



Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved; NAAC Accredited A++

Karunya Nagar, Coimbatore - 641 114, Tamil Nadu, India.

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

A SKILL BASED EVALUATION REPORT

SUBMITTED BY

TANVIK SRI RAM REDDY (URK23CS1261)

COURSE CODE

23CS1007

COURSE NAME

PYTHON PROGRAMMING

APRIL 2024

INDUSTRIAL CERTIFICATION



OpenEDG Python Institute Authorized Academy Program



Statement of Achievement

PCAP: Programming Essentials in Python

The graduate of the *PCAP: Programming Essentials in Python* course, provided by **Cisco Networking Academy®** in collaboration with **OpenEDG Python Institute**:

- knows the universal concepts of computer programming, including variables, data structures, algorithms, control flow, functions, and exceptions;
- can proficiently use the developer tools, the runtime environment, and the syntax and semantics of the Python language;
- can use fundamental programming techniques, best practices, customs, and vocabulary, including the most common standard library functions in Python 3;
- can write Python programs using standard language infrastructure, and knows the means by which to resolve typical implementation problems;
- knows how to work with modules and packages, process text and binary files, and use generators, iterators, and closures;
- understands the fundamentals of object-oriented programming (OOP) and the way they are adopted in Python.

TANVIK SRI RAM REDDY

Student

A handwritten signature in blue ink, appearing to read 'M. Wichary', written over a horizontal line.

Maciek Wichary
VP & CEO, OpenEDG

22 Feb 2024

Date

www.netacad.com | www.pythoninstitute.org

WATER INTAKE FOR EFFICIENT HYDRATION

A REAL TIME APPLICATION REPORT

Submitted by

**TANVIK SRI RAM REDDY (URK23CS1261)
NIVESH R (URK23CS1262)
BIJJAM DINESH REDDY (URK23CS1263)**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed-to-be-under Sec-3 of the UGC Act,
1956) Karunya Nagar, Coimbatore - 641 114. INDIA**

APRIL 2024

ABSTRACT

The "Water Intake for Optimal Hydration" project presents a user-friendly application designed to help individuals effectively manage their daily water consumption for better hydration. With a clean and intuitive graphical interface, users can easily register, log in, and track their water intake over time. The application offers a dynamic dashboard upon login, providing users with a quick overview of their hydration status, goals, and progress. By effortlessly recording their water intake, users can monitor their hydration habits and make necessary adjustments to stay adequately hydrated throughout the day. Moreover, the application allows users to customize settings such as fonts and appearance modes, ensuring a personalized experience tailored to individual preferences. Its responsive design ensures smooth navigation across different devices and screen sizes. Visual elements such as images and icons enhance the overall user experience, making the application both functional and visually appealing. Additionally, robust error logging capabilities are integrated to capture and address any encountered issues promptly. In essence, the "Water Intake for Optimal Hydration" application serves as a convenient tool to promote and maintain healthy hydration habits for improved well-being.

PROBLEM STATEMENT

Many individuals struggle to maintain adequate hydration levels, leading to various health issues. Existing solutions lack user-friendly interfaces and comprehensive tracking features. Users need an intuitive application to monitor and improve their water intake habits effectively. Current options also lack personalized settings and error logging functionalities, hindering user experience and troubleshooting efforts. Thus, there's a critical need for a user-centric water intake application that provides seamless tracking, customization options, and error logging for enhanced hydration management.

METHODOLOGY / ARCHITECTURE

Organization: The project is divided into smaller parts called modules, making it easier to manage.

Classes and Objects: Different sections of the program are represented by classes, which are like blueprints for creating specific parts of the application.

Database Use: Data is stored and retrieved from a database, allowing users to save and access their information.

User Interface Design: The program's appearance and layout are carefully planned to be easy to understand and use.

