colored rectangle

| MAINTENANCEdetail of persons hands with scissors, markers, working  REPORT |  |
| --- | --- |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTScolored contents page background

[1. Introduction](#_heading=h.it2wbzpcmo6u)2

[2. Structure](#_heading=h.iks2dlw0we4o) 2

[3. Prerequisite Development-Set Up](#_heading=h.qqqi281unx6b) 2

[4. SITTOFIT Website](#_heading=h.xr8eoj2l8qxk) 8

[4.1 Website Features: Working with components](#_heading=h.j2se2qitlpl3) 9

[4.1.1 Front-End](#_heading=h.5s90rdpzk1f2) 9

[4.1.2 Risk Meter](#_heading=h.lfds0vzhkli) 10

[4.1.3 Recommendation System](#_heading=h.m3chue5r448c) 12

[4.2 Backup: Git](#_heading=h.cmy4wlqscfg6) 17

[5. System Failures](#_heading=h.5f0sld25fu5e) **17**

[5.1 Bugs & Fixes](#_heading=h.7spooyxba1eg) 19

[5.2 Fixing Bugs](#_heading=h.eqtvzz1hf67d) 21

[6. Testing](#_heading=h.l658ycgbiy9c) 23

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 1. Introduction

It is essential to progressively maintain the application to keep the website updated and future proof. The report logs the common bugs & fixes, and how to restart the website components if required. Moreover, the document will assist the reader in understanding each component of the application to be able to maintain or make changes.

## 2. Structure

The structure of this report serializes the process to start the Maintenance of Sittofit website and will also guide through the process to add new features if required.

Structure:

1. Getting your system ready to start development - We have used a variety of Languages to build the website and as per the requirements, components are required to be installed.
2. Getting started with the development or maintenance - To begin working with Sittofit source code, we have divided the components that are being used in the website for simplicity and usability while making any changes, hence it will assist in making changes and publishing new changes to the website.
3. Maintaining website - The website relies on different components and servers to keep it running healthy. To maintain this, it requires keeping track of bugs and actively testing the website.

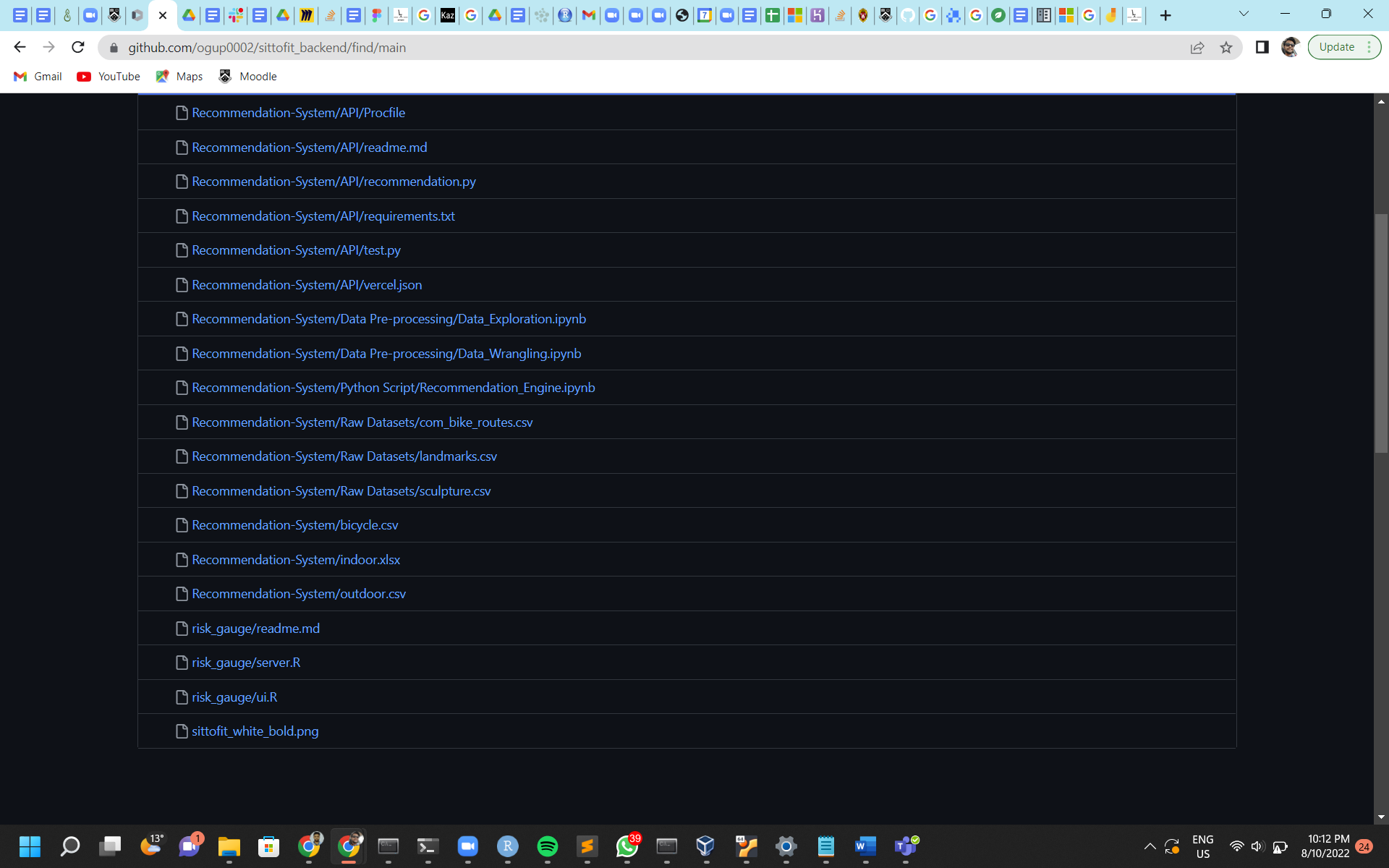
## 3. Prerequisite Development-Set Up

This is Getting your system ready to start development section:

3.1 Local Development Environment

* **Access Git Repository**

1. Download Git repository for Back-End and Front-End to local system
2. Unzip the files
3. List of files in Back-End:



1. List of files in Front-End:

| src  ├── App.vue  ├── assets  │ ├── 404.jpg  │ ├── DietaryPlanView  │ │ ├── Healthy-Diet.jpg  │ │ ├── dukan.png  │ │ ├── intermittent.png  │ │ ├── ketogenic.png  │ │ ├── mediterranean.png  │ │ └── paleo.png  │ ├── FeatureInfoView  │ │ ├── IconLifestyle.svg  │ │ ├── IconRecommendation.svg  │ │ ├── IconReminder.svg  │ │ ├── lifestyle.svg  │ │ ├── recommend.svg  │ │ └── reminder.svg  │ ├── HealthyLifestyleView  │ │ ├── activities.jpg  │ │ ├── diets.jpg  │ │ └── ergonomics.jpg  │ ├── HomeViewV2  │ │ ├── Background.svg  │ │ ├── IconAlarm.svg  │ │ ├── IconBall.svg  │ │ ├── IconPlan.svg  │ │ ├── IconSit.svg  │ │ └── pexels-fauxels.png  │ ├── Logo.svg  │ ├── PhysicalActivitiesView  │ │ ├── Physical-Activities.jpg  │ │ ├── cycling.png  │ │ ├── dog-walking.png  │ │ ├── house-chores.png  │ │ ├── rope.png  │ │ ├── running.png  │ │ ├── walking.png  │ │ ├── weightlifting.png  │ │ └── yoga.png  │ ├── PhysicalErgonomicsView  │ │ ├── IconCrossRed.svg  │ │ ├── IconTickGreen.svg  │ │ ├── Physical-Ergonomics.jpg  │ │ ├── b-1.jpg  │ │ ├── b-2.jpg  │ │ ├── b-3.jpg  │ │ ├── b-4.jpg  │ │ ├── g-1.jpg  │ │ ├── g-2.jpg  │ │ ├── g-3.jpg  │ │ └── g-4.jpg  │ ├── Recommendation  │ │ ├── IconCardio.svg  │ │ ├── IconCycling.svg  │ │ ├── IconSightseeing.svg  │ │ ├── IconWalking.svg  │ │ └── Indoor.png  │ ├── SedentaryInfoView  │ │ ├── background.png  │ │ ├── blood-circulation.svg  │ │ ├── bone.svg  │ │ ├── metabolism.svg  │ │ └── mortality.svg  │ ├── base.css  │ ├── icons  │ │ ├── IconArrowRight.svg  │ │ ├── IconArrowRightWhite.svg  │ │ ├── IconCheckbox.svg  │ │ ├── IconCheckboxBlank.svg  │ │ ├── IconCircleLeft.svg  │ │ ├── IconCircleRight.svg  │ │ ├── IconClose.svg  │ │ ├── IconGreenTick.svg  │ │ ├── IconRefresh.svg  │ │ ├── IconRestart.svg  │ │ ├── IconSetting.svg  │ │ ├── IconSmile.svg  │ │ ├── IconThumbDown.svg  │ │ ├── IconThumbDownFill.svg  │ │ ├── IconThumbUp.svg  │ │ └── IconThumbUpFill.svg  │ ├── images  │ │ ├── AlertReminder.svg  │ │ ├── DietaryPlans.svg  │ │ ├── HomeImage1.svg  │ │ ├── HomeImage2.svg  │ │ ├── HomeText1.svg  │ │ ├── HomeText2.svg  │ │ ├── IconButtonRight.png  │ │ ├── IconClose.svg  │ │ ├── IconFocus.svg  │ │ ├── IconGraph.svg  │ │ ├── IconLeft.svg  │ │ ├── IconMenu.svg  │ │ ├── IconRight.svg  │ │ ├── IconUser.png  │ │ ├── Measure1.png  │ │ ├── Measure2.png  │ │ ├── Measure3.png  │ │ ├── Measure4.png  │ │ ├── Measure5.png  │ │ ├── PhysicalActivities.svg  │ │ └── PhysicalErgonomics.svg  │ ├── logo.svg  │ └── main.postcss  ├── components  │ ├── BreadCrumb.vue  │ ├── Footer.vue  │ ├── Header.vue  │ ├── HeaderV2.vue  │ ├── JourneyHeading.vue  │ ├── JourneyLayout.vue  │ ├── JourneyLayoutV2.vue  │ ├── JourneyMeasure.vue  │ ├── JourneyModal.vue  │ ├── JourneyProgress.vue  │ ├── JourneySection.vue  │ ├── JourneySectionHeading.vue  │ ├── JourneyStat.vue  │ ├── Map.vue  │ ├── ReminderItem.vue  │ ├── SedentaryAdultPowerbiViz.vue  │ ├── SedentaryAdultTableauViz.vue  │ ├── SedentaryAdultViz.vue  │ ├── SedentaryRiskPowerbiViz.vue  │ ├── SedentaryRiskViz.vue  │ └── icons  ├── main.ts  ├── router  │ └── index.ts  ├── stores  │ ├── alert-reminder.ts  │ └── recommendation.ts  ├── sw.js  └── views  ├── AlertReminderView.vue  ├── BecomeNewUserView.vue  ├── HealthyLifestyleView.vue  ├── HomeView.vue  ├── HomeViewV2.vue  ├── JourneyView.vue  ├── NotFoundView.vue  ├── SedentaryLifestyleView.vue  ├── Test.vue  ├── healthy-lifestyle  │ ├── DietaryPlanView.vue  │ ├── PhysicalActivitiesView.vue  │ └── PhysicalErgonomicsView.vue  ├── journey  │ ├── MeasureInfoView.vue  │ ├── RiskMeterView.vue  │ └── SedentaryInfoView.vue  ├── journey-v2  │ ├── FeatureInfoView.vue  │ ├── RiskMeterView.vue  │ └── SedentaryInfoView.vue  └── recommendation  ├── PreferenceView.vue  └── RecommendationView.vue |
| --- |

* **Python**

Prerequisite for - Recommendation System

Steps:

1. Download Python 3.10.7 (most stable build) on Local system
2. Install pip on cmd: ‘python install pip’
3. Install packages using ‘pip install -r requirements.txt’

