TIMESAVVY

- Introduction
  - Project Overview
  - Project Goals
  - Audience
    - Normal User
    - Admin
    - Super User
- Getting Started
  - Technologies Used
  - Installation and Configuration
    - Prerequisites
    - Steps:
- Project Architecture
  - ER Diagram
- Project Implementation
  - Backend (Timesavvy)
  - User Interface (TimesavvyUI)
  - Database Connection Setup (database.py)
  - ORM Models (models.py)
    - Creating Tables
- Feature Overview
    - Login
    - Sidebar
  - Interface for Normal User
    - Profile Section
    - Timesheet Section
      - Timesheet Functionality
      - API Endpoints
    - Downloads Section
      - API Endpoint:
    - Comp Off Section
      - Comp-Off Functionality
      - API Endpoints:
  - Interface for Admin
    - Timesheet Status Section
      - 1. Timesheet Window Status
      - 2. Timesheet Completion Status
    - Upload Data Section
      - 1. Upload Employee Data
      - 2. Upload Monthly Leavesheet Data
      - 3. Upload Holiday Data
      - 4. Download Employee Data
    - Edit Timesheet Section

        - API Endpoints
    - Stoxx Sheet Section
        - Workflow:
        - API Endpoint:
    - Role Allocation (Super User Specific)
- Conclusion
- Contributors
- Other Attachments

Introduction

Project Overview

The TimeSavvy project aims to revolutionize timesheet management by automating manual tasks and offering a user-friendly interface for efficient and accurate timesheet generation. Leveraging the power of Python, FastAPI, and Streamlit, this project seeks to streamline the process, reducing errors and saving time for users. This project will enhance productivity and ensure precise tracking of work hours, making timesheet management more seamless and effective.

Project Goals

- Create an interactive user interface to replace manual Excel timesheets.
- Integrate built-in checks and alerts for accurate data entry.
- Simplify new employee onboarding by eliminating use of Excel, reducing administrative time.
- Enable detailed visibility for approvers to quickly review and approve timesheets.
- Implement automatic format updates across all relevant sheets for consistency.
- Option to download the Stoxx sheet and Summary sheet in its specified format.

Audience

The audience for the TimeSavvy application is divided into three main roles: Super User, Admin, and Normal User. Each role has distinct capabilities and permissions within the application, ensuring that users can perform their tasks efficiently and securely.

Normal User

Normal Users are the primary users of the TimeSavvy application. They have the following capabilities:

- **Fill Timesheets**: Can fill timesheets when the window for timesheet filling is open.
- **Download Timesheets**: Can download any previously filled timesheets.
- **Comp Off**: Can avail Comp Off.

- **Profile**: Can view their details in the profile section of the application.

Admin

Admins have advanced permissions to manage and oversee the timesheet process. Their capabilities include:

- **Manage Timesheet Window**: Freeze/Unfreeze the timesheet filling window.
- **Monitor Timesheet Status**: Can view the status of employees' timesheet completion (e.g., completed, partially completed, not started).
- **Data Management**: Can upload file of employee data, leaves, and holidays to be stored in the database.
- **Edit Timesheets**: Fill/edit the timesheets of any employee, even when the timesheet window is closed.
- **Download Employee Details**: Can download file containing employee details.
- **Download Reports**: Can download a zip file of the Stoxx sheet and Summary sheet by selecting project codes.

Super User

Super Users have the highest level of access within the TimeSavvy application. They have all the rights of an Admin, in addition to the following:

- **Role Allocation**: Can allocate roles (Super User, Admin, Normal User) to any user, managing user permissions and access levels within the application.

This structured approach ensures that each user has the appropriate level of access and functionality, enabling efficient management and use of the TimeSavvy application.

Getting Started

Technologies Used

Streamlit for UI

Streamlit is used to create a user-friendly and interactive interface, making it easy for users to navigate and manage timesheets.

PostgreSQL as Database

PostgreSQL serves as the database, providing a reliable and scalable solution for storing and managing data.