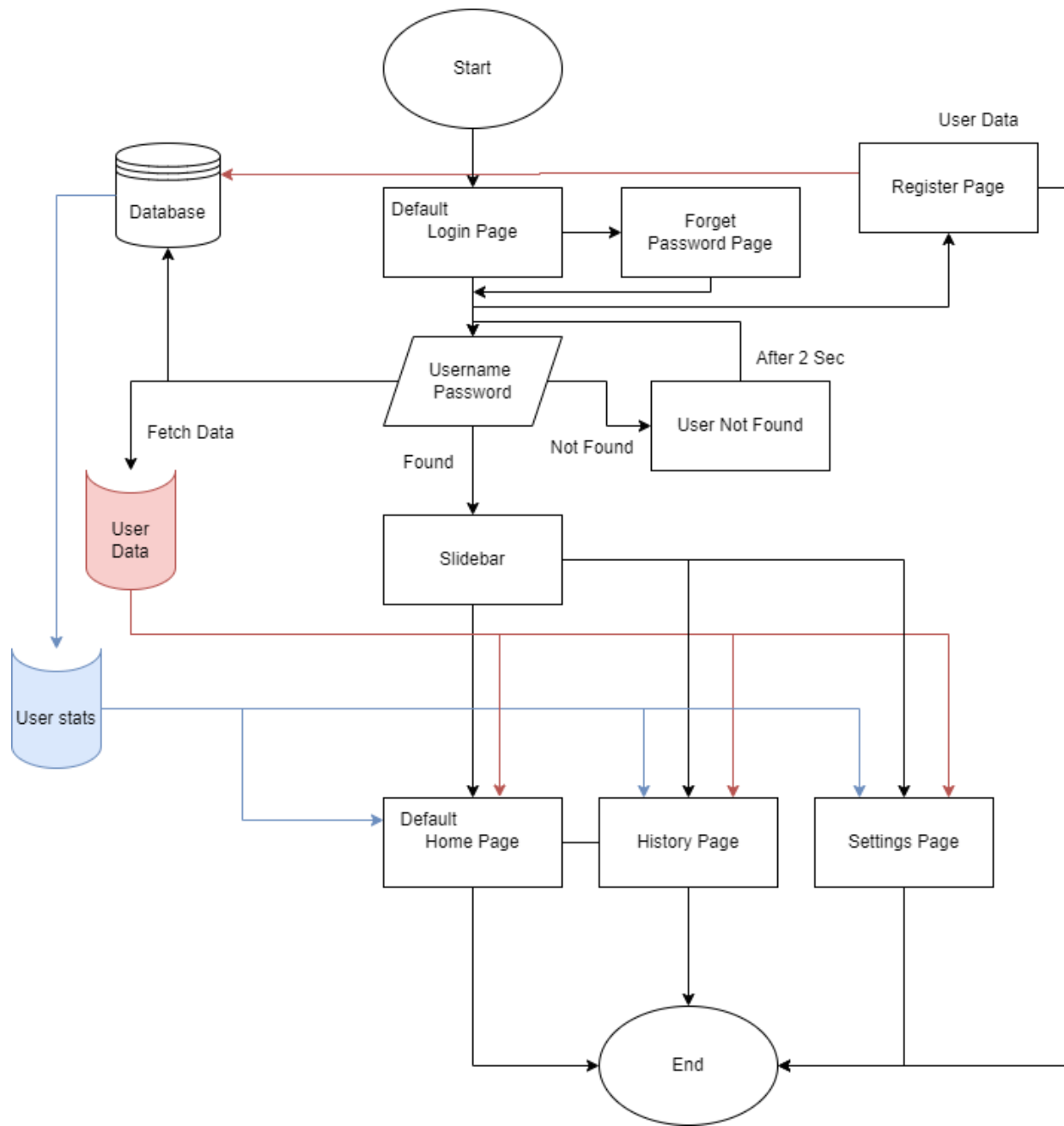
Responding to Actions: The program reacts to what users do, like clicking buttons or entering information.

Error Handling: The code effectively handles errors and logs important events, ensuring smooth operation and facilitating efficient debugging.

Architecture:



Flow Chart:



IMPLEMENTATION

CODE:

start.py:

- The main start file which access the Base class and maintains the CTK instance

```
from Scripts.Base.WaterTracker import WaterIntake
from Scripts.Utilities.window import close_window
if __name__ == '__main__':
    app = WaterIntake()
    app.logger.info("App Started!")
    app.bind("<Control-q>", lambda event: close_window(event=event, app=app))
    # Binding to app instances.
    app.mainloop()
```

The whole code is divided into segments called Base and Utilities.

