on the route, some buses are a total nightmare in peak hours so it depends.

I: Okay, I guess it depends on the situation itself. But general I guess that you prefer driving rather than public transport.

D: Yes, it's more convenient.

I: Okay. Now let's talk a little bit about the parking situation where you go. You said you go to university; do you have an easy time parking there?

D: Yes, they have a yearly pass that I can buy and then I can just park... so it's really not a problem but I do have a struggle when I go somewhere in the city.... for.... in my own time, then it’s.... sometimes it's really hard to find a parking lot and it's really hard to find one that is not extremely overpriced.

I: *laughs* would you say this is the case with the central areas of Copenhagen or like... bigger?

D: Definitely, central area because the prices can get crazy high, above 100 kr. per hour and then they just accumulate over hours and it costs a fortune.

I: Yeah, yes... so would you be... so in the case with the city centre, even though you have to pay so much do you have an easy time finding them or?

D: It's also a struggle because there are some parking areas that are not that expensive to park, but are usually already taken, so at some point I just end up driving around for a good half an hour and then not finding it and having to go to a very expensive area... so.... yeah....

I: Oh, yeah.... I guess that's the worst situation, well I guess you have already answered my next question, because... the time spent looking for a parking spot how high can that go? #

D: well... that was one of those cases when I have been looking for a parking lot for I would say.... even longer than half an hour, and it was a weekend, I think, and of course all the spots were taken and then I had to go to, I think, the most expensive area, it was right next to the central square, and I had to park the car for at least 4 hours and I paid over 500 kr. just for the parking.

I: okay... that sounds like way too much, would you say this was the worst experience you had?

D: yeah, personally yes.

I: that definitely sounds like something very bad. Do you think the situation could have been improved somehow?

D: I would have been really nice to have somewhat of map indicating the parking lots, because I don't drive that often so I don't have the knowledge of where all the parking areas are so I would like to have an overview of where my possibilities are and maybe even the price range, that would be really convenient.

I: Yes, that sounds like quite the feature. Do you have any general concerns when it comes to parking? Are you... do you care about the type of parking: parallel parking, time restrictions, stuff like that?

D: I mean of course, the easier the better, but it's not going to scare me away if I have to parallel park.

I: All right, so as long as you know that you will have a parking spot you will generally go with it regardless of paid, not paid?

D: Paid or not paid not that big of concern but it's more how much it costs, because I am pretty open to paying a small sum per hour if I have to park for a few hours or like a limited amount of money for the whole day like some places do. But I would not be comfortable paying a really big amount.

I: Yes of course. It wouldn't make sense.

I: Okay so, my team and I are in the process of designing a parking application for the city of Copenhagen and we want to create an app that helps you find and locate available parking spots near your destination or near your location on the map.

D: oh... nice.

I; Would you use an app like that?

D: Oh yes, definitely.

I: Would you be interested in... selecting the parking spots yourself or would you trust the app to give one to you?

D: .... depends on the efficiency of the app, because if it works really well then I am more comfortable with that, for example, at Ikea they have those green and red lights that indicate if the parking spot is available and more often than not they are accurate so I really having that guidance system rather than driving up to the spot and looking out because there are sometimes really small cars that park, and you think the spot is free but then you just drive up and you see it's taken and it's a whole new feeling of disappointment *sigh*

I: *sigh* yeah, yeah, I understand you... but yeah.... any suggestion or improvements that you would like to see from a parking app?

D: it would be nice to have the map, preferably... with multiple parking spots in the area or how many there are... maybe have a certain range... already there, on those parking spots, indicating the price range or if it's free or not. Maybe even, how many spots are available at the moment if it's possible, I don’t know how far that technology goes.

I: would you be... do you think it would be better if the app showed you, for example, you have 20 spots available or if the app showed you, 20 percent empty spots.

D: definitely in spots not in percentage.

I: so knowing exactly or approximately how many spots are available is better?

D: yes, I am not a cyborg I don’t do percentages.

I: *laughs* okay. I think that is it. Thank you for your time.

D: Goodbye.

Appendix C: Semi-structured interview 3

I = Interviewer

D = Interviewee / Driver

I: Can you please tell us how your general driving routines are?

D: Hello! Well, because we live in Denmark, and in Copenhagen the driving is a bit expensive, and as an international, I definitely drive less in Copenhagen than in my home country. Also.... it's usually just getting to work or getting to the city centre and that's about it.

I: Yeah, but how often do you drive?

D: A.... it depends a lot on what I have to do that week or that day, but generally I would say on average like.... 3 times a week, probably... 3 - 4 times a week

I: Back and forth..

