IT Management & Audits

Practical Lab Manual

Agile Sprint Simulator

Practical P05

Learning Domain

IT Project Management & Agile Methodology

Course Learning Outcomes

CLO05: Apply agile project management techniques to IT projects

Unit

Unit IV: IT Project Management

Time Allocation: 3 hours

Learning Mode: Hands-on (70%) + Theory (30%)

Difficulty Level: Intermediate

Agile Sprint Simulator

Practical P05

Quick Reference

Practical Code	P05
Practical Name	Agile Sprint Simulator
Slot	T/P-5
Duration	3 hours
CLO Mapping	CLO05
Unit	Unit IV: IT Project Management
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	MEDIUM
Screenshot Count	5 Required

Prerequisites

- Basic understanding of project management concepts (waterfall vs. agile)
- Familiarity with Python programming (variables, functions, loops)
- Command-line interface (CLI) proficiency
- Understanding of JSON data format
- Text editor or IDE installed (VS Code recommended)
- Python 3.8 or later installed on your system

Tools Required

Tool	Version	Free	Notes
Python	3.8+	✓	Core runtime
pip	Latest	✓	Python package manager
Click	8.x	✓	CLI framework
Rich	13.x	✓	Terminal formatting
matplotlib	3.x	✓	Chart generation
Git	Latest	✓	Version control
Web Browser	Latest	✓	Viewing HTML reports

Learning Objectives

- ✓ Understand the Agile Scrum framework and its application to IT project management
- ✓ Create and manage a product backlog with prioritized user stories
- ✓ Plan sprints using velocity-based capacity planning
- ✓ Simulate daily standups with progress tracking and blocker management
- ✓ Interpret burndown charts to assess sprint progress
- ✓ Use Kanban boards for visual workflow management
- ✓ Conduct sprint retrospectives and generate actionable insights
- ✓ Apply agile practices in the context of Indian IT industry

What You Will Learn

By the end of this practical, you will: (1) understand the Scrum framework (roles, events, artifacts); (2) write user stories with acceptance criteria and estimate story points; (3) plan sprints using velocity-based capacity planning; (4) track progress through standups and Kanban boards; (5) generate and interpret burndown charts; (6) conduct retrospectives for continuous improvement; (7) generate professional sprint reports.

Real-World Application

Agile powers India's IT sector. Companies like **TCS**, **Infosys**, and **Wipro** have adopted Agile at scale. FinTech startups such as **Razorpay** and **PhonePe** use Scrum for bi-weekly payment feature releases. This practical gives you hands-on experience with the same techniques used by India's leading technology organizations.

Hands-On Procedure

Part A: Environment Setup

Step 1: Clone Repository and Explore Project Structure

Objective: Obtain the Agile Sprint Simulator source code and understand the project layout.
Instructions:

1. Open your terminal (Command Prompt, PowerShell, or bash)
2. Navigate to your preferred working directory
3. Clone the repository and explore its structure

Commands:

```
# Clone the repository
git clone https://github.com/it-mgmt-labs/agile-sprint-simulator.
    git
cd agile-sprint-simulator

# View project structure
find . -type f -name "*.py" -o -name "*.json" | sort
```

Clone and Explore Repository

Expected Output

```
agile-sprint-simulator/
|- src/
|   |- models.py, backlog.py, sprint_planner.py
|   |- daily_standup.py, burndown.py, kanban.py
|   |- retrospective.py, reporter.py, simulator.py
|   |- cli.py
|- data/
|   |- sample_backlog.json, team.json
|- requirements.txt, README.md
```

Read `src/models.py` first — understanding the data models (Story, Sprint, TeamMember) makes the rest of the codebase easier to follow.

Step 2: Create Virtual Environment and Install Dependencies

Objective: Set up an isolated Python environment and install all required packages.
Instructions:

1. Create a Python virtual environment
2. Activate the virtual environment
3. Install project dependencies from `requirements.txt`
4. Verify the CLI is working by running the help command

Commands:

```
# Create and activate virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# venv\Scripts\activate   # Windows

# Install dependencies and verify
pip install -r requirements.txt
python src/cli.py --help
```

Virtual Environment Setup

Expected Output

```
Usage: cli.py [OPTIONS] COMMAND [ARGS]...
Agile Sprint Simulator - Interactive Scrum Management Tool
Commands:
init    Initialize a new project with backlog and team
plan    Run sprint planning session
run     Execute sprint simulation with daily standups
retro   Conduct sprint retrospective
report  Generate sprint report (HTML/PDF)
```

Ensure Python 3.8+. Check with `python -version`. Some systems require `python3`. Rich needs 3.7+ and matplotlib needs 3.8+.

