**Your Name** <your-ju-id@student.ju.se>

A Project Work in *Web Development Fundamentals*

Jönköping University 2021

Your title here...

*In this template, all italic text should be removed and replaced with your own text (which should not be italic); the italic text is just a placeholder letting you know what to write there.*

*On the cover page, change to your own project name, your own name and your own JU email address.*

*You have a lot of freedom when it comes to writing this report. You do not have to use any parts of this template, but the report you write should in the end somehow (in a good way) provide the same information as indicated in this template. Most students trying to do it in their own way usually fail, so if you try that, be sure to know what you are doing!*

*Use proper sentences, paragraphs, lists, tables, figures, etc. The more figures you use, the less text you need to write, so use many of them.*

*This page should be removed. (Note: it* ***doesn't*** *say "Only the text on this page should be removed").*

*Find more tips on writing a report at* [*https://peppel-g.github.io/course-material/lectures/report-writing/*](https://peppel-g.github.io/course-material/lectures/report-writing/)*.*

*(it's amazing (scary) how many students that delete this text before they have read/understood it...)*

Table of Contents

[Introduction 3](#_Toc17118517)

[Architecture 4](#_Toc17118518)

[Database 5](#_Toc17118519)

[Graphical User Interface 6](#_Toc17118520)

[Web Application 7](#_Toc17118521)

# Introduction

*Introduce your project here. Write text that* ***indirectly*** *(look up what indirectly means if you don't know it) answers questions like:*

* *Why does the project exist?*
* *What is the project about?*
* *Who are involved in the project?*
* *What will the project result in?*
* *Who are interested in the outcome of the project?*
* *How will the outcome of the project be used?*
* *...*

*Especially, all projects are about solving a problem of some kind. Focus on describing the problem, and then derive and describe which features your solution need to contain to solve the problem in a good way.*

*Use at least one UML use case diagram that visualizes how end users will use your website, such as the one at* [*https://www.conceptdraw.com/resources/images/solutions-screens/diagramming/UML\_Use\_Case\_Diagram.png*](https://www.conceptdraw.com/resources/images/solutions-screens/diagramming/UML_Use_Case_Diagram.png)*.* ***DO NOT*** *invent your own notations for your diagrams, stick to well-known notations.*

*After having read this chapter, those that have never heard of the project before should have a good understanding of what it is about. If they would like to learn how it has been implemented, they just need to continue reading the rest of the report.*

*Anyone should be able to read this chapter and understand what is written here, even you grandpa, so don't use very technical terms unless you have to.*

# Architecture

*Give an overview of the components the website consists of (web application, database, web browsers, end-users, etc.). Visualize this in a figure and show how the different components make use of each other.*

*After having read this chapter, the reader should have a broad (but shallow) understanding of the website's internal structure.* ***DO NOT*** *describe implementation details of the individual components, you have other chapters for that.*

# Database

*Describe the database and the resources on the website in detail. What attributes do they consist of? How are they related? Where are they stored (in files on the hard-drive? In a relational database? Etc.). Visualize the resource in an ER diagram or similar, for example as the one found at* [*https://www.researchgate.net/figure/Database-Schema-for-storing-the-processed-version-of-BBC-News-dataset\_fig5\_339168671*](https://www.researchgate.net/figure/Database-Schema-for-storing-the-processed-version-of-BBC-News-dataset_fig5_339168671)*.*

*After having read this chapter, the reader should understand how the data (the resources) on the website is stored. If the reader is a new programmer that should start working on the website, she should now know what she needs to do if she wants to change the resources or add more type of resources (e.g., know how to add a new table to the database with a relation to an existing table in the database, etc.).*

# Graphical User Interface

*Describe the graphical user interface of the website. Initially this can simply be low-detailed pictures, as the ones shown in Figure 1 below (Note: the click at the top is on the "Contact" link, and not the "Home" link (optical illusion)).*



Figure 1, Low-detailed picture of a graphical user interface.

*When you have implemented the website, showing some screenshots of it here is a very good idea. Keep your low-detailed pictures even when you have real screenshots; they give a good/quick overview of the GUI.*

*After having read this chapter, the reader should understand how the end users will be able to accomplish their goals (use-cases from your use-case diagram) through the graphical user interface.*

# Web Application

*Describe implementation details of the web application. Which language have you used? Which framework have you used? Which libraries/packages have you used, and for what purpose? Has all code been written in one file? Or have you somehow structured it in multiple files? Are you using some design patterns (e.g. MVC)? Are you using middlewares? Etc...*

*You do not necessarily need to show any code to describe the implementation, but if you feel that improves the quality of the report, feel free to do that.*

*Try to use many figures. They are much easier to read than a wall of text. Use text to explain details that can't easily be visualized in a figure. File structure, code structure (functions, classes, etc.) can be visualized using various types of UML diagrams, etc.*

*Security is a very important topic when it comes to web applications. Having a sub-chapter here that describes all security vulnerabilities you have thought of (e.g. using HTTPS (and why), hashing passwords (and why), XSS, cookies, session ids, ...) is probably a good idea. It's probably a good idea to have some more sub-chapters as well, but you can figure out them on your own.*

*After having read this chapter, the reader should have a very good understanding of how the web application has been implemented. If the reader is a programmer who should start working on the web application, she should now know where to start when she should implement new features to the web application.*