FastAPI for Backend

FastAPI is utilized for the backend, facilitating quick and efficient API development. It connects the user interface with the database, ensuring smooth data flow and interaction. Also, FastAPI provides its own documentation (Swagger UI, ReDoc)

Python as Framework

Python is chosen as the overarching framework due to its compatibility with Streamlit, PostgreSQL, and FastAPI, allowing seamless integration and development across the entire project.

Installation and Configuration

Prerequisites

Before starting the installation, ensure you have the following prerequisites:

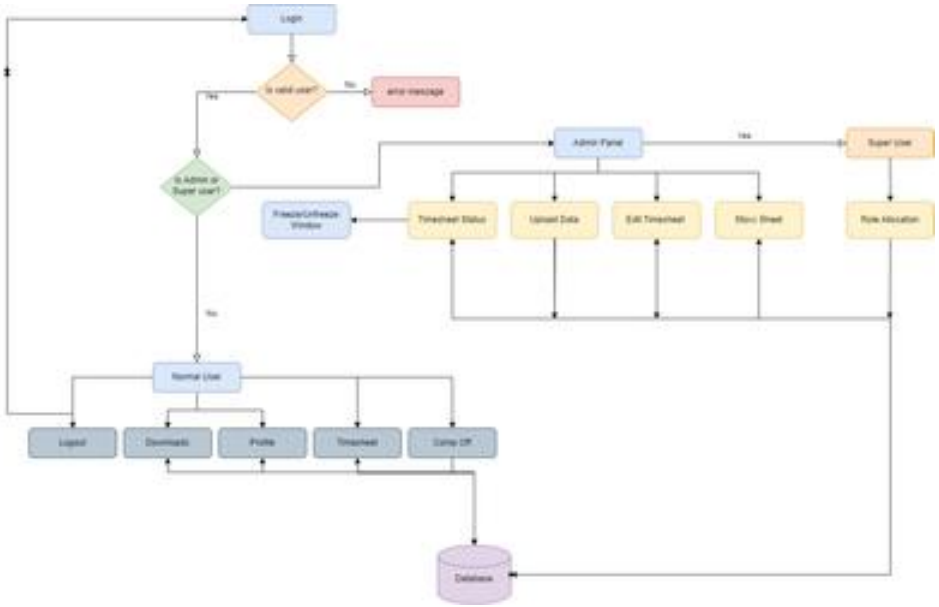
- Python 3.x installed on your system
- PostgreSQL database server with credentials (username, password, host)
- Git installed to clone repositories

Steps:

- **Clone Repositories**
  - Clone the frontend and backend repositories from GitHub.
- **Backend Setup**
  - Install dependencies from `requirements.txt`.
  - Initialize the database schema.
  - Run the FastAPI backend server.
- **Frontend Setup**
  - Install dependencies from `requirements.txt`.
  - Start the frontend development server.
  - Access the frontend UI in your browser.
- **Initial Admin Setup**
  - Use the Swagger UI of FastAPI to upload the initial employee data file.
- **Integration**
  - Verify backend and frontend integration.
  - Test all functionalities for proper operation.

Project Architecture

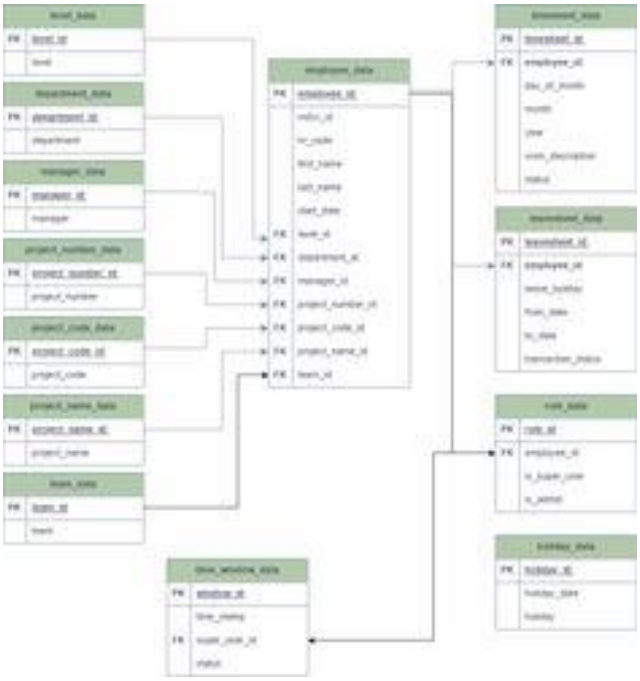
Flow Chart



Database Table Schema

ER Diagram

This shows the schema of the tables formed in the database and their relationships with each other.



