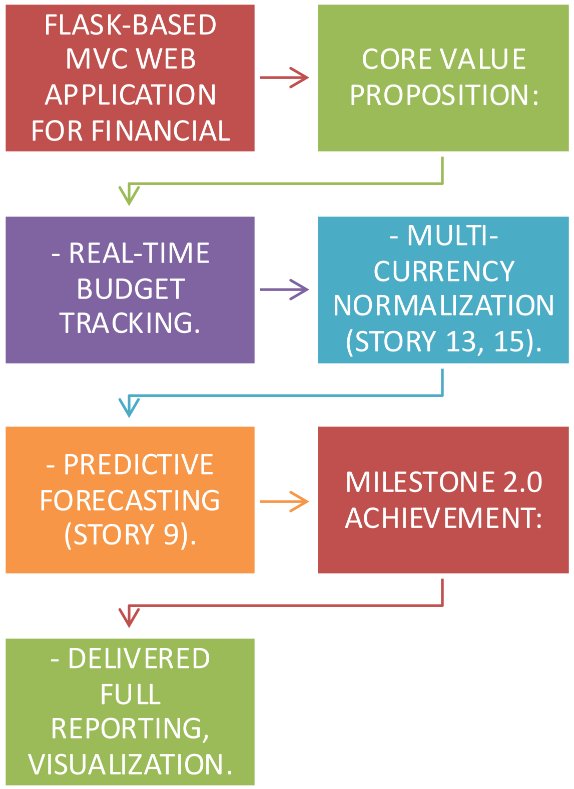
Paldea Finance Tracker Milestone 2.0 Final Presentation

IST 303 – Fall 2025

Team Paldea:
Qiao Huang, Samantha Aguirre, Gerves F. Baniakina, Rachan Sailamai, Manish Shrivastav

Project Overview & Objectives


FLASK-BASED MVC WEB APPLICATION FOR FINANCIAL
CORE VALUE PROPOSITION:
- REAL-TIME BUDGET TRACKING.
- MULTI-CURRENCY NORMALIZATION (STORY 13, 15).
- PREDICTIVE FORECASTING (STORY 9).
MILESTONE 2.0 ACHIEVEMENT:
- DELIVERED FULL REPORTING, VISUALIZATION.

Technical Architecture

Tech Stack: Python (Flask), SQLAlchemy (ORM), SQLite, Bootstrap 5.

Key Models (models.py):

- User: Handles authentication and currency prefs.

- Transaction: Polymorphic design for Income/Expense.

- ScheduledReport: Manages automated background tasks.

Code Deep Dive: Currency Logic

```
• @paldea_app.route('/convert_currency', methods=['POST'])
@login_required
def convert_currency():
    # ... setup ...
    # STRATEGY: Try primary API, fallback to secondary, fallback to 1:1
    try:
        url = f"https://api.exchangerate-
api.com/v4/latest/{from_currency}"
        response = requests.get(url, timeout=5)
        if response.status_code == 200:
            rates = response.json().get('rates', {})
            rate = rates.get(to_currency)
    except Exception as e:
        print(f"API 1 failed: {e}")
        # ... fallback logic ...

    converted_amount = amount * rate
    # ... save to ConversionHistory ...
```

Epic 4: Enhanced Visualization

Objective: Provide actionable insights (User Story 2, 5).

Implementation:
-Compute per-category spending + monthly totals in Flask routes.
-Render charts in templates using Chart.js.
-Progress bar color driven by budget-usage %

- Dynamic Spending Calculation in 'home' route.

- Visual Feedback: Logic determines progress bar colors.

- Green < 70% | Yellow 70-90% | Red > 90%.

Presenter: Samantha Aguirre

Code Deep Dive: AI Forecasting

```
• from sklearn.linear_model import LinearRegression

# STRATEGY: Prepare historical vectors for regression model
if len(expenses_for_forecast) >= 2:
    X = np.array(range(len(expenses_for_forecast))).reshape(-1, 1)
    y = np.array(expenses_for_forecast)

    model = LinearRegression()
    model.fit(X, y) # Train model on last 6 months

    # Predict next month (index = length of array)
    next_month_prediction =
model.predict([[len(expenses_for_forecast)]])[0]

    # Calculate confidence based on R-squared score
    forecast_confidence = min(100, max(0, 100 - abs(model.score(X,
y) * 100 - 100)))
```

Epic 5: Export & Reporting

Objective: Data portability and compliance (Story 10, 12).

Features: Includes PDF and CSV export plus a tax-deduction summary feature.

PDF Generation: Programmatic drawing via ReportLab.

CSV Export: In-memory buffer handling.

Tax Summary: Filtering for deductible categories.

Presenter: Gervas Francois Banjakina

Code Deep Dive: PDF Engine

```
from reportlab.platypus import SimpleDocTemplate, Table, Image

@paldea_app.route('/export_pdf')
def export_pdf():
    buffer = BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=letter)

    # STRATEGY: Generate charts dynamically using Matplotlib
    with tempfile.TemporaryDirectory() as tmpdir:
        fig, ax = plt.subplots(figsize=(6, 4))
        ax.bar(['Income', 'Expenses'], [total_income, total_expenses])
        plt.savefig(chart1_path)

        # Embed chart into PDF flowable
        story.append(Image(chart1_path, width=5*inch, height=3.33*inch))

    doc.build(story) # Build PDF in memory
```

