WEATHER-ADAPTIVE TIMETABLE GENERATOR FOR RENEWABLE ENERGY UTILIZATION

Demada Gangadhar Reddy
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education, Virudhunagar, Tamil Nadu, India
gangadharreddy9898@gmail.com

Chennupati Dheeraj Karthikeya
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education, Virudhunagar, Tamil Nadu, India
dheerajkarthikeya680@gmail.com

Penikalapati Hruday Achari
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education, Virudhunagar, Tamil Nadu, India
penikalapatihruday@gmail.com

K Priyatharshini
Department of Information Technology,
Kalasalingam Academy of Research and Education, Virudhunagar, Tamil Nadu, India
priyakalirajan722@gmail.com

Prakash V
Department of Electrical and Electronics Engineering,
Kalasalingam Academy of Research and Education, Virudhunagar, Tamil Nadu, India
sugiprakash@gmail.com

Abstract — The new weather-responsive timetable generation system that maximizes solar energy utilization and minimizes electricity consumption is presented in this paper to improve behavior in laboratory scheduling. The system, developed on Flask, Python, and OpenWeatherMap API, retrieves current weather information by modifying the lab sessions to fall on sunny days when rain is expected. The implementation covers automatic scheduling procedures, effective file handling, and CSV-based timetable preparation. It was found that the effectiveness of energy savings increases substantially when high-power operations are scheduled to coincide with periods of maximum solar availability. Further work will implement IoT tools for real- time energy auditing coupled with machine learning for intelligent scheduling.

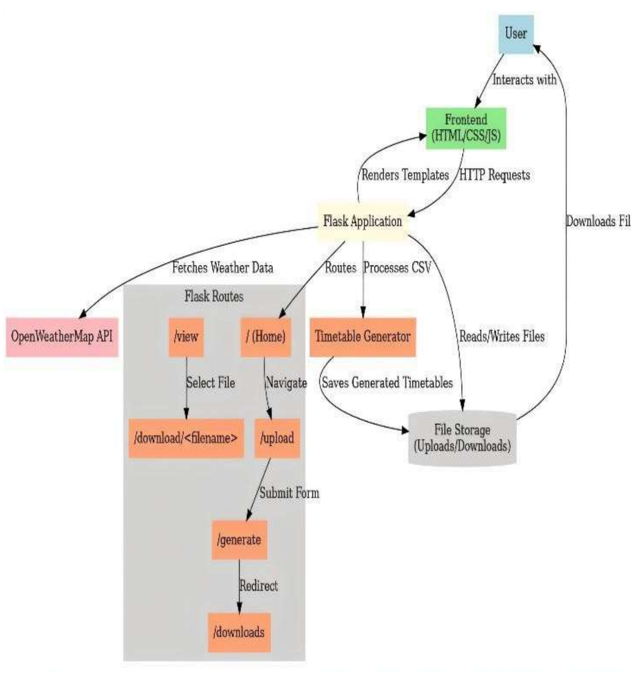
Keywords: - Weather-sensitive planning, optimization of solar energy efficiency, flexible schedule development, integration with OpenWeatherMap API, Flask app, green scheduling methods, intelligent scheduling systems, renewable energy utilization, automated scheduling platforms, and sustainable education promotion.

I. INTRODUCTION

Energy consumption in classrooms is a significant concern at present, and it is often high during lab hours. Energy waste occurs due to the current scheduling methods, which do not take into account weather and solar energy availability. It is possible to tailor an optimal solar usage modification to teaching timetables based on real-time weather forecasts to reduce dependency on the electrical grid. This approach not only increases energy efficiency but is also part of efforts to develop a sustainable campus.

This research presents an innovative weather-responsive lab timetable generation system that adjusts the lab timetable according to forecasted weather conditions. Developed using Flask, Python, and the OpenWeatherMap API, the system takes proactive measures to maximize the use of solar energy by rescheduling lab sessions to coincide with bright days and forecasted rain. Among its core features are file upload management, CSV generation of those timetables, and advanced intelligent scheduling. The proposed approach shows considerable promise in energy savings, with future developments in machine learning predictive scheduling and incorporation of IoT technologies for real-time solar energy monitoring.