D: Yeah, yeah, obviously back and forth. I am not just going to leave my car there

I: Makes sense. How would you describe the parking situation in the areas that you generally drive to? How easy is it to find a spot or how much would you spend looking for a parking spot?

D: Well... whenever I go to work that's incredibly easy, because the workplace has its own parking spots, so I know for a fact that if I go there I will have more than enough parking spots. Because, actually, there are less people with cars in the entire building then there are parking spots in the parking lot. So, that's amazing. But if I have to go to university or I have to go somewhere else, it's usually a real pain...

I: Then do you have a parking story that you would consider horrible?

D: A.... yes, there are a few. For example, I remember one time I went out shopping and I took my car to the city centre and I tried to park somewhere in Kongens Nytorv, I think. And I remember.... I remember looking around... I remember finding a parking spot that was.... it didn't look like it was one of those paid parking lots, so I imagined it was simply free, I looked around like it was legitimate to park there by all driving laws. And then I didn't see any signs from any company owning that parking lot and then I just left. Once... once I came back I saw a fine for like 500 krr and no way to repeal it, because yeah.... And I remember that one time I actually... it was EasyPark I think... probably...I think it was EasyPark the guys who owned the parking lot and I tried calling them and I tried asking them why was there no sign there or anywhere else to say that parking spot was paid... and yeah, they said that that's the law and that's it, nothing they can do about it.

I: Oh okay, so what would you say was the most irritating aspect of the whole story?

D: Not having any idea that, that parking spot is paid even though there is no sign or what so ever around to say it.

I: Mhmmm, and how could this experience have been improved?

D: Aa.... *sigh* well... I don't know. Maybe just have some signs, whenever there is a paid parking lot there should definitely be a sign there and I know that with paid parking lots there is also that ticket booth and I know that nowadays there are making phone apps for that. But the thing is with phone apps.... when you log onto their apps and try to pay for it, they either tell you that you are in a zone that does not belong to them either you are in a zone that belongs to them.... but the zone is the entire Norreport region, so I have no idea if this particular street is owned by them or is free or... I have no idea. It feels like you need to be a good driver, or to have been a driver around Copenhagen for a really long time so you just get the hang of it and then you get used to it.

I: What are your general concerns in regards to parking?

D: .... Whether it's legal or not. *laughs*

I: *laughs*

D: Well... .most of the times, in Copenhagen you have some pretty weird laws in terms of parking, for example I know that if there is no continuous lien on the middle of the road, and you try to park next to the edge of the road, there is a present distance from the edge of the intersection where you can park, and usually this distance is marked by these small yellow triangles, but the thing is, those were literally painted on the concrete a few years ago and they just get removed or washed away by getting used or by rain or stuff like that. So it's just knowing where you are allowed to park and where you are not allowed to park, that's my biggest concern.

I: Well, I can tell you that we are in the process of designing a new parking app for the city of Copenhagen that would help you find and locate available parking spots in the vicinity of your destination. What are your general feeling about having such a service?

D: *sigh* my immediate concern would be how would that work.... but that's up to the app to decide, if it does what you say it does, then yeah sure I am all up for it. I would use it, or at least give it a try.