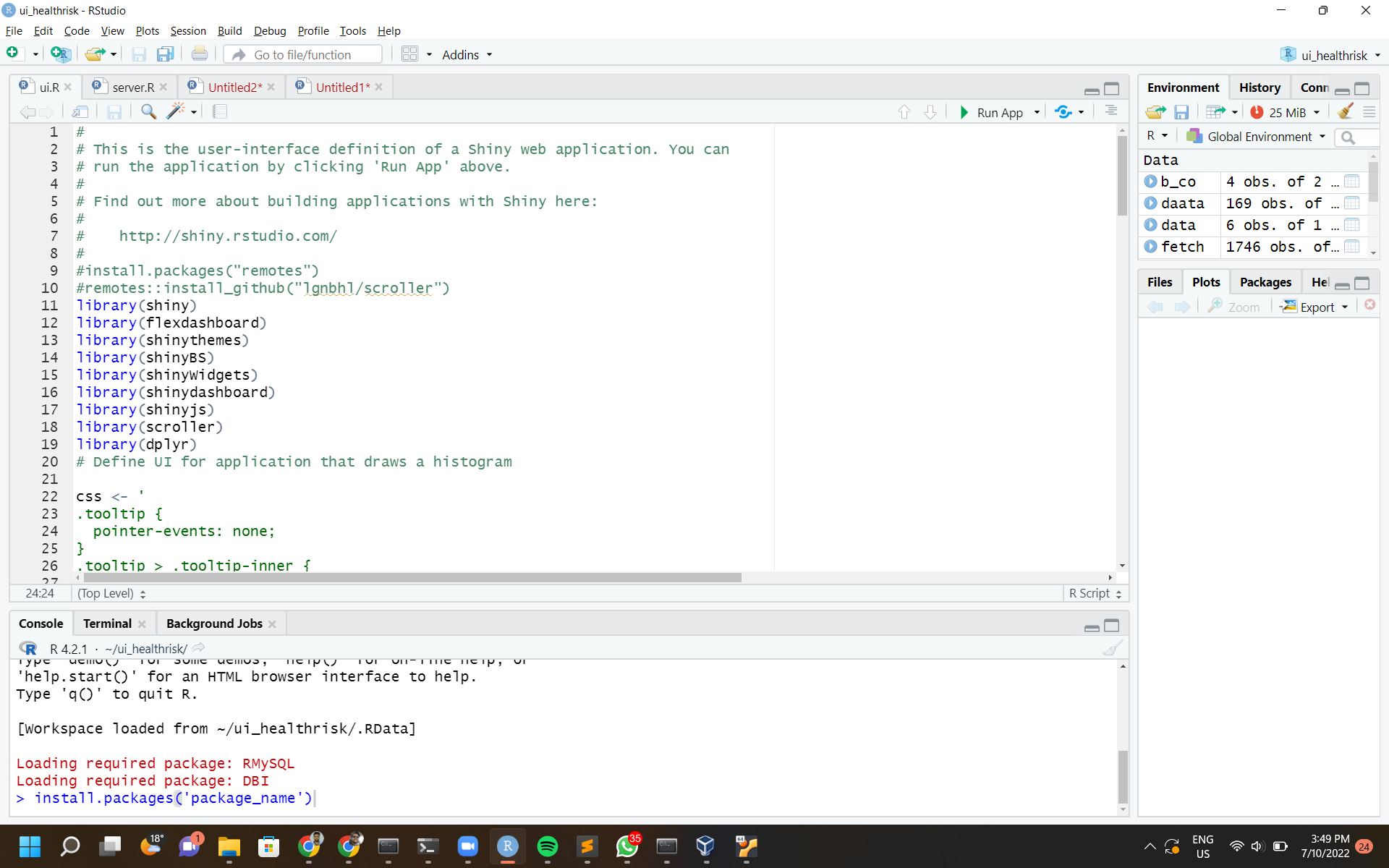
*Note: Open cmd in the directory where requirements.txt file is present?*

* **R**

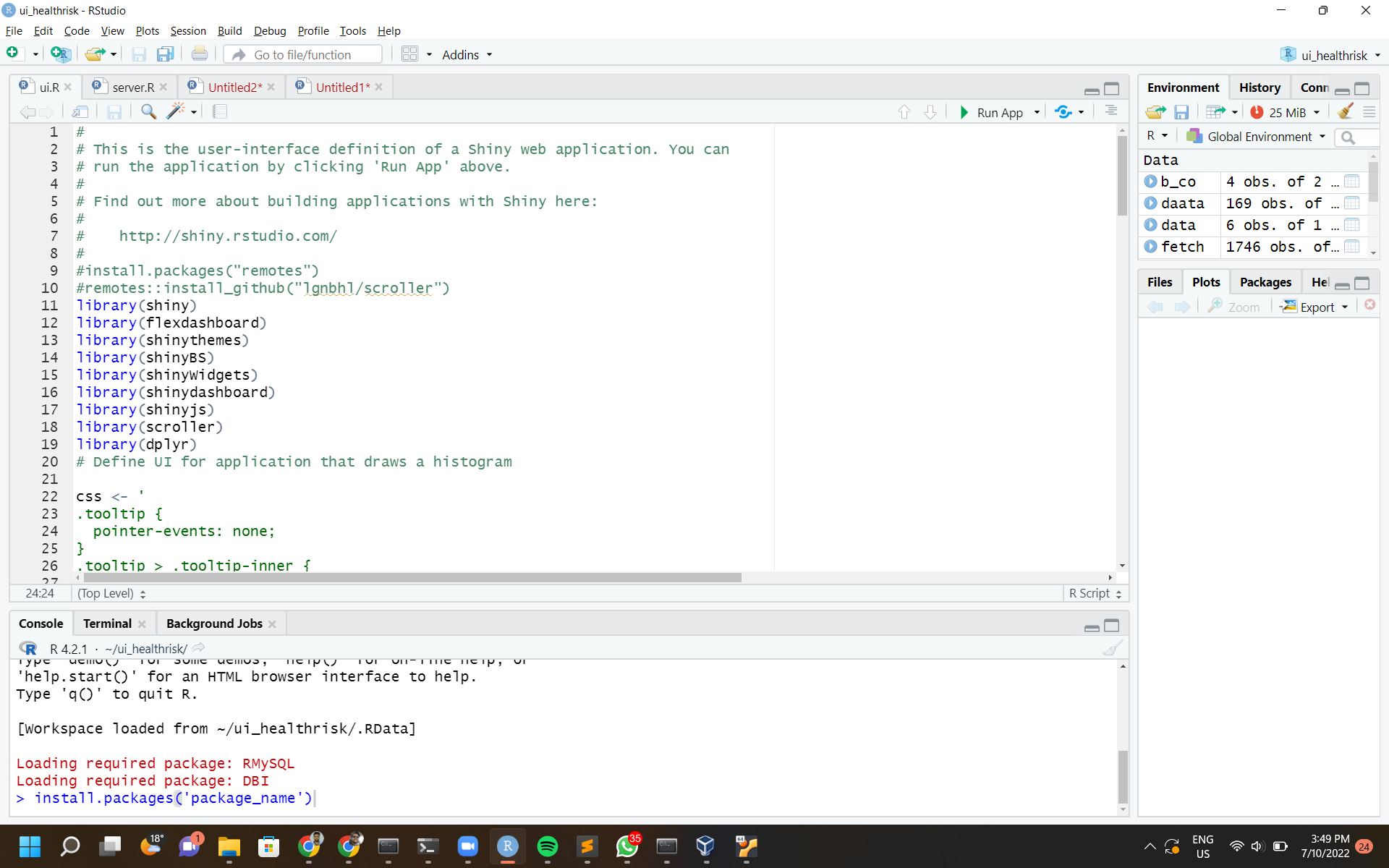
Prerequisite for - Risk Meter Gauge

Steps:

1. Download R 4.2.1 build from <https://cran.r-project.org/bin/windows/base/>
2. Install the .exe file
3. Install packages that are required to run application on local machine



1. Take all the package names in the above list and put in the package\_name position to install the package. For exame, 1st package is ‘shiny’.



* **Nodejs v16.15.1**

Prerequisite for - Front-End development

Steps:

1. Download and install nodejs

Prebuilt files and source code: [Index of /dist/v16.15.1/ (nodejs.org)](https://nodejs.org/dist/v16.15.1/)

Guide to install via package manager: [Installing Node.js via package manager](https://nodejs.org/en/download/package-manager/)

1. Install project dependencies

run command npm install in the root folder of sit to fit frontend project where you can find a package.json file.

How does npm install works: [npm-install | npm Docs (npmjs.com)](https://docs.npmjs.com/cli/v8/commands/npm-install)

* **Editor - Python, R, Front-End development**

For Python: Jupyter Notebook

1. Run on cmd - ‘python pip install jupyter notebook’
2. Run on cmd - ‘jupyter notebook’ to run jupyter notebook instance
3. Browse python file from git file downloaded in above section

For R: RStudio

1. Download R Studio (Editor to run R Script in UI environment) - <https://www.rstudio.com/products/rstudio/download/>

For Front-End development - Visual Studio Code

1. Download visual studio code from [Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com/)
2. Install visual studio code plugins:
   1. Volar for Vue 3
   2. Tailwind CSS Intellisense for Tailwind library

You can also use other web development tools that supports Vue 3 and Typescript such as Webstorm

For Front-End Vue debugging - Vue.js devtools

1. Install Vue.js devtools plugin from chrome web store [Vue.js devtools - Chrome Web Store](https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd)

* **Database Server (MariaDB v10.9.3)**

MariaDB is a RDBMS that maintains high compatibility with MySQL. In this project, MariaDB v10.9.3 jammy is used.

**Steps to setup MariaDB instance in OpenSUSE via Podman**

1. Install Podman in SLES 15, [Podman Overview | Container Guide | SUSE Linux Enterprise Server 15 SP2](https://documentation.suse.com/sles/15-SP2/html/SLES-all/cha-podman-overview.html)
2. Pull MariaDB image with Podman

| podman pull mariadb |
| --- |

[podman-pull — Podman documentation](https://docs.podman.io/en/latest/markdown/podman-pull.1.html)

[mariadb - Official Image | Docker Hub](https://hub.docker.com/_/mariadb)

1. Create a container with Podman, you can also attach a volume to /var/lib/mysql so that files in the container can be accessed in the file system.

| podman run --detach --name mariadb -p 3306:3306 -v ~/mariadb:/var/lib/mysql --env MARIADB\_USER=YOUR\_USER\_NAME --env MARIADB\_PASSWORD=YOUR\_USER\_PASSWORD --env MARIADB\_ROOT\_PASSWORD=YOUR\_ROOT\_PASSWORD mariadb:jammy |
| --- |

[podman-run — Podman documentation](https://docs.podman.io/en/latest/markdown/podman-run.1.html)

1. Configure firewall policy to expose port 3306 (the port for mariadb connection by default)
2. Grant privileges to user

[GRANT - MariaDB Knowledge Base](https://mariadb.com/kb/en/grant/)

[MySQL :: MySQL 8.0 Reference Manual :: 13.7.1.6 GRANT Statement](https://dev.mysql.com/doc/refman/8.0/en/grant.html)

* **Database**

The Database sits in the back-end for Sittofit website that handles the data in the front-end. There are two databases, one for recommendation systems and another for Risk Meter. The recommendation system uses a list of open datasets that are stored on our database.