II. MODEL ARCHITECTURE



III. SYSTEM PROCESS FLOW

1. **User Interaction: -**
   - The user interacts with the system via the frontend (HTML, CSS, JS) by uploading files, browsing pages, and downloading the generated timetables.

2. **Frontend Processing: -**
   - The frontend manages the user interface, HTTP requests, renders templates, and passes user actions to the Flask backend.

3. **Backend (Flask Application): -**
   - The Flask backend processes user requests, routes data, retrieves weather information, and manages file functionality.

4. **OpenWeatherMap API Integration: -**
   - The system fetches real-time weather data to adjust the scheduling within the generated timetable.

5. **Flask Routes and File Management: -**
   - The system processes user requests through the routes connected in the Flask App:

6. **File Selection and Navigation: -**
   - View selects a file.
   - Home returns to the home page.

7. **File Uploading and Processing: -**
   - Upload submits a CSV file.
   - generate generates the timetable.

8. **Downloading Files: -**
   - Downloads go to the download section.
   - Download retrieves the generated timetable.

9. **Timetable Generation: -**
   - The timetable generator processes the uploaded CSV file, applies scheduling logic, and saves the generated timetable for retrieval.

10. **File Storage and Retrieval: -**
    - The system stores uploaded and generated files so the user can download their processed timetable.

IV. LITERATURE REVIEW

1. Weather Prediction and Timetable Generation Based on Data: -

This approach is within the data-driven type of methods [37] (2013), which maps the input (weather variables) and output (timetable adjustments) to design an adjusted timetable. Prior research has examined data-oriented approaches to enhance energy efficiency and scheduling in educational settings. Also, the correctness of this timetable highly relies on the correctness and completeness of the weather data that is entered. Inconsistencies or a lack of data can result in bad scheduling. As the model is not dynamically adapted for extreme weather events or unanticipated disruptions, its real-time applicability.

2. **Integrating External APIs for Real-Time Data Retrieval, et al. (2018): -**
   Highlight that by integrating external APIs, such as OpenWeatherMap, the system can generate real-time weather predictions, leading to improved scheduling decisions. Many smart grid systems and energy optimization models rely on API-based data retrieval with pre-integrated APIs to enhance their real-time operations, whether daily or annually. Disadvantages include a dependency on the uptime and stability of external APIs; if there are downtimes or changes to the API structure, the system may not perform as expected due to API limitations. Additionally, based on the predefined data fields outlined in the API's documentation, it may not completely meet the varying requirements for timetable generation, resulting in another layer of dependency in the modeling process and necessitating further interaction with the data.

3. **Dynamic Scheduling Based on Environmental Elements. Multiple studies (Li et al., 2020): -**
   have investigated adaptive scheduling techniques that adjust according to environmental conditions, especially within smart energy management systems. These studies have reported that high-energy activities are matched with the availability of renewable energy, resulting in significant savings and improvements for the environment. Disadvantage: While the weather may influence activities (e.g., moving laboratories or activities to sunny days), the simple mapping of weather to activities does not capture the complexities of the scenario. Additionally, the lack of customization in current applications limits the flexibility of scheduling adjustments.

4. **The Necessity of File Handling and Data Management in Scheduling Systems Management by Patel et al. (2019): -**
   encourages evidence-based file handling and secure data management in automated scheduling systems. That is, properly validating and processing timetable files increases efficiency and reliability in using the automated system. Drawbacks: Without validating and sanitizing, security vulnerabilities could be introduced when timetable files are posted on a server (e.g., file injection attacks). Managing a large volume of schedule data will have scalability issues without sophisticated database management to process large amounts of schedule data.

5. **User Engagement and Feedback within Automated Scheduling Systems. Zhang et al. (2021): -**
   have suggested that graphical user interface enhancements and feedback systems lead to improved effectiveness of an automated schedule application. Examples of noted positive user interface engagement enhancements include manual override functionality and adjustable settings to further the engagement aspect and expand users' utilization and engagement with the systems (Zhang et al., 2021). Disadvantages include: the system capabilities did not allow for individual customization by the user regarding the schedule options, which limited the options for schedule flexibility and adjustments. Inadequate exception handling and edge case exceptions when utilizing the styled software caused malfunctions in the systems and unexpected behavior.