Part B: Sprint Planning

Step 3: Initialize Project with Sample Backlog

Objective: Load the product backlog and team configuration to initialize a new Scrum project.
Instructions:

1. Initialize the project using the sample backlog and team data files
2. Review the loaded product backlog (20 user stories for a mobile banking app)
3. Examine the team composition (5 members with roles and capacity)
4. Understand story point estimates and priority levels

Commands:

```
python src/cli.py init \
  --backlog data/sample_backlog.json \
  --team data/team.json
```

Initialize Agile Project

Expected Output

```
=== Project Initialization ===
Project:  Mobile Banking App
-- Product Backlog (20 stories) --
US01 | HIGH   | 8 pts  | User login with biometric
US02 | HIGH   | 5 pts  | Fund transfer (NEFT/IMPS)
US03 | HIGH   | 8 pts  | UPI payment integration
US04 | MEDIUM | 3 pts  | Account balance view
US05 | MEDIUM | 5 pts  | Transaction history
US06 | HIGH   | 13 pts | Bill payment module
...  (20 stories total, 134 points)
-- Team (5 members) --
Priya (Scrum Master, 6h), Arjun (Dev, 8h),
Sneha (Dev, 8h), Rahul (Tester, 7h),
Meera (UI/UX, 6h) | Capacity:  35 hrs/day
Project initialized successfully!
```

The backlog uses India-relevant features: UPI payments, NEFT/IMPS transfers, and Aadhaar-based KYC — critical components in Indian digital banking.

Step 4: Run Sprint Planning Session

Objective: Plan a sprint by selecting user stories based on priority and team velocity.
Instructions:

1. Run the sprint planning command with a target velocity of 40 story points
2. Observe how the planner selects stories by priority (HIGH first, then MEDIUM)
3. Review task assignments to team members based on their roles and capacity
4. Examine the resulting sprint backlog

Commands:

```
python src/cli.py plan --velocity 40
```

Sprint Planning

Expected Output

```
=== Sprint Planning (Sprint 1, 2-week) ===
Target Velocity:  40 pts
[SELECTED] US01:  Biometric login (8) | US02:  NEFT/IMPS (5)
[SELECTED] US03:  UPI payment (8)     | US06:  Bill pay (13)
[SELECTED] US04:  Balance view (3)    | US05:  Tx history (5)
[SKIPPED]  US07:  Notifications - velocity limit
Sprint Backlog:  6 stories | 42 points committed
Assignments:  Arjun->US01,US03 | Sneha->US02,US06
Meera->US04,US05 | Rahul->Testing | Priya->Facilitation
```

Velocity stabilizes after 3–4 sprints. The planner over-commits slightly (42 vs. 40 target) because US04 fits remaining capacity — common in real sprint planning.

Part C: Sprint Execution

Step 5: Simulate Sprint Execution with Daily Standups

Objective: Run the sprint simulation with interactive daily standups and observe task movement across the Kanban board.
Instructions:

1. Execute the sprint simulation using the `run` command
2. Observe daily progress updates for each team member
3. Note random blockers that appear during the sprint (simulating real-world impediments)
4. Watch tasks move across the Kanban board columns: To Do, In Progress, Done
5. Pay attention to how blockers affect sprint velocity

Commands:

```
python src/cli.py run
```

Sprint Execution

Expected Output

```
=== Sprint 1 Execution ===
-- Day 1 Standup --
Arjun:  Started US01 | Today:  Auth module | Blockers:  None
Sneha:  Started US02 | Today:  API layer   | Blockers:  None
...
-- Kanban Board (Day 1) --
+--------+-------+---+
| TO DO         | IN PROGRESS  | DONE |
| US03, US04    | US01 (8 pts) |      |
| US05, US06    | US02 (5 pts) |      |
+--------+-------+---+
-- Day 4:  BLOCKER --
Arjun:  US01 75% - Biometric SDK integration issue
Priya:  Resolving blocker for Arjun
```

The Kanban board uses the Rich library for colorful terminal rendering. Real teams often add columns like "In Review" or "Ready for Testing" for more granular visibility.

