Software Requirement Specification

**for**

**“DudeWheresMyBike” application**

Project Code: COMP30830

Document Version: 1.0

****

Prepared by Tomás Kelly / Jörg Striebel / Will O’Donohoe

Student-IDs: 21207079 / 21205773 / nnnnnnnn

UCD School of Computer Science

13th February 2022

**Table of contents**

[**Introduction**](#_heading=h.gjdgxs) **2**

[**Purpose**](#_heading=h.30j0zll) **2**

[**Product Scope**](#_heading=h.1fob9te) **2**

[**Overall Description**](#_heading=h.3znysh7) **3**

[**Product Perspective**](#_heading=h.2et92p0) **3**

[**Product Functions**](#_heading=h.tyjcwt) **3**

[Finding the Best Station to Start a Journey](#_heading=h.d5htbkg179sh) 4

[Finding the Best Station to Return a Bike](#_heading=h.fx9dkj885t3t) 4

[Displaying Information about Recent Occupancy Trends](#_heading=h.nlav2ppkoh5q) 4

[Displaying Weather Information](#_heading=h.5m0nplbhuxct) 4

[**User Classes and Characteristics**](#_heading=h.3dy6vkm) **5**

[**Operating Environment**](#_heading=h.1t3h5sf) **5**

[**Design and Implementation Constraints**](#_heading=h.4d34og8) **5**

[**User Documentation**](#_heading=h.2s8eyo1) **5**

[**Specific Requirements**](#_heading=h.17dp8vu) **6**

[**External Interface Requirements**](#_heading=h.3rdcrjn) **6**

[User Interfaces](#_heading=h.64i66vhlcje4) 6

[**Functional Requirements**](#_heading=h.lnxbz9) **7**

[Finding the Best Bike Station Nearby](#_heading=h.42mlqwlahbg7) 7

[Feature x](#_heading=h.d536fyyse7g5) 7

[**Non-Functional Requirements**](#_heading=h.35nkun2) **8**

[Performance Requirements](#_heading=h.qqjh7lp0d7mk) 8

[Security Requirements](#_heading=h.bxwsv7ewza7s) 8

[xx](#_heading=h.tocl7iy3itq1) 8

[**Software Quality Attributes**](#_heading=h.1ksv4uv) **8**

[Reliability](#_heading=h.qthk005iyrnm) 8

**List of figures**

[Fig. 1 - Wireframe draft 6](#_heading=h.26in1rg)

**Document Revision History**

| Name | Date | Reason For Changes | Version |
| --- | --- | --- | --- |
| Jörg Striebel | 13/02/2022 | The first draft created | 0.1 |
|  |  |  |  |

# Introduction

## Purpose

This document specifies the Software Requirement Specifications (SRS) for the “Dude, where’s my bike” web application. It is aimed at stakeholders, such as our professor, product owner, and developers involved in the project. This document describes both functional and non-functional software requirements of “Dude where’s my bike”.

## Product Scope

The web application aims to help Dublin Bikes users find the closest station with bikes available to start their journey. Moreover, it will also support the user in finding bike stations with free spaces to return their bike.

Another aim of this application is to display relevant and valuable information based on recent occupancy trends. The user will see availability trends for each station in the upcoming days. Furthermore, weather information will also be incorporated into this application to make more precise occupancy predictions.

The service will be freely available, and no registration process will be required to use the service.

# Overall Description

This section aims to provide an overview of the “Dude, where’s my bike” web application and its related systems. Moreover, the core functionality of the app and the interaction between the web interface and other related systems/components will be outlined in this chapter.

## Product Perspective

The application will be implemented as a frontend and backend application. The Amazon Web Services (AWS), more precisely an Amazon EC2, will be hosting components running on the server-side. These components will be comprised of:

* A scrapper scheduler. This scheduler will constantly run a Python script to scrape dynamic data related to bike stations and occupancy in the Dublin centre. This dynamic data will be retrieved every 5 minutes via the JCDecaux developer API. JCDecaux is a multinational advertising company that runs bike-sharing schemes in many cities.
* A MySQL database to store retrieved dynamic data. In addition, this database will also store static data such as names, coordinates, addresses, etc., of all bike stations.
* A Python flask application will provide an API for the HTML/JavaScript frontend application.