Utilities contains sub-operational code file:

1. **window.py:**

- Which contains two functions classed close_window and center_screen.
 - close_window: When a user types (control + z) there will be pop up message for prompting user to confirm close.
 - center_screen: This function takes window instance, window width and height as the parameters, it makes the window to stay in center of the screen.

2. **DatabaseMethods.py:** It contains total 8 functions, update_userstats, update_widgets, last_login, reset_daily_values, inc_value, update_data, waterintakehistory_update and database_extractor. All these functions are related to database somehow.

- Contains:
 - Imports: From json, dumps and loads, from datetime, time delta, datetime, date
 - *update_userstats:* This function creates an instance of Database from WaterDatabase.py file. Then calls update values function in order to update values finally it closes the connection, in case of error it will show the error with logging module
 - *update_widgets:* This function consists of self, page and app as parameters, it hands logic to update the homepage and history page widgets with help of database.
 - *last_login:* This updates the last login time of user in database, this function takes self and userdata as parameters.

- *reset_daily_values*: This function aims to check if the last reset time is loaded into database or not. If not loaded it will set the current time value and goes to the current points and current water intake and its it reset default values for daily set. All this data is from userstats table.
- *inc_value*: This function takes self, valx and measure as parameters, valx is amount of water we drank and measure is either ml or L. This connects to the database and updates increased value by calling the update_userstats function.
- *waterintakehistory_update*: This function sets the values for currentwaterintake, dailypoints, currentpoints and dailyintakegoal and loads them in to the database table. This is mainly used to append new data to the old data.
- *update_data*: This function calls waterintakehistory_update and sets the data and then updates the values of userstats table using database.
- *database_extractor*: This is main function of this whole datamethods file, it takes self, userdata (bool), userstats (bool), username (str), userid (str) as parameters. It will organize the data from tables in dict format without missing anything and return desired table data.

3. Generator.py:

- *UserIDGenerator*: Generates user IDs with a given prefix.
- *VerifyCodeGenerator*: Generates verification codes.
- *EmailSender*: Sends emails for verification or welcome messages.
- *WaterIntake*: Calculates daily water intake recommendation based on user data.

4. Hovertooltip.py: Displays a tooltip when hovering over a widget with text.

5. JsonMethods.py:

- *save_data*: Saves data to a JSON file.
- *load_data_mode*: Loads data from a JSON file based on a mode.

6. LevelingSystem.py:

- *update_exp_lvl*: Updates experience level in userdata table of database.
- *send_notification*: Sends OS-specific notifications.
- *LevelingSystem*: Manages user leveling.
- *generate_level_threshold*: Generates level thresholds.
- *check_level_up*: Checks for level-up eligibility.

7. DateBox.py: A widget for selecting dates with a button to display a calendar for date selection.

8. CalendarToplevelWindow.py: A window for selecting dates using a calendar widget, updating data upon selection.

9. FloatSpinbox: Custom widget for selecting floating-point numbers with buttons for increasing and decreasing the value.

10. GraphManger.py:

- `__setfigure`: Creates a Matplotlib figure with specified size.
- `__extract_data`: Extracts data from a dictionary for plotting.
- `bar_graph`: Plots a bar graph using extracted data.
- `pie_graph`: Plots a pie chart using extracted data.

