**SkyWise**

**📌 Project Title**

**SkyWise Telescope Scheduler**

**📝 Project Description**

**Astronomical observations are limited by weather, moonlight, and telescope time. Professional observatories often rely on advanced schedulers to maximize the quality of data collected in a single night. This project aims to design a lightweight, automated telescope scheduling system that uses real-time weather forecasts, lunar constraints, and astronomical calculations to generate an optimized observing plan for a given site and night.**

**The scheduler ranks targets based on visibility, conditions, and scientific priority, then assigns each target to time slots using a greedy optimization approach. The result is a timeline (CSV + visualization) showing which objects should be observed hour by hour to maximize data quality.**

**This tool can serve amateur astronomers, student observatories, or as a demonstration of optimization and applied astronomy software engineering.**

**🎯 Objectives**

1. **Automate nightly telescope scheduling using weather and astronomical conditions.**
2. **Maximize observing efficiency (cloud-free, high-altitude, low-airmass windows).**
3. **Apply lunar constraints (avoid observations too close to the Moon).**
4. **Prioritize targets based on scientific importance (user-defined).**
5. **Generate human-friendly outputs (CSV table + Gantt chart).**

**📂 Inputs**

1. **Site configuration (lat, lon, elevation, timezone).**
2. **Observation date (single night).**
3. **Targets list (CSV):**
   * **Name**
   * **RA (Right Ascension, degrees)**
   * **Dec (Declination, degrees)**
   * **Priority (1–5 scale)**
4. **Weather data: hourly cloud cover (0–1), optional wind speed (CSV or API).**

**📤 Outputs**

1. **Schedule (CSV): Each row = time slot, target, altitude, airmass, moon separation, weather conditions, score.**
2. **Visualization (Matplotlib): Gantt chart showing targets vs time, with cloud cover background.**
3. **Summary report (TXT/Markdown):**
   * **Total hours per target**
   * **Average airmass**
   * **Fraction of “good hours” used**

**⚙️ Methodology**

**1. Astronomical Computations (Astropy/Astroplan)**

* **Calculate Sun altitude → define astronomical night (sun\_alt < –18°).**
* **Compute each target’s altitude, azimuth, airmass at each time slot.**
* **Compute Moon altitude & separation for lunar constraint.**

**2. Weather Integration**

* **Read hourly forecast (cloud fraction 0–1).**
* **Optionally filter out slots with high winds or high clouds.**

**3. Scoring Function**

**For each target *i* at time slot *t*:**

**score(i,t) = w\_clear \* (1 - cloud\_fraction)**

**+ w\_alt \* max(0, sin(altitude\_rad))**

**+ w\_air \* (1 / airmass)**

**+ w\_moon \* min(1, moon\_sep / 90°)**

**+ w\_prio \* (priority / 5)**

* **Skip slot if: sun\_alt > –18°, target\_alt < horizon, moon\_sep < limit.**
* **Default weights: w\_clear=4, w\_alt=3, w\_air=2, w\_moon=2, w\_prio=1.**

**4. Greedy Scheduling Algorithm**

* **Iterate through available slots.**
* **At each slot, select the highest-scoring target not yet fully observed.**
* **Mark the slot as “taken” and continue until night ends.**

**5. Output & Visualization**

* **Save results in schedule.csv.**
* **Plot timeline chart with:**
  + **X-axis = time (hours of night).**
  + **Y-axis = targets.**
  + **Colored bars = scheduled observations.**
  + **Overlay cloud cover curve.**

**🛠️ Tech Stack**

* **Language: Python**
* **Libraries:**
  + **astropy + astroplan → astronomical calculations**
  + **pandas → data handling**
  + **matplotlib → visualization**
  + **pyyaml → config management**
  + **requests → weather API (optional)**

**📊 Example Workflow**

1. **User provides targets.csv (e.g., M31, M13, M57).**
2. **System pulls weather for Toronto on 2025-09-22.**
3. **Night is from 20:00–05:00 local time.**
4. **Scheduler computes scores for each target per hour.**
5. **Final schedule:**
   * **21:00–23:00 → M31 (high altitude, clear skies)**
   * **23:00–01:00 → NGC7000 (Moon far away, good conditions)**
   * **01:00–03:00 → M13 (priority 3, acceptable airmass)**
6. **Output: schedule.csv + Gantt chart.**

**📈 Resume Value**

**This project demonstrates:**

* **Real-world problem solving (resource optimization under constraints).**
* **Data integration (weather + astronomy).**
* **Algorithm design (scoring + greedy optimization).**
* **Professional output (CSV reports + visualizations).**

**📆 4-Week Plan (2h/day → ~10–12h/week)**

**Week 1 – Setup & Foundations**

**🎯 Goal: Get the project repo ready, load configs & targets, compute basic astronomy times.**

**Day 1–2**

* **Create project folder + GitHub repo.**
* **Make README.md with project name + one-line description.**
* **Install Python + libs: astropy, astroplan, pandas, matplotlib, pyyaml.**
* **Write config.yaml and targets.csv with 3–5 sample targets.**

**Day 3–4**

* **Use astroplan.Observer for your site (lat/lon/elevation).**
* **Compute sunset, sunrise, astronomical twilight for one date.**
* **Print: "Night runs from 20:11 → 05:42 local time".**

**Day 5–6**

* **Build hourly time grid for the night.**
* **For each target & time → compute altitude (degrees).**
* **Print simple table: hour, target, alt.**