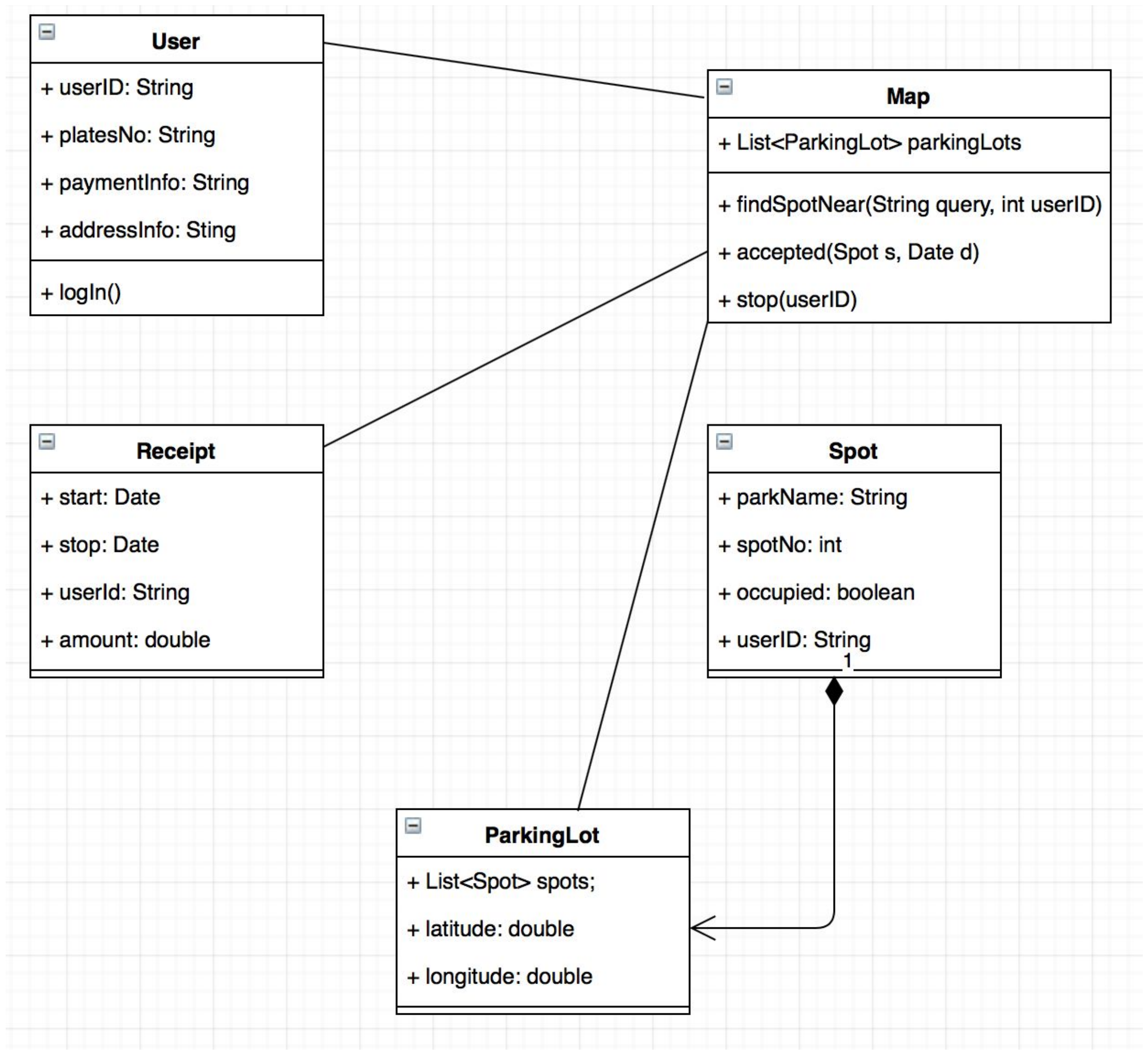
I: And how could you see an app like that change your life as a driver in Copenhagen.

D: I imagine I would get less infuriated and less stressed about not knowing whether what I did was legal or not and it would reduce a lot of the stress of just waiting to see if you got a ticket or not. *laughs*

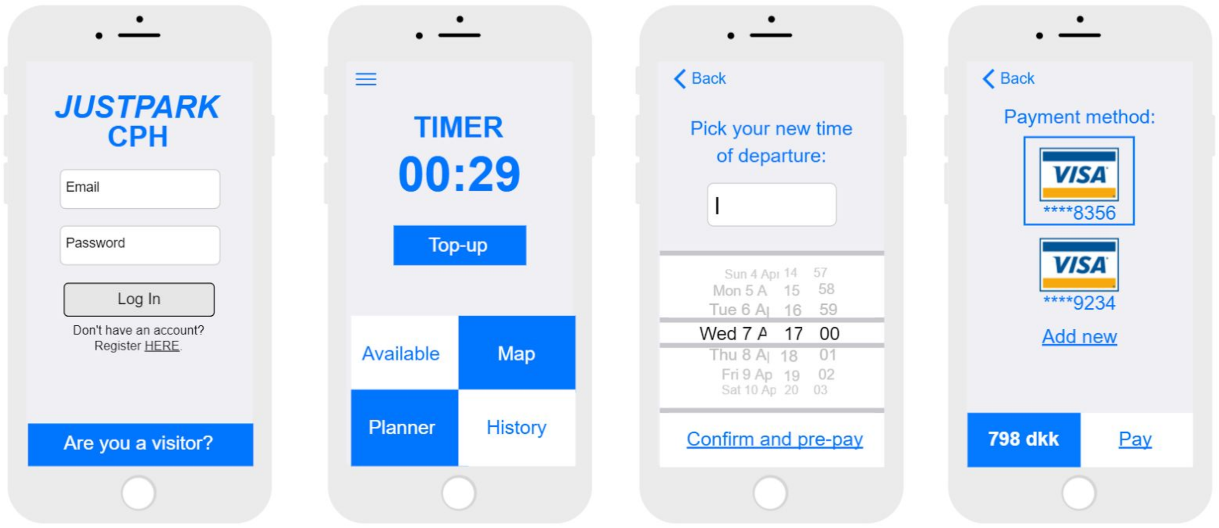
I: Okay, thank you for the interview.