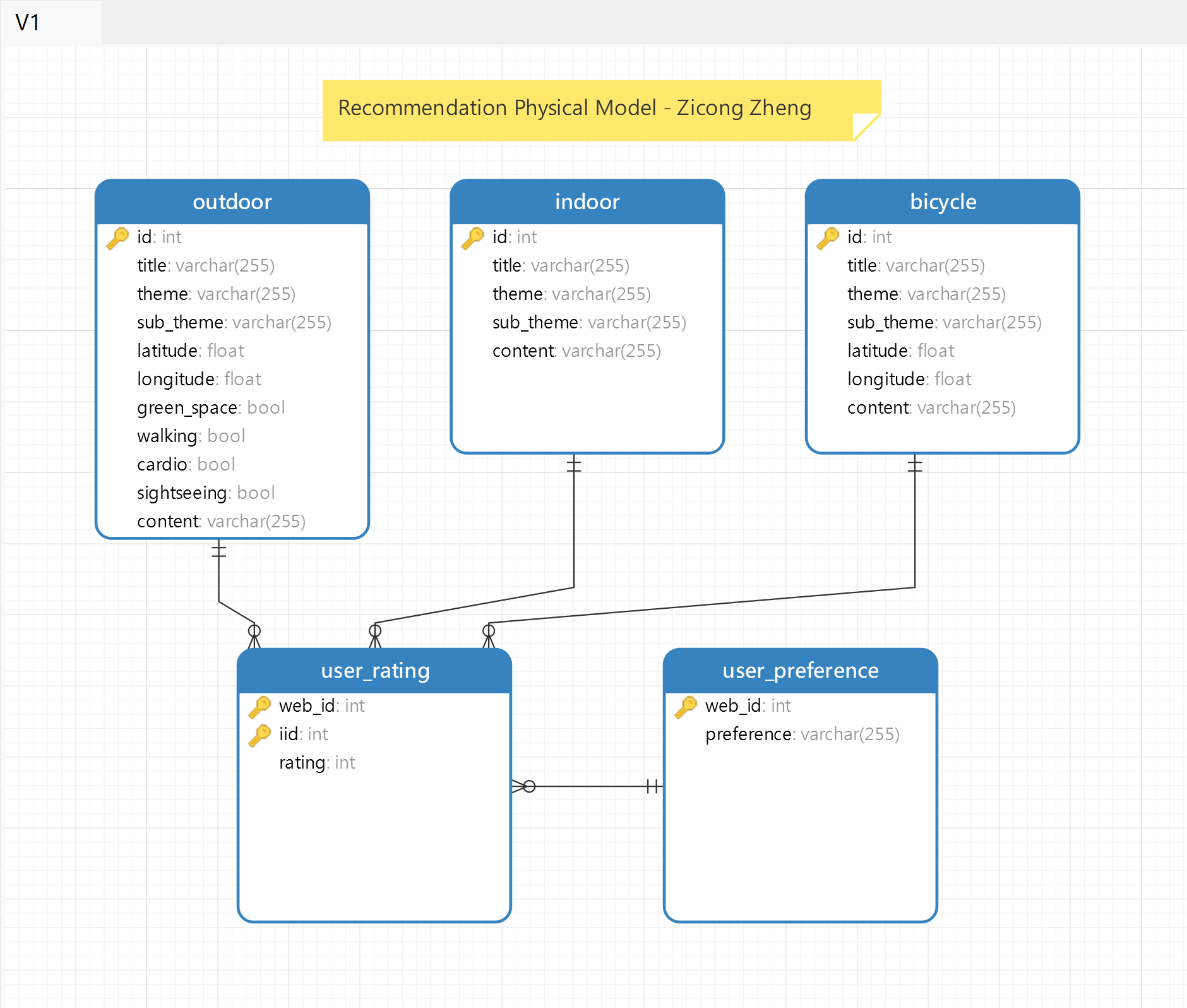
The database activities are available in …/2.1.3 Maintenance Document/Data Management Plan.docx.

With MySQL workbench or other graphical database management tools, the script of databases can be forward engineered from database models, or reverse engineered from existing databases.

[MySQL :: MySQL Workbench Manual :: 9 Database Design and Modeling](https://dev.mysql.com/doc/workbench/en/wb-data-modeling.html)

*Note: The above queries for inserting records are used in Recommendation\_system Python Script. Please refer to source code files.*

**Database Schema (Entity Relationship Diagram) - recommendation**



* **Python API Server**

The Python API server can be configured on Heroku for easy and free-cost setup. The instructions are limited to Heroku Server but other services have a similar setup. Read documentation for server specific setup.

Refer to Heroku documentation for Python API - <https://devcenter.heroku.com/articles/getting-started-with-python>

Steps:

*NOTE: Before commencing creating an application, Push files that are required for Python API on Github root directory.*

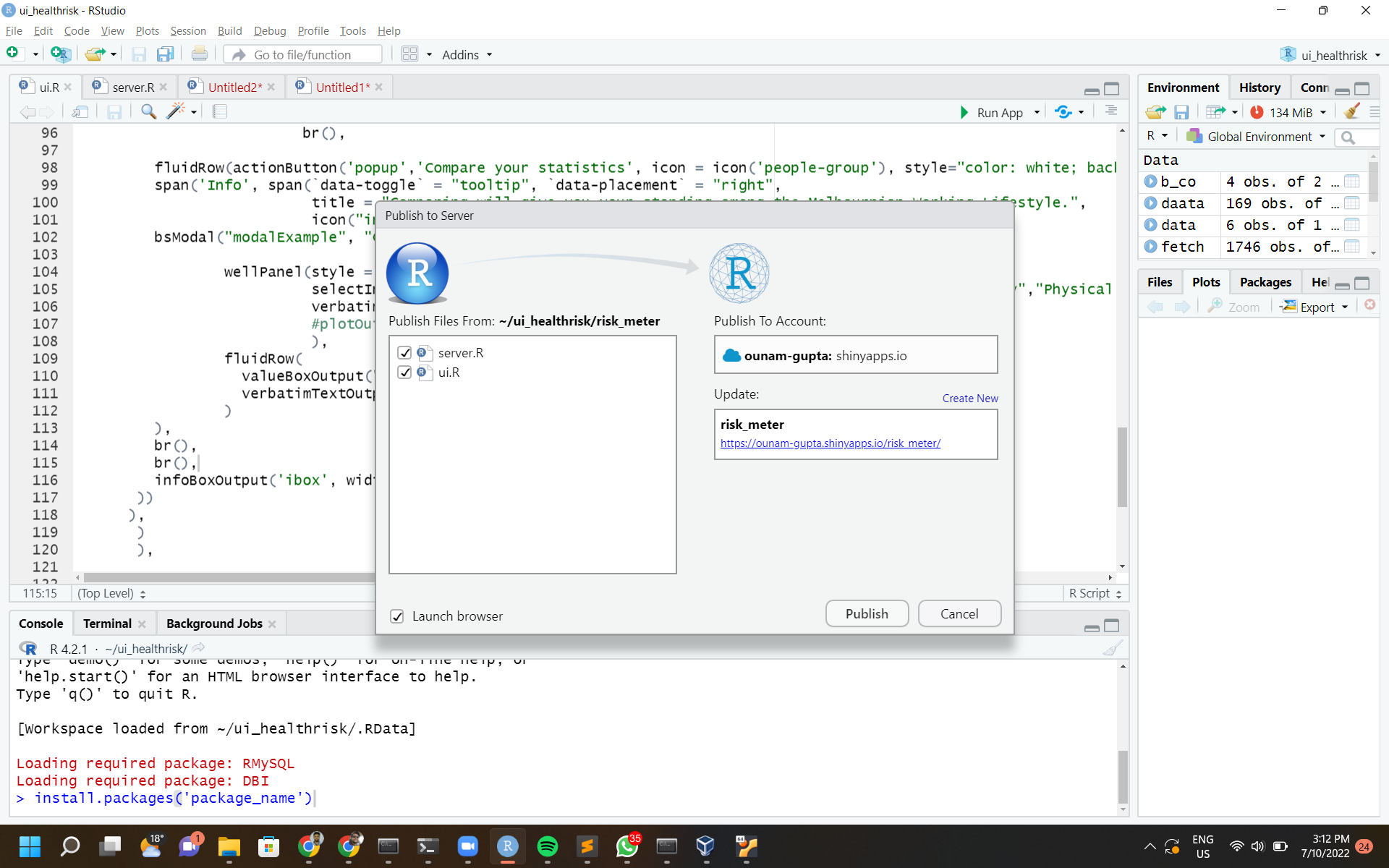
*Note: Enable Auto Deployment to ensure everytime you push changes to Git Repository, it will automatically deploy the changes.*

* **RShiny Server**

The RShiny Server is hosted on the ShinyApp server which allows the RShiny website to be hosted with required R packages seamlessly.

Steps:

1. Create and Connect your Shinyapps server account to RStudio - Refer [Shinyapps documentation](https://docs.rstudio.com/shinyapps.io/getting-started.html#working-with-shiny-for-r)
2. Make the changes and push the Shiny application to the server by clicking on the Publish button - Refer [Publishing Shinyapp documentation](https://docs.rstudio.com/shinyapps.io/getting-started.html#working-with-shiny-for-r)
3. Push both Server.R and UI.R files

****

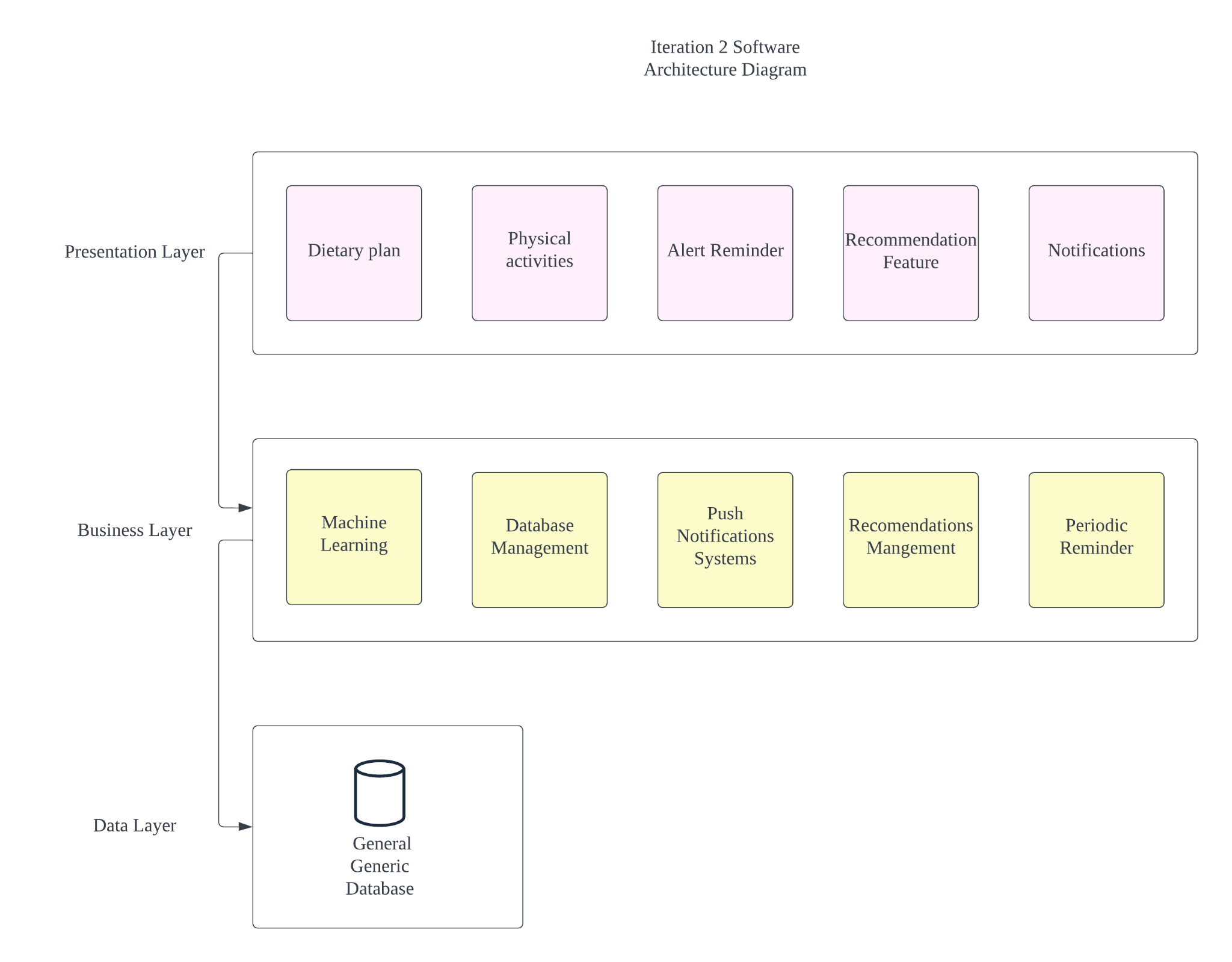
* **Git Repository**

Create a Git account that can be accessed by all the team members.

Using Git for Version Control - Refer to [Git documentation](https://docs.github.com/en/get-started/quickstart/set-up-git) to get started.

## 4. SITTOFIT Website

The Sittofit website product is built with a design to take the user on a journey to learn about sedentary lifestyle and then providing measures to mitigate sedentary lifestyle using information pages and interactive tools. The Software diagram illustrates the functioning of the website in the image below.



