Tampere University of Technology

Department of Pervasive Computing

TIE-13106 Project Work on Pervasive Systems

Group 11

SPARK - Next generation open-space city parking

Project plan

|  |
| --- |
| Joona Luoma |
| Soumya Das  Hanning Zhao  Tanja Lehtovaara  Aurélien Wolz  Anil Özdemir  Perttu Paarlahti |

Version history

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1 | 02.10.2016 | Joona Luoma | First draft |
| 2 | 09.10.2016 | Joona Luoma | Write risk monitoring section, add global and cloud implementation related risks to the risk list |
| 3 | 11.10.2016 | Hanning Zhao | Supplement usability goals section |
| 4 | 12.10.2016 | Tanja Lehtovaara | Added project organization and development constraints |
| 5 | 13.10.2016 | Tanja Lehtovaara | Added project management |
| 6 | 14.10.2016 | Perttu, Tanja, Joona, Soumya, Aurélien, Hanning | Edited whole documents |
| 7 | 15.10.2016 | Joona Luoma | Write functional requirements section. Fix usability goals. |
| 8 | 15.10.2016 | Tanja Lehtovaara | Write rest of the introduction and add to the project management and the project organization |
| 9 | 15.10.2016 | Anil Ozdemir | Added Testing / QA Plan |
| 10 | 16.10.2016 | Joona Luoma | Fix formation and some major typos |

Contents

1 Introduction 4

1.1 Purpose and scope 4

1.2 Product and environment 4

1.3 Customer's current system 5

1.4 Development constraints 5

1.5 Project constraints 7

1.6 Definitions, abbreviations and acronyms 7

2 Project organisation 9

2.1 Group members 9

2.2 Customer 10

2.3 Related organisation 10

3 Project goals and ending/termination 11

3.1 Goals of the project group 11

3.2 Goals of the customer 11

3.3 Goals and deliverable of the project 11

3.4 Quitting (termination) criteria of the project 12

3.5 Ending criteria of the project 12

4 Project management 12

4.1 Methods and tools 12

4.2 Monitoring and guidance 13

4.3 Learning and study plan 13

5 Project iterations and timing 15

5.1 Iterations 15

5.2 Sprint 1 15

5.3 Sprint 2 16

5.4 Sprint 3 16

5.5 Sprint 4 16

5.6 Sprint 5 (Quality assurance) 16

6 Requirements 16

6.1 Functional requirements 16

6.2 Additional functionalities 21

6.3 Quality goals (Non-functional requirements) 21

6.4 User interface requirements 23

7 Test and quality assurance plan 24

7.1 General approach 24

7.2 Definition of done 24

7.3 Special testing 25

7.4 Test documentation 26

8 Risk management 26

8.1 Risk list 26

8.2 Risk monitoring 26

9 Open issues 27

10 References 28

APPENDIX A […Z] 28

# Introduction

## Purpose and scope

This document contains the project plan of the project SPARK, a software development project for the Tampere University of Technology course TIE-13106 implemented by Group 11. The project plan in this document was created during sprint zero, according to the Scrum project management framework that the project uses, and thus only covers the parts that were possible to be documented at the time, i.e. the document covers detailed sprint planning only for sprint 1.

This document is targeted especially to the course personnel to measure group's progress and for the group itself as a plan for the project.

## Product and environment

The working name of the product is SPARK. The purpose of the project is to create a proof of concept for a smarter way to park and check the car's parking validity. "Smart" in this case meaning increasing the benefits by using technology. The project aims to provide an easier way for the parking attendants to check if a car is validly parked (the parking fee has been paid and/or the time limit has not been passed) while also providing benefits for the driver.

The context in more detail is public outdoor parking.There are three different types of parking: paid time-limited parking, free time-limited parking and free parking. As these contexts are similar to the current system they are explained in more detail in section 1.3. Also section 6.1 provides visualization of the contexts in pictures 1 and 2. The first priorities of the project is the context of paid (time-limited) outside parking including the drivers and the attendants point of view. The context of parking disc (time-limited) outside parking is considered in case there is time left. Free parking is out of the scope of the project.

In this phase the system is planned to consist of application for the driver, application for the attendant, smart parking meter, a beacon in the car and a cloud service. In the context of paid time-limited parking, the idea is to provide an easy way for the driver to pay for the parking with the driver application. The paying feature in this itself will be a mockup but the relevant part of the process is to register that the car is parked. When the attendant comes to check the cars' parking validity, the attendant's application can use the cars' parking information that was given by the drivers' applications and smart parking meter. The context of time-limited parking is similar to paid time-limited parking but it does not include payment or a smart parking meter.

There is also a possibility to provide a map for the driver and for the attendant using a cloud service. Driver map would show available parking places in certain areas and attendant map would show places where there are lot of cars parked. The maps would be part of driver's and attendant's applications respectively.

This system would improve the current system described in section 1.3 by providing information about free/taken parking places, informing the attendant if there are parking violations and making it easier for the attendant to check the validity of a individual car (e.g. if the snow is covering the whole windshield attendant does not have to wipe the snow in order to see the parking receipt/parking disc). In the future, if the system can be improved to detect the car parking automatically, there would also be more benefits for the driver (for example not having to walk to the parking meter in order to pay).

## Customer's current system

As the project is closer to a research project the customer does not have any current system.

As the goal of SPARK is to find an easier way for outdoor parking, a closer look for how parking is managed currently is needed. There are three kinds of public outdoor parking types: paid time-limited parking, free time-limited parking and free parking. In free parking, the driver can park without payment and without time limitations to marked parking spots or to roadside. Free time-limited parking allows the driver park for free but only for a certain amount of time. The driver marks the arrival time by using a parking disc. When it comes to paid time-limited parking the driver has to pay for the parking spot for the time they are planning to stay and leave the receipt on the car windshield.

Currently parking is moderated by Tampereen pysäköinninvalvonta. The parking attendant will check if the cars are parked correctly by either checking the time on the parking disc or on the parking receipt. To do this the attendant has to walk around to check each car individually. The cars can also be invalidly parked according to other laws but that is out of the scope of this project.

When the attendant finds a car that is invalidly parked they will leave a written fine on the windshield of the car or, if possible, give it to the car driver/owner as stated in the law (Finlex, 2011). They also take a picture of the car as a proof in case there is a complaint afterwards.