Project Implementation

The TimeSavvy project is divided into two main components: the backend and the user interface (UI), maintained in separate GitHub repositories named **Timesavvy** and **TimesavvyUI**.

Backend (Timesavvy)

The backend handles all computational tasks, manages the database, and provides APIs for the UI to interact with the database. Key aspects include:

- **API Development**: Using FastAPI, the backend provides endpoints to handle operations like data retrieval and processing, and updates, all in JSON format.
- **Database Connection**: The backend of the TimeSavvy project connects to a PostgreSQL database using the psycopg2 module. The connection is established and managed through SQLAlchemy, which provides a higher-level interface for interacting with the database.
- **Data Exchange**: JSON format is used for data exchange between the backend and the UI, ensuring seamless communication.
- **Error Handling**: Robust error handling and validation mechanisms are in place to manage data integrity and provide meaningful error messages.

User Interface (TimesavvyUI)

The UI is developed using Streamlit, offering an interactive and user-friendly interface. Key aspects include:

- **Interactive Interface**: Users can fill timesheets, request compensatory time off, view profiles, and download timesheets. Admins and Super Users have additional management functionalities.
- **API Integration**: The UI communicates with the backend via APIs to submit and retrieve data, ensuring real-time updates and dynamic content rendering.
- **Role-Based Access**: The UI enforces role-based access control, providing appropriate functionalities based on user roles (Normal User, Admin, Super User).

By separating the project into backend and UI components, and ensuring clear communication through APIs, TimeSavvy achieves a modular, maintainable, and efficient timesheet management system.

Database Connection Setup (database.py)

- **create_engine**: This function from SQLAlchemy creates a connection to the database using the provided URL. The `engine` object manages the connection pool.
- **SessionLocal**: This is a factory for creating new SQLAlchemy sessions. A session is used to interact with the database. `autocommit` and `autoflush` are set to `False` to control transaction management and data consistency.

- **Base**: This is the base class for all ORM models. Models will inherit from this base class to be recognized by SQLAlchemy.
- **get_db**: This function provides a local session for each request. It yields a database session and ensures it is closed after use, preventing connection leaks.

ORM Models (models.py)

- **Defining Models**: Classes represent database tables. For example, `EmployeeData` represents the `employee_data` table, with columns defined using SQLAlchemy's `Column` class.
- **Relationships**: Models can have relationships with other models to enable complex queries and maintain data integrity. For instance, `EmployeeData` has relationships with `LevelData`, `TeamData`, and `DepartmentData`.
- **Example Model:** This model defines the `employee_data` table with columns for employee details and relationships to other tables.
- `class EmployeeData(Base):`
- `    __tablename__ = "employee_data"`
- `    employee_id = Column(Integer, primary_key=True, autoincrement=True)`
- `    first_name = Column(String(50), nullable=False)`
- `    last_name = Column(String(100))`
- `    level_id = Column(Integer, ForeignKey("level_data.level_id"))`
  `    level = relationship("LevelData", back_populates="employees")`

Creating Tables

```
models.Base.metadata.create_all(bind=engine)
```

This line generates the necessary SQL to create the tables in the database based on the defined models, if they do not already exist.

In summary, this setup ensures efficient database connectivity and operations using SQLAlchemy ORM models, with automatic table creation and robust session management.

Feature Overview

Login

The very first page of the TIMESAVVY application is the Login page, where users log in by entering their Indxx ID. The UI takes this Indxx ID as input and calls a simple API to verify it. The API fetches and returns the employee's information based on their Indxx ID and store it in “user_profile”. If an invalid Indxx ID is entered, the application displays an error message.

