University of Pretoria

COS 301

CLIENT APPLICATION PROGRAMMER INTERFACE AND DATABASE

User Management Team Gamma

Student Name	Email Address
Mohammed Gangat	u17058849@tuks.co.za
Isobel Bosman	u18020519@tuks.co.za
Thato Tshukudu	u18010408@tuks.co.za
Ihlaam Abrahams	u17320462@tuks.co.za
Mxolisi Mkanzi	u18059326@tuks.co.za
Amber Grill	u18164022@tuks.co.za
Orifha Mbedzi	u17097071@tuks.co.za

Contents

1 Introduction **2**
1.1 Resolution 2

2 Database **2**
2.1 Overview 2
2.2 Design 2

3 Entity Relationship Diagram **2**
3.1 Diagram$_A$ 2
3.2 Diagram$_B$ 3

4 API Usage Information **3**

5 HTTP Requests Currently Supported By Server **3**

6 Secuirty Methods Being Revised **3**

7 References **4**

1 Introduction

1.1 Resolution

The MouthPiece Project is a simple app which will be designed, with the intent of a relaxed stress free user experience. As User Management Team it is our interest that users data is captured safely, and stored securely within a client profile database. Our application programmer interface will be built with the intent to give all moving parts of the application or web interface such as login, registration, modifications of accounts, password recovery and many more a method of streamlined authenticated access to a simplified database.

2 Database

2.1 Overview

Our database was designed to store only relevant data such as personal info and preferences. All tables are preferably normalized to Boyce-Codd Normal Form.

2.2 Design

Tables:

1. User Profile - Used to store simple user data i.e. user id (unique/primary key), first name, last name, email, username, password, date created and date last modified
2. User Template - Used to store template ids' of mouth templates in the web database paired with the user ids' that own them
3. Templates - Mock table to represent
4. Extensions - more tables can be added for normalization of authentication as well as adding more preferences

3 Entity Relationship Diagram

3.1 Diagram$_A$

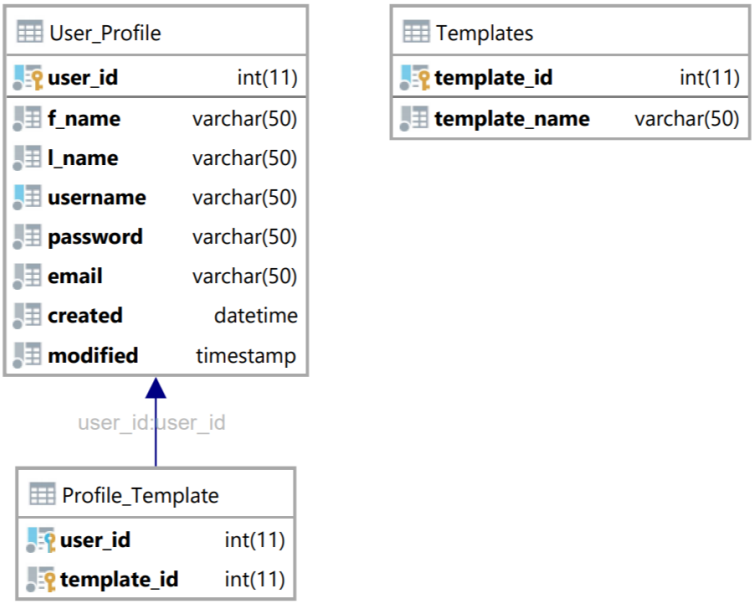

Figure 1: Basic Conceptual Entity Relation Diagram

3.2 Diagram$_B$

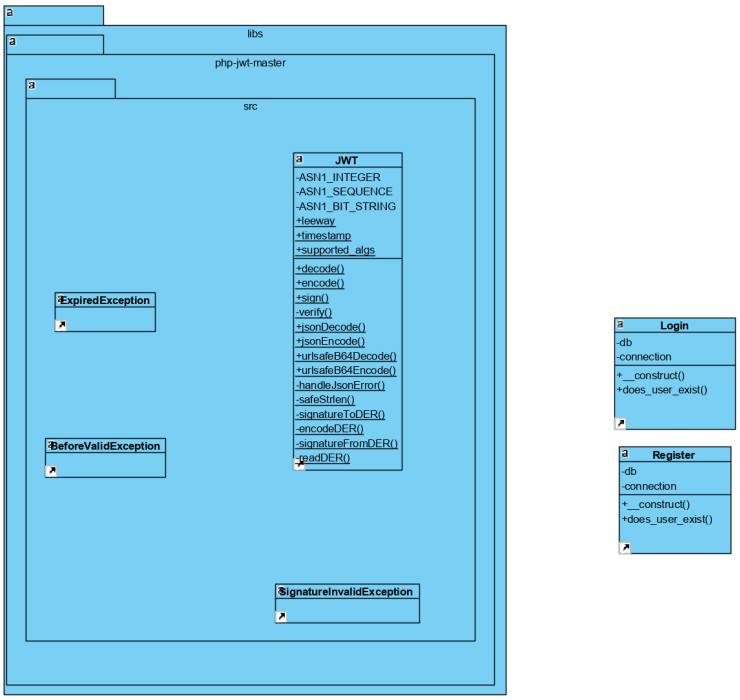

Figure 2: Login and Registration Classes with Auth Key Security Solution

4 API Usage Information

1. This API strictly only allows JSON objects via POST requests.
2. GET and un-encoded raw data will produce a failure to query the API.
3. Headers include: 'Content-Type: application/json' "Access-Control-Allow-Origin: : http://teamgamma.ga/"
4. Login API - http://teamgamma.ga/api/api/login.php -Pass a json object containing a username and password
5. Registration API - http://teamgamma.ga/api/api/register.php -Pass a json object containing name, surname, email, username and password

5 HTTP Requests Currently Supported By Server

1. Login Request example json "username": "Johndoe", "password": "doejohn"
2. Registration Request example json "l$_n ame$" : "$John$""$f_n ame$" : "Doe""$email$" : $johndoe@gmail.com$"$username$" : "$Johndoe$","$password$" : "$doejohn$"
3. List templates owned by user internal app reqest example json "user$_i d$" : "$uniqueID1$"

6 Secuirty Methods Being Revised

1. Encryption of password will be done by username salting and SHA512 encryption so as to protect sensitive info in the event of a database compromise this method maybe used for any other sensitive information

2. JWT/firebase authorization key generation. With each app being installed with its own private secret. On authorization the app will have to produce its secret to generate an authorization key which will be returned to the app via a redirect URI similar to OAuth 2.0. Skeleton code as well as libraries added and the topic is being discussed currently.

7 References

https://www.php.net/manual/en/index.php

https://auth0.com/learn/json-web-tokens/

https://www.w3schools.com/tags/ref$_h$*ttpmethods.asp*