# **Introduction**

1.1 **Purpose**

This document describes the architecture and design for the Dive Logger mobile application. The document contains an overall view of the technologies used, architecture, design and features. Document will provide in depth information about this software project and act as an introduction and guide to the software for all stake holders.

1.2 **Scope**

Dive Logger is a cross-platform mobile application which aims to bring PADI(Professional Association of Diving Instructors) paper log book features to mobile platform. The target users of this product are divers, diving instructors or anyone interested in diving who might find it useful.

Application let’s user log his dives, store them at the device and retrieve them with ease later. This feature could be used as a main record storage for the diver, or as a backup to the dive log book.

Other features include diver’s post board and a map which view’s known dive sites on the map.

Divers post board: a feature of Dive Logger application where user of an app can post their messages to the public message board for other users to see. Messages are saved to remote database and could be used by users for various reasons.

Dive sites map: a distinctive attribute of this application is a dive site location map view. Taking and using mobile device capability of tapping into GPS data and providing its location, application can view dive sites near user. Also, user can browse the map and focus on location of interest and view dive sites on map. Locations of sites are retrieved from http://divesites.com website.

1. **System overview**

The system is designed to take advantage of mobile device provided functions and to enable user to utilize them. Location and networking services are two main points of focus of Dive Logger application, providing two core features of an app.

Networking is used for user registration and authorization, message posting and map feature. Application depends on internet connection and would have a very limited ability if network connection not available. Each system user must create an account, information will be stored at remote database. Authorization request implemented and process data stored at user database to authenticate user. Two other distinct functions of Dive Logger app - message board and dive site locations viewing - solely relies on network connection being available.

Implementation of mobile platform capability to provide GPS coordinates is a core function of Dive Logger application. Dive Sites page uses location services to track user’s current position and implements map to provide information to user. Using mobile device native tools to get its location, which is then used to populate map with dive site information, that is a default behavior of a map page. To fully utilize mobile device attribute, users can browse the map and system updates information depending on location of view. Information is acquired from divesites.com webpage using HTTP request and processing response.

1. **System Architecture**

System architecture is based on MVVM design pattern. This way the system would be easy maintainable and expandable. Responsibilities of a system were partitioned and then assigned to one of the MVVM parts. This application provided a need for two more separate abstractions – Utilities and Validations.

The system composes of:

View – a view of an application which is presented for the user. This is a UI implementation and provides way present data for user and receive it.

Model – holds the data and business logic.

View Model – presentation of data to and from model.

Utilities – a helper objects for other components of a system. Includes Validations helper subsystem which deals with validations of various actions.

Rationale for choosing this MVVM design approach was that mobile devices are always constantly and rapidly changing, and this design provides loose coupling between components and changes could be reflected in system implementation quickly.

**3.1 Architecture decomposition**

**View**

This part of the system is solely responsible for presenting applications output to user and in turn receiving it. Mainly created using XAML declarative language. The input received from user is sent to View Model component for processing. To forward data for View Model data binding is used.

**View Model**

Stands behind View and is connected using data binding. This part of the system gets information from View and processes it as required per its function. Another responsibility of this component is to send that data to Model for saving. View Model acts as a bridge between View and Model and provides loose coupling between the too.

**Model**

Holds data about the concept it represents. In this system design it has very few operations on data it holds.

**Utilities**

A collection of helper instances of the designed system. Each instance is responsible for very specialized task or action and is loosly coupled with other parts of the system.

1. **Technologies used**

Dive Logger application makes use of technologies like Xamarin.Forms.Maps, Geolocation, MongoDB and FireBase API.

**Maps –** native maps of each device are used to provide visual interactive map for user. It is closely used with Geolocation library to provide experince for user.

**Geolocation –** used to tap into GPS data and receive user device coordinates whicih is represented in map.

**Mongo DB -**  used for user registration and login(later changed to **FirebaseAPI**). Also it is used for holding user posts. The interaction between application and Mongo DB is done in JSON format, so it is widely accpeted and easy to change.

**Firebase API –** in later stages of system design it replaced Mongo DB in user athentification and registration functionality. It took away a lot of work in user input validation and proved to be faster. In future it should replace the Mongo DB entirely as it really reduces amount of work and is faster in providing respones to requests.