colored rectangle

| SUPPORT Documentdetail of persons hands with scissors, markers, working |  |
| --- | --- |
|  |  |  |  |
|  |  |  |  |

**TABLE OF CONTENTS**colored contents page background

[1. Introduction](#_heading=h.e1nbbi1wk3da) **2**

[2. Functional Components](#_heading=h.e31ou16h0rbu) **2**

[3. Support Operations for Sittofit Website](#_heading=h.kgpihybx7gtp) **2**

[3.1 Required Activities](#_heading=h.6spzxyiibmkj) **2**

[4.1.1 Database Updation](#_heading=h.ahy0kcofqjj) 3

[4.1.2 Testing](#_heading=h.rz9onfoyputg) 5

[4.1.3 Back-End](#_heading=h.hb4c1dvvvc50) 5

[**3. Primary Operations for Sittofit Website**](#_heading=h.e2zq2xmspuer) **6**

[3.1 Start up](#_heading=h.wy5so363a1rn) 6

[1. Start Development on your Local System](#_heading=h.ssecphbw3tws) 7

[2. Setup Production Servers](#_heading=h.wx7llym1ut2w) 9

[3. Push Website Live](#_heading=h.73x4cx8oj3bv) 10

[3.2 Backup](#_heading=h.hwyxdrvnb6v7) 10

[3.3 Restore](#_heading=h.ddqx31568v31) 11

[3.4 Shut down](#_heading=h.tkxp59kqtpx) 11

[System Failures](#_heading=h.wjsieoi7m92x) 12

[5.1 Failure Points](#_heading=h.mq5buio2qcp7) 12

[5.2 Fixing Processes](#_heading=h.jid5wcqgmmg) 13

[6. Training](#_heading=h.eowmlzas3xaq) **13**

[7. Change Management Control](#_heading=h.jvi1zwicgvb3) **14**

[7.1 Extend Features](#_heading=h.9fs5llobn5a8) 14

[1. Login/Registration system](#_heading=h.4x0jqp444s5n) 14

[2. Recommendation tool](#_heading=h.180vydqbyl3m) 14

[3. Alert Reminder](#_heading=h.kn2rolbjzlcx) 15

[7.2 Transition Process](#_heading=h.3o98t0uwtpls) 15

## 

## 1. Introduction

The document describes the information required to understand the working of Sittofit website product, which encompasses the functional operations that builds the entire website. This document is intended for potential developers/project lead wherein the report will assist in technical support in order to simplify the processes associated with Sittofit website day to day activities.

The report will detail the basic operations that are required to work with the different interfaces of the website. It will guide the reader to handle the operations seamlessly with visual representations.

## 2. Functional Components

1. Displayed Content
2. Risk Analysis Meter
   1. Record user statistics
   2. Output Risk Analysis
3. Recommendation System for Physical Activities
   1. User Preference
   2. Recommendations
   3. User likes and dislikes
   4. Popular Recommendations
4. Alert Reminder
   1. Snooze/Do not disturb
   2. Reminder for Water/Activity

## 3. Support Operations for Sittofit Website

The website requires frequent maintenance to ensure the website is up-to-date and functions properly. As the website is hosted on servers that automatically update the application packages, it can lead to possible bugs due to an update in the package. Regularly updating and maintaining your site is crucial to ensuring that your site is running at full capacity.

## 3.1 Required Activities

The website will require regular maintenance activities which may occur weekly, monthly or yearly depending on the component.

| Item | Component | Description | Activity Frequency |
| --- | --- | --- | --- |
| 1 | Database Updation | Open data needs to be updated on the system | Yearly |
| 2 | Database Backup | Backup user data with logs for version | Daily |
| 3 | Log Website | Check logs of the website to see if there is any error on any of the components in loading on the user’s end. | Daily |
| 4 | Testing | * Regression Testing * Unit Testing to check if the content is accurately populated on the website | Weekly |
| 5 | Backend Server Check | * Check log on API servers to ensure API is responding * Restart API server to clear * Check database limit for space exhaustion | Daily |

### 3.1.1 Database Updation

The database houses the Open datasets that the website uses for populating information on several components of the website and the user data that is collected in Recommendation System. *The current database server is on AWS MySQL.*

The database needs to be updated as per the requirements of the open dataset.