## Development constraints

### Technology constraints

The customer requires us to use Bluetooth Low Energy (BLE, Bluetooth 4.0) technology for communication between driver's or attendant's application and in-vehicle beacons. BLE is also to be used in communication between smart meters and driver's application.

The customer initially wanted us to use Google Brillo as operating system for beacons and smart meters, Google Weave as communication technology between smart meters and the cloud, and Google Cloud Platform as the cloud. The project is intended to be part of Google's developer contest, where the goal is to use said technologies in a new creative way. Technology constraints derive from the contest. However, technology constraints from the customer may change, since we have encountered several issues with them.

Google has provided the customer with five Intel Edison development boards, and we are expected to use them as beacons and smart meters. In addition, the customer provides us with access to ten Raspberry Pi 3 computers and BLE USB-dongles, which we may use if necessary.

When getting familiar with Brillo, it turned out that Brillo does not provide hardware abstraction layer for Bluetooth. Communicating over Bluetooth is essential in our project, so lacking support for Bluetooth makes Brillo useless for our purposes. We may use other third party libraries with Brillo to access Bluetooth, but there are other frameworks for Intel Edison that do support BLE without other libraries. Brillo is an experimental project for Google and still under pre-release development, so it has very limited support, and finding compatible third party libraries may be difficult.

Weave also has its problems. To enable Weave on a device requires the device to be provisioned using special tools provided by Google. Currently, Weave Device Manager tool used for provisioning is under closed beta testing, and we have no access to it. There may be an alternative way to provision Weave devices using another tool, but at the moment of writing, alternative way is still under investigation. Google's Weave developer support has not answered our questions regarding unavailable tool, and documentation for the other tool seems incomplete, so it is possible that we will not be able to find solution within course time limits.

We have agreed with the customer to keep investigating Brillo and Weave, but if these technologies proof unusable for the project, we decide alternative technologies with the customer.

The customer has provided us with access to Google Cloud Platform. The cloud platform is supposed to be used for storing parking event data provided by smart meters and attendant gears. Then analysis steps are executed on the data, analysis results being the data needed by the parking availability map. Based on the initial test runs, we have gotten access to it and there is a quite good support and community behind the platform. We also considered Microsoft Azure for the platform, but for now it seems that Google Cloud Platform indeed is sufficient for us. For the Dataflow processes included in Google Cloud Platform, that actually transform and move data from on place to another, one has to choose a programming language to be used. Out of all supported languages, Java and Python seem to have the overall best support considering general use and discussion around the languages, the tutorials and the documentation. From these two, Java seems to be even better supported than Python, and at least initially, Java is our choice, though the cloud lead is much more experienced with Python.

We will choose technologies that support the goal of our project and have the features needed to implement the goals. The technologies are not set by the customer or the course but the group is free to change the technologies used if there is a need to.

### Standards and laws

The law concerning the project is Laki pysäköinninvalvonnasta (The parking law) (Finlex, 2011). Especially the parts of the project that concern the work of a parking attendant should be in accordance with the law.

As the developed system will only use a mockup for the paying functionality, there is no need for considering standards and laws about mobile payments specifically. At the time of writing this document we are not planning to store any personal identifying information so there is also no need for considering laws about storing and personal information.

### Other constraints

The prototype should be implemented in a way that the attendants could hypothetically use it at the same time as the current system. In a scenario that the SPARK product would be taken into use not all cars and drivers would start using SPARK immediately. In fact, it is likely that some drivers would use spark and some would use the current system. Therefore, the attendant would have to check the validity in both traditional way and by using SPARK. Taking this into account SPARK should work in a way that would not make the traditional checking more difficult but, if possible, make it more simple.

## Project constraints

The software resulting from the project will be open source with MIT license.

## Definitions, abbreviations and acronyms

|  |  |
| --- | --- |
|  |  |
| Attendant | Parking attendant patrolling parking areas |
| Beacon | BLE-BEACON gadget positioned in cars whose drivers use SPARK. Responds with METADATA about the parking situation of CAR |
| BLE | Bluetooth Low Energy, Bluetooth Smart or Bluetooth 4.0, a wireless communication technology for short distances. |
| Car | A vehicle that gets parked. Either has a BEACON or not |
| Cloud | Google Cloud Platform receiving data from SMART METERs and providing parking map information to DRIVER APP. |
| Driver | A person parking CAR and using DRIVER APP |
| Driver app | A mobile application that, at the time of parking CAR, communicates with SMART METER and BEACON respectively. |
| Gear | A handheld smart tool wielded by ATTENDANT. Connected to CLOUD |
| Map | A functionality in DRIVER APP and GEAR. A Google Maps -map with a heat map overlay that presents the user the predicted availabilities of parking lots |
| Metadata | Used vaguely to stand for “all metadata necessary at this point”. Specified in detail whenever purposeful. |
| MIT license | A Permissive software licence |
| Parking availability map | See **Map** |
| Scrum | Agile software development framework |
| Smart meter | A parking meter, which accepts payments both traditionally (i.e. the person parking goes at the meter, pays physically with cash and the meter prints a parking permit for the parker) and using SPARK. When requested, sends METADATA about the parking context to DRIVER APP. Sends METADATA about payments to CLOUD. |
| SPARK | This application (of many applications). A system consisting of SMART METERS, BEACONS, DRIVER APPS, GEARS and CLOUD. |
| Token | A mean of authentication between multiple parties without shared memory. In our context, TOKEN is used for validating parking permit advertised by BEACON, making sure that permits cannot be forged. |
| UAT | User acceptance testing |
| UI | User interface |
| UX | User Experience Design |
| UAT | User Acceptance Test |
| CAT | Customer Acceptance Test |

# Project organisation

## Group members

**Tanja Lehtovaara, Project manager**

Experience, interests:

* Professional: Testing and UI design
* Education: C++, Java script, Python, Java, User Experience, Project management
* Interested in the process of making a good and working software. This includes the project management, requirements management, designing a usable UI, testing and the programming itself.

Estimated minimum weekly hours for the course: 7 h/week

**Soumya Das, Scrum master**

Experience, interests:

* Professional**:** 2years in database and data warehousing domain with SQL, Informatica (ETL tool).
* Personal: R, Python, Statistical Analysis
* Mostly interested with data analytics and feature engineering. Scrum, agile development of software also interests me.