Base contains code files

1. WaterTracker.py :

- This file is base of the whole script, It contains imported modules like customtkinter, pillow, logging and other modules which are created for this project.
- It contains a single class for the whole instance of the app called WaterTracker which take CTK as parent class.
- It contains methods
 - `logger_setup`: This is used in order to create a logging instance and console handler, it returns logger as object.
 - `__init__`: This method is used to import Assets files and json_mode_files using try and except block. It configures the instance window and changes the colors, grid row and column configuration, main frame and side page frame.
 - `set_sidebar`: This function contains sidebar frame; its functionalities are hiding (deletes the frame) the side bar and if it hidden it shows (create the frame) like trigger logic. Default is displays sidebar, where everything will be added to sidebar frame. And adds menu bar image for user-friendly.
 - `show_pages`: This is an important part of the code where we change the code files, for deeper understandings: The side frame is provided as instance for base folder program files like homepage, loginpage, registerpage, settingspage, forgotpage, historypage.
 - ```
from .HistoryPage import HistoryPage
from .RegisterPage import RegisterPage
from .LoginPage import LoginPage
from .HomePage import HomePage
from .SettingsPage import SettingsPage
from .ForgetPage import ForgetPage
```

```

self.pages = {
 "HomePage": lambda: HomePage(master=self.side_page,app=self),
 "LoginPage": lambda: LoginPage(master=self.side_page,app=self),
 "HistoryPage": lambda: HistoryPage(master=self.side_page,app=self),
 "RegisterPage": lambda:
RegisterPage(master=self.side_page,app=self),
 "SettingsPage": lambda: SettingsPage(master=self.side_page,app=self),
 "ForgetPage": lambda: ForgetPage(master=self.side_page, app=self)
}

```

Every instance of this frame classes is traced to object called `current_page`.

- It Contains small logic for resizing screen and instance trace.
- Default is set to loginpage

## 2. LoginPage.py:

- This file contains a class `LoginPage` where it takes `CTkFrame` from `customtkinter` as parent class. It contains basic layout for login page.
  - *Import:* Database methods `last_login`, `database_extractor` from Utilities folder.
  - *\_\_init\_\_:* Initializing layout for all the basic structure for login page with frame and labels. This takes `self`, `master` and `app` as parameters, `app` is the `WaterIntake` base instance, `master` is the current side frame from the base instance. It contains login frame where it has labels like `username`, `password`, `confirm password`, `eye image (off/on)`. Some password effects like `show password`, `hide password` automatically after release of button and enter, exit of forgot password label, sign up label. Forgot password label is connected to `ForgetPage`, While Sign up label is connected to `RegisterPage`.
  - *login\_button:* This function is activated when user click on the login button, it checks if the username and password is provided if not it shows and whiteframe (custom error) and if it is valid, it checks if database tables are present or not, if not it will create and show error that user not found. If user found it will trace to the `HomePage`. It does contain some special effects on entry box of username and password.

## 3. RegisterPage.py:

- This file contains `RegisterPage` class where it takes `CTkFrame` as the parent class. It contains basic logic for register page and its layout.
  - *Import:* It imports everything from `customtkinter`, `Image` from `pillow`, `datetime` from `datetime`, Utilities files: `Hovertooltip`, `CalendarTopLevelWindow`, `FloatSpinbox`, `Generator- UserIDGenerator`, `JsonMethods`. Base files: `WaterDatabase`.

- `__init__`: Configuring basic items, creating a dict for storing user input data,
 

```
self.register_data = {
 "userid": "",
 "name": "",
 "username": "",
 "password": "",
 "confirm_password": "",
 "dob": "",
 "gender": "",
 "email": "",
 "activity_level": "",
 "weight": "",
 "height": "",
 "register_time": "",
 'level': "",
 'experiencepoints': "",
 'lastlogin': "",
 'verificationcode': "",
 'verificationtime': "",
 'verifiedcode': "",
 'recoveryemail': "",
 'recoverycode': ""}
```

  - Creating scroll\_frame in order to group labels and subframes, it is quite complex design to elaborate. It contains heading Aqua Sign Up where it is mixture of one frame and two labels with bg and fg configuration.
  - It contains hovetooltip where it shows text when certain label or entry is bind with <Enter>. It contains name, username, username progressbar (below the username entry), username level, password, password progress bar, password level, confirm password, eye(on/off) for password and confirm password, validation command for username and password. Reveal, hide password and confirm password effects. Date of birth entry is two level entry and label on top of it, it contains select dob function where it opens a Calendar Top level Window and gets input from user. Gender checkbox and its event handler function, email, activity level is two level entry and option menu on top of it. Height, weight and finally register button which is bind to register\_user function.
- `register_user`: This function handles the logic to check if all the necessary data is provided from user or not, if no it will show whiteframe(custom error) and shows success if data is successfully loaded into database or else error in logger handler. If successful it will load Homepage.