**Open Dataset**

| OPEN DATASET | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Names** | **Physical Access** | **Frequency of Source Updates** | **Frequency of Iteration system updates** | **Granularity** | **Copyright/Licensing Details** |
| Outdoor artworks  https://data.melbourne.vic.gov.au/People/Outdoor-artworks/ue3p-kdsu | CSV | Not Provided (Latest by 2020) | Yearly | Precise location of activity | CC-BY |
| Parks and green spaces in Melbourne  <https://data.melbourne.vic.gov.au/People/Landmarks-and-places-of-interest-including-schools/j5vt-ppat/data> | CSV | Monthly | Yearly | Precise location of activity | CC-BY |
| Bicycle tracks in Melbourne  <https://data.melbourne.vic.gov.au/Transport/Bicycle-routes-including-informal-on-road-and-off-/24aw-nd3i> | CSV | Quarterly | Yealy | Coordinate data for track | CC-BY |
| Public Memorials & Monuments & Fountains  <https://data.gov.au/dataset/ds-melbourne-https%3A%2F%2Fdata.melbourne.vic.gov.au%2Fapi%2Fviews%2Fuqhf-q5h7/details?q=Melbourne%20city> | CSV | Static | Yearly | Location and data about activity | CC-BY |
| Social Indicators for City of Melbourne Residents 2020  <https://data.melbourne.vic.gov.au/People/Social-Indicators-for-City-of-Melbourne-Residents-/x4k3-uj4j/data> | CSV | Static | Yearly | Statistics by category of Physical activity | CC-BY |
| Current Weather  <https://openweathermap.org/> | API | Live | Live | Melbourne current weather | CC-BY |

**Note:** *The open datasets are updated by the source periodically.*

The above table : Open Datasets list the open datasets with the frequency of update that will be required in order to keep the data on the website up-to-date. The required personnel for this activity are Data Engineers or Data Scientists.

According to the updates on the source, the data needs to be adjusted on the system.

**Support: Changes in Database:**

Making changes on the database or adding new open data will require amending the database on the server. This will also require data processing steps to utilize the new data which is mentioned in our Data Management Plan.docx.

Note: See sections as per requirements, the data management will allow you to understand the process to connect to the database and make changes as required.

### 3.1.2 Database Backup

The database needs to be backed up with the user data as it is updated daily with new user interactions for user preferences and user ratings. Perform SQL dump on the user\_preference, user\_rating tables (part of recommendation system) by following the [documentation for AWS MariaDB instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/MySQL.Procedural.Importing.NonRDSRepl.html).

### 3.1.3 Log Website

Check logs of the website to see if it is connected to the API servers and produce the output without errors. If any error occurs, check the system failures to respond to the error appropriately.

### 3.1.4 Testing

The website and its components require frequent log checks to ensure the server is handling the traffic load and service is running optimally.

Testing documentation with steps and logs of testing phases is provided in …/2.1.2 Support Document/Testing.docx.

**Support: Regular Testing Phases**

| Type of testing | Testing Result | Procedure |
| --- | --- | --- |
| Manual Testing - Test different component | Checks the content and button response on the website is checked.  Result - Test is cleared | Each click to mouse was done for every button/ clickable section. |
| Automated Testing - Regression | As some of the new features were added during the iterations. Automation testing was done to save time.  Result - Website performs optimally in each iteration | All the click to mouse on the buttons were recorded with a tool called TestProject. |
| Automated Testing - Load | Check the performance of the website under heavy traffic load. Result - Test is cleared  Sample size - 1000  Result -   * Throughput - 601.28/min * Deviation - 39   This means our server is able to handle 601.286/ requests per minute. This is sufficient for our website therefore we have passed this loadtest. | Apache Jmeter was used to conduct Load testing for the Homepage for Sit-To-Fit.  Number of Threads: 100 (Number of users connects to the target website: 100)  Loop Count: 10 (Number of time to execute testing)  Ramp-Up Period: 100 |

### 3.1.5 Backend Server Checks

The back-end remains constant to the current deployed version and requires attention weekly in order to oversee the performance on the server. The backend logs need to be backed up weekly to review the performance. The Back-end side will be requiring attention while updating the website content or database.

The logs of the API server will contain if there is any failed response from the server, in which case the Maintenance team (developers) need to debug the condition.  
Server checks:

* For Heroku Server - Restart all Dynos of the API server (Refer [Heroku dyno manager documentation](https://devcenter.heroku.com/articles/dynos))
* For RShiny Server - Restart the application (Refer [ShinyApp server documentation](https://docs.rstudio.com/shiny-server/#schedulers-application-restarts))
* Database Server needs to be checked for space as it may exhaust, in which case the user data needs to be migrated to a new server or expand the current server limit (currently it is running on free AWS service)

## 4. Primary Operations for Sittofit Website

The purpose of this section is to cover the basic operations that are required to maintain or change the state of the application. The website is built over several components and requires all these components to work together in order to make the site active. Similarly, in order to shut down the website, these components are required to be shut to ensure all the existing nodes are closed and removed safely.