Step 6: View Burndown Chart

Objective: Generate and analyze a burndown chart showing sprint progress against the ideal trajectory.
Instructions:

1. Run the sprint simulation with the burndown chart flag enabled
2. A matplotlib window will open displaying the burndown chart
3. Compare the ideal burndown line (straight diagonal) with the actual burndown
4. Identify any periods where progress stalled (flat sections indicate blockers)
5. Save the chart image for your report

Commands:

```
python src/cli.py run --show-burndown
# Chart saved automatically to output/burndown.png
```

Generate Burndown Chart

Expected Output

A matplotlib chart window opens showing "Sprint 1 Burndown Chart" with ideal line (blue, dashed: 42 pts to 0) and actual line (red, solid: deviates around Day 3–4 due to blockers). Chart is saved to `output/burndown.png`.

On headless environments, set `export MPLBACKEND=Agg` before running. The chart will still save to file without interactive display.

Part D: Review and Retrospective

Step 7: Run Sprint Retrospective

Objective: Conduct a sprint retrospective to analyze sprint performance and identify areas for improvement.
Instructions:

1. Run the retrospective command to analyze the completed sprint
2. Review key metrics: velocity achieved, stories completed, carry-over items
3. Examine team performance statistics per member
4. Review the auto-generated "What went well" and "What to improve" categories
5. Note the action items suggested for the next sprint

Commands:

```
python src/cli.py retro
```

Sprint Retrospective

Expected Output

```
=== Sprint 1 Retrospective ===
Planned:  40 pts | Achieved:  35 pts | Rate:  83%
Completed:  5/6 stories | Carry-over:  US06
Blockers:  3 encountered, 2 resolved
-- Team Performance --
Arjun:  16 pts (2 stories) | Sneha:  10 pts (1 story)
Meera:  8 pts (2 stories)  | Rahul:  4 stories tested
-- What Went Well --
+ Auth module on time, UPI ahead of schedule
-- What To Improve --
- Bill payment underestimated, API dependency delays
-- Action Items --
1.  Break stories >8 pts into sub-tasks
2.  Buffer for external dependencies
3.  Technical spikes before planning
```

Retrospectives are time-boxed to 1.5 hours for 2-week sprints. Focus on actionable improvements, not blame. Indian IT teams commonly use the "Start / Stop / Continue" format.

Step 8: Generate Sprint Report

Objective: Generate a professional HTML sprint report with charts, metrics, and recommendations.
Instructions:

1. Run the report generation command with HTML output format
2. Open the generated HTML file in a web browser
3. Review the report sections: summary, burndown chart, velocity metrics, team performance, and recommendations
4. Note how the report can be shared with stakeholders

Commands:

```
python src/cli.py report \
  --format html --output sprint_report.html

# Open in browser (use start/open/xdg-open)
start sprint_report.html
```

Generate Sprint Report

Expected Output

```
Generating:  summary...metrics...burndown...velocity...
team performance...story details...recommendations...
Report generated:  sprint_report.html (42 KB)
Open in browser to view the formatted report.
```

The HTML report embeds charts as base64 images for single-file sharing. In professional environments, reports are presented during Sprint Reviews and shared on Confluence or Jira.

Screenshots

Screenshot 1

What to capture: Repository structure showing the project files (src/ directory with all Python modules) and the output of `python src/cli.py -help` displaying the five available commands (init, plan, run, retro, report).

Paste your screenshot here

Screenshot 2

What to capture: Sprint planning output showing the selected user stories with their priority levels, story point estimates, and task assignments to team members. The sprint backlog summary should be visible.

Paste your screenshot here

Screenshot 3

What to capture: Terminal Kanban board during sprint execution showing tasks distributed across the three columns: To Do, In Progress, and Done. Include at least one daily standup summary with blocker information.

Paste your screenshot here

Screenshot 4

What to capture: Burndown chart generated by matplotlib showing both the ideal burndown line (straight diagonal from total points to zero) and the actual burndown line (showing real progress with deviations due to blockers).

Paste your screenshot here

Screenshot 5

What to capture: The generated HTML sprint report opened in a web browser, showing the executive summary section with sprint metrics, embedded burndown chart, and team performance data.

Paste your screenshot here

Conceptual Background

The Agile Manifesto

The Agile Manifesto, published in 2001, established four core values and twelve principles that guide agile software development.

