NASA API Internship - Excel Reporting Guide

Table of Contents

Overview

This document provides detailed instructions for all 50 students participating in the NASA API Implementation internship. You will be using eight interconnected tracking sheets to document your progress throughout the 30-day program. Accurate tracking is essential for:

- Documenting your personal growth and contributions
- Enabling instructors to provide targeted guidance
- Creating a portfolio of your work for future job interviews
- Learning professional project tracking practices

General Guidelines

1. **Update Frequency**:
    - **Daily Task Tracking (Sheet 1)**: Update twice daily (start and end of class)
    - **Code Contributions (Sheet 5)**: Update after each commit/PR
    - **Technical Skill Development (Sheet 7)**: Update weekly
    - **Code Quality & Testing (Sheet 6)**: Update after code reviews
    - **Sprint Retrospective (Sheet 8)**: Update at the end of each sprint
    - **Other sheets**: Updated primarily by instructors
2. **Accuracy**: Always provide accurate information. The goal is to track real progress, not to inflate numbers.

3. **Specificity**: Be detailed in your descriptions and comments.
4. **Questions**: If unsure about how to record something, ask your instructor during class time.

Sheet 1: Daily Task Tracking

Purpose: Track your daily activities and progress.

Update Frequency: Twice daily (beginning and end of class)

Required Fields:

1. **Date**: Current date (MM/DD/YYYY format)
2. **Name**: Your full name
3. **Task Assigned**: The task assigned to you by your instructor (from GitHub issues)
4. **Status**:
    - "To Do" (not started)
    - "In Progress" (currently working on)
    - "Done" (completed)
5. **Comments**: Notes about progress, challenges, etc.
6. **Story Points**: Filled in when task is complete (ask instructor for point value)
7. **Pull Requests**: PR number if submitted (e.g., "PR-007")
8. **Commits**: Number of commits made that day
9. **Hours Logged**: Time spent on the task that day

Example Morning Update:

```
Date: 04/23/2025 | Name: Priya Patel | Task Assigned: User
Registration Form | Status: In Progress |
Comments: Starting work on validation logic | Story Points: - | Pull
Requests: - | Commits: - | Hours Logged: -
```

Example Evening Update:

```
Date: 04/23/2025 | Name: Priya Patel | Task Assigned: User
Registration Form | Status: In Progress |
Comments: Completed validation, working on error handling | Story
Points: - | Pull Requests: - | Commits: 2 | Hours Logged: 3
```

Sheet 2: Sprint Planning & Capacity

Purpose: Plan sprint goals and allocate team capacity.

Primary Updater: Instructors (students will have view access)

What you need to know: Each sprint is 7 days long with a specific goal:

- **Sprint 1 (Days 1-7)**: Project setup and architecture
- **Sprint 2 (Days 8-14)**: APOD feature implementation
- **Sprint 3 (Days 15-21)**: Mars Rover feature implementation
- **Sprint 4 (Days 22-28)**: Testing, documentation, and refinement

Your individual capacity is defined here, showing how your time should be allocated between:

- Development work
- Code review
- Learning
- Meetings

Review this sheet at the beginning of each sprint to understand expectations.

Sheet 3: User Stories & Features

Purpose: Track high-level user stories and their status.

Primary Updater: Instructors (students will have view access)

What you need to know: This sheet contains the user stories you'll be working on. Each story has:

- A unique ID (e.g., US-001)
- Description in user-centric format
- Acceptance criteria
- Story points (measure of complexity)
- Priority level
- Assigned developer(s)
- Current status

Reference this sheet to understand how your individual tasks relate to larger features.

Sheet 4: Task Breakdown

Purpose: Break down user stories into specific, actionable tasks.

Primary Updater: Instructors with student input

What you need to know: This sheet breaks down user stories into specific tasks:

- Each task has a unique ID (e.g., T-001)
- Tasks are linked to parent user stories
- Each task has estimated hours and complexity rating
- Tasks will be assigned to specific students
- Dependencies between tasks are documented here

Review this sheet daily to understand your task details, including:

- Technical requirements
- Estimated time
- Complexity level
- Dependencies (what needs to be completed first)

Sheet 5: Code Contributions

Purpose: Track your actual code contributions.

Update Frequency: After each commit or pull request

Required Fields:

1. **Date**: When code was committed
2. **Developer**: Your name
3. **Feature/Task**: What you worked on (reference Task ID if possible)
4. **Files Changed**: Number of files modified
5. **Lines Added**: Lines of code added
6. **Lines Deleted**: Lines of code removed
7. **Commit Hash**: Git commit identifier (e.g., "8f4e96d")
8. **PR Number**: Pull request number if applicable (e.g., "PR-007")
9. **PR Status**: Current status ("Pending", "In Review", "Changes Requested", "Approved", "Merged")
10. **Code Review Feedback**: Summary of feedback received

How to Calculate Code Metrics:

See the Appendix: Calculating Code Metrics for detailed instructions.