Estimated minimum weekly hours for the course: 7 h/week

**Hanning Zhao, UX lead**

Experience, interests:

* Education: Java, Python, HTML 5 and CSS, JavaScript, UX field, MySQL
  + - Major: User experience
    - Minor: Software Engineering

Estimated minimum weekly hours for the course: 8 h/week

**Perttu Paarlahti, Hardware lead**

Experience, interests:

* Professional: software developer on machine control domain, worked as a research assistant at TUT.
* Own projects: Open source smart home Auxilo2 and some other stuff <https://github.com/PerttuP>
* Skills: C, C++, C#, Java, Python, SQL.
* Most interested in hardware related software and IoT.

Estimated minimum weekly hours for the course: 10 h/week

**Aurélien Wolz, App lead**

Experience, Interests:

* Professional: Big data analysis using Hadoop clusters, Web app development (MySQL / PostgreSQL, HTML & CSS, PHP Symfony 2, Javascript, JQuery, RESTful web services)
* Education: C/C++, Java, Python, Javascript, SQL, Android, Front-end design.
* Personal experience : Hybrid mobile applications using Ionic, AngularJS, Cordova for client side, Node.js, Express for server side and Firebase for real-time database.
* Passionate by mobile application development, database architecture, data analysis. I also have a particular interest for blockchain and I would like to improve my skills with IoT technologies.

Estimated minimum weekly hours for the course : 10 h/week

**Joona Luoma, Cloud lead**

Experience, interests:

* Professional: +2 years of C#, SQL Server and Web services
* Personal: C#, Machine learning, Python (including sklearn), R, TypeScript
* Education: Machine learning, Python, R, C++, JavaScript, Testing/QA, Project management, Architecture design, Front-end design.
* Passionate about database development, data modelling, machine learning, analytics, data intensive applications. Interested also in integration and web services.

Estimated minimum weekly hours for the course : 7 h/week

**Anil Özdemir, QA lead**

Experience, interests:

Education : C++, Java, Machine Learning

Estimated minimum weekly hours for the course: 7 h/week

## Customer

The contact personnel are:

* Bill Silverajan
* Antti Kolehmainen

Both of them work for TUT in information technology department. In addition to being the customer of the project Bill Silverajan and Antti Kolehmainen are available for coaching, development support and advice.

## Related organisation

The customers organization is Tampere University of Technology.

In addition, Tampereen Pysäköinninvalvonta is a organization that could give us valuable information and useful statistics as they are currently responsible for checking the parking validity in Tampere area. Otherwise they are not related to the project.

In case Google's Brillo and Weave will be chosen as technologies for the project, Google will be related organization as the technologies were provided by them and there might be a need for support.

# Project goals and ending/termination

## Goals of the project group

Through this project, the development team (composed by students from Tampere University of Technology) wants to put theoretical knowledge acquired during their studies into practice.

There are many personal goals, because everyone in the group hopes to discover and deepen subjects related to their passions through this project. But in general, this project is an important opportunity for everyone to work with the world of the Internet of Things.

From the perspective of the project's conception, our goals are:

* making a product conforming to specifications
* to respect imposed deadlines
* customer satisfaction
* to be proud of our work.

And for sure, the common goal is to have fun through this project!

## Goals of the customer

The customer wants a functioning prototype for the proposed subject. The customer has declared that the main purpose of the project is to make parking validity checking easier for parking attendants, especially in winter time.

Project features are divided into several parts, which constitute an objective hierarchy to achieve. The goal for the customer is the realization of a functional product for drivers and attendants.

## Goals and deliverable of the project

The primary objective of this project is to create a prototype of an application that covers at least the minimum requirements, thus making the application to meet customer's goals, while still respecting the course deadlines. Minimum requirements are defined in chapter 6 on the requirement part of this document.

The deliverables for this project are:

* source code for parts of the project;
* technical documentation;
* functional prototype.

## Quitting (termination) criteria of the project

The project may be abandoned by the customer if the development does not follow the forecasts. If delay is proven, it is possible, on a joint decision with the client and the project manager, to stop the project and evaluate available solutions.

This is a research project, so we are really open for technology and termination criteria of the project. If a technology does not work for our purpose, we are free to change it and adapt the project.

## Ending criteria of the project

The project ends if one of the following criteria is fulfilled:

* Customer's goals are achieved
* Product is functional
* Product is approved by the customer
* The course ends

Project is considered successful, if customer's minimum requirements are achieved.

# Project management

## Methods and tools

**Project management**

The project follows scrum. Agilefant is used for user story backlog and for estimating the difficulty and priority of the tasks. Agilefant is also used for dividing the stories for sprints and keeping track of the progress of the tasks.

**Communication**

For internal communication the group uses slack and for communicating with customer, Gitter and email are used.

**Documentation**

Internally the group uses Google Drive and OneDrive for sharing and creating documents. The code and the documents shared with the customer will be uploaded in Github.

All meetings with customer and important internal group meetings must be documented in the form of a meeting report which will be uploaded in Github.

**Hour reporting**

Google sheets that everyone in the group is able to edit.

**Version control**

Github is used for version control.

**Change management**

If there is a need for a change in the group's internal working methods the case is discussed within the group. If the change concerns the requirements or directly affects the customer the case is discussed with the group and with the customer. If the decision replaces an earlier documented decision the change is also documented. The reason for change should also be documented. The changed requirement will be added to the backlog.

**UI prototyping**

Balsamiq mockups is used for prototyping the user interface. Any other prototyping tool can also be used depending on the need.

**Design visualization**

Draw.io will be used for design visualization as it provides good collaboration possibilities.

**Time management**

Google calendar is used for marking all important deadlines and meetings. Also if a group member is unavailable for a longer period of time, it should be marked.

**Development environment**

Working with Intel Edison development boards, Brillo and Weave requires a desktop with a 64-bit Linux operating system. Both Google and Intel recommend Ubuntu 14.04 Trusty as the Linux distribution. We have a couple of actual Ubuntu desktops, but to make sure that everyone is able to work with Edison boards if necessary, we have put up a virtual development environment on WMWare Workstation 12 Player (non-commercial licence).

## Monitoring and guidance

The group members will monitor each other to make sure the work gets done. Especially, the project manager takes care that the deadlines are met, the scrum master takes care that scrum methods are followed, product owner makes sure that the features are scheduled for implementation and that they are done within time limits and according to customers requirements and the QA lead monitors that the testing is appropriate. All group members will help each other if there is a problem.