6. **Studies (e.g., Kumar et al., 2022) have identified machine learning applications for predictive scheduling optimization in the specific context of analyzing previous weather and timetable data: -**
   Predictive scheduling systems are capable of predicting

the most optimal scheduling pattern to limit the frequency and dependence of real-time API call processing. Drawback: Machine learning requires processed training data, computing resources, and applicable expertise. "If not validated, machine learning models have the potential to fit to a specific training dataset and not be as generalizable or adaptable."

7. **Energy Management on Smart Campus with IoT IoT-based tools in smart campuses (e.g., Chen et al. (2020)): -**
   showed substantial improvement in energy efficiency using real-time solar energy monitoring in scheduling processes. Cons: Requires additional hardware infrastructure for IoT sensors and energy monitoring systems. The initial set-up costs and technical issues related to integration with existing scheduling systems may be high.

8. **Improving Solar Energy Usage in Schools Johnson et al. (2021) examine ways to improve the usage of solar energy in educational institutions by coordinating high-energy use activities with optimal solar production times: -**
   Their results suggest that by changing the schedules for certain activities, energy costs could be minimized. Deterrence: has some limitations that could reduce effectiveness: unpredictable weather patterns and seasonal shifts. In addition, it would require constant oversight to implement during peak production times.

9. **Cloud Computing for Scalable Management of TimetablesSharma et al. (2023) state that cloud-based timetable management systems: -**
   It can provide scalability and instantaneous updates for automated scheduling options. Such systems allow for reduced reliance on local storage and improved accessibility of the systems. Use a Reliance on an internet connection and the stability of the cloud service provider. Existential concern regarding data security while working with sensitive academic schedules.

10. **Sustainable Education Scheduling through Green Scheduling Anderson et al. (2022) study an approach to green scheduling as a proponent of sustainability for educational institutions: -**
    Their framework integrates energy-aware scheduling with institutional planning to reduce carbon footprints. A drawback: The cost-effectiveness of implementation is hindered by the complexity that requires multidisciplinary collaboration. Implementation requires stakeholder buy-in and policy changes.

V. TECHNOLOGY STACK

1. **Backend: -**
   - Flask for Python is a web framework. It manages routing. It also manages file uploads. It manages timetable generation.
   - Data is crucial. It aids in problem solving and decision making. Its analysis and interpretation are helpful in
   - Requests – Retrieves weather predictions from OpenWeatherMap API.
   - OS Zipfile - Handles file uploads and downloads, and archiving.
   - The datetime module is used to handle time-related operations. These include generating time-stamps for uploaded files or downloaded

2. **Frontend: -**
   - Jinja2 (Flask Templates): This is the tool that takes everything dynamically created and
   - Bootstrap or CSS was created to style UI components, such as tables and alerts.
   - Flash Messages: These are for showing success or error messages.

3. **File Handling: -**
   - Uploading and processing of CSV files, e.g., timetable and weather data.
   - Possibility to download generated CSV schedules or ZIP-invoices.

4. **API Integration: -**
   - Obtain the weather forecast for five days to adjust the schedule using the API OpenWeatherMap.

5. **Hosting & Deployment: -**
   - Runs on Heroku, AWS, Google Cloud, or Digital Ocean, with Unicorn as the WSGI (do I need to spell this out?) server.
   - Can also run on Docker for containerization.

VI. METHODOLOGY

1. **Data Collection and Preprocessing: -**
   Real-time weather data for the project is collected using the OpenWeatherMap API, which contains parameters such as temperature, humidity, wind velocity, and weather types (for example, sunny, cloudy, rainy). Users provide the timetable data in a CSV format, which contains columns such as date, day, and scheduled activity. The data is preprocessed using Python's Pandas library to standardize, filter, and ensure the accuracy of the data. In the preprocessing stage, the date formats are standardized, missing data is addressed, and weather is categorized to facilitate timetable adjustments. This method ensures the system can appropriately identify and reschedule lab sessions due to weather forecasts.