Four Core Values

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

Twelve Principles (Summary)

1. Satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development
3. Deliver working software frequently (weeks rather than months)
4. Business people and developers must work together daily
5. Build projects around motivated individuals with the support they need
6. Face-to-face conversation is the most efficient communication method
7. Working software is the primary measure of progress
8. Promote sustainable development at a constant pace
9. Continuous attention to technical excellence enhances agility
10. Simplicity—maximizing the amount of work not done—is essential
11. Best architectures emerge from self-organizing teams
12. Regular reflection on effectiveness and adjustment of behavior

The Scrum Framework

Scrum is the most widely adopted agile framework, providing a structured approach to iterative and incremental product development.

Scrum Roles: (1) **Product Owner** — maximizes product value, manages backlog, prioritizes stories; (2) **Scrum Master** — ensures framework adherence, facilitates ceremonies, removes impediments; (3) **Development Team** — self-organizing, cross-functional group that delivers the increment.

Scrum Events and Artifacts

Event	Time-box	Purpose
Sprint Planning	4–8 hours	Select stories, create sprint backlog
Daily Standup	15 minutes	Synchronize work, identify blockers
Sprint Review	2–4 hours	Demo to stakeholders, gather feedback
Sprint Retrospective	1.5–3 hours	Reflect and identify improvements

Key Artifacts: *Product Backlog* (ordered list of all needed work, managed by PO), *Sprint Backlog* (items selected for the sprint plus delivery plan), and *Increment* (sum of completed items meeting the Definition of Done).

User Stories

A user story follows the format: *"As a **[role]**, I want **[feature]**, so that **[benefit]**."* Each story must have **acceptance criteria** using Given-When-Then format (e.g., *Given* a user is logged in, *When* the user initiates a UPI transfer, *Then* the amount is debited and recipient credited).

Story Points measure relative effort using scales such as Fibonacci (1, 2, 3, 5, 8, 13, 21 — most popular), T-shirt sizing (XS–XXL), or powers of 2. They account for **complexity**, **uncertainty**, and **effort**.

Velocity and Capacity Planning

Velocity is the total story points completed in a sprint: Velocity $= \sum$ Story Points of Completed Stories

Capacity Planning Steps:

1. Calculate available team hours (members $\times$ hours/day $\times$ sprint days)
2. Account for meetings, holidays, and planned leaves
3. Use historical velocity (average of last 3 sprints) to determine commitment
4. Select stories from the prioritized backlog until velocity is reached

Daily Standups

A 15-minute time-boxed event where each member answers: (1) What did I accomplish yesterday? (2) What will I work on today? (3) Are there any blockers? The Scrum Master facilitates and resolves blockers.

Burndown and Burnup Charts

A **burndown chart** tracks remaining work (story points) against time. The *ideal line* runs diagonally from total committed points to zero; the *actual line* shows real progress. If actual is above ideal, the team is behind schedule; below means ahead.

A **burnup chart** tracks completed work against total scope, making scope changes visible. It shows a *scope line* (total points, may change) and a *work done line* (cumulative completed points).

Kanban: Visual Workflow Management

Kanban boards visualize work items through columns representing workflow stages. Core principles: (1) **Visualize the workflow** using columns (To Do, In Progress, Done); (2) **Limit WIP** to prevent multitasking overhead; (3) **Manage flow** by monitoring cycle time; (4) **Make policies explicit** for column transitions; (5) **Implement feedback loops**; (6) **Improve collaboratively**.

With a WIP limit of 3, no new stories enter "In Progress" until one moves to "Done," preventing overload and improving focus.

Sprint Review and Retrospective

The **Sprint Review** is a meeting where the team demonstrates completed work to stakeholders, collects feedback, and refines the product backlog. The **Sprint Retrospective** focuses on

process improvement. Common formats include: *What went well / What to improve / Action items* (used in this simulator), *Start / Stop / Continue* (popular in Indian IT), *Mad / Sad / Glad*, and the *4Ls* (Liked, Learned, Lacked, Longed For).

Agile in the Indian IT Industry

India is one of the largest Agile adopters globally. Major companies: **TCS** (Agile Vision, SAFe, 50,000+ practitioners), **Infosys** (enterprise Agile@Scale), **Wipro** (Agile Advantage + DevOps), **HCL** (Mode 1-2-3).