SOFTWARE ARCHITECTURE DIAGRAM

### 4.1 Website Features: Working with components

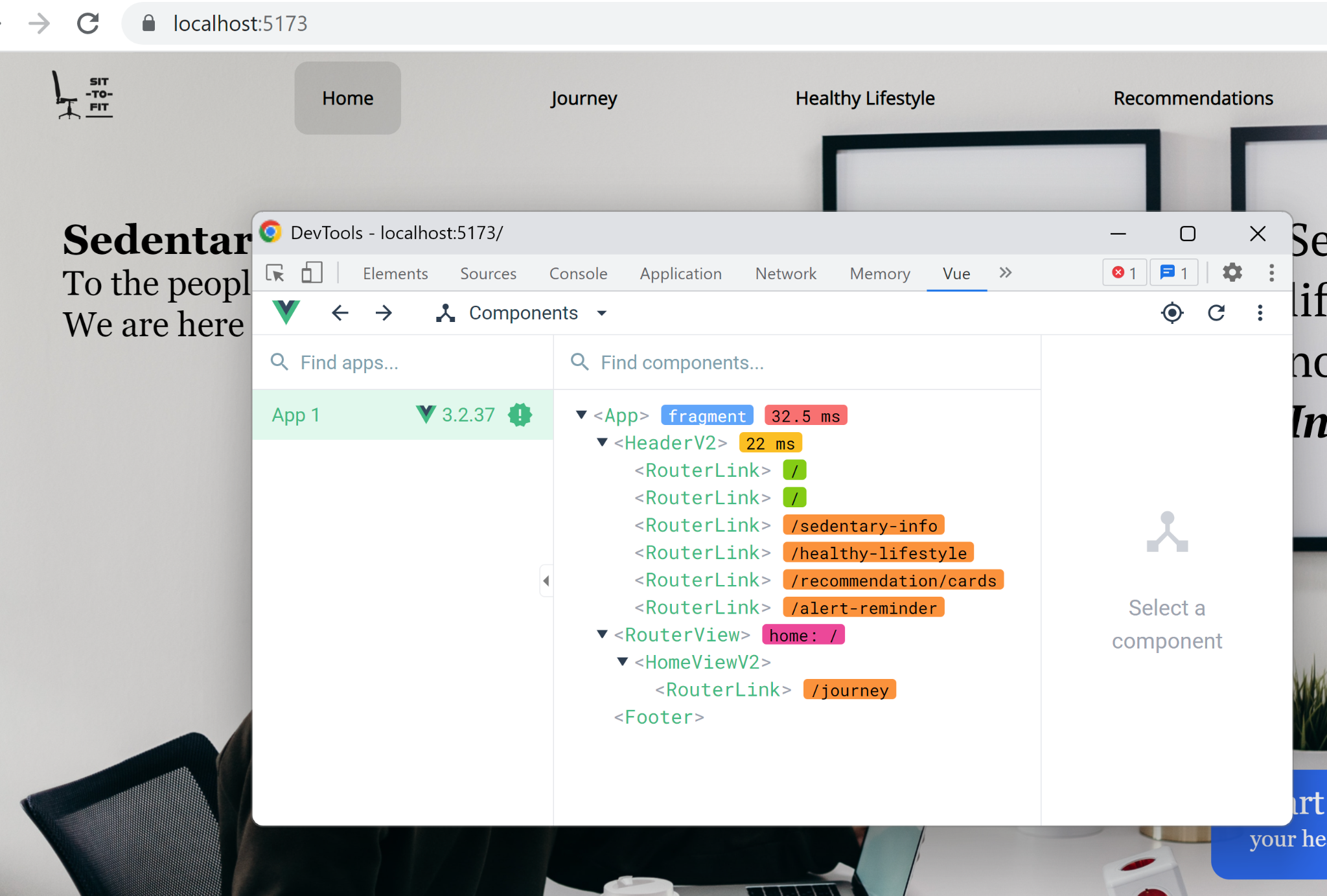
This section covers the information on how to make changes to each component of the website. After working or making changes on any component, check the [Backup:Git section](#bookmark=kix.se0qda9fnzpk) to safely backup your changes on your Git Repository for version control.

#### 4.1.1 Front-End

* **How to find the component that needs to be modified**

Without being familiar with the project file structure, the component used in the page can be easily inspected with Vue.js devtools plugin in Chrome. The plugin can also show the component level tree to select.

It is recommended to use developer tools in browser and Vue.js devtools plugin to check any frontend errors, including routing errors.

****

* **Learn about view and view models**

Vue v3.2.37 is used for creating listeners and managing properties. In this project, composition API and SFC is used with the Vite build tool.

[Introduction | Vue.js (vuejs.org)](https://vuejs.org/guide/introduction.html)

[Getting Started | Vite (vitejs.dev)](https://vitejs.dev/guide/)

* **Learn about layout and styling**

Tailwind v3.1.8 and postcss v8.4.16 are used in this project for styling. Tailwind is a utility first framework which helps simplify css code, postcss allows developers to use the latest css syntaxes with prefixing and transpiling. Flex and Grid layout is very common in this project, it is recommended to understand these before changing stylings.

[Utility-First Fundamentals - Tailwind CSS](https://tailwindcss.com/docs/utility-first)

[postcss/postcss: Transforming styles with JS plugins (github.com)](https://github.com/postcss/postcss#usage)

[CSS Flexible Box Layout - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout)

[CSS Grid Layout - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout)

There is a tailwind.config.js file in the root folder of the frontend project that customizes the global configuration of tailwind. It can be seen that there are some abbreviations set up for breakpoints: 600px, 840px, 1240px, 1440px. The default font family of sans is set as Open Sans which is a free font family under open source license.

[Configuration - Tailwind CSS](https://tailwindcss.com/docs/configuration)

[Open Sans - Google Fonts](https://fonts.google.com/specimen/Open+Sans)



* **Learn about state management**

Pinia v2.0.17 is a store library for Vue, it is easier to manage shared states of components with Pinia than passing props directly. Store created with Pinia is a model-like singleton that can be accessed from any vue component with an import statement.

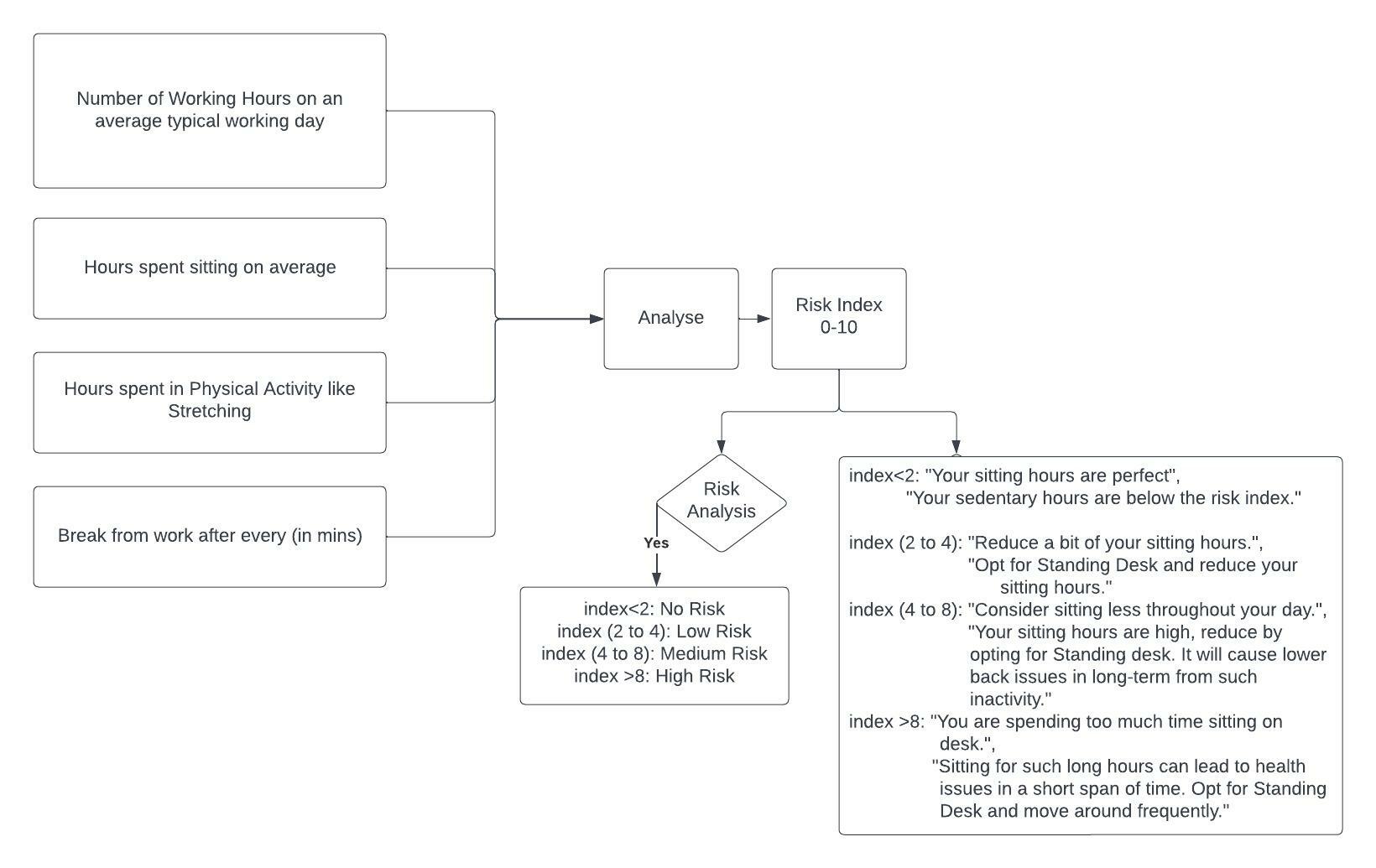
In the frontend project, recommendation feature and alert reminder feature use stores to manage states.

[Introduction | Pinia (vuejs.org)](https://pinia.vuejs.org/introduction.html)

#### 4.1.2 Risk Meter

The Risk meter is built on R and RShiny.