The frontend web application will display the station data on a Google map, indicating the occupancy and availability through coloured/sized markers. Additionally, hourly/daily details regarding occupancy will be shown when a station is clicked or hovered over. Moreover, the station will also display the weather forecast and the predicated occupancy, relating to a Machine Learning (ML) algorithm that will correlate occupancy trends with weather information.

## Product Functions

The primary product functions of this application can be categorised into groups: finding the best station to start a journey, finding the best station to return a bike, displaying information about recent occupancy trends, and incorporating weather information.

### Finding the Best Station to Start a Journey

Starting a new bike journey should provide the following features:

* Displaying all available bike stations with bikes available on a map so that the user can easily find the closest station nearby.
* Showing how many bikes/e-bikes are currently available when clicking or hovering over a particular bike station.
* Displaying the selected station's name, address, and location when clicking or hovering over a particular bike station.

### Finding the Best Station to Return a Bike

Finishing a bike journey should provide the user with the following features:

* Displaying all available bike stations with free spaces available so that the user can quickly return their bike to the closest station nearby.
* Showing how many free spaces are currently available when clicking or hovering over a particular bike station.
* Displaying the selected station's name, address, and location when clicking or hovering over a specific bike station.

### Displaying Information about Recent Occupancy Trends

Based on recently stored occupancy trends in the database, the user should be provided with an occupancy trend comprised of the following features:

* Displaying predicted occupancy trend based on data retrieved within the last 14 days.
* Allowing the user to select the occupancy trend shown for each day of the week individually.
* Once the day is set, the user can view availability trends in one-hour intervals.
* Adjusting the hourly availability can be done by utilising a slider.

### Displaying Weather Information

The aforementioned occupancy trends should correlate with the weather data based on recently stored weather information in the database.

* This information could be yielded by correlating the timestamps of occupancy trends and weather data.
* Also, the current weather should be displayed for each station when clicking or hovering over it (for simplicity, it can probably be assumed that the weather information displayed will be the same for all existing stations in Dublin).
* Moreover, the weather forecast for the next three days should be displayed for each station when clicking or hovering over it.

## User Classes and Characteristics

xx

## Operating Environment

xx

## Design and Implementation Constraints

xx

## User Documentation

xx

# Specific Requirements

This chapter specifies the functional and non-functional requirements of the application. A requirement has several properties, chief among which are:

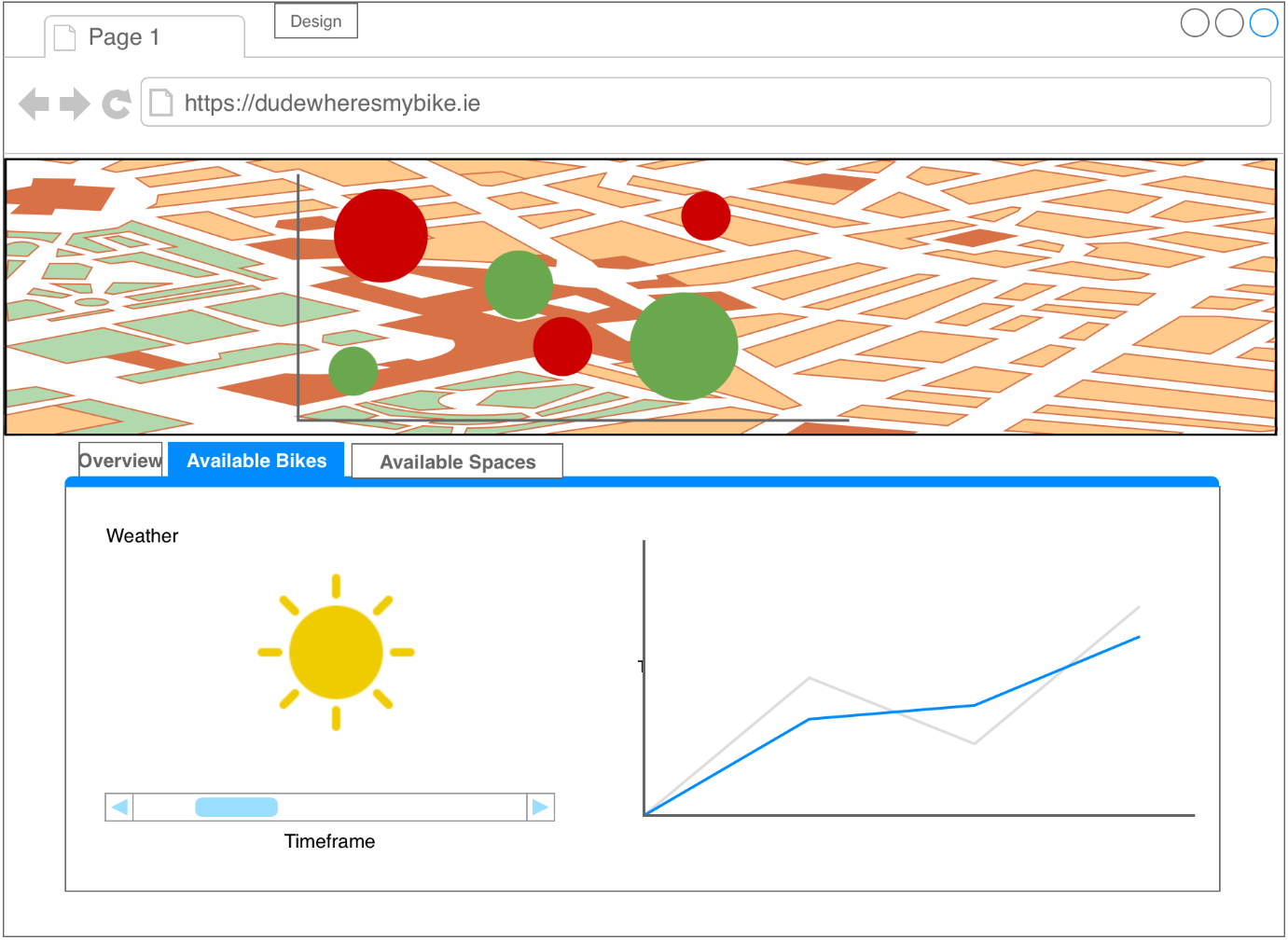
**Priority**: Priorities are ordered in descending order, starting with the highest priority (1). Functional requirements with priority (1) must be implemented in the first release.

**Risk:** The risks and consequences of not implementing a specific requirement are categorised as follows:

* **Critical:** The system cannot be used without this feature
* **High:** It highly impacts the main functionality; system parts might be inaccessible
* **Medium:** Some systems features will be affected, but the overall system is operational
* **Low:** Only minor issues in terms of usability

## External Interface Requirements

### User Interfaces



*Fig. 1 - Wireframe draft*

## Functional Requirements

This section comprises functional requirements, which are organised by functionality.

### Finding the Best Bike Station Nearby

This paragraph describes the requirements for finding the best bike station nearby.

#### Feature x

| **Req. ID:** | R1.01.01 |
| --- | --- |
| **Title:** |  |
| **Description:** |  |
| **Priority:** |  |
| **Risk:** |  |
| **Dependency:** |  |

## Non-Functional Requirements

This section describes non-functional requirements, such as performance and security.

### Performance Requirements

| **Req. ID:** | Rx.xx.xx |
| --- | --- |
| **Title:** |  |
| **Description:** |  |
| **Priority:** |  |
| **Risk:** |  |
| **Dependency:** |  |

### Security Requirements

This section specifies requirements regarding data privacy and the use of data created by the system.

#### xx

| **Req. ID:** | Rx.xx.xx |
| --- | --- |
| **Title:** |  |
| **Description:** |  |
| **Priority:** |  |
| **Risk:** |  |
| **Dependency:** |  |

## Software Quality Attributes

This section specifies quality measures such as reliability and availability.

### Reliability

| **Req. ID:** | Rx.xx.xx |
| --- | --- |
| **Title:** |  |
| **Description:** |  |
| **Priority:** |  |
| **Risk:** |  |
| **Dependency:** |  |