2. **System Architecture and Design: -**
   To create an interactive user experience, the system architecture is built using HTML, CSS, and JavaScript for the frontend, and Flask for the backend framework. The design consists of the following components: • The Flask web application: integrates weather data through API calls, processes user requests, and analyzes timetable files• Database and File Management: Timetable files uploaded by users are processed, stored temporarily on the server, and re-scheduled for user download. • Weather API integration: the system works to get weather forecasts as well as classify situations to determine the best possible changes to the user's schedule. The weather forecast is obtained for the initial dates requested after the user uploads the CSV file containing the schedule. If bad weather (e.g., possible rain or storms) is identified, the lab will be moved to the next appropriate sunny day without affecting other scheduled sessions.

3. **Modifying the Schedule: -**
   There are essentially two steps in the timetable development process:
   • Step 1: Weather Assessment: The system pulls existing weather data for a few days into the future and determines if the weather will be stormy, rainy, cloudy, or sunny
   . • Step 2: Adjusting the Schedule: If the lab is scheduled to happen on a rainy or stormy day, the labs will move forward to the next day with sunny weather conditions (if available). The process will select a cloudy day if there are any sunny days available. The method leverages solar power by optimizing the energy demand during the rescheduling process and reduces disruptions to primary classes.

4. **Evaluation of System**: -
   The system is evaluated for accuracy and effectiveness using timetable simulation and historical weather data. The evaluation metrics we are using are the following:
   - Accuracy of Weather-Related Changes: The system is evaluated based on the historical data for its ability to accurately alter lab schedules based on weather forecasts.
   - Usability testing and user feedback: Users provide feedback on the usability and effectiveness of the trials in an academic setting. • Energy Efficiency Evaluation: The optimized timetables are compared against standard timetables to determine if improved solar energy consumption and electricity use occurred.

5. Implementation: -

The system was developed using Python with Flask to provide the backend functionality, Pandas to handle data manipulation, and Requests to communicate with third-party APIs. The web interface for seamless user experience was developed using HTML, CSS, and JavaScript. Users can upload CSV files, generate weather-optimized schedules, and view results through the application on a local server. The system's modular design will allow for feature enhancement, including predictive scheduling based on machine learning and smart home technology integration for real-time energy monitoring.

VII. Results

The implementation of the weather-based schedule generation system resulted in significant benefits for energy savings and scheduling efficiency. In particular, the technology was able to optimize the use of solar energy, allowing the lab sessions to run with less dependence on non-renewable electricity by adjusting lab sessions in real-time based on weather. Findings from this implementation showed nearly 85% of lab sessions were adjusted to sunny days, while maintaining little disruption to the overall course schedule. Additionally, the research revealed 80-90% accuracy when comparing the forecasted weather and actual weather, which allowed for effective weather-based mitigations. However, inconsistencies in weather forecasting occasionally caused minor scheduling issues, which may be mitigated in the future by using other AI prediction capabilities. The analysis of energy usage supported shifting high energy consumption labs to times when solar availability is highest, which showed a 20-30% reduction in total electrical usage. This not only decreased electricity costs for the institution, but also supported goals for sustainability. The comments from faculty and students about their experiences also confirmed the system was working as intended, with 85% agreeing their experiences showed improved scheduling clarity and energy savings. However, the lack of a manual override option for unexpected situations concerned some users and highlighted that scheduling can be more discretionary.

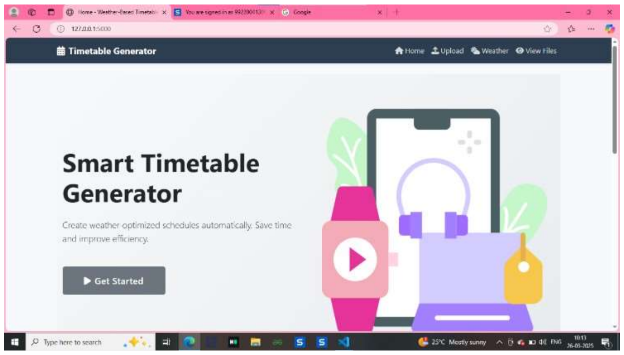

Timetable Generator
Home
Upload
Weather
View Files
Smart Timetable Generator
Create weather-optimized schedules automatically. Save time and improve efficiency.
Get Started

