<Company Name>

<Company Name>

Bike portal
Supplementary Specification

Version 1.0

Revision History

Date	Version	Description	Author
18/Mar/18	0.1	Housekeeping	Ács Dávid
19/Mar/18	1.0	Added five non-functional requirements	Ács Dávid

Table of Contents

Supplementary Specification

1. Introduction

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:
Legal and regulatory requirements, including application standards.
Quality attributes of the system to be built, including modifiability, testability, conceptual integrity, usability and performance requirements.
Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

2. Non-functional Requirements

2.1 Modifiability

- **Quality attribute definition:** How easy (or how costly) it is to change parts of the system.
- **Source of stimulus:** Developer
- **Stimulus:** Change database management system
- **Environment:** system is in development
- **Artifact:** Data access layer.
- **Response:** Database is changed without affecting the rest of the project.
- **Response measure:** amount of time spent making the change.
- **Tactics:** keep coupling low and cohesion high.

2.2 Testability

- **Quality attribute definition:** Testability is concerned with how easy it is to create tests and execute the tests.
- **Source of stimulus:** Developer
- **Stimulus:** new feature is added, needs to be tested
- **Environment:** system is in development
- **Artifact:** part of the system to be tested
- **Response:** developer can inspect the state values.
- **Response measure:** amount of time spent writing and executing the tests
  Test coverage
- **Tactics:** Keeping the design modular, so objects can be mocked.

2.3 Conceptual Integrity

- **Quality attribute definition:** How consistent and coherent is the design and code of the application.
- **Source of stimulus:** Maintenance developer
- **Stimulus:** bug found in part of the program.
- **Environment:** system is deployed.
- **Artifact:** buggy code.
- **Response:** system is easily comprehended, and bug corrected.
- **Response measure:** amount of time spent making the change.
- **Tactics:** coding standards, variable naming conventions

2.4 Usability

- **Quality attribute definition:** How intuitive is the application is to the average user. Satisfying all the requirements of the user. How easy it is to accomplish a desired task.
- **Source of stimulus:** end user
- **Stimulus:** user makes a mistake
- **Environment:** normal operation

- **Artifact:** system
- **Response:** system provides and undo or a cancel
- **Response measure:** user satisfaction
- **Tactics:**

2.5 Performance

- **Quality attribute definition:** performance can be broken down to two subcategories: throughput and latency. Throughput is concerned with how many events/requests are handled on a large period, while latency is how fast individual events are handled/reflected to the user.
- **Source of stimulus:** User
- **Stimulus:** Button click
- **Environment:** system is operating normally
- **Artifact:** complete system.
- **Response:** system performs operation related to button click.
- **Response measure:** the amount of time between button click and response.

3. Design Constraints

The software language used for this project is C#.

As the main development tool Visual Studio 2017 Community edition will be used.

As Relational Database management system MSSQL will be used.

The Entity framework will be used to make the life of the developer easier and the developer process less error prone.