#### 4. HomePage.py:

- This file contains the class HomePage where it takes CTkFrame as parent class, it contains basic layout for home page.
  - *Import*: It imports everything from customtkinter, Image, ImageDraw from pillow, Utilities files:DataMethods, LevelingSystem, GraphManger – pie\_graph, Generator-WaterIntake. Base files: WaterDatabase, from tkinter, ttk.
  - *update\_pie\_graph*: It not method in HomePage class but it does contain self, and app parameters. It doesn't have scope limit because of self. It updates the pie Chart whenever there is a change in userstats table.
  - *\_\_init\_\_*: Configuring basic items, extracting data from tables and modification of data like water stats etc.
    - Resetting data values, leveling system, and updating all this changes in table. Creating scroll bar frame, it contains sub frames top, mini bar, mid and down frames.
    - Top frame contains frames and labels for title, level progress bar, profile picture, username.
    - Mini bar frame contains 3 frames right, left and center with drink, drank and need to drink respectively.
    - The concept behind them is quite simple fetching data from table, organize and drive water intake from generator.
    - Center frame contains two progress bars with and six labels for start, end, zero to hundred percentage values.
    - Down Frame contains glass image where its bind to increase the value of currentdrink value in database table by using database methods inc\_value, it contains two buttons plus and minus to adjust the value of increment which is text of the label on top of the glass image.
    - Finally pie chart on left bottom to represent today's status in graphical representation. When ever a value in data changes it calls update\_pie\_graph function.

#### 5. HistoryPage.py:

- This file contains the class HisotryPage where it takes CTkFrame as parent class, it contains basic layout for history page.
  - *Import*: It imports everything from customtkinter, Image from pillow, Utilities files:DataMethods, DateBox, GraphManger – bar\_graph, datetime, timedelta, date from datetime, Calendar from tkcalendar.

- *update\_bar\_graph*: It not method in HistoryPage class but it does contain self, and app parameters. It doesn't have scope limit because of self. It updates the pie Chart whenever there is a change in userstats table.
- *\_\_init\_\_*: It contains top scrollable frame, title label, box from utilities, calendar from tk and other things in a systematic way
  - Inherits from the parent class and sets up the logger and userdata attributes.
  - Extracts userdata and userstats from the database using the provided app instance.
  - Configures the main scrollable frame and the application's color scheme.
  - Contains the title label for the water intake history page.
  - Includes a calendar widget for date selection.
  - Provides options for selecting different date ranges.
  - Divides the page into frames for displaying various statistics such as daily water intake goals, amount drank, points gained, and points lost.
  - Contains a frame for displaying graphs related to water intake statistics.
  - Configures the grid layout for all the frames and widgets. Defines functions for updating data based on user selections. Handles events such as calendar date selections with option\_selector .
  - It shows the graph at the bottom and calls
- *option\_selector*:
  - Retrieves history data from the user stats.
  - Initializes variables and converts date formats.
  - Defines a custom function to filter data based on start and end dates.
  - Defines a function to handle date selection events and update data accordingly.
  - Handles different options selected by the user:
    - 'Single': Retrieves data for a single day.
    - '7 days ago': Retrieves data for the past 7 days.
    - '30 days ago': Retrieves data for the past 30 days.
    - 'From the start': Retrieves data from the registration date to the selected date.
    - 'Custom': Allows the user to select custom date ranges.
  - - Updates widgets and bar graphs based on user selections.

## 6. SettingsPage:

- This file contains the class SettingsPage where it takes CTkFrame as parent class, it contains basic layout for settings page.
  - *\_\_init\_\_*: It contains top scrollable frame, title label, LevelingSystem, JsonMethods from utilities, ttk, filedialog from tk and other things in a systematic way
    - Initializes the settings interface.
    - On figures the app's color scheme.
    - Retrieves user data and user stats from the database.
    - Creates frames for profile settings, user details, and settings.
    - Sets up a scrollable frame for profile settings.