### 4.1 Start up

In order to perform a Deployment cold start (Start every component of the website for further development) to publish the website live, it is important to start all the components associated with the application. This is because the website may not function properly if some of the components are missing.

The two step process to start the further development of the Sittofit website product are:

1. **Start Development on Local System**
   1. Check the list for all the components of the website in table1.
2. **Setup Production Servers**
3. **Push Website Live**

A list of components is provided in the table 1: Components list that contains all the components for the website that are to be kept in running state.

| Component | Action |
| --- | --- |
| Database | MariaDB is deployed through a podman container with MariaDB image in Docker Hub. The podman container is running in an OpenSUSE instance hosted by Amazon Web Services.  See the steps in the …/2.1.3 Maintenance Document/Maintenance Document Section 3.1. |
| Feature1: Risk Analysis Gauge  Deployment Container | The feature is deployed over shinyapps.io provided by R. The server is running on [ShinyApp Server](https://www.shinyapps.io/admin/#/dashboard) which is a UI based dashboard that allows the access to hosted/published applications.  See steps in …/2.1.3 Maintenance Document/Maintenance Document Section 3.1. |
| Feature2: Recommendation System  Deployment Container | The recommendation system is hosted on Github, where a list of files needs to be in place in order to start the feature to be accessible for the website.  See the steps in the …/2.1.3 Maintenance Document/Maintenance Document Section 3.1. |
| Feature2: Recommendation System  API Hosting Server | The API for the application is hosted on Heroku Server from where the application needs to be up and running.  See steps in …/2.1.3 Maintenance Document/Maintenance Document Section 3.1. |

#### Start Development on your Local System

The current development source code can be accessed by taking over the git repository and downloading the progress onto the local system (Refer to documentation section to [Access Git](https://docs.github.com/en/repositories/creating-and-managing-repositories/transferring-a-repository) and [Download Progress](https://www.npmjs.com/package/download-git-repo)).

To start the development process for Maintenance or extending the current deployment of the website, follow the Website functional Dependencies Section to build the development environment.

* **Back-End**

The Back-End source code is available on Github [Sittofit-Backend](https://github.com/ogup0002/sittofit_backend). The back-end comprises functionalities of the website, including [Risk Meter in Journey Page](https://sittofit.tk/risk-meter), [Recommendation System tool](https://sittofit.tk/recommendation/cards), Database and Alert Reminder.

* Feature1: R-Shiny

| Requirements | Title |
| --- | --- |
| Platform | R Studio |
| Language | R, Rshiny |
| Server | RShiny ShinyApp Server |
| Packages | library(shiny)  library(shinydashboard)  library(flexdashboard)  library(RMySQL)  library(shinyjs)  library(plyr)  library(shinyWidgets)  library(ggplot2)  library(dplyr) |
| Other | Account on the ShinyApp server is required in order to host the application. |
| Source Code location | [Risk Analysis Gauge Application](https://github.com/ogup0002/sittofit_backend/tree/main/risk_gauge) (Git Repository)  Local file - riskmeter.R  Address - /1.3 Code and Database Folder/1.3.1 Source Code Files/1.3.1.1 Risk Meter Gauge/… |

* Feature2: Recommendation System

| Requirements | Title |
| --- | --- |
| Platform | Editor for Python |
| Language | Python, Flask, |
| Server | Heroku |
| Packages | flask -> Flask, Response  Flask\_restful -> Api, Resource, reqparse  Pymysql  Pandas  Requests  Sklearn.model\_selection -> train\_test\_split  Sklearn.naive\_bayes -> GaussianNB |
| Other | Account on Heroku is required. |
| Source Code location | [Recommendation-System-Feature](https://github.com/ogup0002/sittofit_backend/tree/main/Recommendation-System) (Git Repository)  The current running instance of  Recommendation System API is hosted on a separate repository as Heroku requires the files for the application to be in the root directory. The location is [Recommendation-System-API](https://github.com/ogup0002/recommendation_engine) (Git repository) for the source code used for the API.  Local file - recommendation.py  Address - /1.3 Code and Database Folder/1.3.1 Source Code Files/1.3.1.2 Recommendation System/… |