* **System Architecture Diagram**



* **Maintaining Recommendation System:**

1. Working with code

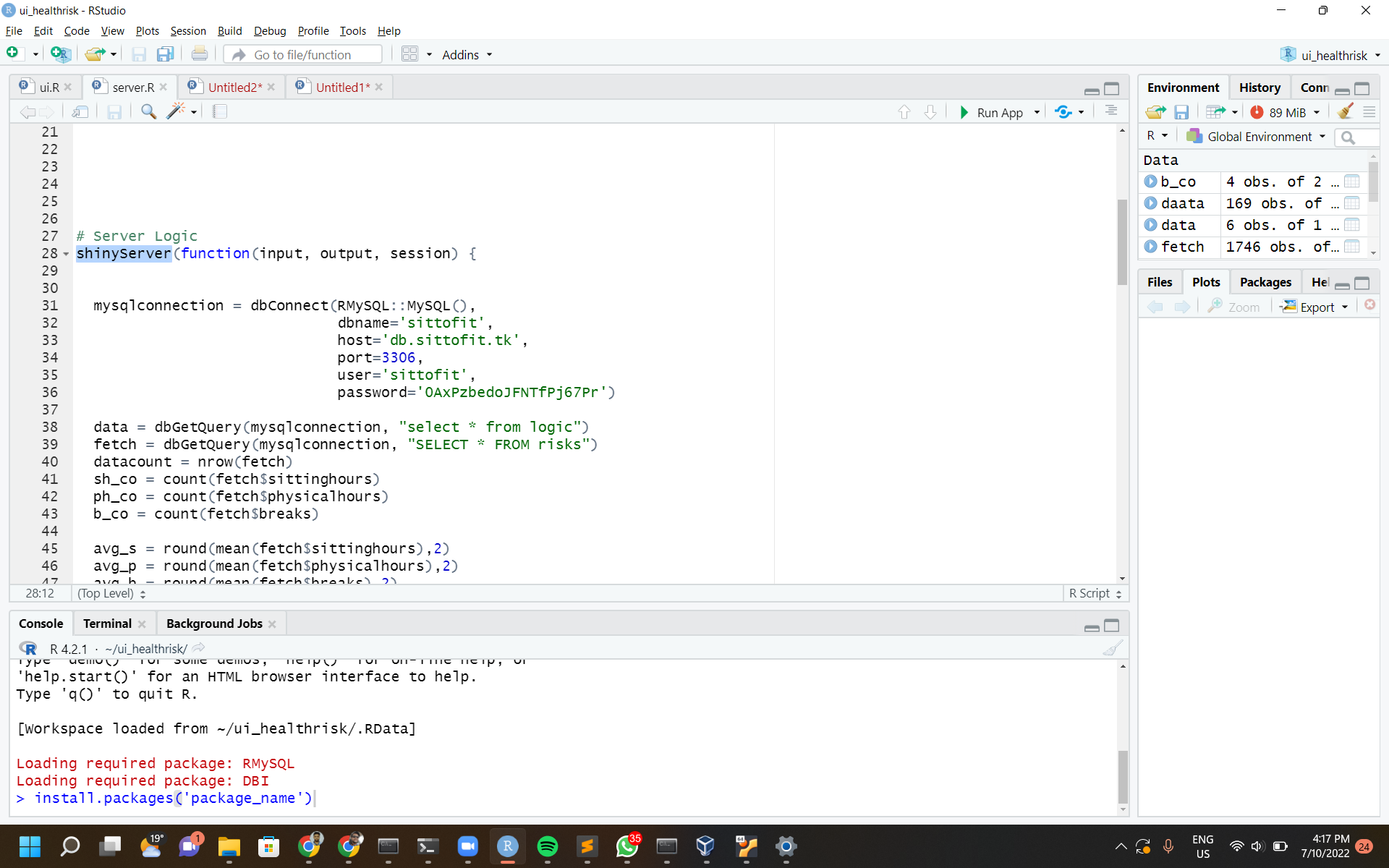
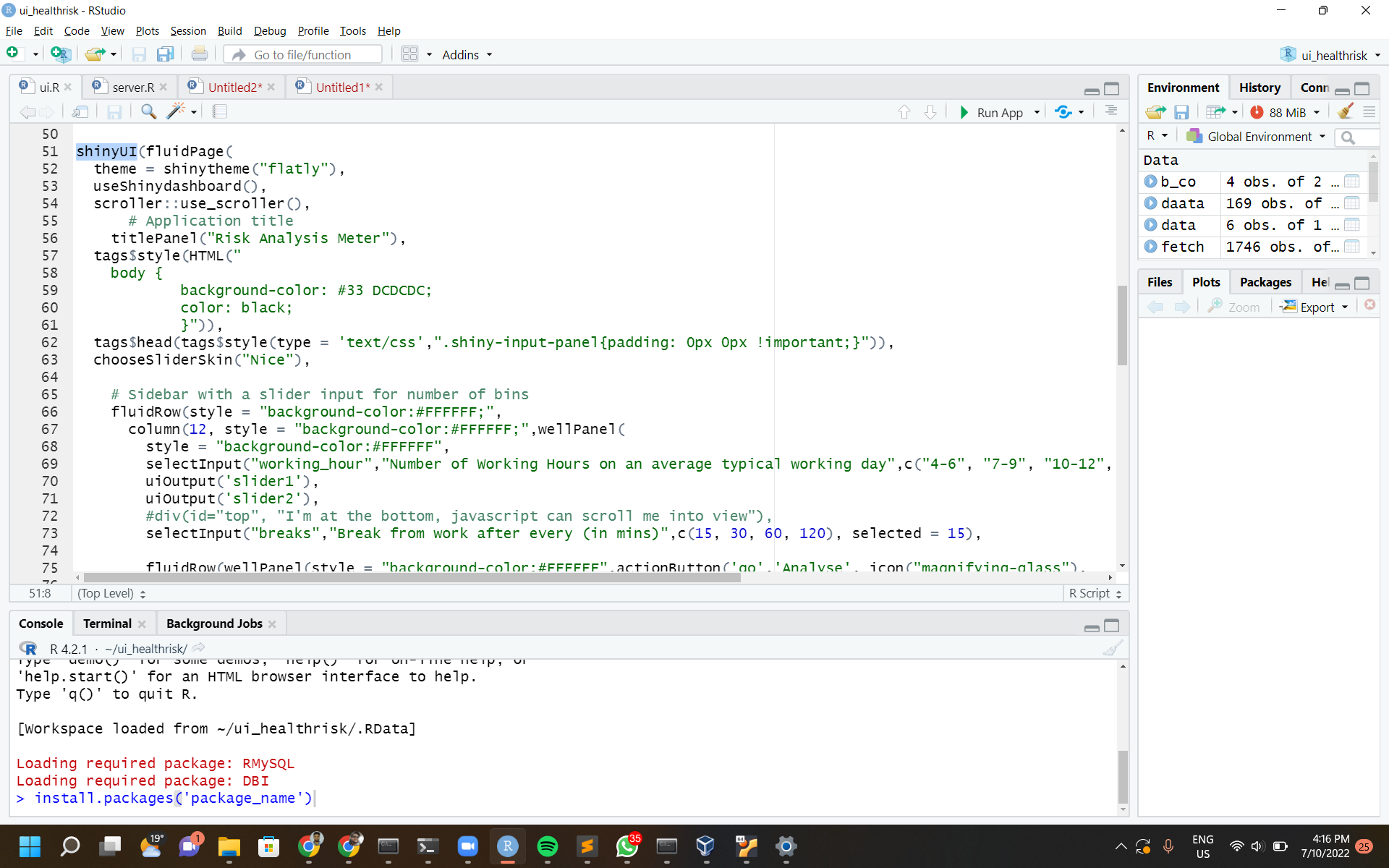
The source code for the Risk Analysis Meter is /1.3 Code and Database Folder/1.3.1 Source Code Files/1.3.1.1 Risk Meter Gauge/riskmeter.R which needs to be divided in UI.R and Server.R.

* To alter by changing a function:

Coding standard:

* Line commenting (comment above each line of code)
* Block commenting (comment for a section of code)
* Line coding

1. RShiny application requires two set of files: UI and Server



UI Server

1. UI.R contains the front-end part of the application

Server.R contains the back-end processing of the application.

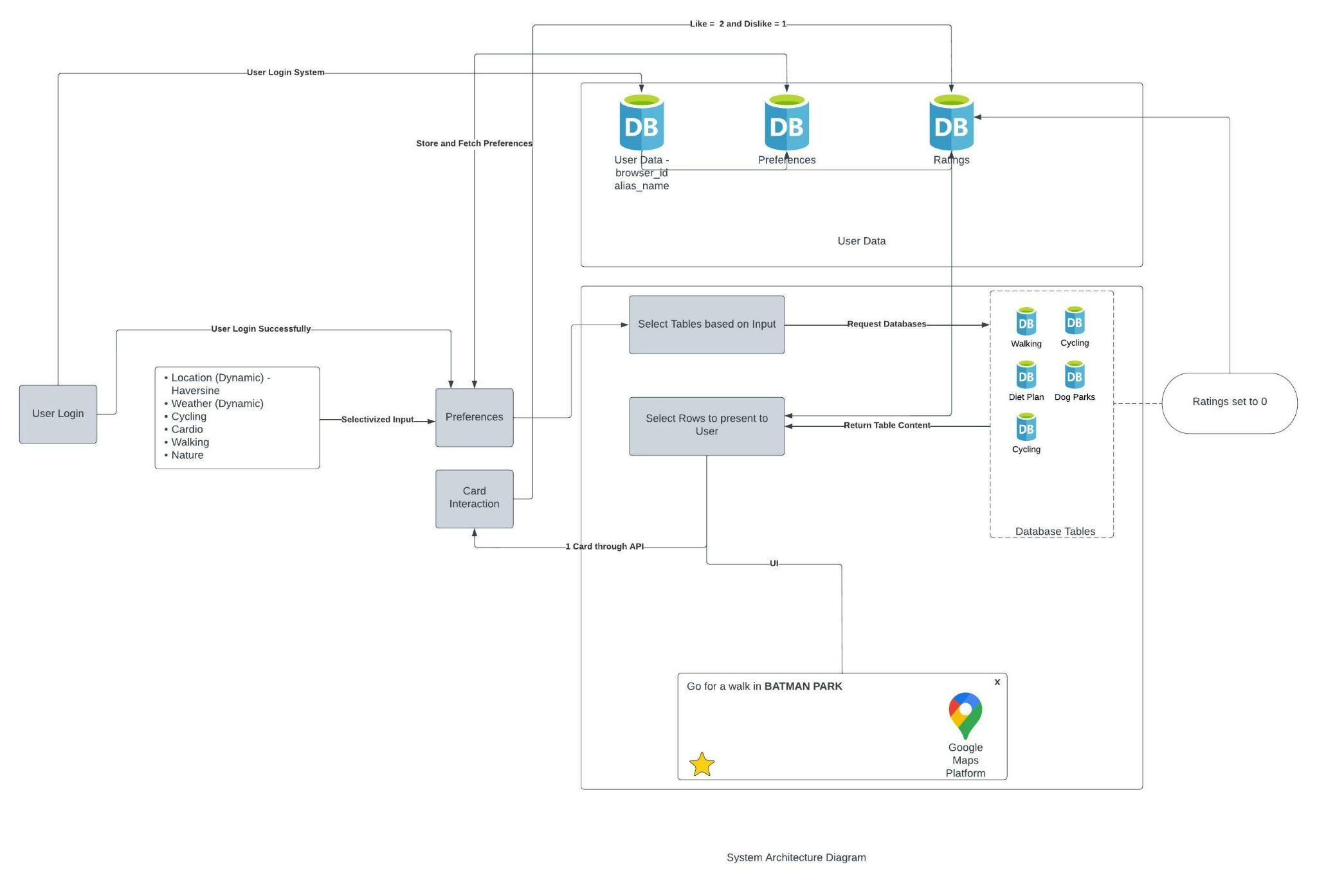
1. Components were discarded from the application but can be re-

#### 4.1.3 Recommendation System

The Recommendation System is built over Python for development and Flask for API.

System Architecture Diagram

The system architecture of Recommendation System explicates the process flow of a user interaction with the tool.

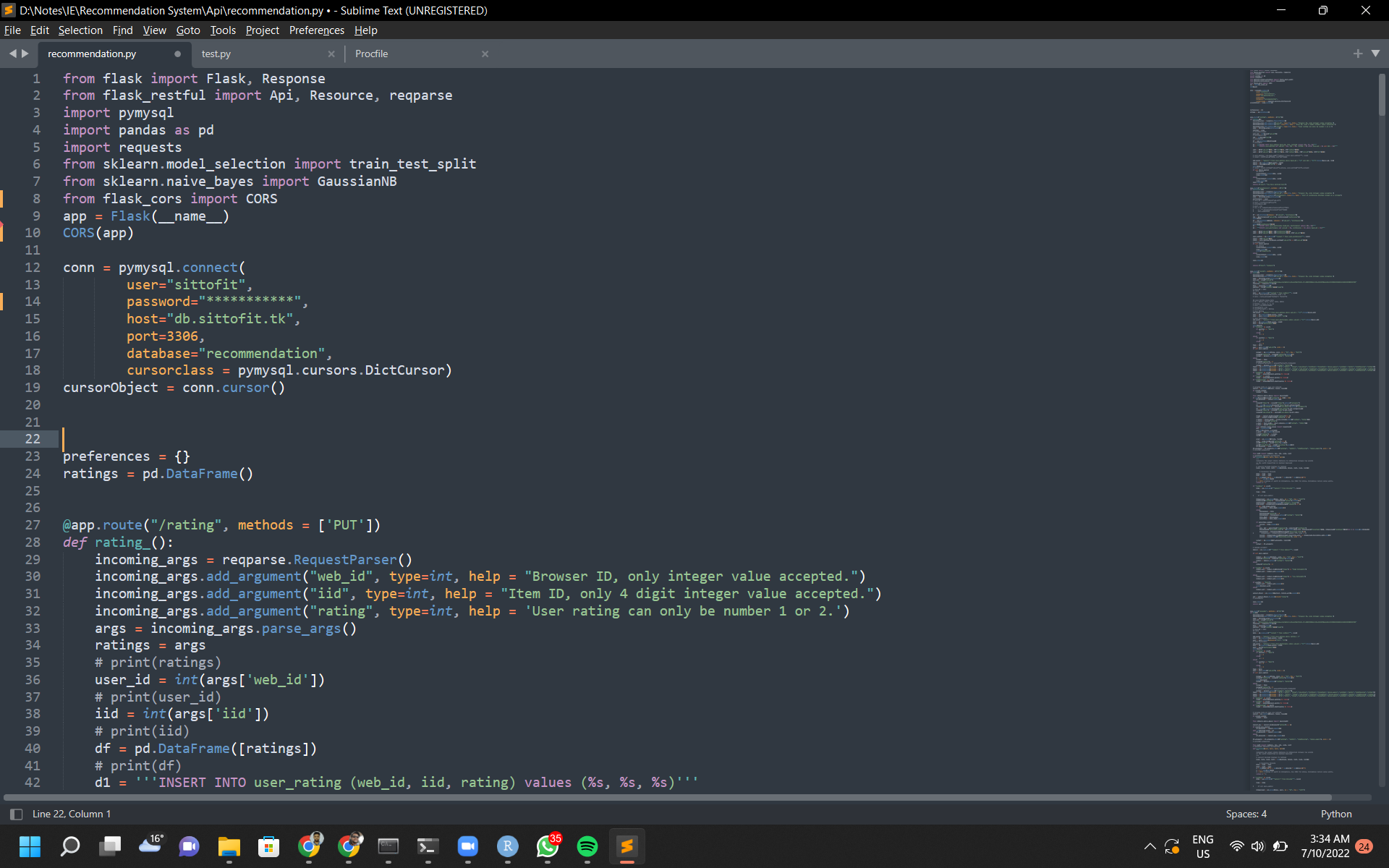


**Maintaining Recommendation System:**

What is required to Maintain the recommendation system and how it can be achieved.