Monitoring and guidance from outside the group comes from the course personnel or form the customer. The course personnel will monitor that the group does all required tasks to pass the course and help the team to reach their goal. The customer wants to have their goals reached so they monitor the progress in that sense. In our case, the customer has also mentioned that they are able to provide help and guidance in our project.

## Learning and study plan

As the technologies used for the project are not yet defined also there is no definite study plan yet. However, the group members need to learn about potential technologies in order to choose the appropriate technologies and tools for the project. This is done as early as possible, in sprints 0 and 1.Researching technologies is mostly done by the hardware lead, the app lead and the cloud lead. After the technologies have been chosen, everyone in the group who is responsible for that part in the project has to learn the technology as well. The ones with knowledge about the technologies will share their information with others to get everyone started with learning process.

Google Cloud Platform is studied by the Cloud Lead. If Google Cloud Platform or other related technologies turn out to be insufficient, then Microsoft Azure is studied by the Cloud Lead. Studying the cloud platforms is done by going through tutorials, by getting familiar with documentation and by doing some hands-on testing with the platforms. Studying the cloud platforms is done mostly during the sprints 0-1.

The app side of the project is studied by the App Lead. If initial technologies turn out to be insufficient, we are free to change them in order to create apps which work. Studying the app side is done by going through tutorials, by realizing small applications related, by working with documentation. Studying is done mostly during the sprints 0-1.

The HW lead has studied Brillo, Weave, Bluetooth standard and working with Intel Edison boards during sprint 0. So far we have managed to compile Brillo and install it on Intel Edison with an example application. Adding Weave integration to Edison has failed due to inaccessible Weave Device Manager tool. After discovering problems with Brillo and Weave, Perttu and Aurèlien have studied alternative ways to use Intel Edison's Bluetooth peripherals. Intel XDK development kit with Node.js looks promising. We will still continue studying Brillo and Weave and finding solution to the problems in end of sprint 0 and in early sprint 1.

Most urgent task in sprint 1 in HW perspective is to get Bluetooth communication working. After that, we will start looking into cloud integration and other technologies required for the stretching target.

The customer has promised to provide training on hardware related issues. We have not used this option by now, but if we can't progress with the hardware by ourselves, we will definitely ask for training from the customer.

Except for HW lead, not many other team members have experience on working with development boards. Initial plan is that HW lead will do all the tasks related to the development boards, but introducing others to the subject is necessary in case of HW lead being unavailable for reason or another. The plan is to have an introduction session with the team, and make sure that everyone has development environment set up, and a basic idea about the work with the development boards.

Especially the scrum master and the project manager has to get to know more about the practicalities of Agilefant, since we will be doing the project tracking using Agilefant. And also will be trying to focus on scrum. Since the project is more of agile development process. Risk scenarios will also be taken into account, whatever risk scenarios can happen in the project. Like if the team is unable to complete all the planned user stories in a sprint, how is it going to figure out the next sprint, and design the user stories for next sprint. Though we will try to follow scrum, but practical scenario may make us following a scrum but methodology.

# Project iterations and timing

This chapter describes iterations, their starting and ending times, contents, important dates and deadlines and responsibilities for of each team member.

## Iterations

The project will last 15 weeks. The first 13 weeks are reserved for design and implementation and the last two weeks for quality assurance. Weeks reserved for design and implementation are divided into four sprints. Quality assurance has its own two week sprint at the end of the project. Starting and ending dates for each sprint are listed in the table below. Content of each sprint will be decided in detail at the beginning of the sprint. This document will be updated at the beginning of every sprint.

Table 1. Sprint starting and ending dates.

|  |  |  |
| --- | --- | --- |
| Sprint | Starts | Ends |
| 1 | 17th October 2016 | 6th November 2016 |
| 2 | 7th November 2016 | 27th November 2016 |
| 3 | 28th November 2016 | 18th December 2016 |
| 4 | 19th December 2016 | 15th January 2017 |
| 5 (QA) | 16th January 2016 | 29th January 2017 |

Following sections describe goals and responsibilities for each sprint.

## Sprint 1

Main focus of the first sprint is to produce a demo for the mid-presentation. We have agreed to demonstrate a functioning Bluetooth beacon with a placeholder data, and mock-up implementations for both driver's and attendants application in the presentation. Beacon also has to be tested in its real environment, inside a car, to verify that technology is fitting its purpose. Table 2 lists important dates related to sprint 1. Sprint backlog in table 3 describes first sprint's contents and responsibilities for each group member in more detail.

Table 2. Important dates.

|  |  |
| --- | --- |
| Date | Description |
| 8th November 2016 | Mid-presentation, demo |

Table 3. Sprint 1 backlog.

|  |  |  |  |
| --- | --- | --- | --- |
| Story | Story Responsibles | Story Description | Story points |
| Beacon | Perttu Paarlahti | Receive and broadcast parking information | 4 |
| Configure your informations (car model, size, ...) | Aurélien Wolz | Set configuration such as car registration number, model, beacon pin number ... | 1 |
| Create App Engine Platform for Data Retrieval and Analytics | Joona Luoma |  | 2 |
| Create parking lot availability map visualization | Aurélien Wolz, Joona Luoma | Draw interactive map based on informations receive from the cloud | 2 |
| Create usable user interface | Aurélien Wolz, Soumya Das, Hanning Zhao |  | 2 |
| Create usable user interface | Aurélien Wolz, Soumya Das, Hanning Zhao |  | 2 |
| Detect BLE devices around you | Aurélien Wolz |  | 2 |
| Implement Cloud Pub/Sub | Joona Luoma |  | 3 |

## Sprint 2

Description of sprint 2.

## Sprint 3

Description of sprint 3.

## Sprint 4

Description of sprint 4

## Sprint 5 (Quality assurance)

Description of quality assurance sprint.

# Requirements

## Functional requirements

The functional requirements are presented in the form of user stories. For readability, the most important concepts are written with capital letters in the stories. Namely, these concepts are (in alphabetical order): ATTENDANT, BEACON, BLE, CAR, CLOUD, DRIVER, DRIVER APP, GEAR, MAP, METADATA, SMART METER, SPARK, TOKEN. The concepts that are especially presented by this project or that are technical terms are presented in the section 1.6. of this document.

