**User API**

**Reference Implementation**

**Design Document**

Version 0.1

Mind2Mobile

12 June 2019

|  |  |  |  |
| --- | --- | --- | --- |
| Document Revision History | | | |
| Revision | Date | Description of Change | Originator |
| 0.1 | 06/12/19 | Created | Charles Stack |
| 0.2 |  |  |  |
| 0.3 |  |  |  |

|  |  |  |
| --- | --- | --- |
| Approvals | | |
| Role | Approval | Date |
| Author |  |  |
|  |  |  |

**Table of Contents**

[License and Copyright 4](#_Toc11241250)

[Introduction 5](#_Toc11241251)

[Document Purpose 5](#_Toc11241252)

[Audience 5](#_Toc11241253)

[Related Documents 5](#_Toc11241254)

[Conventions 5](#_Toc11241255)

[Acronyms & Definitions 5](#_Toc11241256)

[Design Considerations 6](#_Toc11241257)

[Technology Stack 6](#_Toc11241258)

[Platform 6](#_Toc11241259)

[Implementation Language 6](#_Toc11241260)

[Package Management 6](#_Toc11241261)

[Database 6](#_Toc11241262)

[Authentication 7](#_Toc11241263)

[Security 7](#_Toc11241264)

[Source and Version Control 7](#_Toc11241265)

[User Management Service API Details 8](#_Toc11241266)

[Overview 8](#_Toc11241267)

# License and Copyright

The content of this document is published and licensed in accordance with the MIT License. The terms and conditions are outlined below:

Copyright (c) 2019, Charles Stack. All Rights Reserved

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Introduction

## Document Purpose

This document describes the design of the User Service API Reference Implementation Service. The reference implementation exposes the User Service API and provides a set of REST based APIs intended to provide a general-purpose service for managing the users of a more complex system.

The reference implementation is not intended to be deployed “as-is”. It does, however, demonstrate how build a version of the API which is functional and provides a working version by which other implementations can test against.

## Audience

This document is intended for use by Software Development and Software Testing Teams.

## Related Documents

1. **Swagger\_UserAPI.yaml** – A Swagger 2.0 document describing the User Management Service API.
2. **OAS\_UserAPI.yaml** – An OAS 3.0 document describing the User Management Service API.
3. **UserManagement.yaml** – Contains just the User Management aspects of the Swagger\_UserAPI.yaml
4. **UserAccess.yaml** – Contains just the User Access aspects of the Swagger\_UserAPI.yaml
5. **PasswordManagement.yaml** – Contains just the Password Management aspects of the Swagger\_UserAPI.yaml
6. **AccountSecurity.yaml** – Contains just the Account Security aspects of the Swagger\_UserAPI.yaml

## Conventions

N/A

## Acronyms & Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| JSON | Javascript Object Notation |
| MIT License | A permissive free software license originating at the Massachusetts Institute of Technology (MIT) |
| OTP | One-Time-Password. A unique code generated either by an application or device the user holds or sent to the user via an Out-Of-Band (OOB) method (such as SMS). |
| REST | Representational State Transfer is a software architecture style that defines a set of constraints to be followed when creating web services. Web services that confirm to the REST conventions are known as RESTful web services and provide interoperability with other systems on the Internet. |
| SWAGGER/OAS | An open-source software framework that helps developers, design, build, document and consume RESTful Web Services. See <https://swagger.io/>  There are two specifications: Swagger (Version 2.0) and Swagger 3.0 (now known as the Open API Specification (OAS). |
| UUID | Universally Unique Identifier |
| YAML | Yet Another Markup Language |

# Design Considerations

## Technology Stack

The technology stack chosen is highly scalable and open-source. The Swagger and its successor, Open API Standard (OAS), provide a mechanism to clearly define the API in a manner that is both human and machine readable.

Platform

The reference implementation was developed on a Macintosh Pro laptop with 16 GB memory. The Operating System is, currently: macOS Mojave Version 10.14.5

## Implementation Language

The User Management Service API is, generally, programming language and platform neutral. While Swagger/OAS is used to define the API, the API can be implemented using the developer’s language and platform of choice.