Agile in Indian FinTech Startups

Indian FinTech companies heavily rely on Scrum for rapid feature development:

- **Razorpay:** Uses 2-week sprints for payment gateway features. Separate Scrum teams manage UPI, card payments, and lending products, shipping features bi-weekly.
- **PhonePe:** Independent feature teams (payments, insurance, mutual funds) run parallel sprints with cross-team coordination for shared infrastructure.
- **CRED:** Small, focused teams use Kanban for operational work and Scrum for feature development on their credit card rewards platform.
- **Zerodha:** India's largest stock broker uses lean agile practices with a compact team, demonstrating agile scales down as well as up.

Real-World Example: A typical sprint at a payment company: PO prioritizes UPI 2.0 features; team commits 45 points; daily 9:30 AM IST standups; developers integrate NPCI APIs; blockers (e.g., sandbox downtime) escalated by SM; sprint review demos to stakeholders; retrospective feeds action items into next sprint.

Assessment

Assessment Questions

Answer the following questions in your submission:

Q1. Explain the four core values of the Agile Manifesto. How do they differ from waterfall?

Q2. Describe the three Scrum roles and their key responsibilities.

Q3. Write a user story for "mobile check deposit" with three Given-When-Then acceptance criteria.

Q4. Team velocity is 35 pts/sprint. Backlog has 180 pts. How many 2-week sprints to complete? What assumptions?

Q5. Analyze your burndown chart. Where did progress deviate from ideal? What caused it?

Q6. Compare Kanban and Scrum. When would you choose one over the other?

Q7. Team over-commits by 15–20% each sprint. What actions improve estimation accuracy?

Q8. How do TCS, Infosys, Wipro adapt Agile for 100+ member projects? Describe SAFe.

Deliverables

Submit: (1) All 5 screenshots, (2) Written answers to 8 questions, (3) Generated `sprint_report.html`, (4) Burndown chart (`output/burndown.png`), (5) Reflection (200–300 words).

Verification Checklist

- ☐ Repo cloned, venv created, dependencies installed, CLI help verified
- ☐ Project initialized with 20 stories and 5-member team
- ☐ Sprint planning completed with velocity-based selection
- ☐ Sprint simulation run with standups and Kanban board
- ☐ Burndown chart generated (ideal vs. actual lines)
- ☐ Retrospective completed with metrics and action items
- ☐ HTML sprint report generated and viewable in browser
- ☐ All 5 screenshots captured and assessment questions answered

Grading Rubric

Criteria	Description	Points	Score
Environment Setup	Repo cloned, venv created, deps installed	10	___/10
Project Init	Backlog loaded, team configured correctly	10	___/10
Sprint Planning	Stories selected by priority, assignments made	15	___/15
Sprint Execution	Simulation run, standups observed, Kanban shown	15	___/15
Burndown Chart	Chart generated, ideal vs. actual lines visible	10	___/10
Retrospective	Metrics reviewed, improvements identified	10	___/10
Sprint Report	HTML report generated with all sections	10	___/10
Screenshots	All 5 screenshots submitted clearly	10	___/10
Assessment Answers	All 8 questions answered thoughtfully	10	___/10
	TOTAL	**100**	___/100

Appendix A: Scrum Events Quick Reference

Event	Time-box	Participants	Output
Sprint Planning	4–8 hrs	Scrum Team	Sprint Backlog
Daily Standup	15 min	Dev Team, SM	Updated daily plan
Sprint Review	2–4 hrs	Team + Stakeholders	Feedback, revised backlog
Retrospective	1.5–3 hrs	Scrum Team only	Action items
Refinement	≤10% sprint	PO + Dev Team	Estimated stories

Sprint Duration: 1 week (startups), **2 weeks** (most common, used here), 3 weeks (complex projects), 4 weeks (maximum recommended).

Appendix B: User Story Template and Examples

User Story Template and Example

```
# TEMPLATE:
# As a [role], I want [feature], so that [benefit].
# Acceptance: Given [precond] When [action] Then [result]
# Definition of Done: reviewed, tested, documented, deployed

# EXAMPLE (Mobile Banking):
Title: UPI Payment Integration
ID: US-003  |  Priority: HIGH  |  Points: 8

As a bank customer,
I want to send money via UPI using a VPA,
So that I can make instant payments.

Acceptance Criteria:
1. Given user is logged in with linked bank account,
   When user enters valid VPA and amount,
   Then payment is processed within 30 seconds.
2. Given recipient VPA is invalid,
   When user attempts to send money,
   Then error is shown and no money is debited.
```

User Story Template and Example

Appendix C: Estimation Techniques

Planning Poker

The most widely used agile estimation technique. Each team member holds Fibonacci-valued cards (1, 2, 3, 5, 8, 13, 21). After the Product Owner reads a story, members privately select a card. All cards are revealed simultaneously. If estimates vary significantly, the outliers explain their reasoning, and the team re-estimates until consensus.

