Dynamic Scheduler Agent

A intelligent time management system designed to optimize a user's daily schedule by integrating with Google Calendar, Google Tasks, and Gmail. The system aims to maximize high-leverage outputs while protecting time for deep thinking, leadership, and personal wellbeing.

Features

- **Energy-Based Scheduling**: Tasks are aligned with the user's natural energy patterns throughout the day
- **Strategic Prioritization**: Activities are prioritized based on their importance and urgency
- **Protected Time Blocks**: Critical time for deep work, personal wellbeing, and family is preserved
- **Goal Alignment**: Daily activities are connected to strategic goals
- **Adaptive Rescheduling**: Meetings without clear outcomes are candidates for rescheduling
- **Morning Brief**: Daily email with schedule overview, key metrics, critical tasks, and meeting intelligence

System Architecture

The Dynamic Scheduler Agent consists of the following components:

1. **Google API Integration**: Connects to Google Calendar, Tasks, and Gmail
2. **Prioritization Engine**: Scores tasks and meetings based on importance, urgency, energy alignment, and goal alignment
3. **Schedule Optimizer**: Transforms prioritized items into an optimal daily schedule
4. **Morning Brief Generator**: Creates a comprehensive daily brief with schedule and insights
5. **Protected Time Manager**: Ensures essential blocks of time are preserved

Setup Instructions

Prerequisites

- Python 3.8 or higher
- Google account with Calendar, Tasks, and Gmail
- Google Cloud project with API access

Installation

1. Clone the repository:

```
git clone https://github.com/yourusername/dynamic-scheduler.git
```

```
cd dynamic-scheduler
```

2. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

3. Set up Google API credentials:
   - Go to the Google Cloud Console
   - Create a new project or select an existing one
   - Enable the Google Calendar API, Google Tasks API, and Gmail API
   - Create OAuth 2.0 credentials
   - Download the credentials JSON file and save it as `credentials.json` in the project directory

Configuration

Edit the `config.py` file to customize your preferences:

- Energy patterns throughout the day
- Work location preferences
- Protected time blocks
- Strategic goals
- Meeting preferences
- Email settings

Usage

Running the Scheduler

To generate a schedule for tomorrow:

```
python scheduler.py
```

To generate a schedule for a specific number of days ahead:

```
python scheduler.py --days 2
```

To generate a schedule without sending the morning brief:

```
python scheduler.py --no-brief
```

Morning Brief

The morning brief is delivered at 6 AM daily and includes:

1. **Today's Schedule Overview**: Timeline visualization of the day
2. **Key Metrics**: Deep work time allocation, progress toward North Star goal, balance score
3. **Critical Tasks**: Top 3 most important tasks for the day
4. **Meeting Intelligence**: Preparation notes, reschedule candidates, decision points

5. **Recent Context**: Important emails, follow-ups, upcoming deadlines

Components

scheduler.py

Main script that orchestrates the scheduling process by integrating all components.

google_api.py

Handles authentication and interactions with Google Calendar, Tasks, and Gmail.

prioritization.py

Implements the prioritization algorithm to score tasks and meetings.

schedule_optimizer.py

Transforms prioritized tasks and meetings into an optimal daily schedule.

morning_brief.py

Generates the morning brief with schedule overview and key information.

config.py

Contains configuration settings and user preferences.

License

Acknowledgements

- Google API Client Libraries
- Python community for excellent libraries and tools