* Feature3: Alert Reminder

| Requirements | Title |
| --- | --- |
| Platform | Vue |
| Language | Typescript |
| Server | AWS Amplify |
| Packages | Vue, Typescript |
| Other | @vueform/toggle for toggle component |
| Source Code location | [z-sx/sit-to-fit (github.com)](https://github.com/z-sx/sit-to-fit) repository  Local file - |

* Database

| Requirements | Title |
| --- | --- |
| Platform | MariaDB, Podman |
| Language | SQL, Bash |
| Server | OpenSUSE, AWS Lightsail |
| Packages | [mariadb - Official Image | Docker Hub](https://hub.docker.com/_/mariadb) |
| Other | AWS Account is required |
| Source Code location | UI Dashboard |

#### Setup Production Servers

Once all the installation is done in the local environment, the development can begin. In order to publish the website live, it will require a set of new or existing servers (depending on the compatibility with the organizational server setup).

New Server Setup  
The website will require a set of servers listed in table : Required Servers for Website. The listed server is a personal preference and can be replaced with a different server.

**NOTE:** This list is based on Free Services that are available for a component type, for example, the Python API for recommendation system is set up on Heroku which offers a free tier plan that handles the traffic more optimally.

| Component | Hosting Server | Instructions |
| --- | --- | --- |
| Front End UI | AWS | Refer [AWS Amplify Documentation (amazon.com)](https://docs.aws.amazon.com/amplify/) |
| Database | OpenSUSE, AWS Lightsail | Refer [AWS Documentation](https://aws.amazon.com/lightsail/) |
| Recommendation System | Heroku | Refer [Heroku Documentation](https://www.heroku.com/pricing) |
| Risk Meter Gauge | ShinyApp Server | Refer [Shiny Server Documentation](https://www.rstudio.com/products/shiny/shiny-server/) |

Existing Server Setup

If the organisation has a dedicated server or existing server set up on the cloud services, there are a few checks that need to be done before migrating the services to the organization’s server.

This is based on the platform that the component/application is built upon.

| Component | Requirements |
| --- | --- |
| Front-End | The frontend is a Single Page Application built over Vue, Typescript, Pinia, and Vite. It should be compiled and bundled as a set of files with the build process. Therefore the server should at least support static site hosting. For the CICD pipeline, Nodejs environment and NPM is needed to install and use packages. |
| Database | The website component is built over a MySQL compatible database, and the server must allow exposing the MySQL port 3306. |
| Recommendation System | The tool is built over Python & Flask and uses machine learning algorithm, therefore the server should support processing and high scalability as a website instance (single user using this tool) makes multiple calls to the server. |
| Risk Meter Gauge | The Server should support packages of R and RShiny and allow third party packages to be installed on the server. |

#### Push Website Live

When the servers are set up (completed till step 2) and the container for source code is ready, the server needs to be configured to establish a connection with the source code container to deploy the component.

This can be done via connecting the server to the Github repository as displayed in [Heroku Integration documentation](https://devcenter.heroku.com/articles/github-integration).

Note: The example showcases a connection example using Heroku server, but this remains almost the same for every other server, check the documentation for server specific guidelines.

### 4.2 Backup

During the development phase, the development files must be backed up on Git server to ensure source code or development is secured during developer’s system crashes/failure.

Steps to follow to create backup of Source Code:

1. Create Git Repository (Refer [Git documentation](https://docs.github.com/en/get-started/quickstart/create-a-repo)[)](#bookmark=kix.e902v92n4sgj)
2. Push source code (Refer [Git documentation](https://docs.github.com/en/get-started/using-git/pushing-commits-to-a-remote-repository))

A list of servers and containers for the development components is in table 2: Website component accessibility.

| Server Instance | Back-up Description |
| --- | --- |
| Heroku | If using Heroku Postgres, you can create a backup by Heroku PgBackups. More detail can be read in the following link: <https://devcenter.heroku.com/articles/heroku-postgres-backups> |
| Github | API or 3rd-party tool can be used to backup the repository. A number of tools are available in the market. For more information regarding backups click on the following link:  <https://docs.github.com/en/repositories/archiving-a-github-repository/backing-up-a-repository> |
| AWS | Backups by AWS are provided which is a cost-effective solution. More information can be seen here:  <https://aws.amazon.com/backup/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc> |
| ShinyApp Server | Periodic backup can be done on your local machine. This can be further understood in the following link: <https://statistics.sciences.ncsu.edu/resources/it/shiny-server/> |

Current Deployment - The components of the website product are securely running on servers that provide functionality to Backup the running instances. Moreover, the current deployment source code for Backend services is available on the [Sittofit-backend repository on Github](https://github.com/ogup0002/sittofit_backend) and the Frontend services is available on [Sittofit-frontend repository on Github](https://github.com/ogup0002/sittofit_backend) that maintains the accessibility for the code easily and safely.