Why Fibonacci? Increasing gaps reflect growing uncertainty in larger estimates.

T-shirt Sizing

Useful for initial rough estimation: **XS** (1 pt, trivial), **S** (2–3 pts, small feature), **M** (5 pts, some complexity), **L** (8 pts, cross-component), **XL** (13 pts, significant uncertainty), **XXL** (21+ pts, epic-level — should be broken down).

Stories at 13 points or above should be broken down before sprint planning. Large stories carry high risk of not completing within a single sprint.

Affinity Mapping

Affinity mapping is useful for estimating a large backlog quickly. Write each story on a card, place the first as a reference, and position subsequent stories to the left (smaller) or right (larger). Once all stories are placed, assign Fibonacci values to the groups.

Appendix D: Agile Glossary

Term	Definition
Backlog	An ordered list of work items to be completed
Burndown Chart	A chart showing remaining work versus time in a sprint
Capacity	The total available working hours of a team in a sprint
Carry-over	Stories not completed in one sprint that move to the next
Daily Standup	A 15-minute daily meeting for team synchronization
Definition of Done	Criteria that must be met before a story is considered complete
Epic	A large body of work broken down into multiple user stories
Impediment	Any obstacle preventing the team from making progress
Increment	The sum of all completed backlog items during a sprint
Kanban Board	A visual tool for managing work as it flows through stages
Product Owner	Person responsible for maximizing the value delivered by the team
Scrum Master	Servant-leader who facilitates Scrum events and removes impediments
Sprint	A time-boxed iteration (typically 1–4 weeks) producing a shippable increment
Story Points	A unit of measure for the relative effort of a user story
User Story	A description of a feature from the user's perspective
Velocity	The total story points completed by a team in a sprint
WIP Limit	A constraint on work items in a particular workflow stage

Appendix E: Troubleshooting Guide

Common Issues and Solutions

Problem: "python: command not found" or "pip is not recognized."
Solutions: (1) Try `python3` / `pip3` instead; (2) Ensure Python is in PATH; (3) On Windows, reinstall with "Add to PATH" checked; (4) Verify venv is activated (`(venv)` in prompt).

Problem: `TclError: no display name and no $DISPLAY environment variable`
Solutions: (1) Set backend: `export MPLBACKEND=Agg`; (2) On WSL, install VcXsrv and set `DISPLAY=:0`; (3) Chart still saves to `output/burndown.png` without display.

Problem: Kanban board appears garbled with broken characters.
Solutions: (1) Use modern terminal (Windows Terminal, iTerm2); (2) Set encoding: `chcp 65001` (Windows) or `export LANG=en_US.UTF-8`; (3) Widen terminal to 120+ columns; (4) Update Rich: `pip install -upgrade rich`.

Appendix F: Additional Resources and Tools

Documentation and Reading

→ Scrum Guide: `https://scrumguides.org`

→ Agile Manifesto: `https://agilemanifesto.org`

→ Atlassian Agile Coach: `https://www.atlassian.com/agile`

→ SAFe Framework: `https://scaledagileframework.com`

Books: "Scrum" (Sutherland), "User Stories Applied" (Cohn), "Agile Estimating and Planning" (Cohn), "The Phoenix Project" (Kim et al.)

Agile Tools Comparison

Tool	Best For	Free Tier	Notes
Jira	Enterprise Scrum/Kanban	✓ (10 users)	Industry standard
Trello	Simple Kanban boards	✓	Visual, easy to start
Azure DevOps	Microsoft ecosystem	✓ (5 users)	Integrated CI/CD
GitHub Projects	Open-source projects	✓	Tight Git integration
ClickUp	All-in-one workspace	✓	Feature-rich free tier

Tools Used in This Practical

Tool	Purpose	Cost
Python 3.8+	Core programming language	Free (open source)
Click	Command-line interface framework	Free (open source)
Rich	Terminal formatting and Kanban display	Free (open source)
matplotlib	Burndown chart generation	Free (open source)
pip	Python package manager	Free (included)
Git	Version control and repo cloning	Free (open source)
Web Browser	Viewing HTML sprint reports	Free

—END OF LAB MANUAL—

Document Version: 1.0

IT Management & Audits – Practical Lab Series