**Deliverable by end of week:**

* **Repo with configs, targets, and a script that prints when night starts + target altitudes over time.**

**Week 2 – Astronomy Calculations**

**🎯 Goal: Add all astronomy features (airmass, moon separation, constraints).**

**Day 1–2**

* **Compute airmass from altitude (astroplan has helper functions).**
* **Add check: skip times when target altitude < horizon.**

**Day 3–4**

* **Compute moon altitude + separation from each target.**
* **Add constraint: skip if moon separation < 20°.**

**Day 5–6**

* **Implement scoring function:**
  + **(1 – cloud fraction)**
    - **sin(alt)**
    - **1/airmass**
    - **moon separation factor**
    - **priority weight.**
* **Use mock weather file (CSV with random 0–1 cloud fractions).**

**Deliverable by end of week:**

* **Script that prints a scoring table: each target × each hour with alt, airmass, moon\_sep, cloud, and final score.**

**Week 3 – Scheduling Logic**

**🎯 Goal: Build the actual scheduler (greedy algorithm) + CSV export.**

**Day 1–2**

* **Write greedy loop:**
  + **For each time slot, pick the highest-score target available.**
  + **Respect exposure length (block N slots if needed).**

**Day 3–4**

* **Export results to schedule.csv with columns:**
  + **time\_start, time\_end, target, score, alt, airmass, moon\_sep, clouds.**

**Day 5–6**

* **Add summary stats:**
  + **Total hours per target.**
  + **Average airmass.**
  + **Percentage of “good hours” scheduled.**

**Deliverable by end of week:**

* **schedule.csv and summary.txt for one night’s run.**

**Week 4 – Visualization & Polish**

**🎯 Goal: Make the project look professional with plots, documentation, and tests.**

**Day 1–2**

* **Plot Gantt chart (target vs time, colored bars).**
* **Overlay cloud fraction curve on the same plot.**

**Day 3–4**

* **Run on 2–3 different nights with different weather files.**
* **Save results in examples/.**
* **Compare fixed schedule vs your optimized one (show ~20–30% gain).**

**Day 5–6**

* **Write a README.md with:**
  + **Project description + screenshots.**
  + **How to run (python run.py --date 2025-09-22).**
  + **Example schedule + chart.**
* **Push final repo to GitHub.**

**Deliverable by end of week:**

* **Polished repo with code, data, README, and plots.**
* **Final demo: one command generates schedule + chart + summary.**

**✅ End Result**

**By the end of 4 weeks you’ll have:**

* **A working telescope scheduler that uses weather + astronomy data.**
* **CSV schedules, plots, and reports that look professional.**
* **A GitHub repo ready to showcase to employers.**

**Perfect 🙌 here’s a day-by-day checklist (28 tasks) — you can literally tick each box as you go. Each item is sized for ~2 hours of focused work.**

**📆 4-Week / 28-Day Checklist**

**Week 1 – Setup & Foundations**

**✅ Day 1: Create project folder + GitHub repo, add empty README.md.  
✅ Day 2: Install Python libs (astropy, astroplan, pandas, matplotlib, pyyaml), test imports.  
✅ Day 3: Write config.yaml with site info (lat, lon, elevation).  
✅ Day 4: Create targets.csv with 3–5 sample targets (RA, Dec, priority).  
✅ Day 5: Use astroplan.Observer to compute sunset/sunrise, print night interval.  
✅ Day 6: Generate hourly time grid (20:00–05:00), print to console.  
✅ Day 7: For each target, compute altitude over the night, output table.**

**Week 2 – Astronomy Calculations**

**✅ Day 8: Compute airmass from altitude (skip if below horizon).  
✅ Day 9: Format output: time, target, alt, airmass.  
✅ Day 10: Compute Moon position (altitude + RA/Dec) using astropy.  
✅ Day 11: Calculate moon separation from each target.  
✅ Day 12: Add rule: skip slots if moon separation < 20°.  
✅ Day 13: Create mock weather.csv with hourly cloud fractions.  
✅ Day 14: Implement scoring function (clouds, altitude, airmass, moon, priority).**

**Week 3 – Scheduling Logic**

**✅ Day 15: Implement greedy scheduler: pick highest score each hour.  
✅ Day 16: Add support for exposures (block multiple hours if needed).  
✅ Day 17: Write schedule results to schedule.csv.  
✅ Day 18: Add columns to schedule: alt, airmass, moon\_sep, clouds, score.  
✅ Day 19: Compute summary stats (total hours per target, avg airmass).  
✅ Day 20: Save stats to summary.txt.  
✅ Day 21: Test with 2–3 different dates & targets.**

**Week 4 – Visualization & Polish**

**✅ Day 22: Implement Gantt chart (targets vs time, colored bars).  
✅ Day 23: Overlay cloud cover curve on chart.  
✅ Day 24: Add plots to examples/ folder.  
✅ Day 25: Compare fixed schedule vs optimized one (show improvement).  
✅ Day 26: Write full README.md: description, features, usage, screenshots.  
✅ Day 27: Clean code (split into modules: astronomy.py, scheduler.py, plot.py).  
✅ Day 28: Final test run, commit & push repo → 🎉 project complete!**

**🏆 Final Deliverables**

* **schedule.csv → Final observing plan.**
* **summary.txt → Nightly stats.**
* **plots/ → Gantt chart & cloud cover.**
* **README.md → Project overview + screenshots.**
* **GitHub repo → Ready to show employers.**