- Adds labels for user details and creates label-value pairs.
  - Displays the last login, last profile update, and last reset time.
  - Provides options to change appearance mode.
  - Allows users to upload a profile picture.
  - Configures row and column weights for layout.
  - Calls the `img_upda` method to update the profile picture.
- *img\_upda*: Checks if a profile picture exists for the user. Updates the profile picture displayed on the interface.
  - *upload\_picture*: Opens a file dialog for users to select an image file. Copies the selected image to the UserProfile folder. Updates the status label with a success message. Updates the last profile update time. Calls the `img_upda` method to update the profile picture.
  - *change\_appearance\_mode\_event*: Updates the appearance mode based on user selection. Saves the new appearance mode to the database.

## 7. WaterDatabase:

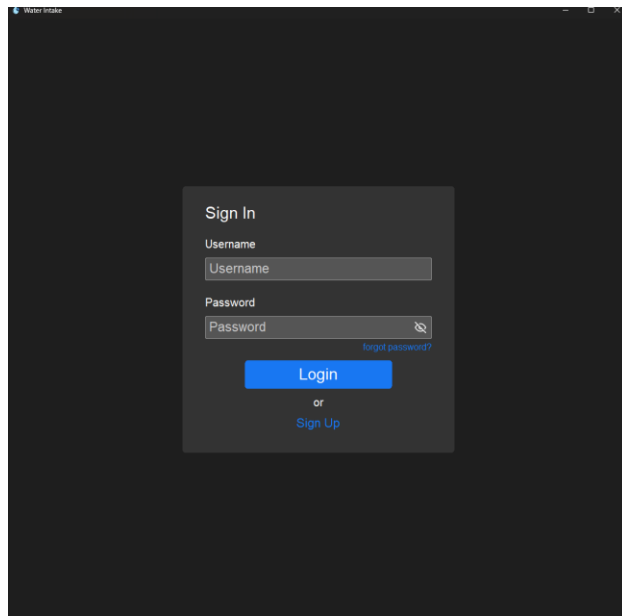
- This file contains class Database which communicate and has functions related to database.
  - *Import*: Imports logging and psycopg2
  - *\_\_init\_\_*: Initializes the Database class instance by checking the database connection and setting up logging.
    - Checks database connection status.
    - Sets up logging for debugging purposes. Configures logger settings such as log level, format, and handlers. Creates both console and file handlers for logging. Returns a logger object for logging events.
  - *Logger\_setup*: Sets up the logger for the Database class instance.
    - Configures logger with the specified name. Sets log level to DEBUG. Defines a logging format including timestamp, log level, filename, and message. Creates a console handler and sets its formatter. Creates a file handler with the specified filename and sets its formatter. Adds both console and file handlers to the logger. Returns the configured logger object.
  - *\_\_connection\_check*: Checks the database connection status and establishes a connection if possible.
  - *table\_exists*: Checks if a table exists in the database.
  - *create\_userstats*: Creates the 'userstats' table if it doesn't exist, based on certain conditions.
    - Checks if the 'userstats' table does not exist but the 'userdata' table exists.

- Executes SQL queries to create the 'userstats' table with appropriate columns and constraints. Sets datestyle for date formatting. Commits the transaction after table creation.
- *create\_tables*: Creates necessary tables ('userdata' and 'userstats') if they don't exist.
- *get\_all\_values*: Retrieves all rows from a specified table in the database.
- *get\_values*: Retrieves specific row(s) from a specified table based on user ID or username.
- *set\_values*: Inserts new data into the specified table, checking for existing data based on user ID or username.
- *close\_connection*: Closes the database connection.



