Salil Belapurkar

Bike Tour API Architecture V 1.0

## Purpose:

This document provides a comprehensive architectural overview of the Bike Tour Web API. It is intended to capture and convey the significant architectural elements to provide a high level overview of the Bike Tour Web API. It is intended to capture decisions that have been made for the Bike Tour Web API. The document provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases.

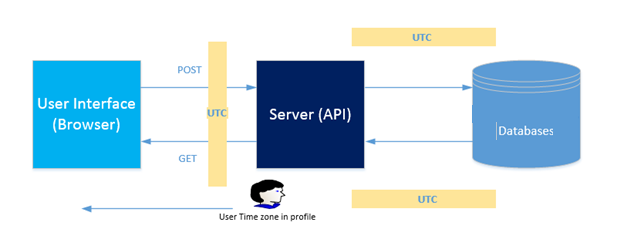
This document aims to provide technology and architecture drivers and constraints for the Bike Tour Web API.

## Scope:

The scope of this document is to detail out the architecture of 2 API’s

1. Login
2. Upload .GPXs files.

## Architecture Diagram:



* To scale to allow addition of new features in future without impacting the existing application architecture and design
* Application to be designed using SOLID principles to address separation of concerns. Each layer will be implemented independently so that UI rendering does not depends on the way data computation is being performed. It is also expected to have new features being added in future which should can be done without impacting the existing components
* Addresses security considerations – Authentication and authorization for application users, the application authenticates that users are who they say they are, and that they take only authorized actions.
* Provides customizable, extensible and maintainable application-framework. Should allow support team to modify, extend and troubleshoot the framework (as well as application) as appropriate for the given requirement.
* It should provide ready availability to common operational services (e.g. exception handling, logging, security).
* Should be multi-layered.
* Should be designed adhering Object Oriented Principles
* Should use industry standard practices for routine framework activities (e.g. exception, logging, validation etc.)
* Should be based on Loosely Coupled Layers and Components
* Should use industry-standard common patterns and practices at appropriate levels and layers.