The user stories are described from the point of view of the two *main users* of SPARK: DRIVER and ATTENDANT. DRIVER parks their CAR and eventually ATTENDANT will come and check the validity of the CAR's parking permit. The user whose point of view the user story is described from is expressed in the title of each user story.



Picture 1. Paid outside parking in Lindforsinkatu, Tampere.



Picture 2. Parking disc outside parking in Insinöörinkatu, Tampere.

There are two *main parking contexts* that the SPARK application considers: *paid outside parking* and *parking disc outside parking*. The application and the project only covers these contexts, and therefore, for example, free of charge outside parking and parking hall parking are not in the scope of the project. Each user story that takes place in one of these contexts has the context name shown in the title of the story. The paid roadside parking in Lindforsinkatu, Tampere, is an example location for paid outside parking (Picture 1). The parking disc parking in front of Speakeasy restaurant in Insinöörinkatu, Tampere, is an example of parking disc outside parking (Picture 2).

The main goal is to implement paid outside parking context from driver's and attendant's point of view. Parking disc outside parking is a secondary goal.

Some of the user stories refer to stories that do not cover **functional requirements** for this project, but that would be necessary for the application to be functional and usable in practice. These referred stories are:

* **DRIVER: Paid outside parking without SPARK**
* **ATTENDANT: Paid outside parking without SPARK 1b**
* **ATTENDANT: Parking disc outside parking without SPARK 1b.**

These stories can be found in the section 6.2.

### DRIVER: Configuration

1. DRIVER configures his CAR’s register number to the DRIVER APP and bonds the BEACON with the DRIVER APP. DRIVER APP may have multiple sets of register numbers and BEACON bonds, since DRIVER may have multiple CARS. Likewise, BEACONS may be bonded with multiple DRIVER APPs, since multiple people may be using the same car. DRIVER APP configuration needs to be only once per CAR-BEACON pair.

### DRIVER: Parking lot availability map

1. DRIVER opens the DRIVER APP and goes to the map view with the DRIVER APP’s menu
2. DRIVER APP pulls from CLOUD a view of Google Maps -map with a custom heat map overlay that has been built based on the latest available parking lot availability information received from SMART METERs and GEARs. This is called the MAP
   * In the overlay, green areas mean that the CLOUD analysis has deduced that in these areas it is likely that there are free parking lots. As the color turns from green to red, the likelihood of finding a free parking lot decreases.
     + These exact colors are mere placeholders. I.e. in later phases they may be changed to some other colors for UX reasons.
3. DRIVER searches through the MAP for a place around which there are likely free parking lots by zooming and moving the MAP view
4. Having an idea about where to park, DRIVER closes the DRIVER APP.

### ATTENDANT: Parking lot availability map

ATTENDANT has access to the same MAP with the GEAR as was described in **DRIVER: Parking lot availability map**.

### DRIVER: Paid outside parking with SPARK

1. DRIVER drives his CAR to a paid outside parking lot and parks his CAR
2. DRIVER exits the CAR and goes to a SMART METER 20 metres away from the CAR
3. DRIVER opens the DRIVER APP on his way to the SMART METER and configures the time he would like to park his CAR for into the DRIVER APP
4. When the DRIVER is at the SMART METER, he confirms the configured time with the DRIVER APP by tapping a “Confirm time”-button on the DRIVER APP. The DRIVER APP connects with the SMART METER by bluetooth and receives METADATA about the parking area. This METADATA would include at least
   * Parking area identifier
   * Parking price per hour
   * The price that the DRIVER has to pay for staying for the configured amount of time.
5. DRIVER APP presents this information to the DRIVER and DRIVER confirms the payment by tapping “Confirm payment”
6. SMART METER sends DRIVER APP a TOKEN that authorises the DRIVER to park his CAR for the allotted amount of time
7. SMART METER sends METADATA about the parking situation to CLOUD. This event is called “parking registration to CLOUD”. The METADATA would include at least
   * Parking area identifier
   * Parking time
   * CAR’s register number
   * TOKEN.
8. DRIVER APP instructs the DRIVER to walk back at his CAR. The DRIVER walks back to his CAR with his smart device in his hand
9. Back at the CAR, the DRIVER taps “Send parking info to car”-button on the DRIVER APP (without necessarily entering the CAR)
10. DRIVER APP sends the TOKEN to the BEACON inside the CAR
11. BEACON receives the TOKEN and starts broadcasting it
12. DRIVER APP informs the DRIVER that parking has been completed
13. DRIVER closes the DRIVER APP and leaves the CAR.

### DRIVER: Parking disc outside parking with SPARK

1. DRIVER drives his CAR to an outside parking disc parking lot and parks the CAR
2. DRIVER opens the DRIVER APP with his smart device, chooses the parking disc parking -option and taps “Park here”
3. DRIVER APP sends a point of time when parking was initiated to the BEACON
4. BEACON begins broadcasting the point of time when parking was initiated
5. DRIVER closes the DRIVER APP and leaves the CAR.

### ATTENDANT: Paid outside parking

1. ATTENDANT arrives to the parking area
2. ATTENDANT taps “Get parking area information” on the GEAR and GEAR gets METADATA about all the registered-as-parked CARs in the area. A registered-as-parked CAR in this context means a registration of a payment related to a CAR for a certain amount of time. This does not mean that CAR should still be physically in the parking lot, but at least its parking time has not yet exceeded. The METADATA would include at least
   * TOKENs
   * Register numbers of CARs
   * Points of time when parking registration to CLOUD was done
   * Allotted time (i.e. time-spans that were actually paid).
3. ATTENDANT walks by CAR and taps “Scan” button on GEAR
   * IF GEAR receives a broadcasted TOKEN from the BEACON inside the CAR, then the user story continues in **ATTENDANT: Paid outside parking with SPARK 1a**
   * IF NOT, then the user story continues in **ATTENDANT: Paid outside parking without SPARK 1b**.

### ATTENDANT: Paid outside parking with SPARK 1a

1. ATTENDANT looks at GEAR.
   * IF CAR’s staying time has exceeded the allotted time, then the user story continues in **ATTENDANT: Paid outside parking with SPARK 2a**
   * IF NOT, then the user story continues in **ATTENDANT: Paid outside parking with SPARK 2b**.