### 4.3 Restore

The Restore process for all the website components can be easily achieved as the deployment instance is generated with the latest code of the master branch in the Github repository. In case of system failure, the latest workable version will not be impacted and the build failure for deployment can be tracked in AWS Amplify portal. The latest commit can be seen with git or the github portal and can be easily fixed through git reset. As the hosting servers are directly connected to the Github repositories to automatically deploy the latest version of the code (in case of Front-End, the setup of AWS amplify pipeline has to be manually done.

### 4.4 Shut down

The shut down process follows the process of similarly accessing the components mentioned in the table1: Components list of the Startup process. However, the development code can be left as it is unless required to remove the source code in which case the repository for Front-End and Back-End needs to be deleted.

### 

### 5. System Failures

The website uses dynamic features and UI that are prone to malfunctioning and can cause the website to possibly crash. The failure points refer to potential components in a website that can have a malfunction. The identified failure points in different components during the development and testing phase are listed in table 7: Failure points in development and testing phase.

#### 5.1 Failure Points

| Component | Description | Causes | Category |
| --- | --- | --- | --- |
| Content Formatting | The layout of the content on the website front-end is not ideal for user experience. | 1. The consistency in using font family.  2. Font size, line height, letter spacing, and margin. | MINOR |
| Static Images loading error/delay | The images on the website are loading improperly or with an extensive delay. | 1. The file size of the image is too large.  2. CORS issue.  3. Request HTTP resource in HTTPS page. | MINOR |
| Page layout | The layout of the website is not fit for different screen sizes. | 1. Incorrect settings of CSS box model.  2. Browser compatibility issues. | MINOR |
| Risk analysis gauge | App fails to send content to the website or sends incorrect content. | 1. Incorrect logic while processing user input  2. Incorrect graphical representation | MAJOR |
| Recommendation System | App fails to send content to the website or sends incorrect content. | 1. Incorrect logic while processing user input (preferences, rating, userid)  2. Incorrect logic while calculating predictions for recommendation  3. Weather API fails to respond  4. | MAJOR |
| API unresponsiveness | API is down due to failure in source code. | 1. API is unable to handle traffic (API is hosted on free cloud service)  2. DDOS attack  3. Server allows a limited number of automatic deploys and will revert back to last working version | MINOR |
| Incorrect API Procedure | API calls made from the website are incorrectly processed. | 1. The API route called is incorrect  2. The API call carries incorrect data format | MAJOR |
| Database unresponsiveness | The database is unable to query the required output. | 1. Permission is not granted  2. Deadlock  3. The instance may be shut down by a maintenance event in AWS. | MINOR |

#### 5.2 Fixing Failure: Processes

In case of an imminent failure of the website, a sequential process to identify the bug and possible root cause of the failure point needs to be addressed and fixed.

The system failure can be responded with the category type of the failure as reported in the table: Failure points wherein Minor bugs can be addressed by support team whereas the Major bugs needs to be tackled by Maintenance team.

The documentation for minor system failures is available in Maintenance.docx.

## 6. Training

| Position | Roles | Skills |
| --- | --- | --- |
| Data Engineer | Create and maintain features with open datasets. Integrate with the front-end of the website and database. Update datasets,data cleaning and data wrangling. | Data cleaning, Data wrangling, MySql |
| Front-end developer | Fix UI/UX design on the website. Integrating features with the back-end. API testing. | Vue, CSS, HTML, Javascript  Internet fundamentals, Web security fundamentals, frontend build tools, Typescript |

## 7. Change Management Control

There exists possibilities to improve the processes for day-to-day operations.

* Database updation

The process can be done by employing automated scripts that load and deal with the data as they are updated by the source by using data API to fetch data rather than checking with the source for an update. This will require Data Engineer to implement a script that can be deployed on the server to run the script on a scheduler to check the data sources for updates and in the case where data is updated, it will run the data processing script and push the new data on the database.