### Important Files for the Timetable Generator Project

Here are the essential files you need to continue working on this project, along with their roles and relationships:

1. \*\*`app.py`\*\* - The main Flask application

- Entry point for the web application

- Handles routing and workflow:

- `/` : Upload page

- `/download\_template` : Template download

- `/generate` : Timetable generation

- Coordinates all other components

2. \*\*`scheduler.py`\*\* - Core scheduling algorithm

- Contains the `TimetableScheduler` class

- Implements intelligent scheduling:

- L-T-P interpretation

- Session-based scheduling

- Constraint management

- Conflict avoidance

- Key method: `generate\_schedule()`

3. \*\*`validator.py`\*\* - Data validation

- Validates input data before scheduling

- Checks:

- Faculty qualifications

- Slot compatibility

- Faculty workload

- L-T-P format validation

- Prevents invalid schedules

4. \*\*`visualizer.py`\*\* - Output generation

- Converts schedule to Excel format

- Handles:

- Main timetable view

- Faculty workload summary

- Session-based output

- Key function: `generate\_excel\_bytes()`

5. \*\*`data\_loader.py`\*\* - Data handling

- Loads and processes input Excel files

- Handles:

- Courses data loading

- Faculty data loading

- Data type conversions

- Ensures proper data formatting

6. \*\*`templates/upload.html`\*\* - User interface

- File upload form

- Template download button

- Simple, clean interface

7. \*\*`templates/error.html`\*\* - Error handling

- Displays validation errors

- Shows processing errors

- Provides "Try Again" option

8. \*\*`data/slot\_master.xlsx`\*\* - Slot definitions (not code)

- Contains:

- Slot definitions (time, type, duration)

- Slot groups (if applicable)

- Critical for automatic slot assignment

9. \*\*`requirements.txt`\*\* - Dependencies

- Lists all required Python packages

- Ensures consistent environment setup

### Project Documentation

\*\*Project Name:\*\* Institute Timetable Generator

\*\*Goal:\*\* Automatically generate optimal timetables based on course requirements and constraints

\*\*Key Features:\*\*

1. \*\*Automatic Slot Assignment:\*\* Intelligently assigns time slots based on course type and requirements

2. \*\*L-T-P Interpretation:\*\* Parses lecture-tutorial-practical requirements

3. \*\*Constraint Management:\*\* Enforces faculty availability, room constraints, and session limits

4. \*\*Conflict Prevention:\*\* Prevents scheduling conflicts between courses

5. \*\*Balanced Workload:\*\* Distributes sessions evenly across the week

\*\*Workflow:\*\*

1. User downloads template Excel file

2. User fills in course and faculty data

3. User uploads filled template

4. System validates input data

5. System generates optimal timetable

6. User downloads generated timetable as Excel

\*\*Key Components Relationship Diagram:\*\*

```

[User]

│

▼

[app.py] ──┬─► [data\_loader.py] (loads data)

├─► [validator.py] (validates input)

├─► [scheduler.py] (generates schedule)

└─► [visualizer.py] (creates output)

│

▼

[Browser] ◄──┐ (upload.html)

└── (error.html)

```

\*\*Data Flow:\*\*

1. User upload → `app.py`

2. `app.py` → `data\_loader.py` (load courses/faculty)

3. `app.py` → `validator.py` (validate data)

4. `app.py` → `scheduler.py` (generate timetable)

5. `app.py` → `visualizer.py` (create Excel output)

6. `app.py` → User (download timetable)

\*\*Setup Instructions:\*\*

1. Create virtual environment:

```bash

python -m venv venv

source venv/bin/activate # Linux/Mac

venv\Scripts\activate # Windows

```

2. Install dependencies:

```bash

pip install -r requirements.txt

```

3. Create directory structure:

```

project-root/

├── data/

│ └── slot\_master.xlsx

├── templates/

│ ├── upload.html

│ └── error.html

├── app.py

├── data\_loader.py

├── scheduler.py

├── validator.py

├── visualizer.py

└── requirements.txt

```

4. Run the application:

```bash

flask run

```

\*\*Key Classes/Functions:\*\*

1. \*\*`TimetableScheduler` (scheduler.py)\*\*

- `parse\_ltp(ltp\_str)`: Converts L-T-P string to components

- `recommend\_slots()`: Suggests slots based on course type

- `find\_compatible\_slot()`: Finds available slots

- `schedule\_course()`: Schedules all sessions for a course

- `generate\_schedule()`: Main scheduling method

2. \*\*`generate\_excel\_bytes()` (visualizer.py)\*\*

- Converts schedule dictionary to Excel bytes

- Creates two sheets: Schedule and Faculty Workload

\*\*Future Enhancement Ideas:\*\*

1. Add room capacity constraints

2. Implement student batch-specific views

3. Add conflict resolution algorithms

4. Include holiday/break support

5. Add calendar integration

6. Implement multi-department scheduling

7. Add historical schedule comparison

\*\*Troubleshooting Tips:\*\*

1. Ensure `slot\_master.xlsx` has all required slots

2. Verify L-T-P format in input data (e.g., "3-0-0")

3. Check faculty qualifications match course codes

4. Ensure no faculty exceeds MaxHours

5. Validate room preferences for lab courses

This documentation provides all necessary information to continue the project with another AI or team member. The system is designed to be modular, with each component handling a specific responsibility in the timetable generation workflow.