1. The recommendation system uses a list of packages pre-built in Python by importing them into a jupyter notebook instance or recommendation.py (git files)
2. To Establish database connection and create a database element.

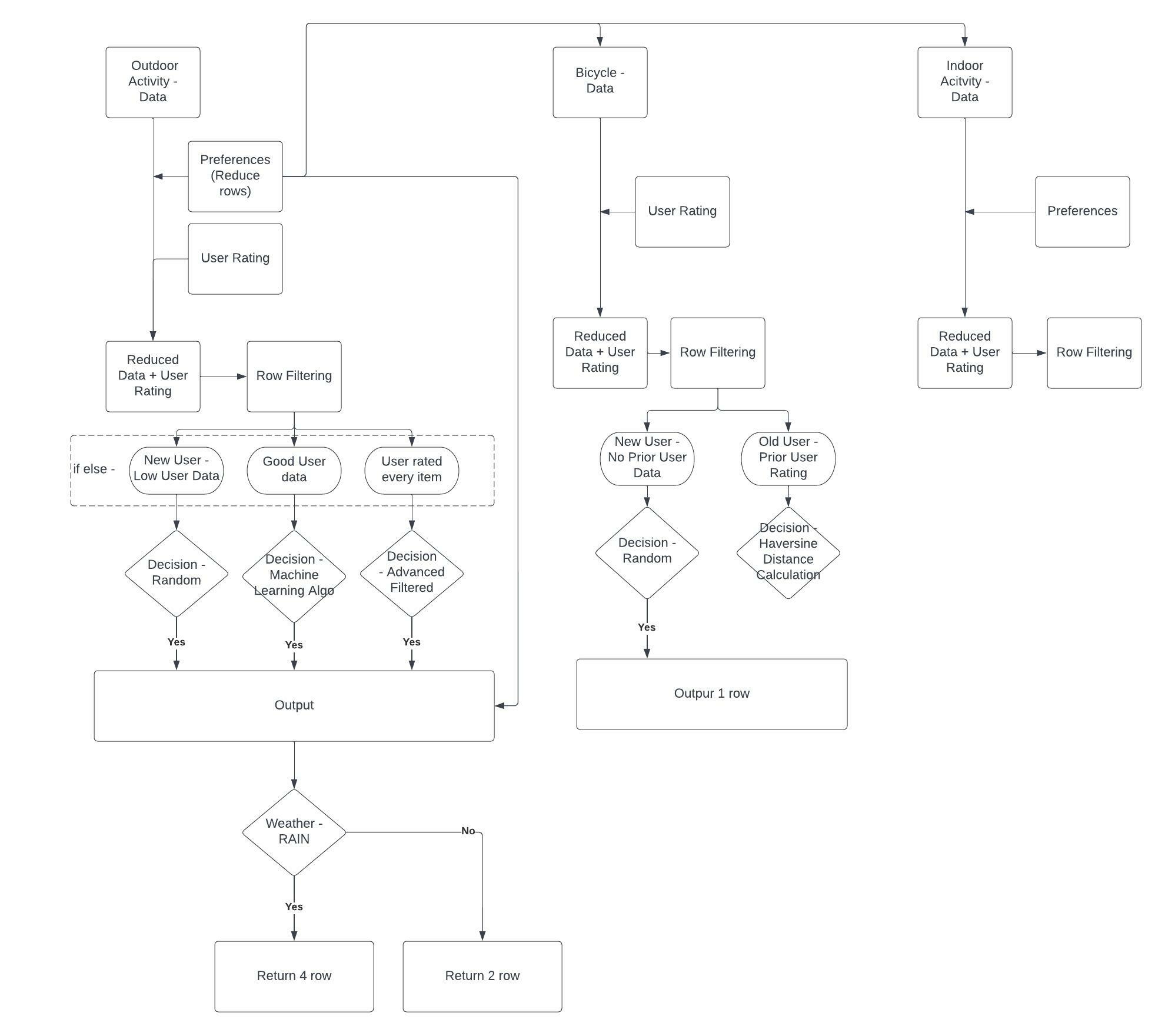
* The ‘conn’ will take the pymysql package and create a connection object.
* cursorObject will be used to call sql like query in python.



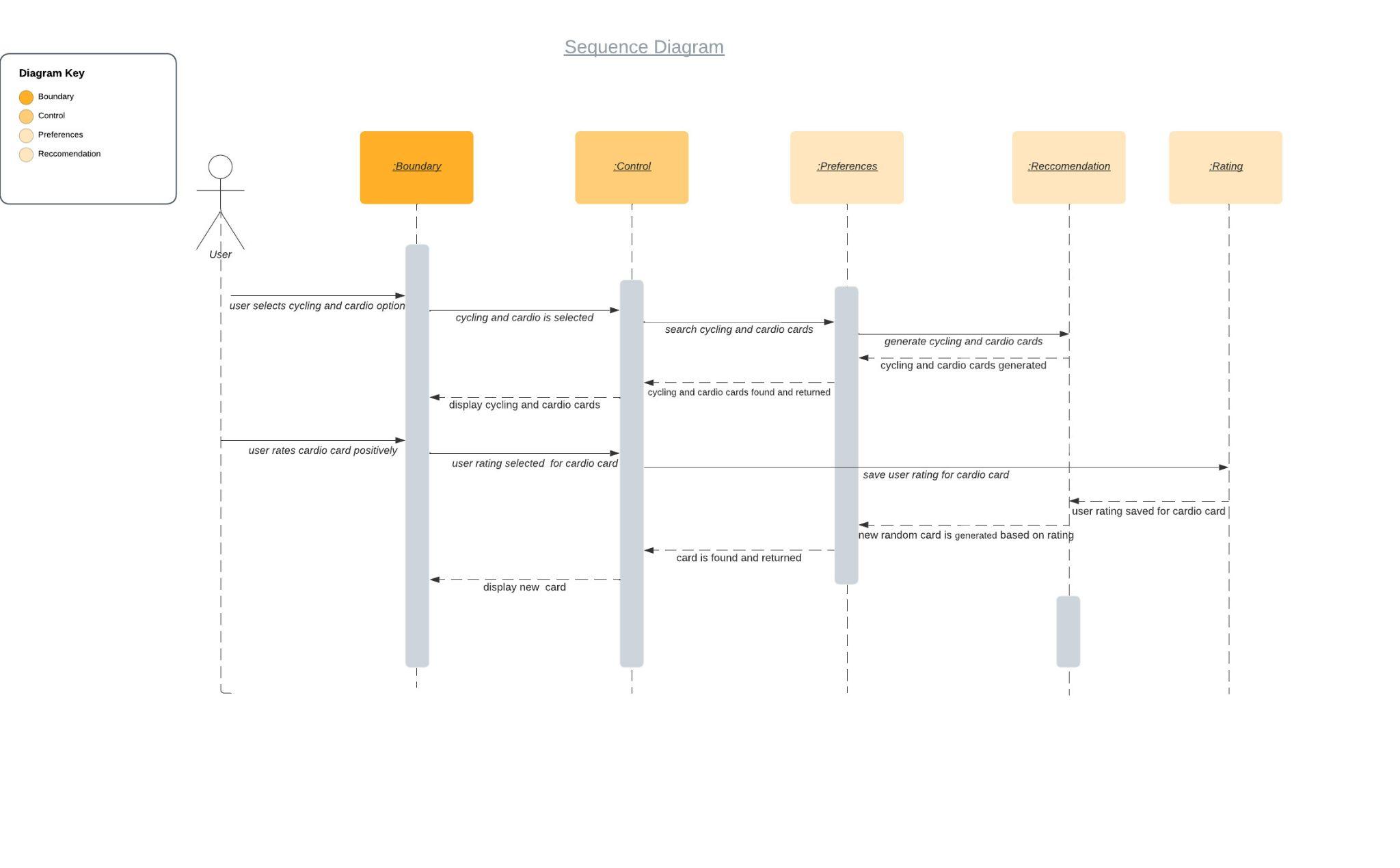
* Use the new database created in the above section and enter the details as per your database.

1. Logic for Recommendation System

The flow diagram below shows the logic behind the output for populating cards on the Recommendation System tool on the front-end.

