# High Level Design Report

# 1. Introduction

## 1.1 Purpose of the system

## 1.2 Design goals

## 1.3 Definitions, acronyms, and abbreviations

## 1.4 Overview

# 2. Proposed software architecture

## 2.1 Overview

There will be two main components of the SUN system: a backend component to answer the requests and a mobile application that will be a client to the backend. For this reason a client-server architecture is selected.

## 2.2 Subsystem decomposition

Aforementioned system components had three main responsibilities. There is a component that keeps the data for the system, a component that manages the application logic and finally a presentation component. For this reasons the Three-Tier architecture was a logical choice.

On Three-Tier architecture every layer acts a client to the layer below and as a server to the layer above. This approach is very suitable for SUN since the presentation layer is a remote layer and cannot accept requests from the logic layer. Moreover, Three-Tier architecture also reduces the coupling between the components, which is also an important for the system.

Backend of the system will contain the data and the logic layers of the system while the mobile application is only responsible for the presentation. However, iOS’s system architecture encourages Model-View-Controller architectural pattern. Hence, the presentation layer of the SUN is also divided into three components.

Model part of the mobile application is responsible for fetching the data from the server and sending the updates on the data. Moreover, this part also contains the visible representation of the data classes. Since the server will only send the relevant information to the mobile application, data representation on the mobile application will be similar to the actual data on the server but with some missing or additional information.

View part of the application will be implemented using the Interface Builder

## 2.3 Hardware/software mapping

## 2.4 Persistent data management

## 2.5 Access control and security

Every SUN user has a username and password pair for identification. When the user wants to use the mobile application, the correct username and password pair must be provided. A token-based authentication system will be utilized for authentication and authorization purposes.

As the first step of the authentication, API client of the mobile application will send the credentials of the user to the login end point of the API. Login system will perform the checks and generate a token for the user if the entered information is correct. Any further requests to the API will contain the authentication token in the HTTP header. Backend of the application will match the authentication token and respond the request.

While responding the request, the backend will honor the authorization information of the data. Private information about users or groups will not be included in the responses given to unauthorized users.

There will be benefits of using a token-based authentication over Basic HTTP Authentication or a similar strategy that sends username and password with each request. These benefits are, shrinking the window of opportunity of capturing the password, eliminating the need of saving the credentials of the user to the mobile device, easy revocation of authentication tokens.

In order to implement an auto login feature on the mobile application, authentication information must be kept on the mobile device. In case of the username and password based system, the authentication information will contain the personal password of the users and using a token-based system will prevent this.

Furthermore, if an attacker somehow captures the authentication token, generating a new token will revoke the captured one and the captured token will be useless.

The passwords of the users will be kept on the database by salting and hashing. Salts of the users will be unique and be generated using a secure random number generator. Then salt will be prepended to the user’s password and hashed with SHA256 hashing algorithm.

Finally, all data communication between mobile application and the backend will use HTTPS protocol for safety reasons.

## 2.6 Global software control

## 2.7 Boundary conditions

# 3. Subsystem services

# 4. Glossary

# 5. References