### ATTENDANT: Paid outside parking with SPARK 2a

1. GEAR informs the ATTENDANT about CAR’s staying time having exceeded the allotted time and offers the ATTENDANT METADATA about the CAR. This METADATA would include at least
   * Register number
   * Point of time when parking registration to CLOUD was done
   * Allotted time
   * Amount of time exceeded.
2. ATTENDANT checks that the register number of the CAR matches the register number that the GEAR offers, and if they match, the ATTENDANT fines the CAR
3. ATTENDANT continues patrolling.

### ATTENDANT: Paid outside parking with SPARK 2b

1. GEAR informs the ATTENDANT that CAR has not exceeded the allotted time
2. ATTENDANT continues patrolling.

### ATTENDANT: Parking disc outside parking

1. ATTENDANT arrives to the parking area
2. ATTENDANT taps “Get parking area information” on the GEAR and GEAR gets METADATA about the parking area from CLOUD. This METADATA would include at least
   * Parking area identifier
   * Maximum allowed parking time.
3. GEAR informs ATTENDANT that the current parking area is a parking disc area
4. ATTENDANT walks by CAR and taps “Scan” button on GEAR and GEAR sends to CLOUD information about “Scan”-button having been pressed in that parking disc area
   * IF GEAR receives the broadcasted TOKEN from the BEACON inside the CAR, then the user story continues in **ATTENDANT: Parking disc outside parking with SPARK 1a**
   * IF NOT, then the user story continues in **ATTENDANT: Parking disc outside parking without SPARK 1b**.

### ATTENDANT: Parking disc outside parking with SPARK 1a

1. ATTENDANT looks at GEAR.
   * IF CAR’s staying time has exceeded the allotted time, then the user story continues in **ATTENDANT: Parking disc outside parking with SPARK 2a**
   * IF NOT, then the user story continues in **ATTENDANT: Parking disc outside parking with SPARK 2b**.

### ATTENDANT: Parking disc outside parking with SPARK 2a

1. GEAR informs the ATTENDANT about CAR’s staying time having exceeded the allotted time and offers the ATTENDANT METADATA about the CAR. This METADATA would include at least
   * Allotted time
   * Amount of time exceeded.
2. ATTENDANT fines the CAR and continues patrolling.

### ATTENDANT: Parking disc outside parking with SPARK 2b

1. GEAR informs the ATTENDANT that CAR has not exceeded the allotted time
2. ATTENDANT continues patrolling.

## Additional functionalities

This section covers those proposed functionalities, whose conceptualization and/or implementation is not part of the required functionalities, and to which the project team will come back to in case there is enough time. Whether or not these functionalities are conceptualised and/or implemented does not have an effect on team's, customer's or course personnel's views on whether the project was successful or not.

### ATTENDANT's additional MAP features

ATTENDANT could have some additional features in his version of the MAP, for example the absolute numbers of SPARK using CAR's in certain areas.

### DRIVER: Paid outside parking without SPARK

1. DRIVER drives his CAR to a paid outside parking lot and parks the CAR
2. DRIVER exists the CAR and goes to the SMART METER
3. DRIVER purchases a ticket from the SMART METER and METADATA about the purchase is sent to CLOUD. This METADATA would include at least
   * Parking area identifier
   * Parking time
4. DRIVER receives a parking ticket from the SMART METER
5. DRIVER walks back to his CAR, puts the parking ticket on the dashboard and leaves the CAR.

### ATTENDANT: Paid outside parking without SPARK 1b

1. GEAR informs the ATTENDANT that the scanning found no SPARK using CARs
2. ATTENDANT checks if there’s a valid parking permit on the windshield of the CAR
   * IF THERE is, then ATTENDANT continues patrolling.
   * IF NOT, then ATTENDANT fines the CAR and continues patrolling.

### ATTENDANT: Parking disc outside parking without SPARK 1b

1. GEAR informs the ATTENDANT that the scanning found no SPARK using CARs
2. ATTENDANT checks if there’s a valid parking disc on the windshield of the CAR
   * IF THERE is, then ATTENDANT continues patrolling
   * IF NOT, then ATTENDANT fines the CAR and continues patrolling.

## Quality goals (Non-functional requirements)

### Usability goals

In SPARK, usability is seen especially a concern of the two smart device applications: driver app and attendant app. In general, usability concern different aspects, for example, an application should be effective to use, efficient to use, engaging, error tolerant and easy to learn. Therefore, usability goals are set for each application.

Assuming the goals for driver application, drivers are able to complete a parking work with no errors, more specific, they can find out a parking place effectively and start parking immediately in an easy way. Moreover, they are able to find relevant information on application during parking period, finally, they are supposed to pay for the parking fee smoothly. Drivers should be able to use the parking availability map while driving.

Assuming the goals for attendant application, attendants can check validity of parking successfully, which means no errors (i.e. no false positives or true negatives). In addition, they can know the time of parking for each car and if there is an invalid parking, attendants are able to complete tasks involved with this scenario (i.e. fining) without the usage of our application interfering the process too much.

In each view and after each user action, each applications will give some response to the user in at least two seconds. The response is either the actual result of the action that the user is executing, or if the execution cannot be finished in two seconds, the view will show the user a loading indicator.

###### User experience goals

Another point is user experience goals which differ from usability goals, as they focus more on the feeling of user when interacting with product. The user experience goals for each application are as follows:

Considering the driver application, drivers should experience

* That it is easy and quick to find a sufficient parking place by using the parking occupancy map
* That the application facilitates the paid/parking disc parking process making it more smooth
  + Possibly making it also less time consuming, though this is not guaranteed
* Autonomy to check the state of parking, for instance, the time of parking and cost on it
* That the application and the beacon system is easy and intuitive to get started and to use
  + At least as easy as the traditional method.

Considering the attendant application, attendants should experience

* A feeling of control in terms of checking cars
* That it is easy to check cars and the response is quick enough
  + On average should not make work more uncomfortable
  + Should make validity checking in winter conditions more comfortable
* That the application is easy and intuitive to get started and to use
  + Assuming that all cars would use SPARK, it should be at least as easy as the traditional method. As requested by the customer, the parallel usage of the traditional method and SPARK is not considered as a high priority required functionality of this project, so the user experience of this scenario is not relevant to be discussed.

###### Usability testing

Moreover, it is essential to make usability goals measurable and conduct test with real users, after all, user is the one who makes final judgement on the product. Generally speaking, once the interactive prototypes completed, it can be validated by real users through feedback. Furthermore, there are serval methods usually employed in validation stage:

* Observation
* Usability testing
* User interview
* Survey.