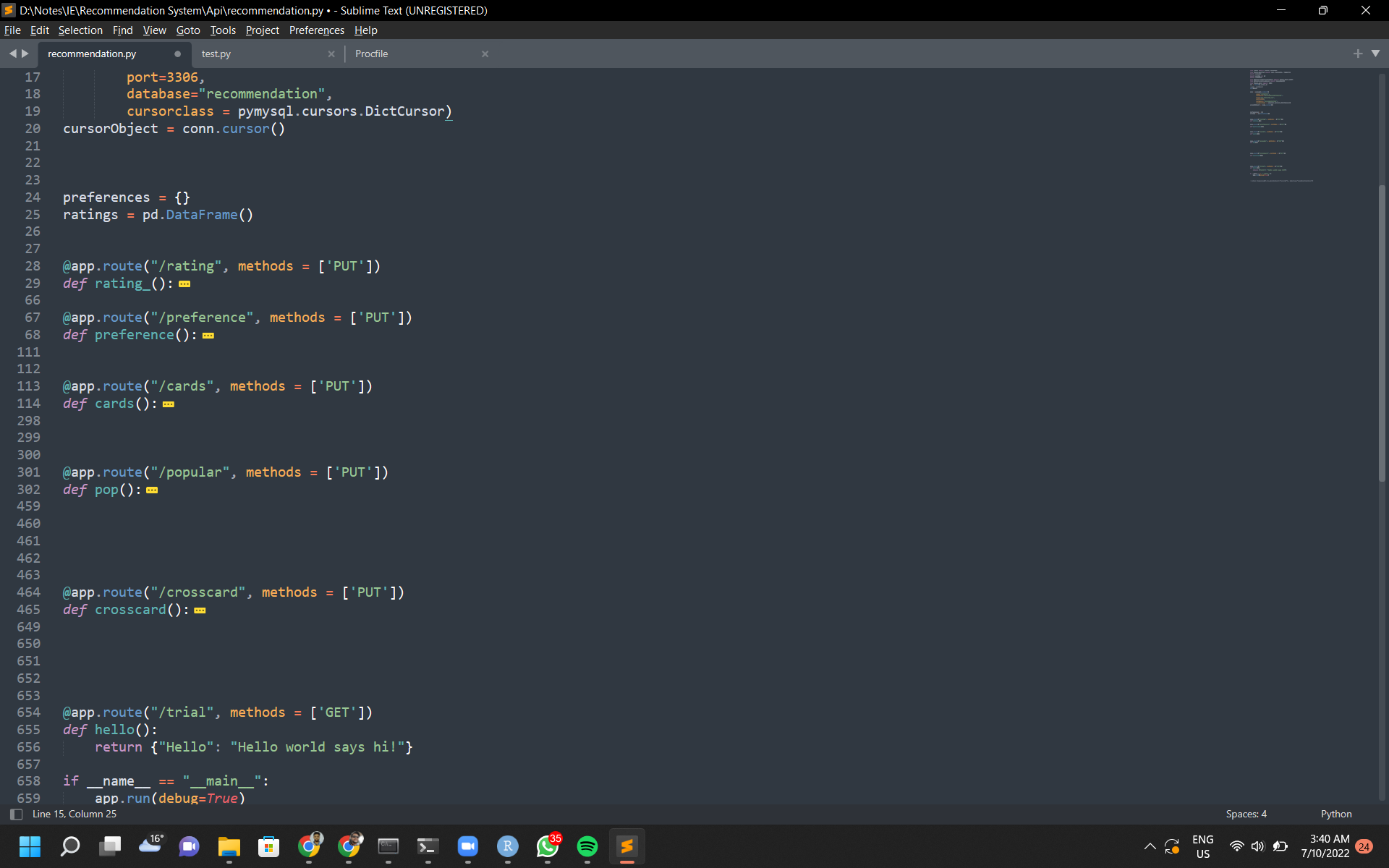


1. Working Model

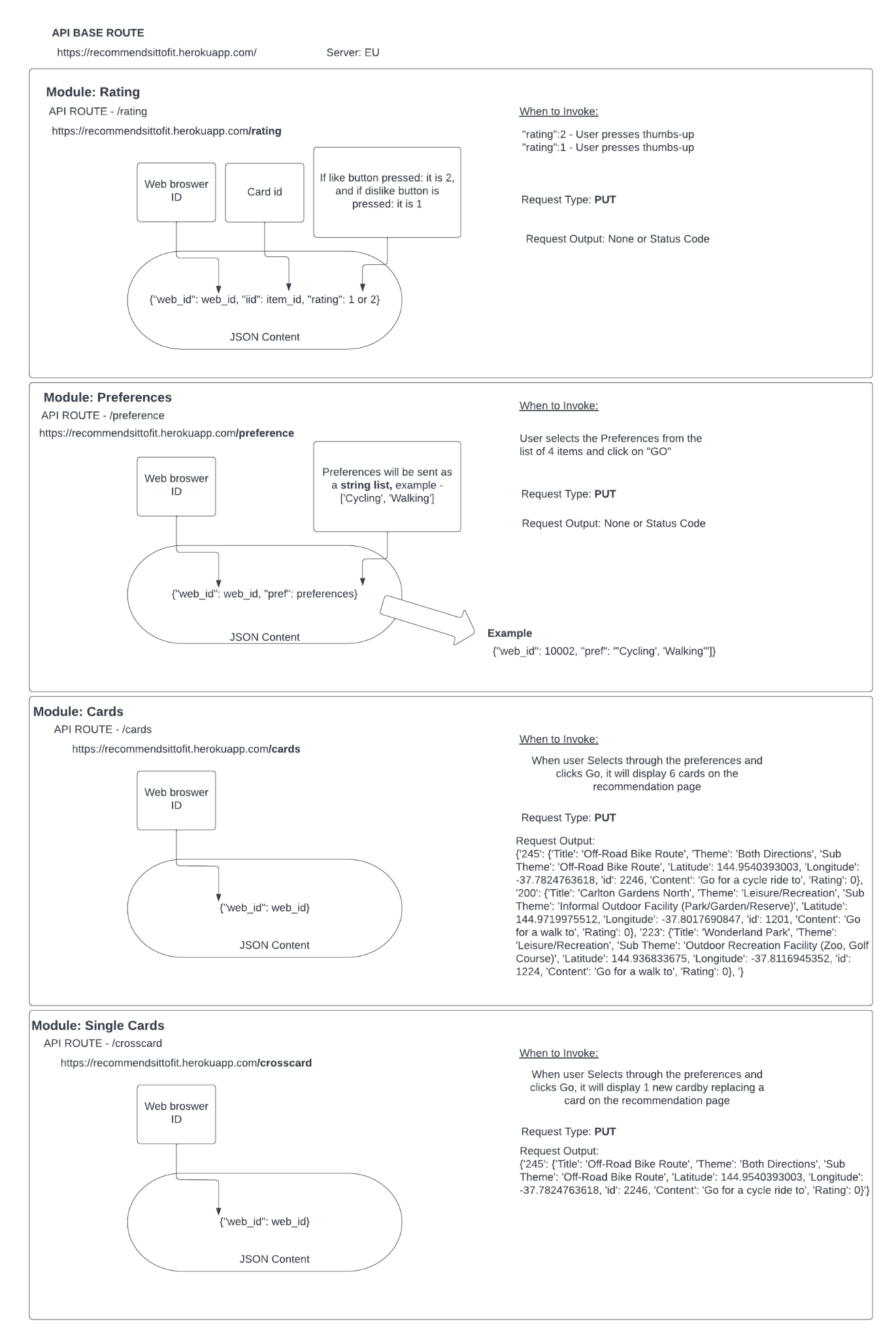


1. Route addresses and method

Currently, the Recommendation System API routes to 5 different functionality in the application. All of the routes use the ‘PUT’ method that describes that the route requires an input in JSON format while calling this route. The different route names and function names serve different functionalities.



1. Input & Output format for different API Routes



1. Working with code

The source code for the Recommendation System is /1.3 Code and Database Folder/1.3.1 Source Code Files/1.3.1.2 Recommendation System/recommendation.py.

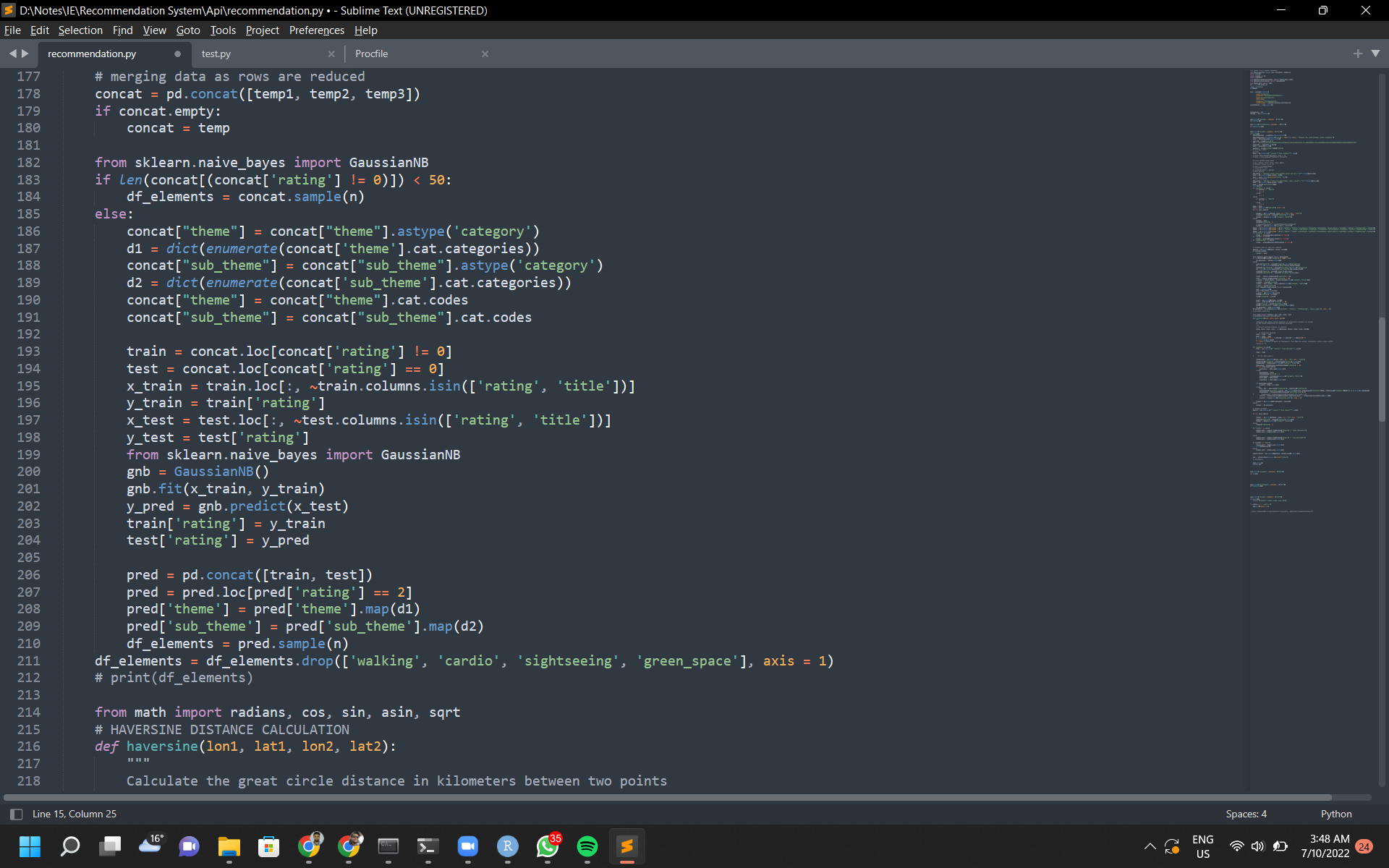
* To alter by changing a function:

Coding standard:

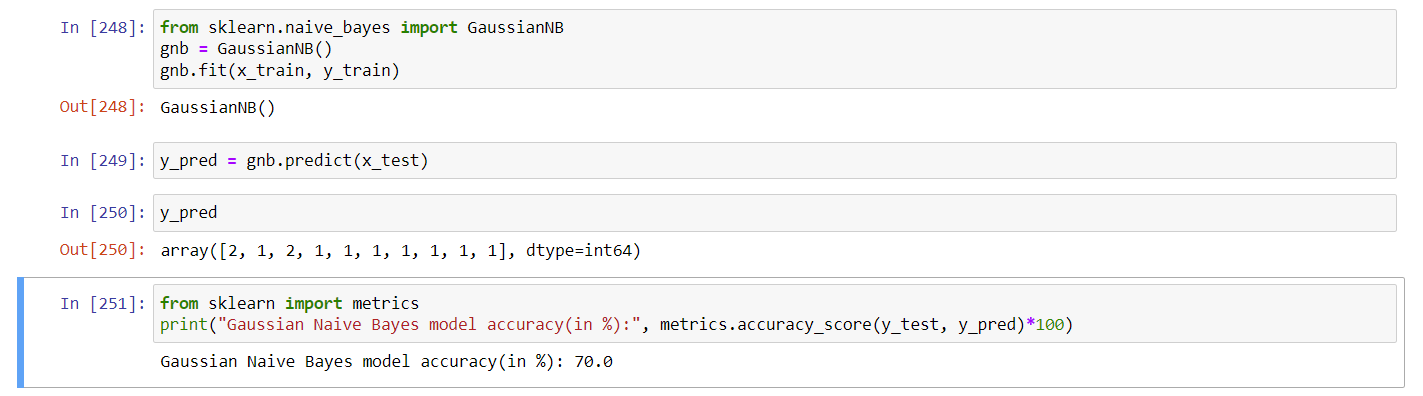
* Line commenting (comment above each line of code)
* Block commenting (comment for a section of code)
* Line coding
* Code compressing is not followed to make it simple to extend small changes.

1. Changing Prediction Model

Currently, the prediction model to determine the recommendation for a user is *Gaussian* *Naive Bayes* Machine Learning model. To change, see this section of the code to modify the model for hyper tuning.



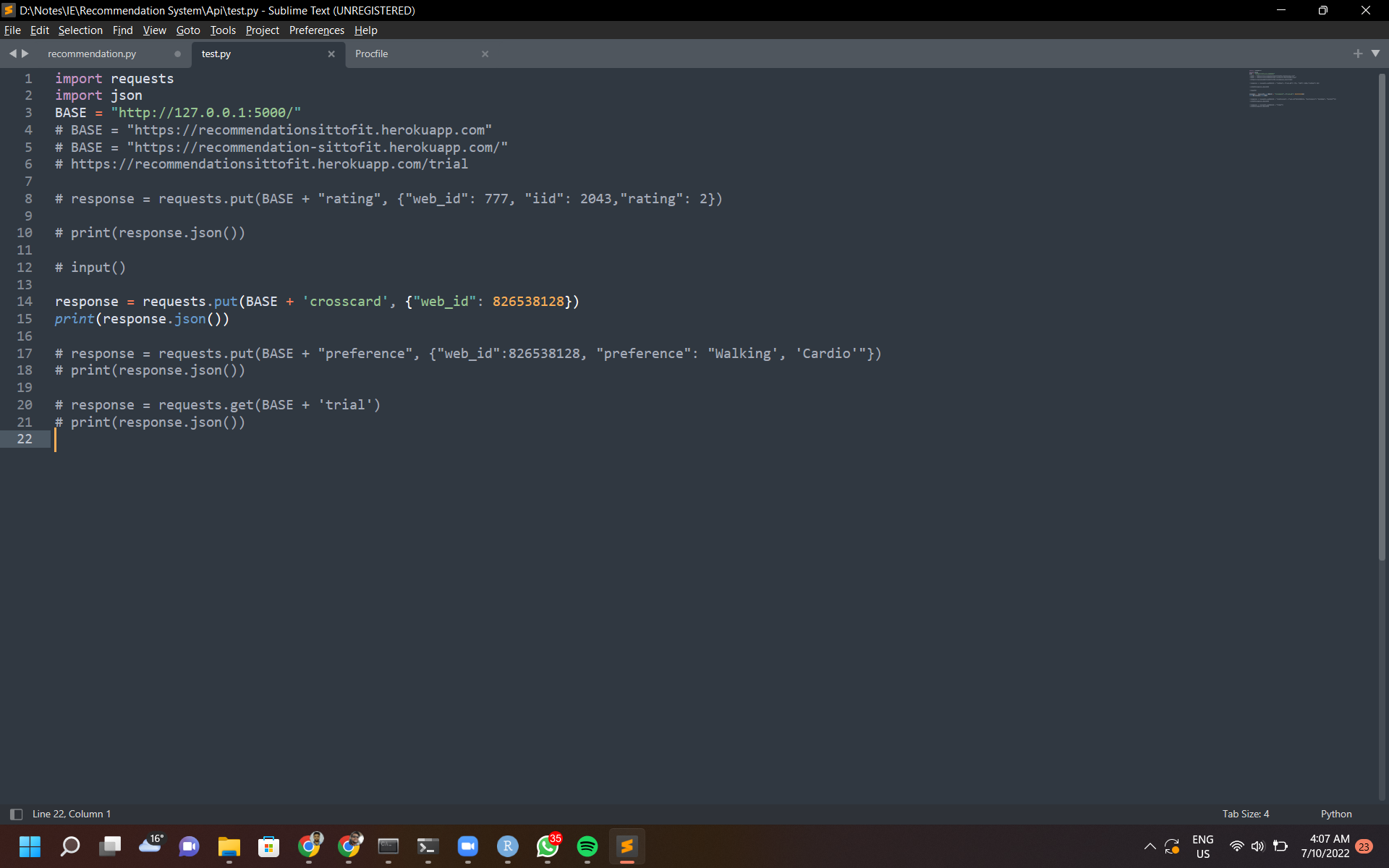
The current model gives the score metrics:



1. Testing Changes

Create a new file for testing the recommendation system on the local machine using localhost.

