Title page

Abstract

Table of contents

Introduction 1

Background – Kike? 1

Purpose – Kike? 1

Material – Hampus? 1

Method- Sergiu? 1

Body/result – ALL? 2

Conclusion and disscusion – ALL? 3

Sources – Kevin? 4

# Introduction

## Background

With many types of resources in the area, with plenty of seafood from the coast, fresh vegetables form the south and mushrooms from the northeast it is not weird that Gothenburg is rich in its food culture. There are more than 650 restaurants in the Gothenburg city area and over 300 more restaurants in the surrounding areas, so Gothenburg is definitely a restaurant rich city. There are four exclusive Michelin stars awarded restaurants. There are plenty of streets that are food and entertainment oriented and the culture incentivizes you to enjoy them. With the classis Swedish fika, the brunch, lunch and dinner, restaurants and cafes have a big influence in the social life of the Gothenburg inhabitants.

## Purpose

The purpose of the project is to have a more centralized and simple way to find the type of restaurant you want to visit at a specific point in time. The application will help the customer find a restaurant based on different parameters. It will also provide a way to rate them and to store those reviews in order to remember which restaurants they have visited and what they thought about it. From the restaurant owner point of view, the application will help the owner to see first hand what customers thought about their experience and to modify the information provided along with the restaurant. The application is targeted for restaurant owner who want to reach their customers in a better way and for people who want a tool to find the right restaurant experience they were looking for.

## Material – Hampus?

Netbeans

Stackoverflow

Java docs

Github (the end)

Facebook

Maven

Dropbox

Email

## Method- Sergiu?

Spring

MVC

Agile process

Given the nature of the project,

Scrum

# Body/result – ALL?

**Initial thoughts on software process**

Our initial choice of a software process was agile, due to the fact that the developing team was inexperienced.

Due to the fact that the team was inexperienced, the risk of requirements and design changes was pretty high. In order to minimize it, agile was the adequate choice as a software process. Incremental approach was proposed as a common technique of development for all members, but each member could tailor their own process when working on the scheduled tasks.

A team leader has not been chosen since none of us possessed enough knowledge to take this responsibility.

Problem – cause – solution discussions section:

Structure: Problem, cause (What caused the problem), solution (which was applied, or should have been applied).

Any or all of the subparts (problem/cause/solution) should be detailed.

* The scheduled tasks were too broad. The tasks should have been more specific so they can be assigned to team members and be monitored more easily.
* The project plan was not updated after the beta project. There was no schedule after we did the beta presentation. This activity should have been an assigned responsibility.
* Revision control was very difficult. We used facebook. We decided pretty late to use github. We decided late because we were scared of integrating one more tool into our process and having to invest more time in training. We should have used it from the very beginning because even with the time spent on learning to use github, we still could have saved a lot of time that was wasted “manual” version control and bug fixes that were caused by it.

**Mention version control on the database.**

* Test driven development was non-existent. This lead to a lot of hours wasted on testing the software for bugs.
* We lacked a team leader
* We needed a more specific and tailored software process to follow.
  1. What did we miss out exactly?
  2. We needed more prototyping
  3. We needed an iterative-incremental approach
* Dividing work proved impossible. We were lacking an architectural pattern. We chose MVC
* A lot of work was invested for developing simple functionality. The use of netbeans’ window builder and spring framework were proposed for minimizing boiler plate coding.
* Tools that were identified to increase productivity were not used. Training proved challenging and time consuming. They were identified late in the project. They should have been identified right from the start so that there would be enough time for training.
* Dependency handling proved difficult. [give examples for when we needed solution for automatic dependency handling]. [explain that Maven was used]
* Sprint workflow inspired from SCRUM was introduced later in the project. [why was it needed?] [examples of cases when it could have been used]

For the beta version we had our project plan

No common deadlines

Tasks were too broad

Monitoring proved difficulty

STORYLINE OF GROUP TASK DIVISION AND THE GROUP LOGS

Pre-Beta Version

The initial process was to divide the group into two teams. One team would take care of the GUI and the other team would set up the database. This structure was used in the mentioned incremental approach with only Facebook as a communication tool. Version control was practically non-existent. Because of correlation between the database and the program itself both teams were crossing borders all the time and trying to do what was needed to finish a part of its own team. So for example the database team did some GUI because they needed to try out their database and the GUI team did a bit of the database because they needed to test their GUI.

Beta Version

At the time the beta consisted of a database with the users, owners, and administrators table and the restaurants table. There would be a capability to filter the restaurant through different possibilities and there would be an option to add and delete restaurants. Also the users would be able to log in and out of the program and they would be able to register as new users.

Post-Beta Version

After the beta version, we received some feedback from our classmates on things to improve. The group acknowledged the fact that a lot of how things were done was not very productive. The team decided to move towards the MVC pattern basing our project on the Spring framework. Maven would be used as a dependency tool and the group would work using SCRUM as the software process. Also it was agreed better communication through the Facebook group was needed and that meetings would happen twice a week on Mondays and on Thursdays to discuss the advance and to distribute new tasks.

Final Version

By this point the advance was much better. The biweekly meetings helped to update on what advancements where being done. There was a decision to start working more closely on version control. Because of the lack of sharing of versions, the task to put the application together was becoming a nightmare. At the ends there would be four people making updates from different fronts and therefore a better version control method was needed. The group decided to adopt Guthub as the version control manager and the application grew and got debugged at a very fast pace.

what you did

how you did it

# Conclusion and disscusion – ALL?

- what major problems you faced

- how you overcame these

- how would we do it differently?

# Sources – Kevin?