Endpoint:

```
GET /users/{indxx_id}
```

Sidebar

After a successful login, the TIMESAVVY UI displays a sidebar menu with different options based on the user's role. The sidebar menu is implemented using the `option_menu` component from Streamlit. For Admins and Super Users, the sidebar includes an "Admin Panel" option. The code for sidebar is written in the app.py file of the repository.

Interface for Normal User

For normal users, the sidebar menu includes the following options:

Profile Section

The Profile option in the sidebar menu displays basic information about the user, such as name, joining date, manager name, project name, and other relevant details. All the information is taken from “user_profile” which was created while login.

Timesheet Section

The Timesheet section allows users to manage their timesheets by providing inputs for the month and year. Depending on the state of the timesheet window (freezed/unfreezed) and whether the timesheet has already been generated, users can fill out or view their timesheet data.

The timesheet data is displayed using the `aggrid` framework in Streamlit, which presents the information in an interactive grid format. All the columns of the timesheet except the 'Work Description' is not editable. The days of the month has a “0” row in it, it is for the summary of the work description.

Day of month	IN	OUT	Work Descript...	Status
0				
1	10:00	19:00		
2	10:00	19:00		
3	10:00	19:00		
4	10:00	19:00		
5	10:00	19:00		
6	10:00	19:00		Saturday
7	10:00	19:00		Sunday

Timesheet Functionality

1. **Input for Month and Year**:
    - If the timesheet window is open, users can input the month and year.
    - If the timesheet window is closed, no input options are available.
    - Then the user should click on the generate button to generate timesheet grid.
2. **Current Month/Year**:

    - If the user is generating the timesheet for the first time for the current month/year, an empty template is shown for the user to fill out with the “status” prefilled from the data of leaves and holidays provided beforehand.
    - If the timesheet has already been generated, the stored timesheet data is fetched from the database.
3. **Filling and Saving Timesheet Data**:
    - Users fill in the work description in the timesheet.
    - After pressing the save button, the data is subjected to certain checks and validations. The work description corresponding to “Saturday”, “Sunday”, “Holiday” and “Leave” is made blank, even if it is filled by the user.
    - The validated data is then stored in the database.
    - After the save button is clicked then a preview table of the timesheet stored is displayed.

API Endpoints

1. **Fetch Timesheet Data**: `GET /time_sheet_data`
2. **Store Timesheet Data**: `POST /add_timesheet`

Downloads Section

The Downloads section allows users to download their timesheet data in CSV format from the database. Users provide the month and year as input, and if the data exists in the database, it is fetched and displayed using the `aggrid` framework, else an error message is displayed. Users can view and download the timesheet, but they cannot edit it.

API Endpoint:

```
GET /time_sheet_data
```

Comp Off Section

The Comp-Off section allows users to apply for compensatory off (comp-off) leave by providing the required input details. Users can specify the duration of the comp-off by entering the start and end dates and selecting the status as “AVAILED“ or “NOT AVAILED“. The provided data is then stored in the database and reflected in the timesheet as leave.

For admin users, the functionality extends to adding or updating comp-off data for any user by providing the Indxx ID as input in the UI as well as can view the Comp-Off data stored in database.

If the Comp Off is to be cancelled then the user or admin can select the start and end dates for that particular employee and choose status to “NOT AVAILED“.

Comp-Off Functionality

1. **Input Details**:
    - Users provide the start date (`from_date`) and end date (`to_date`) for the comp-off leave.
    - Users select the status of the comp-off leave as “AVAILED“ or “NOT AVAILED“ (`transaction_status`).
2. **Storage and Update**:
    - The input data is stored in the database using the `update_comp_off_data` API.
    - The comp-off data is also updated in the timesheet as leave for the specified period.
3. **Admin Role**:
    - Admin users can add or update comp-off data for any user by providing the Indxx ID as input in the UI.

API Endpoints:

```
GET /comp_off

POST /update_comp_off_data
```

Interface for Admin

For Admin/Super User role, the sidebar menu includes these additional options:

Timesheet Status Section

This section in admin panel provides additional functionalities for admins/super-users to manage timesheet status and monitor the completion status of timesheets. The panel includes the following sections:

1. **Timesheet Window Status**
2. **Timesheet Completion Status**

1. Timesheet Window Status

This option allows administrators to freeze or unfreeze the timesheet filling window for users.