This Reference Implementation has been developed using node.js. At the time of this writing, the current version of Node.js for Macintosh is 12.4.0. This version contains all the latest features. Similarly, version 10.16.0 LTS is also available.

You can download the appropriate node.js implementation from: <https://nodejs.org/en/>

## Package Management

The standard Node Package Manager, npm, is utilized to install and manage packages for this service.

npm is installed with node. At the time of this writing, the current version of npm is 6.9.0.

## Database

The design is database agnostic. However, for the reference implementation, MongoDB has been selected. Mongoose has been selected to provide object modelling.

MongoDB can be downloaded from: <https://www.mongodb.com/what-is-mongodb>

The current version of MongoDB server is: 4.0.10

The easiest way to install mongoDb is to use brew:

>brew install mongodb

At the time of this writing, brew will install: 4.0.3

Alternatively, the following link describes how to install mongoDB manually:

<https://treehouse.github.io/installation-guides/mac/mongo-mac.html>

Mongoose can be downloaded: <https://mongoosejs.com/>

MongoDB Compass provides a user friendly interface to MongoDB. The community edition can be downloaded from: <https://www.mongodb.com/download-center/compass>

To have launchd start mongodb now and restart at login:

brew services start mongodb

Or, if you don't want/need a background service you can just run:

mongod --config /usr/local/etc/mongod.conf

## Testing

In order to facilitate unit testing and promote the concept of Test-Driven Development (TDD), the reference implementation shall utilize the Mocha and Chai packages for node.js.

## Development Environment

The choice of a development environment is, usually, a personal choice. While some may prefer using a terminal and text editor, others prefer interactive development environments (IDEs).   
  
The preference for this project was Jetbrain’s WebStorm IDE. This tool has excellent support for Node.js and Swagger development.

## Source and Version Control

The API source document, source code, and related documentation is maintained on GitHub.

# Building the User Management Service

## Overview

The initial source code was generated using the Codegen tool from within the Swagger Editor ([https://editor.swagger.io](https://editor.swagger.io/)). The Swagger code file, was imported into the editor. Then, the “Generate Server” menu, the “nodejs-server” option was selected.

The output from this operation is a file called “nodejs-server-server-generated.zip” and contains a complete mock server using the Swagger tool suite.

While other options exist, such as Express, for creating a REST server, this project uses the Swagger libraries. As a side-note, it is still possible to build the microservice using Express and incorporate SwaggerUI to present a way to document and test the server without a whole lot of effort. This, however, is outside the scope of this project.

The zip file was expanded in the developer directory, imported into the IDE, built and run. This performed all the necessary npm installs and create a working server that returns mock data.

## Establishing the Project Structure

The nodejs\_usermgmtapi.zip file, when expanded, contains the following directories and files:

<root>

* + <.swagger-codegen>
* VERSION
  + <api>
* swagger.yaml
  + <controllers>
* AccountSecurity.js
* PasswordManagement.js
* UserAccess.js
* UserManagement.js
  + <node\_modules>
    - .. lots of files
  + <service>
* AccountSecurityService.js
* PasswordManagementService.js
* UserAccessService.js
* UserManagementService.js
  + <utils>
* writer.js
  + .swagger-codegen-ignore
  + index.js
  + package.json
  + package-lock.json
  + README.md

The following should then added to produce the complete project structure

<root>

* + <models>
  + <test>
  + Dockerfile

If a Jenkins server is installed and available for use, a Jenkinsfile would also be added to implement CI/CD capability.

**Note - Implementing a Jenkins pipeline is outside the scope of this document.**

The way the Swagger file was created for the User Management API resulted in the following four services:

* AccountSecurity
* PasswordManagement
* UserAccess
* UserManagement

were implemented into this one project.

To ease development effort, the original swagger file was refactored to include only the elements relevant for each service. It is from these four Swagger files that each service was generated into its own zip file. The collection of these four (4) services are then grouped under APIs/UserAPI/Reference. Each service is modified accordingly to adhere to the skeleton structure.