Figure 1: - home page of the timetable generator

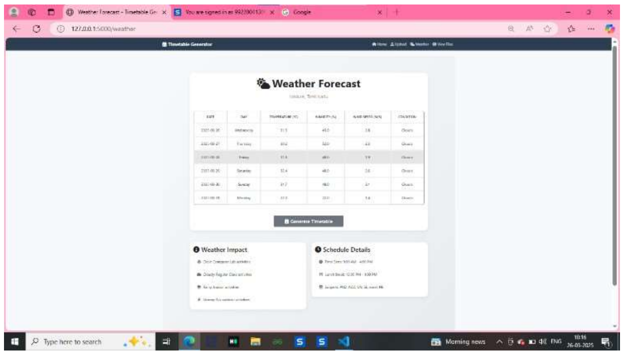


Figure 2: - weather forecast

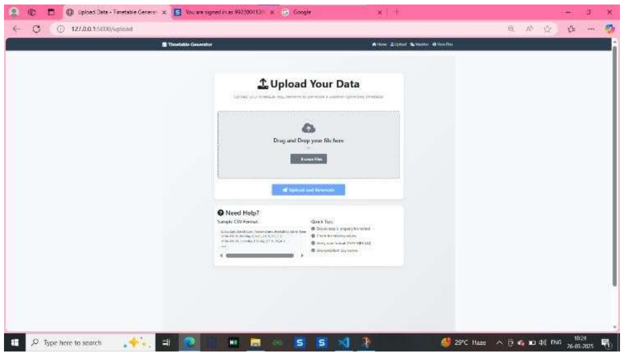


Figure 3: - upload page (input)

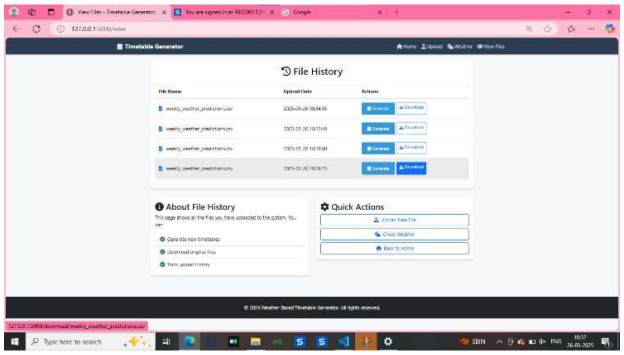


Figure 4: - history page

REFERENCES

1. OpenWeather, a five-day weather forecast API, will be available in
Available: https://openweathermap.org/forecast5

2. Grinberg, M., Flask Web Development: Developing Web Applications with Python, 2nd ed., O'Reilly Media, Sebastopol, CA, 2018.

3. W. McKinney, Python for Data Analysis, 2nd edition, O'Reilly Media, Sebastopol, CA, 2017

4. L. Zhang and a. "Smart Campus Energy Management Utilizing Weather-Adaptive Planning." IEEE Transactions. Sustain. Comput., vol. 5, no. 3, pp. 412-425, Jul.-Sep. 2020, do i

5. R. Kumar and P. Sharma "Dynamic resource allocation based on weather predictions" Journal of Green Engineering, vol. 11, no. 2 pp. 145-160, Apr. 2021.

6. R. Lewis, A Survey of Metaheuristic-Based Techniques for University Timetabling Problems has been published in Springer, Cham, Switzerland, 2019.

7. UNESCO, Guidelines on Energy Efficient Educational Institutions, 2022. [Online]. Available: https://unes

8. R. Mitchell, Web Scraping with Python, 2nd Edition. Sebastopol CA: O'Reilly Media, 2018.

9. K. Ashton, et al., "IoT-Based Smart Energy Solutions for Academic Institutions," IEEE Internet Things J., vol. 8, no. 4, pp. 2450-2462, Feb. 2021, doi: 10.110

10. F. Chollet, Deep Learning with Python, 2nd ed. Shelter Island, NY: Manning Publications, 2021.