The window automatically unfreezes at the start of the month.

API Endpoints:

```
POST /update_time_window_status

GET /get_time_window_status
```

2. Timesheet Completion Status

This option allows administrators to view the status of timesheet completion by selecting respective project codes of the users. The completion status is displayed as a donut chart, which shows three divisions: completely filled, yet to start, and partially filled. Admins can also view the list of completely filled, yet to start, and partially filled by clicking the corresponding buttons available in the UI.

API Endpoint:

`POST /timesheet_status`

Upload Data Section

The upload data section in the admin panel allows administrators to manage employee data, leavesheet data, and holiday data by uploading CSV files. This section provides functionalities to upload and download employee data, and to upload monthly leavesheet and holiday data. Additionally, administrators can download sample files in CSV format to ensure the data matches the required format by clicking the Download sample file button.

1. Upload Employee Data

API Endpoint: `POST /add_employee_data`

2. Upload Monthly Leavesheet Data

API Endpoint: `POST /upload_leavesheet`

3. Upload Holiday Data

API Endpoint: `POST /upload_holidaysheet`

4. Download Employee Data

API Endpoint: `GET /employee_data`

Edit Timesheet Section

The Edit Timesheet section in the admin panel allows administrators to make corrections or edits to the timesheet of any user, even after the timesheet filling window is closed. This feature ensures that any necessary adjustments can be made to the timesheet data, even outside the regular timesheet submission period.

API Endpoints

1. **Fetch Timesheet Data**: `GET /time_sheet_data`
2. **Store Timesheet Data**: `POST /add_timesheet`

These capabilities ensure that any corrections or adjustments needed in the timesheet data can be made efficiently by administrators, maintaining the accuracy and integrity of the timesheet records.

Stoxx Sheet Section

The "Stoxx Sheet" option is the final stage of the TIMESAVVY project. This feature allows administrators to generate and download the Stoxx sheet and summary sheet based on the timesheet data stored in the database. Admins can select the month, year, and project codes to create a zip file containing the formatted sheets.

Workflow:

1. **Admin Inputs:**
    - Admins provide the month and year.
    - Admins can select one or more project codes from the available list.
2. **Generate Stoxx and Summary Sheets:**
    - The selected inputs are used to fetch the corresponding timesheet data from the database.
    - The data is formatted into the predefined Stoxx sheet and summary sheet formats.
    - The formatted sheets are compressed into a zip file.
3. **Download Zip File:**
    - The generated zip file is made available for download via the UI.

API Endpoint:

```
POST /get_stoxx_sheet
```

Role Allocation (Super User Specific)

The "Role Allocation" option is exclusively available to super users within the hierarchy. This feature allows super users to manage roles, including assigning or removing admin and super user roles. This functionality represents the highest level of hierarchy and control within the project.

API Endpoint:

```
POST /create_role
```

This feature ensures that role management is centralised and controlled by super users, maintaining the integrity and security of the TIMESAVVY system by regulating access and administrative capabilities.

Conclusion

TIMESAVVY is a comprehensive timesheet and worklog management system designed to streamline employee time tracking and administrative tasks. Key features include a secure login system, an intuitive sidebar menu with distinct user and admin options, and detailed sections for timesheets, downloads, and comp-off requests. The admin panel offers advanced functionalities such as freezing/unfreezing timesheet windows, monitoring timesheet status via pie charts, and uploading essential data like employee records, leaves, and holidays. Additionally, admins can edit timesheets post-submission, download consolidated Stoxx and summary sheets, and manage user roles, with super users having exclusive access to role allocation.

TIMESAVVY's user-friendly interface and robust APIs ensure data integrity and seamless interactions, enhancing efficiency and accuracy in time management for both employees and administrators. This project exemplifies the integration of functionality and ease of use, making it an invaluable tool for organizations aiming to optimize their time tracking processes.

Contributors

This project was developed by `Avinash Shukla`, `Aryan Jalkote`, and `Himanshu Mittal` under the mentorship of `Mr. Shaine Ansari` with invaluable contribution from `Mr. Sweet Ramteke` and `Mr. Tushar Shrivastava`.