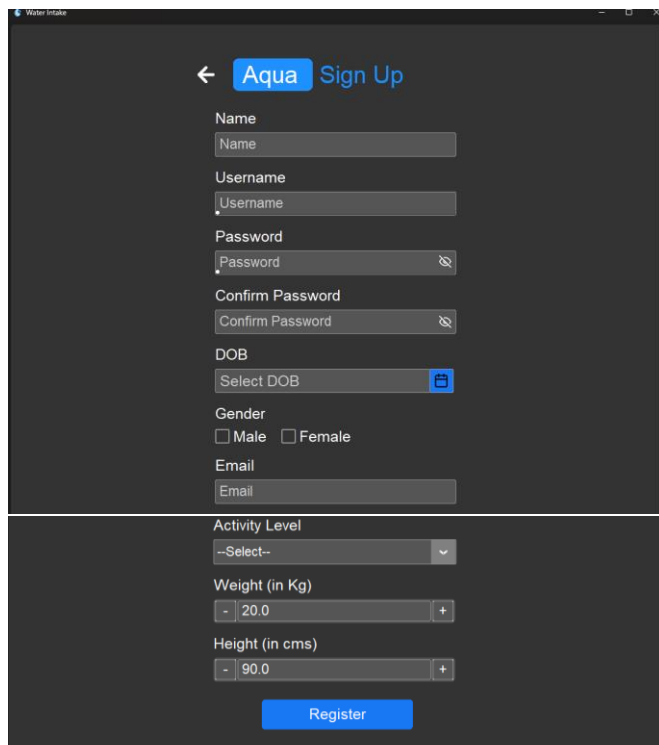
# OUTPUT SCREENSHOT:

## 1. Login Page:



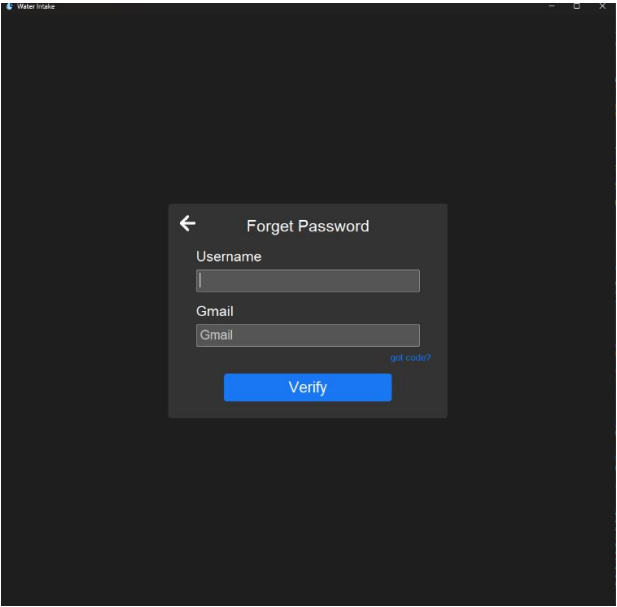
The screenshot shows a login interface titled "Sign In". It features two input fields: "Username" and "Password". The "Password" field has a toggle icon for visibility. Below the fields is a blue "Login" button. A link "forgot password?" is positioned to the right of the password field. Below the login button, the text "or" is displayed, followed by a blue "Sign Up" link.

## 2. Register Page:

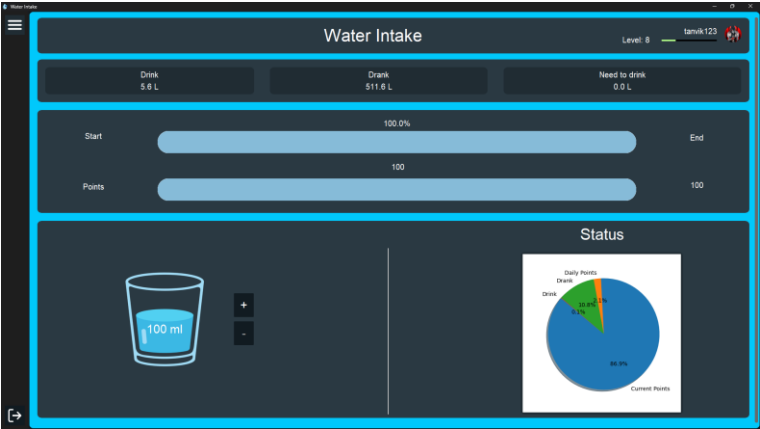


The screenshot shows a registration interface titled "Aqua Sign Up". It includes a back arrow and the title. The form contains several input fields: "Name", "Username", "Password", "Confirm Password", "DOB" (with a calendar icon), "Email", "Activity Level" (a dropdown menu), "Weight (in Kg)" (with minus and plus buttons), and "Height (in cms)" (with minus and plus buttons). There are also checkboxes for "Male" and "Female". A blue "Register" button is located at the bottom.