However, some test methods may cannot be usable in this project, as it seems hard to contact to end user, like finding out an attendant to test on application, so attendant app testing can be done likely by project members through simulation attendant job. Also the project course schedule is too tight for an actual exhaustive usability testing. Similarly, driver app testing done by ourselves or if possible real users, and interviews can be involved in test phase as well.

Attendant should face the need to wipe snow off from cars' windshields less often when using SPARK. This is measured by testing the parking validation in snowy conditions in the winter time after we have gotten the beacon system working.

### Performance goals

* Applications (attendant gear and driver application) have to be fluid and reactive which mean without noticeable slowdown
* Applications' performance are identical on all supported versions of Android
* The updates of parking lot occupancy are done automatically every five minutes and synchronized with applications in real-time

### Reliability goals

Following requirements are deliberately low, since our product is just a prototype. Finalized product would have higher reliability expectations.

* Communication between beacons and driver applications must work over 90% of the time from inside of a car to outside of the car regardless of weather conditions.
* Communication from beacons to attendant gears must work over 90% of time regardless of weather conditions.
* Communication between smart meter and applications must work 90% of time regardless of weather conditions.
* If the wireless connection is assumed to be perfect and online, then the connection between the applications/smart meters and the cloud should work at the rate that Google Cloud Platform's service lever agreement promises the service's uptime rate to be.
* The predictive capacity of the analysis for the parking availability map is guaranteed to be as good as the null model. I.e. searching for an available parking lot with the map is guaranteed to be at least as good method for finding a free parking lot as looking for a free parking lot randomly.

### Security goals

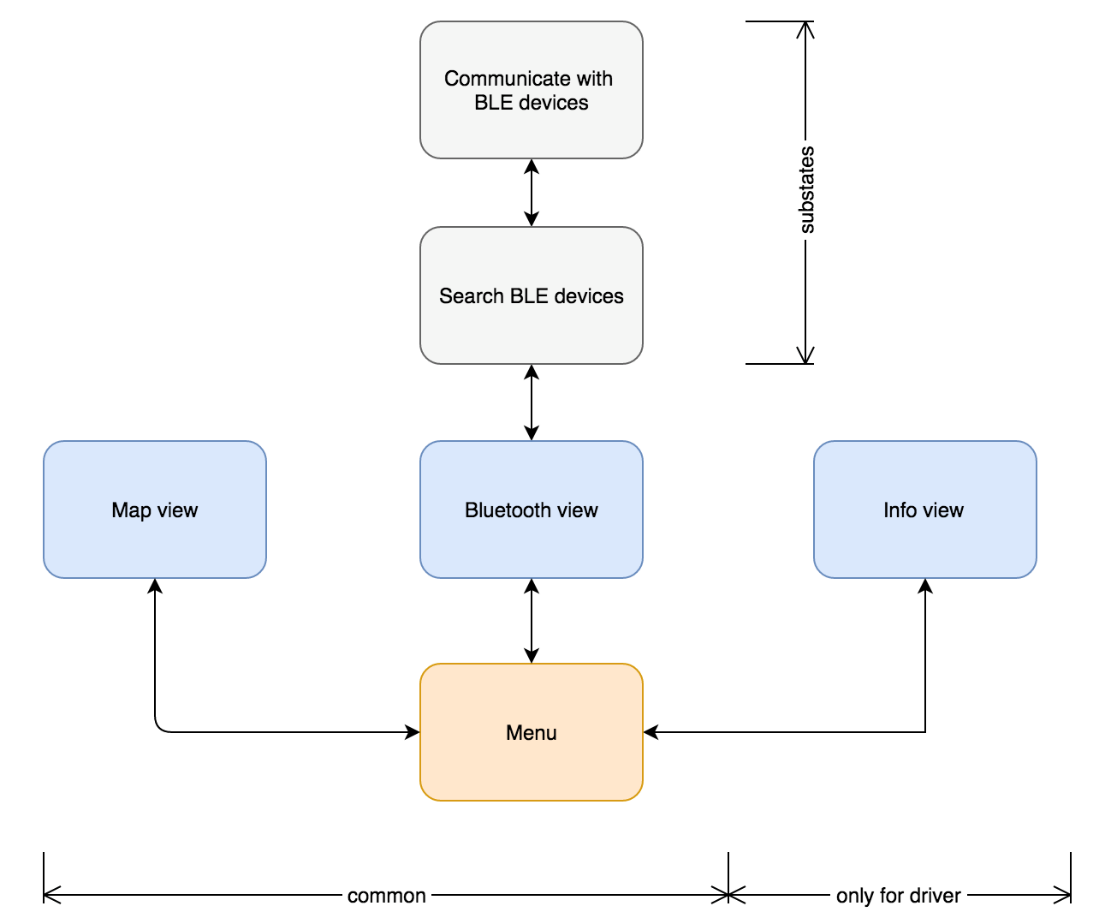
Communication between beacons and driver applications must be secure. Only the driver applications bonded with the beacon are able to send data to the beacon. Bonding process with the beacon has to be authenticated.

Tokens verifying parking validity must be digitally signed to prevent forging tokens. Exchanging tokens with smart meter should be based on encrypted communication.

There will be no personal information stored in the cloud, so no legislation regarding personal data registers needs to be taken into consideration.

## User interface requirements

Interface of the attendant application have to be simple and usable in any circumstance. User must use it (in particular check a token validity) with only one hand. The general flow of the views in the two applications is shown in Picture 3.



Picture 3. General flow of the applications.

# Test and quality assurance plan

## General approach

The testing will start on component basis as objects are developed within each part. Tests will continue with system and integration testing as system is integrated within in a bottom-up fashion.

Open source testing tools will be used for Cloud and Security testing. Cloud testing will be done with Grinder/Gatling and for security testing tools there might be few used depending on the situation and technologies used.

## Definition of done

If every test case for the requirement in the traceability matrix is passed and customer/user score is satisfactory from the UAT the feature will be considered done. Unless it the case that customer want changes made to the specific requirement or the feature. Then the necessary changes to the testing will be done and re-evaluated accordingly.

## Special testing

Tests will include three different types of testing environments to ensure system is working correctly. These include ; testing of Bluetooth devices to ensure Bluetooth devices are communicating correctly, UX testing to ensure the usability of the application meets the non-functional requirements and Cloud testing to make sure the system is responding adequately to the needs of the system and non-functional requirements.

