PROJECT ABSTRACT

Scrum Sprint Planner using Django

A Django web application for agile teams: a kanban sprint board with WIP limits, burndown charts against the ideal line, velocity tracking, in-app planning poker, auto-generated standup summaries and team capacity views — delivered as a built-to-order student project with code, tests, report, PPT and viva kit.

01. Title

Scrum Sprint Planner using Django

02. Introduction / Background

Student project teams typically coordinate over chat messages and discover in the final month that half the planned work was never started. Professional software teams avoid this with Scrum: fixed-length sprints, a visible task board, and a burndown chart that makes slippage impossible to ignore. Building a Scrum planner is a strong full-stack project because it combines data modeling (projects, sprints, stories, estimates), real-time UI (the board), and small but real algorithms (burndown snapshots, velocity averages, WIP enforcement). This project implements the complete loop in Django 4.2 with a DRF API layer.

03. Problem Statement

Agile student and small teams lack an affordable, self-hosted tool that covers the whole sprint loop — estimation, planning, daily tracking and review — instead of just a bare task list. The project addresses this by building a Django application where stories are estimated with in-app planning poker, committed into time-boxed sprints, tracked on a WIP-limited kanban board, and reviewed through burndown charts, velocity history and auto-generated standup summaries.

04. Objectives

- Model projects, sprints, epics, stories, estimates and team capacity in Django with role-based access (Scrum Master, Product Owner, Developer).
- Build a five-column kanban board (Backlog, To-Do, In Progress, In Review, Done) with drag-and-drop and server-enforced WIP limits.
- Compute burndown charts from daily remaining-point snapshots, plotted against the ideal line with mid-sprint scope-change annotations.
- Track velocity across completed sprints and use the rolling average as the next sprint's commitment guide.

- Implement in-app planning poker (Fibonacci deck, simultaneous reveal, averaged estimates written back to the story).
- Generate daily standup summaries from card movements and blockers; add a pytest suite for the sprint math and permissions, plus the complete student kit.

05. Proposed Methodology

The methodology has four stages. (1) Data model: Django models for Project, Sprint, Epic, Story, Estimate and TeamMember with the state machine governing story transitions. (2) Board: DRF endpoints for card moves with WIP-limit enforcement in the view layer, plus a Redis-channels WebSocket layer for live updates with a polling fallback. (3) Analytics: a nightly snapshot job records remaining points per active sprint; burndown and velocity are computed server-side and rendered as SVG charts. (4) Validation: a pytest suite covers burndown math, velocity averaging, WIP enforcement and role permissions; the demo is seeded with a realistic sprint (Sprint 7, campus portal, 21 dev-team cards).

06. System Architecture / Working

The system flows as: browser board -> DRF API -> Django services (sprint engine, estimation, snapshots) -> PostgreSQL; Redis channels push board updates to connected clients. A scheduled job snapshots remaining story points nightly for the burndown. Planning poker runs as a live session object with reveal-on-all-voted semantics. The standup summary is a rules-based aggregation over the last 24 hours of card events. All sprint math lives in tested service functions, not in views, so the report can present the algorithms independently of the UI.

07. Hardware / Software Requirements

- Hardware: any machine that runs Python; 2 GB RAM suffices for the demo seed.
- Software: Python 3.10, Django 4.2, DRF, PostgreSQL, Redis (optional — polling fallback without it); a modern browser.
- Seed data script included for the demo sprint.

08. Expected Outcomes

A working planner where a full sprint can be run end to end: stories estimated, sprint committed, board moved daily, burndown tracking the ideal line, velocity computed, and standup summaries generated. The pytest suite passes on the student's machine; the demo walkthrough shows Sprint 7 at 21 of 34 points on day 9. The report documents the data model, the sprint algorithms and the test results.

09. Applications

Student final-year project teams, small startup teams wanting a self-hosted agile tool, college software-engineering labs teaching Scrum by doing, and as a portfolio piece demonstrating full-stack Django plus real-time features.

10. Future Scope

Multi-project portfolio rollups, cumulative-flow diagrams, Monte-Carlo sprint forecasting, GitHub/GitLab integration linking commits to stories, a native mobile companion app, and LLM-assisted story splitting (kept clearly separate from the rules-based standup summary).