Example Entry:

```
Date: 04/24/2025 | Developer: Ahmed Khan | Feature/Task: T-004 Login
Form UI | Files Changed: 3 |
Lines Added: 126 | Lines Deleted: 7 | Commit Hash: a1b2c3d | PR
Number: PR-005 |
PR Status: In Review | Code Review Feedback: Need to add form
validation
```

Sheet 6: Code Quality & Testing

Purpose: Track code quality metrics and testing efforts.

Update Frequency: After code reviews and when adding tests

Required Fields:

1. **Date**: Date of review or testing
2. **Developer**: Your name
3. **Component**: Specific component or class being reviewed/tested
4. **Unit Tests**: Number of unit tests written
5. **Coverage %**: Test coverage percentage (see below for calculation)
6. **Code Smells**: Number of code quality issues identified
7. **Performance Issues**: Number of performance concerns
8. **Security Issues**: Number of security vulnerabilities
9. **Reviewer**: Who reviewed the code
10. **Review Score**: Rating from 1-5 (5 being excellent)

Calculating Test Coverage:

1. In Eclipse, right-click on your project
2. Select "Coverage As" → "JUnit Test"
3. View the coverage report
4. Record the percentage shown

Example Entry:

```
Date: 04/25/2025 | Developer: Priya Patel | Component: UserService |
Unit Tests: 8 |
Coverage %: 78% | Code Smells: 2 | Performance Issues: 0 | Security
Issues: 0 |

Reviewer: Rohit Sharma | Review Score: 4
```

Sheet 7: Technical Skill Development

Purpose: Track your growth in technical skills.

Update Frequency: Weekly (end of day Friday)

Required Fields:

1. **Developer**: Your name
2. **Skill Area**: Specific technology or concept (e.g., "Spring Security", "REST APIs")
3. **Starting Level (1-5)**: Your proficiency at the beginning of the internship
4. **Current Level**: Your current proficiency (updated weekly)
5. **Progress**: Will auto-calculate based on current vs. starting level
6. **Learning Resources**: What you used to improve (documentation, tutorials, etc.)
7. **Applied In**: Which tasks you applied this skill to (reference Task IDs)
8. **Mentor Feedback**: Comments from your mentor about your progress

Proficiency Scale:

- **1**: Basic awareness, need significant guidance
- **2**: Fundamental understanding, need regular guidance
- **3**: Practical application, occasional guidance needed
- **4**: Deep understanding, can work independently
- **5**: Mastery, can teach others

Example Entry:

```
Developer: Rohit Sharma | Skill Area: Spring Security | Starting
Level: 2 | Current Level: 3 |
Progress: +1 | Learning Resources: Spring Security Documentation |
Applied In: T-006, T-007 | Mentor Feedback: Good progress on
authentication flow
```

Sheet 8: Sprint Retrospective

Purpose: Reflect on each sprint and plan improvements.

Update Frequency: At the end of each sprint (Days 7, 14, 21, 28)

Required Fields:

1. **Sprint**: Sprint number (1-4)
2. **Developer**: Your name
3. **What Went Well**: List achievements and successes
4. **What Could Be Improved**: Areas where you faced challenges
5. **Action Items**: Specific improvements to make in the next sprint
6. **Story Points Committed**: Points you committed to at sprint start
7. **Story Points Completed**: Points actually completed
8. **Velocity**: Will auto-calculate (Completed/Committed percentage)

Example Entry:

```
Sprint: 1 | Developer: Ahmed Khan | What Went Well: Completed UI
components on schedule |
What Could Be Improved: Need better understanding of Thymeleaf |
Action Items: Review Thymeleaf documentation | Story Points
Committed: 8 |

Story Points Completed: 5 | Velocity: 63%
```

Appendix: Calculating Code Metrics

Using Git Commands

To calculate metrics for your latest commit:

bash
```bash
# View statistics for your last commit
git show --stat

# Count files changed in last commit
git show --name-only HEAD | wc -l

# Count lines added/deleted in last commit

git show --stat HEAD
```

The output will look something like:

```
src/main/java/com/openapi/nasa/controller/LoginController.java | 45
+++++++++++++++++++++++++++++++++++++++++++--
 src/main/resources/templates/login.html                       | 78
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++

 2 files changed, 121 insertions(+), 2 deletions(-)
```

In this example:

- Files changed: 2
- Lines added: 121
- Lines deleted: 2

Using GitHub Web Interface

1. Go to your Pull Request on GitHub

2. Click on the "Files changed" tab
3. At the top, you'll see "Showing X changed files with Y additions and Z deletions"
4. Record these numbers in your tracking sheet

Eclipse Git Integration

1. Right-click on your project → Team → Show in History
2. Select your commit
3. The bottom pane will show files changed and line changes

Remember to update Sheet 5 promptly after making commits or pull requests to ensure accurate tracking of your contributions.