**3.1 Testing of Bluetooth devices**

Ensuring the system runs smoothly requires the Bluetooth devices are communicating correctly. Tests will be done to validate that these BLE beacons are working correctly and relaying correct information first and foremost.

**3.2 UX Testing**

Application for the driver and attendant will include a GUI which will be tested by users. These tests will be mostly based on UAT, interviews and feedbacks. Both expert users and non-expert users will be used in these tests. As it might be difficult to find attendants to test the systems, we might have to simulate attendant work by ourselves.

Tests will be conducted by 2 test members and a expert or non-expert user where test personnel will tell user to complete certain tasks while documenting and recording the movement.

The data gotten from the user will be put in different perspectives depending on if the user is considered an expert or non-expert user.

**3.2.1 Metrics for UX Testing:**

* + - **Completion Rate :** Rate of tasks completed by user successfully.
    - **Number of clicks :** No of clicks by user to complete a certain task.
    - **Number of errors :** Number of errors made by user for a certain task.

**3.3 Cloud Testing**

As the system will have an cloud environment communicating with the other parts, there will be tests performed on cloud environment as well. These will include different types of tests however focus will be on performance aspect.

**3.3.1 Stress Testing:**

Stress testing will be performed if necessary tool will be available. It will be performed to determine the breaking point of the system.

**3.3.2 Load/Performance Testing:**

Standard load testing will be performed on the system to generate user traffic and get measurements. These measurements should be matching the requirements specified on the non-functional requirements document.

**3.3.3 Compatibility Testing :**

Compatibility testing will be used to generate traffic from different types of devices to ensure that system can respond adequately.

**3.4 Security Testing**

These security tests will consist of general integrity tests that will ensure receivers and other parts of the application will have correct information. Additionally there will be authentication and authorization tests required to ensure the application and system will run without any interferences and abuses.

There might be different open-source tools used for these tests.

## Test documentation

Test cases will be created from requirements, test cases will identify which requirement they are verifying and the test data used for testing. Depending on the result and the expected result the test case will pass or fail.

Additionally all the test cases will be documented on the test reports along with the traceability matrix where you can track every requirement that is verified or failed with regards to the requirement that is mentioned in requirements document.

# Risk management

## Risk list

An exhaustive matrix about risks involved with the project is maintained in a separate Excel workbook in the project team’s OneDrive workspace. The workbook is based on the “Risk list Excel tool”-risk list template provided by the course. For each risk, the matrix presents the title and description of the risk, the size and probability of the risk, the people monitoring that risk, the risk time window and actions for avoiding and controlling the risk. The titles and descriptions of each risk in the matrix is presented in Table 4.

Table 4. Titles and descriptions of the risks involved with the project.

|  |  |
| --- | --- |
| **Title** | **Description** |
| Motivation problems | Team member not doing his work because of motivation problems. Not attending to the meetings; not doing planned work on time; not responding to messages. |
| Inability to continue with the project for personal reasons | Team member having to quit the project for personal reasons. |
| Customer changes the requirements | Customer changes the requirements in a too late phase compared to the size of the change and the work already done. Customer is not seemingly happy with features or plans and maybe keeps asking unrelated or unexpected questions. |
| Customer not happy with the results | Customer is not happy with the actual implemented results. Customer does not think that the result follows the requirements or the requirements are not comprehensive enough. |
| Customer being too busy | Customer being too busy for doing their part of the project: not being able to assist, monitor or give feedback. |
| Not having access to Google Cloud Platform anymore | The cloud implementation in Google Cloud Platform runs out of credits or the free trial time ends |
| Team member not having enough work | Team member does not have enough work. His time is not utilised as well as it could be. |
| Bluetooth may not work inside car. | Bluetooth communication with bluetooth beacons may not succeed from outside of the car to inside of the car. |
| Brillo and Weave | Using Brillo and Wave may have unforseeable challenges. |
| Centralized HW expertice | If HW lead drops out for reason or another, rest of the team may not have neccessary skills to complete project, |
| Communication problems | There are communication problems inside the group. For example everyone not getting relevant information, misunderstandings. |
| Customer is not sure what they want | Customer keeps changing requirements or not having a clear idea of the project/ goals. |
| Customer is not responding | Customer not responding to any type of contact efforts |
| Centralized App expertice | If App lead drops out for reason or another, rest of the team may not have necessary skills to complete project. |
| Having access to devices for apps development | Members which works on the apps side of the project must have access to mobile devices for the development. |
| Someone from the app team does not understand all or any part of the code created | Members working on mobile applications need to understand the overall functioning of the source code |
| Goals/requirements not understood correctly or agred well enough | The team and the customer have different opinion of what was agreed. |
| Difficulties on usability testing | Hard to contact end user and find the real user (attendant, driver) to test the applications |

## Risk monitoring

Most of the risk-specific monitoring mechanisms were explained in section 8.1. Though, in general, the risk list will be read through by the people in charge of risks on a frequent basis. The frequency depends on the time window of each risk. If, for example, a risk's time window is one week, then the risk should be read through at least once a week.

This reading process includes:

* Updating risk information if necessary
  + If people responsible of the risk are changed, those people are informed about this
* Reminding himself/herself about the risks that should be monitored on a daily basis and to what extent. The extent is defined by the size of a risk.

The monitoring means following the signs of the risk on a daily basis. The person in charge is responsible of monitoring and making sure that he/she or someone else both keeps on doing the necessary actions for avoiding the risk and/or takes action if the risk occurs.

# Open issues

Due to missing Bluetooth hardware abstraction layer for Brillo and inaccessible tools for Weave we may not be able to use these technologies. The team keeps investigating for solution for the problems, but if there is no solution, we need to discuss with the customer about the substitute technologies.

Google Cloud Platform has been successfully tested and remote jobs ran on the platform. Yet, the final architecture of the cloud implementation is still an open issue, as the technology itself is still studied. Also the programming language to use for Google Cloud Dataflow is a question mark, as Java would be a superior choice to Python as there is a better support for it, yet there has been issues with our Eclipse environment getting authenticated with the platform. The remote jobs already ran were ran using Python, and there were no authentication issues with them.

# References

Finlex, 2011. Laki pysäköinninvalvonnasta. http://www.finlex.fi/fi/laki/ajantasa/2011/20110727

# APPENDIX A […Z]