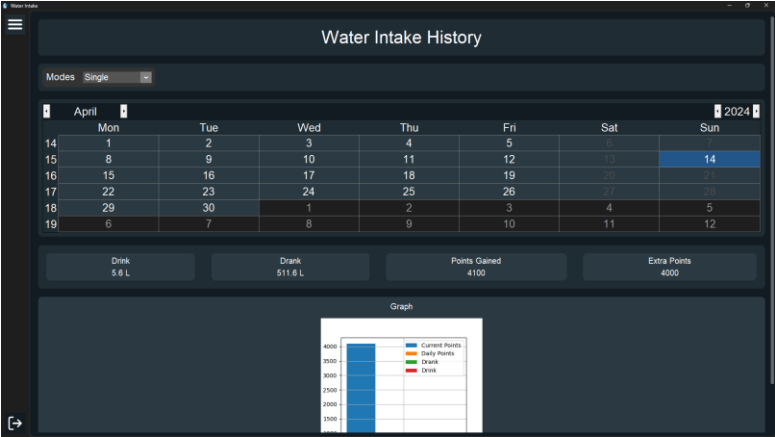
3. Forget Password:



4. Home Page:



5. History Page:



6. Settings Page:

Settings

Name:

tanvi123

Email:

tanvikarim@gmail.com

Weight:

60

Height:

160

Date of birth:

2005-03-02

Gender:

Male

Level:

8

Experience Points:

226 / 960

Activity Level:

Low Level

Last login:

14-04-2024 21:32:25

Last DP Upload:

14-04-2024 14:40:35

Last reset Time:

14-04-2024 00:00:00

Appearance Mode

Dark

Profile Picture:

Upload Picture:

Upload

## **CONCLUSION**

In conclusion, the development of the Water Intake for Efficient Hydration application addresses the pressing need for a user-friendly solution to track and enhance hydration levels. With its intuitive interface, personalized settings, and error logging features, the application offers a seamless experience for users to monitor and improve their water intake habits. By providing comprehensive tracking and customization options, this application aims to empower individuals to achieve optimal hydration for better health and well-being. Overall, Water Intake for Efficient Hydration is a valuable tool in promoting healthier lifestyles and preventing dehydration-related issues.

## REFERENCES

- Matplotlib
  - <https://www.analyticsvidhya.com/blog/2020/02/beginner-guide-matplotlib-data-visualization-exploration-python/>
- Notification:
  - [https://www.youtube.com/watch?v=g\\_Gjq1pmv-k](https://www.youtube.com/watch?v=g_Gjq1pmv-k)
- Datetime
  - [https://www.youtube.com/watch?v=g\\_Gjq1pmv-k](https://www.youtube.com/watch?v=g_Gjq1pmv-k)
  - <https://java2blog.com/check-if-date-is-between-two-dates-python/>
- Customtkinter
  - <https://customtkinter.tomschimansky.com/documentation>
  - <https://github.com/TomSchimansky/CustomTkinter/issues>
- Tk Calendar
  - <https://tkcalendar.readthedocs.io/en/stable/documentation.html>
- Tk Events
  - <https://tkdocs.com/tutorial/windows.html>
- Colors
  - <https://htmlcolorcodes.com>
- Pillow
  - [https://www.tutorialspoint.com/python\\_pillow/index.htm](https://www.tutorialspoint.com/python_pillow/index.htm)
- Postgresql
  - <https://www.geeksforgeeks.org/python-postgresql-update-table/>
- Ctk Progress Bar
  - <https://python-hub.com/ctkprogressbar-progress-bar-in-customtkinter/>

## EVALUATION SHEET

**Reg.No :** URK23CS1261

**Name:** TANVIK SRI RAM REDDY

**Course code:** 23CS1007

**Course Name:** Python Programming

| S.No  | Rubrics                         | Maximum Marks | Marks Obtained |
|-------|---------------------------------|---------------|----------------|
| 1     | Industrial Certification        | 10            |                |
| 2     | Real – Time Applications Design | 30            |                |
| Total |                                 | 40            |                |

**Signature of the Faculty-in-charge**

**Signature of the Examiner1**

**Signature of the Examiner2**