**TIMETABLE GENERATOR PROJECT REPORT**

**Herry Patel 23075054**

**INTRODUCTION**

Efficient scheduling of classes is essential for educational institutions to optimize resource utilization and ensure smooth academic operations. Manual timetable creation is labor-intensive and error-prone, necessitating automated solutions. This project develops a timetable generator that schedules classes for multiple year groups, subjects, and teachers, adhering to constraints such as teacher availability, venue availability, and fixed lecture times. Built using Python and integrated into a Django web application, the system generates conflict-free timetables and exports them as Excel files that are displayed on the website, providing an accessible interface for administrators to manage scheduling tasks effectively.

**METHODOLOGY**

**(i) Data Input:** The system uses Excel files to collect input data, including subjects (e.g., MA101, PHY101, H106), teachers (e.g., ANOOP, PRASUN), year groups (1st, 2nd, 3rd Year), venues (e.g., LT-1.11, LT-2.21), and constraints like teacher and venue availability (Monday to Friday, 8-12 and 13-18). Fixed lectures, such as H106 on Thursday 16-18 at LT-3, are specified to ensure immutable scheduling.

**(ii) Algorithm:** The project employs the OR-Tools constraint solver to generate timetables. OR-Tools optimizes the scheduling by enforcing constraints like no overlapping classes for teachers or venues, maximum consecutive class hours, and fixed lecture times. The solver ensures balanced distribution of lectures, with options to schedule them consecutively on one day or distributed across days.

**(iii) Implementation:** The timetable generator is implemented in Python using libraries like pandas for data handling, openpyxl for Excel operations, and OR-Tools for constraint solving. The logic processes inputs, validates constraints, and generates schedules, which are saved as Excel files (e.g., timetable\_with\_constraints.xlsx) for easy access.

**(iv) Web Integration:** The system is integrated into a Django web application, providing a user-friendly interface at the local host. The frontend, built with React and Tailwind CSS, displays timetables with time slots in the first column and days (Monday to Friday) as headers, showing subjects and venues without teacher names. A dropdown allows users to select scheduling modes: "all together" for consecutive lectures or "distributed among days" for spread-out scheduling. The backend handles input processing and timetable generation, ensuring seamless functionality.

**RESULT**

The timetable generator successfully produces conflict-free schedules that adhere to all specified constraints, including fixed lectures and availability. The Excel output organizes timetables by year group, with clear formatting for time slots, days, subjects, and venues. The web interface displays timetables accurately, with the updated format and scheduling mode options working as intended. Testing confirmed that the "distributed among days" mode avoids back-to-back lectures, and the system handles inputs for multiple year groups efficiently.

**LIMITATIONS**

**(i) Input Dependency:** The system relies on correctly formatted Excel inputs. Errors in input files, such as mismatched teacher or venue availability, can lead to generation failures or incorrect schedules.

**(ii) Scalability:** While effective for small to medium-sized institutions, the solver’s performance may degrade with very large datasets (e.g., hundreds of teachers or venues) due to increased computational complexity.

**CONCLUSION**

The timetable generator effectively automates class scheduling by leveraging OR-Tools and a Django web application. It addresses key challenges in manual scheduling, offering a scalable and user-friendly solution. Future improvements could focus on enhancing input validation and optimizing the solver for larger datasets to broaden its applicability in diverse educational settings.