The test file will have the following test for recommendation card:



*NOTE: Uncomment the batches to test different modules.*

Run the following files in a sequence:

* Run recommendation.py on cmd - ‘python recommendation.py’
* Run test.py on cmd - ‘python test.py

### 4.2 Backup: Git

Backing your progress on git will create a version control of your development environment, which is helpful in case of failure in the deployment. This will help in having a safe restore point for your changes, i.e., to fall back on your last working deployment.

- To backup your work, use Git Desktop or Command line to push your changes to the directory.

Refer to [Git documentation](https://docs.github.com/en/get-started/using-git/pushing-commits-to-a-remote-repository%20.).

After saving your changes follow the steps below for command line version:

1. Open Git bash in the working directory
2. Type - ‘git add .’
3. Type - ‘git commit -m “commit message” ‘
4. Type - ‘git push -u origin master’

This will send your latest changes to the git repository.

*Note - Different Editor supports Git repository connection that allows you to directly push the changes. You can connect your repository and push changes directly from the application.*

*To check your logs, access Git on the website and navigate to your repository.*

## 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.

| Component | Description | Causes |
| --- | --- | --- |
| 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. |
| 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. |
| 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. |
| 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 |
| 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 |
| 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 |
| 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 |
| 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. |

### 5.1 Bugs & Fixes

The reported bugs during the development and testing phase are listed in table :Bugs and Fixes with the status of the bug. If the bug has been fixed, it will detail the fix for the bug.

Reported Bugs

| Bug ID | Bug | Description | Status |
| --- | --- | --- | --- |
| RM001 | Risk Meter Gauge  Database connection drops | Connection to the AWS Mysql database can not be accomplished resulting in failure to load the required data | Resolved |
| RM002 | Risk Meter Gauge  Database number of connections to db server limit exhausts | Number of connections from RShiny server to a database is limited to 16 | Resolved |
| RM003 | Risk Meter Gauge  Logic outputs incorrect risk index | Logic miscalculates risk index | Resolved |
| RM004 | Risk Meter Gauge  Delay in loading user input on Compare statistics | When user clicks on Compare statistics, it fails to load the value of user input on first instance | Pending  (Omitted from application) |
| RM005 | Risk Meter Gauge  Alert Pop crashes the app and fails to load on sending user data | Alert pop crashes the application due to package error with other Shiny package | Resolved (Omitted from application) |
| RM006 | Risk Meter Gauge  UI Layout does not fit screen size | UI layout was set to fit for half the width of screen | Resolved |
| RM007 | Risk Meter Gauge  Unsupported package on server | Package like scroller were incompatible with server side Shiny version | Resolved (Omitted from application) |
| RM008 | Risk Meter Gauge  Auto-Scroll to new elements on the UI | When new elements or output comes on UI, it should automatically come in view but it is not working as it is an embedded application into the website and does not support scrolling the website | Pending |
| RS001 | Recommendation System  User preference mismatch | User preference was mismatched to another user id | Resolved |
| RS002 | Recommendation System  Server unresponsive | The server is not reachable through API call | Resolved |
| RS003 | Recommendation System  Database connection drops | Connection to the AWS Mysql database can not be accomplished resulting in failure to load the required data | Resolved |
| RS004 | Recommendation System  Database connection limit exhausts | Number of active connection to the database in an instance is limited | Resolved |
| RS005 | Recommendation System  Database multithreading not supported | PyMysql supports single threading | Resolved |
| RS006 | Recommendation System  User Preference incorrect data type | User preference table on the database has different data type | Resolved |
| RS007 | Recommendation System  User Rating is empty | If User rating is not there, it will cause the app to crash as data can not be populated with empty table | Resolved |
| RS008 | Recommendation System  Incorrect user id format | The format of user id was set to alphanumeric whereas database was set up for Integer (numeric) | Resolved |
| RS009 | Recommendation System  Incorrect API route | The route to different function of API was incorrect | Resolved |
| RS010 | Recommendation System  Duplicate API function name | The functions in API have duplicate names which results in application failure | Resolved |
| RS011 | Recommendation System  Incorrect Number of cards to output | Number of cards to be output on the front-end was incorrect | Resolved |
| RS012 | Recommendation System  API call blocked by CORS policy | CORS policy blocks connection from a website | Resolved |
| RS013 | Recommendation System  User rating mismapped | Rating for different items mismapped | Resolved |
| AM001 | Alert Reminder |  |  |

### 5.2 Fixing Bugs

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. Some of the common issues that can occur are listed in the table

| Bug ID | Bug | Fix/Solutions |
| --- | --- | --- |
| RM001 | Risk Meter Gauge  Database connection drops | 1. Check for password  2. Check for details to the AWS Server |
| RM002 | Risk Meter Gauge  Database number of connections to db server limit exhausts | Clear connections made from RStudio in an instance.  Code:  > lapply( dbListConnections( dbDriver( drv = "MySQL")), dbDisconnect)  Use this in the “server script” or “console while debugging” |
| RM003 | Risk Meter Gauge  Logic outputs incorrect risk index | The logic relies on if-else loop in Python, compare syntax in R uses inclusive value  value < 2 will have values less than 2, counted till 1.9. |
| RM004 | Risk Meter Gauge  Delay in loading user input on Compare statistics | Not fixed yet.  Discarded in future iteration |
| RM005 | Risk Meter Gauge  Alert Pop crashes the app and fails to load on sending user data | Not fixed yet. The package is not supported by the Shinyapp server.  Refrain from using pop-up Alert.  Suggestion:  Use browser pop-up |
| RM006 | Risk Meter Gauge  UI Layout does not fit screen size | Set width to 12 for column, infobox, valuebox  syntax: width = 12 |
| RM007 | Risk Meter Gauge  Unsupported package on server | Third party packages (eg. Github package installation) may be outdated which are not supported by Shiny app server.  Example - Scroller package (<https://github.com/lgnbhl/scroller>) |
| RM008 | Risk Meter Gauge  Auto-Scroll to new elements on the UI | Use dplyr t  shiny::a() %>% shiny::tagAppendAttributes(href = "##item\_id") |
| RS001 | Recommendation System  User preference mismatch | User id is fetched from debugging mode where default id is called. Comment user\_id in the development environment. |
| RS002 | Recommendation System  Server unresponsive | Check server logs at:  Heroku Logs for error code  AWS for MySQL database server |
| RS003 | Recommendation System  Database connection drops | Remove conn.close() |
| RS004 | Recommendation System  Database connection limit exhausts | Use single connection  Set connection as global for PyMySql |
| RS005 | Recommendation System  Database multithreading not supported | PyMysql only supports single thread, do not create multiple cursor object |
| RS006 | Recommendation System  User Preference incorrect data type | User preference supports int for user\_id and string format for preference |
| RS007 | Recommendation System  User Rating is empty | Check if user ratings is empty  if not user.empty:  …  else:  merged['rating'] = 0 #set all item ratings as 0 |
| RS008 | Recommendation System  Incorrect user id format | JSON object while calling the API should only have Integer values  {web\_id: 1000022, …} |
| RS009 | Recommendation System  Incorrect API route | Check base API route on Heroku server, and add the module to be called as given in the API routes |
| RS010 | Recommendation System  Duplicate API function name | Function name in the script should be different  @app.route("/cards", methods = ['PUT'])  def cards():  ….  @app.route("/crosscards", methods = ['PUT'])  def cards():  …. |
| RS011 | Recommendation System  Incorrect Number of cards to output | output\_final = pd.concat([output, indoor\_act]).sample(4)  Error occurs: If the sample does not have the required number of rows  The sample should append enough rows |
| RS012 | Recommendation System  API call blocked by CORS policy | from flask\_cors import CORS  CORS(app)  Note: Add this package to requirements.txt |
| RS013 | Recommendation System  User rating mismapped | Use left join.  merged = pd.merge(activity\_data, user\_ratings, on = 'item\_id', how = 'left') |

## 6. 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 Address - /1.3 Code and Database Folder/1.3.1 Source Code Files/1.3.1.1 Risk Meter Gauge/Testing.docx.

**Support: Regular Testing Phases**

| Type of testing | Component | Procedure |
| --- | --- | --- |
| Manual Testing - Unit Testing | For this small sections of the website were tested periodically. | 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. | 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.  Sample size - 1000  Result -   * Throughput - 601.28/min * Deviation - 39 | 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: 10 |

### 7. Extend Features

#### Login/Registration system

For further enhancement of the Sit-to-Fit website system, a login and registration feature will be desirable for the user. This will allow users to save their preferences for the recommendation tool in their account and can go back to it whenever they want. The login system will allow us to implement more success measures for the users such as a point system to complete physical activities in the recommendation tool. A daily/weekly leaderboard can then be added as a motivation tool for users to compete with one another and break their sedentary habits and improve their physical health.

**Personnel involved:**

1. Front-end developer

The front-end developer will be required to make the UI/UX changes for the login page and add it to the Sit-to-Fit website. They will be required to create a user database that will store information about the user details such as username, password, and other user details required for the registration process. The database must also save the user preferences for the recommendation system.

1. Data scientist

The data scientist must work with the front-end developer to integrate the recommendation tool preferences with the login system according to each user thus backend development will be required for this.

#### Recommendation tool

The recommendation tool is currently limited for users within or near Melbourne CBD as the physical outdoor activities are currently for Melbourne CBD only. Further enhancement can be made from the recommendation tool by adding physical activities for the whole of Melbourne or Victoria. This will expand the target audience and allow more users to participate in using the recommendation tool and benefiting from it.

**Personnel involved:**

1. Data Scientist

The current API and datasets used for outdoor physical activites recommendation tools are currently only for Melbourne CBD locations. Further expansion of the area to the whole of Melbourne or Victoria will require new dataset to be discovered, cleaned and integrated with the recommendation tool API.

1. Front-End Developer

The front-end developer will need to collaborate with the data scientist to make sure the new API and datasets are integrated with the website and the database of Sit-to-Fit.

#### Alert Reminder

The alert reminder can be further expanded by adding reminder alerts linked with the recommendation tool for physical activities that the user likes. This will require further integration between the alert reminder and recommendation tool to integrate the physical activities into the alert reminder feature. There also can be a snooze button that can be added in the alert reminder so that the alert reminder does not give notifications after a set period of time chosen by the user.

**Personnel Involved:**

1. Front-End Developer

Front-end developers will be responsible for adding the snooze button for the alert reminder. They will need to work with the data scientist to integrate the recommendation tool physical activities with the alert reminder feature.

1. Data Scientist

The data scientist will have to collaborate with the front-end developer to integrate the alert reminder with the recommendation tool for outdoor activities

## 8. Training

| Position | Roles | Skills |
| --- | --- | --- |
| Data Scientist | Create and maintain features with open datasets. Integrate with the front-end of the website and database. Update datasets,data cleaning and data wrangling. | R, Rshiny, Custom API, Agile development, Data cleaning, Data wrangling, |
| UI/UX designer | Design the UI/UX of the website and the features of the system. Conduct usability testing and obtain feedback. | Figma, UI/UX Design principles, Agile development |
| Front-end developer | Implement UI/UX design on the website. Integrating features with the back-end. Establishing and maintaining the databases. | React, CSS, HTML,MySql, Javascript, AWS, |
| Cyber Security architect | Help build and design a system that protects critical data of the system and protects user data. | Establishing security protocol, Pen testing, security incident handling, encryption technologies. |