D: Thank you.

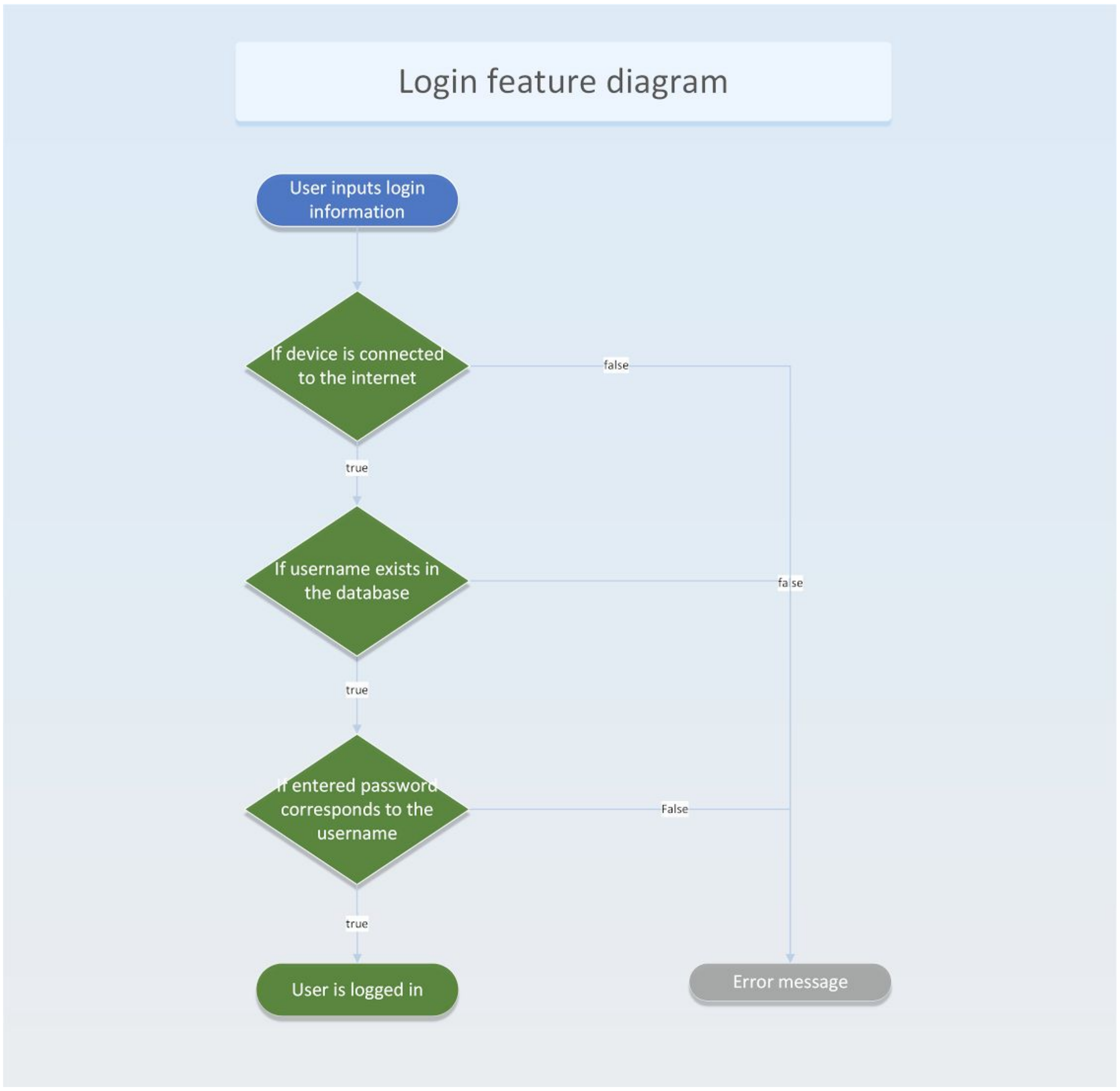
Appendix D: Class diagram - detailed



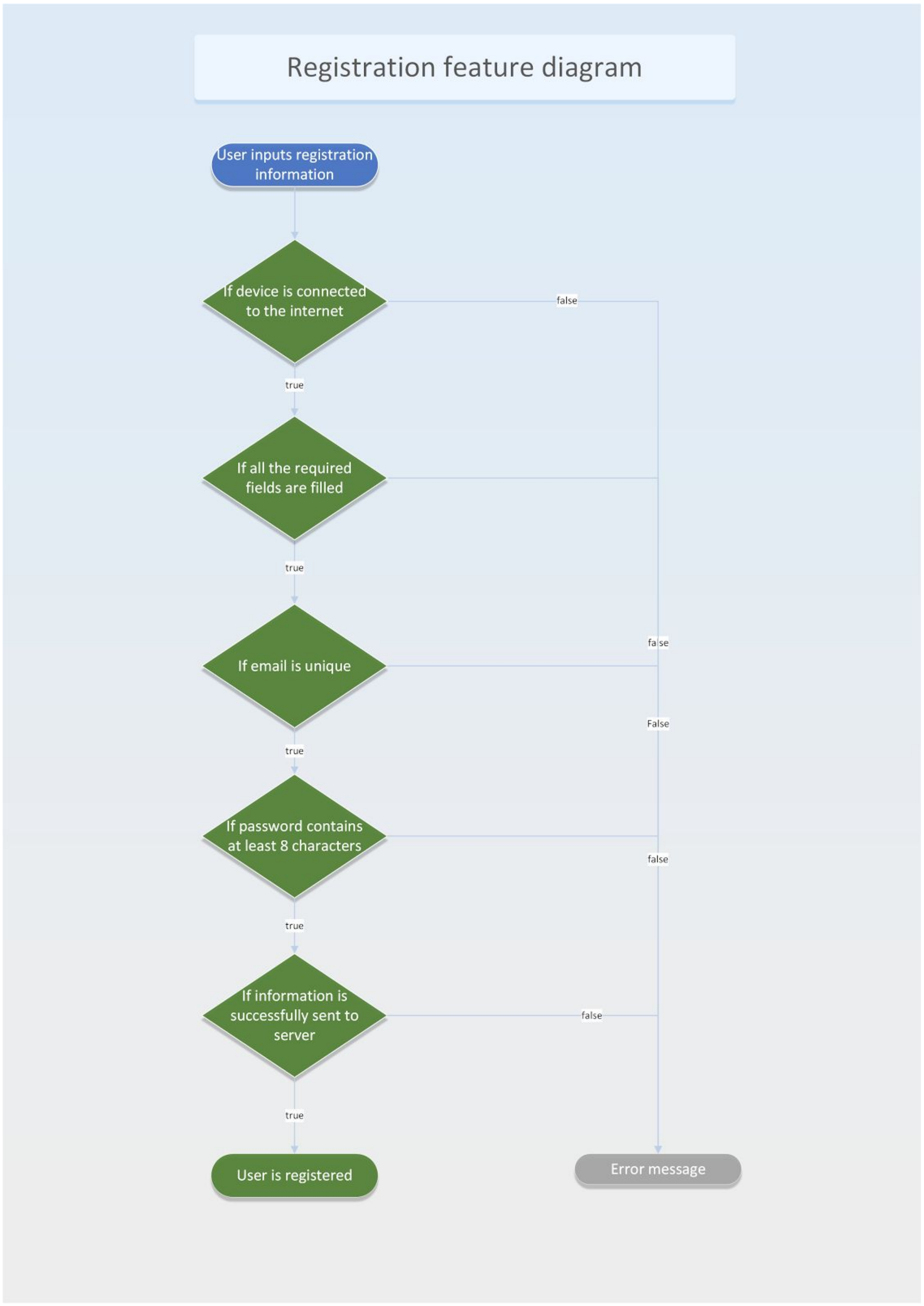
Appendix E


JUSTPARK CPH
Email
Password
Log In
Don't have an account?
Register HERE
Are you a visitor?
TIMER
00:29
Top-up
Available
Map
Planner
History
Back
Pick your new time of departure:
Sun 4 Apr 14 57
Mon 5 A 15 58
Tue 6 A 16 59
Wed 7 A 17 00
Thu 8 A 18 01
Fri 9 Ap 19 02
Sat 10 Ap 20 03
Confirm and pre-pay
Back
Payment method:
VISA
****8356
VISA
****9234
Add new
798 dkk
Pay

Appendix F - Flowchart 1 / Test case #2: User registration


Login feature diagram
User inputs login information
If device is connected to the internet
false
true
If username exists in the database
false
true
If entered password corresponds to the username
False
true
User is logged in
Error message

Appendix G - Flowchart 2


Registration feature diagram
User inputs registration information
If device is connected to the internet
false
true
If all the required fields are filled
true
If email is unique
false
False
true
If password contains at least 8 characters
false
true
If information is successfully sent to server
false
true
User is registered
Error message