Agile Methodology & Process

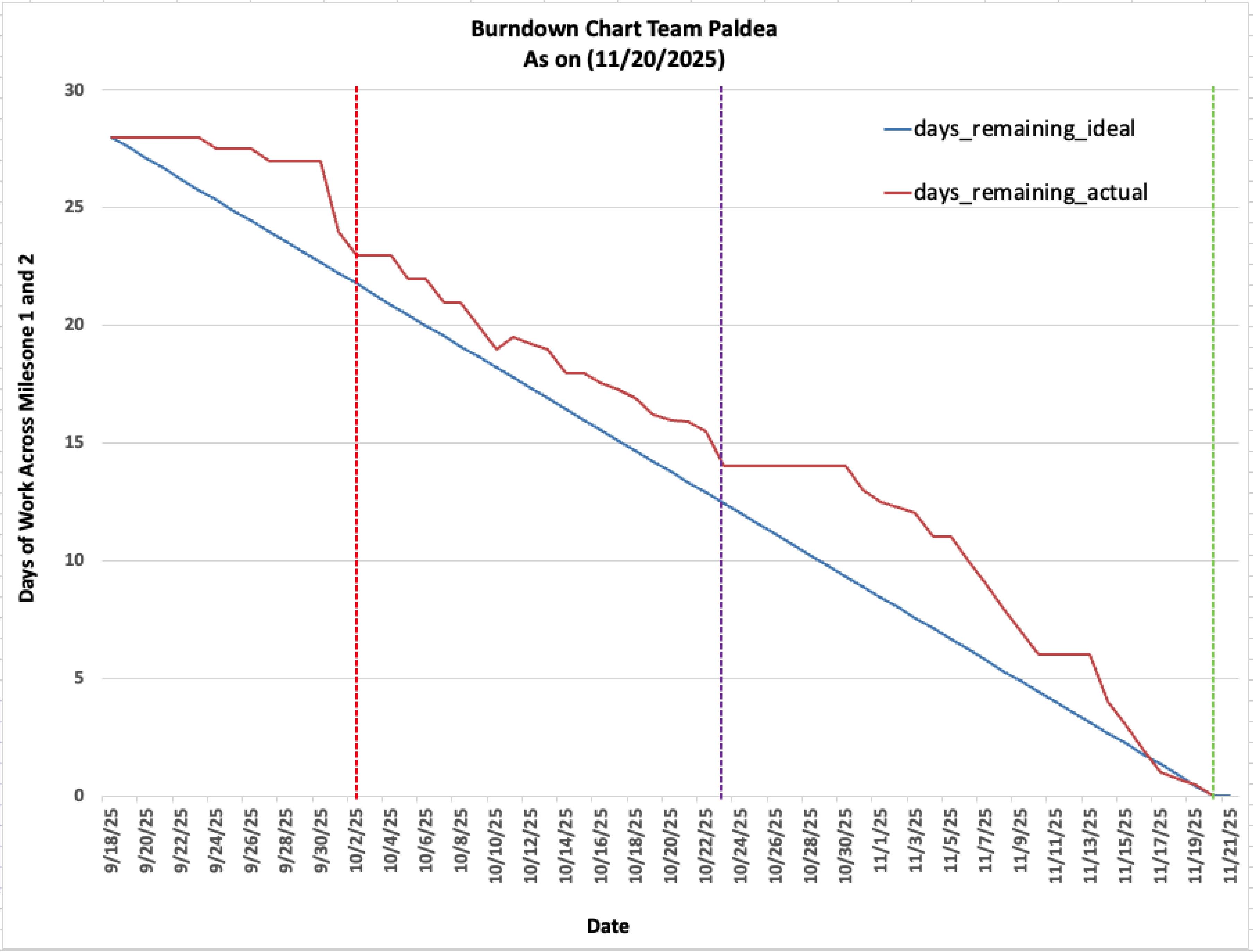
Process Overview

- Structured the project around short, iterative sprints aligned with milestone deadlines.
- Converted high-level user stories into granular engineering tasks for clearer ownership and estimation.

Velocity & Tracking

- Maintained daily burndown tracking, enabling early detection of delays and unbalanced workload.
- Velocity varied initially due to onboarding and architecture setup, then normalized starting Sprint 3 as responsibilities became more defined.

Presenter: Rachan Sailamai


Burndown Chart Team Paldea
As on (11/20/2025)
days_remaining_ideal
days_remaining_actual
Days of Work Across Milesone 1 and 2
Date
30
25
20
15
10
5
0
9/18/25
9/20/25
9/22/25
9/24/25
9/26/25
9/28/25
9/30/25
10/2/25
10/4/25
10/6/25
10/8/25
10/10/25
10/12/25
10/14/25
10/16/25
10/18/25
10/20/25
10/22/25
10/24/25
10/26/25
10/28/25
10/30/25
11/1/25
11/3/25
11/5/25
11/7/25
11/9/25
11/11/25
11/13/25
11/15/25
11/17/25
11/19/25
11/21/25

Tests Summary

Item	Result
Test framework	pytest + pytest-cov
Total test files	5
Total tests run	8
Tests passed	8
Tests failed	0
Overall execution time	~4.5 seconds
Third-party warnings	2 DeprecationWarnings (pyasn1)

⚠️ **Warnings**

There are two **DeprecationWarning** messages from pyasn1. These come from a third-party library (not your code) and are **not test failures**.

Tests Summary

- Code Coverage by Module

Module / File	Statements	Missed	Coverage
`my_paldea/config.py`	16	0	**100%**
`my_paldea/paldea_app/models.py`	116	9	**92%**
`my_paldea/__init__.py`	46	8	**83%**
`my_paldea/utlities.py`	10	3	**70%**
`my_paldea/paldea_app/views.py`	546	286	**48%**
`my_paldea/finSystem.py`	76	69	**9%**
TOTAL	**810**	**375**	**55%**

Coverage summary

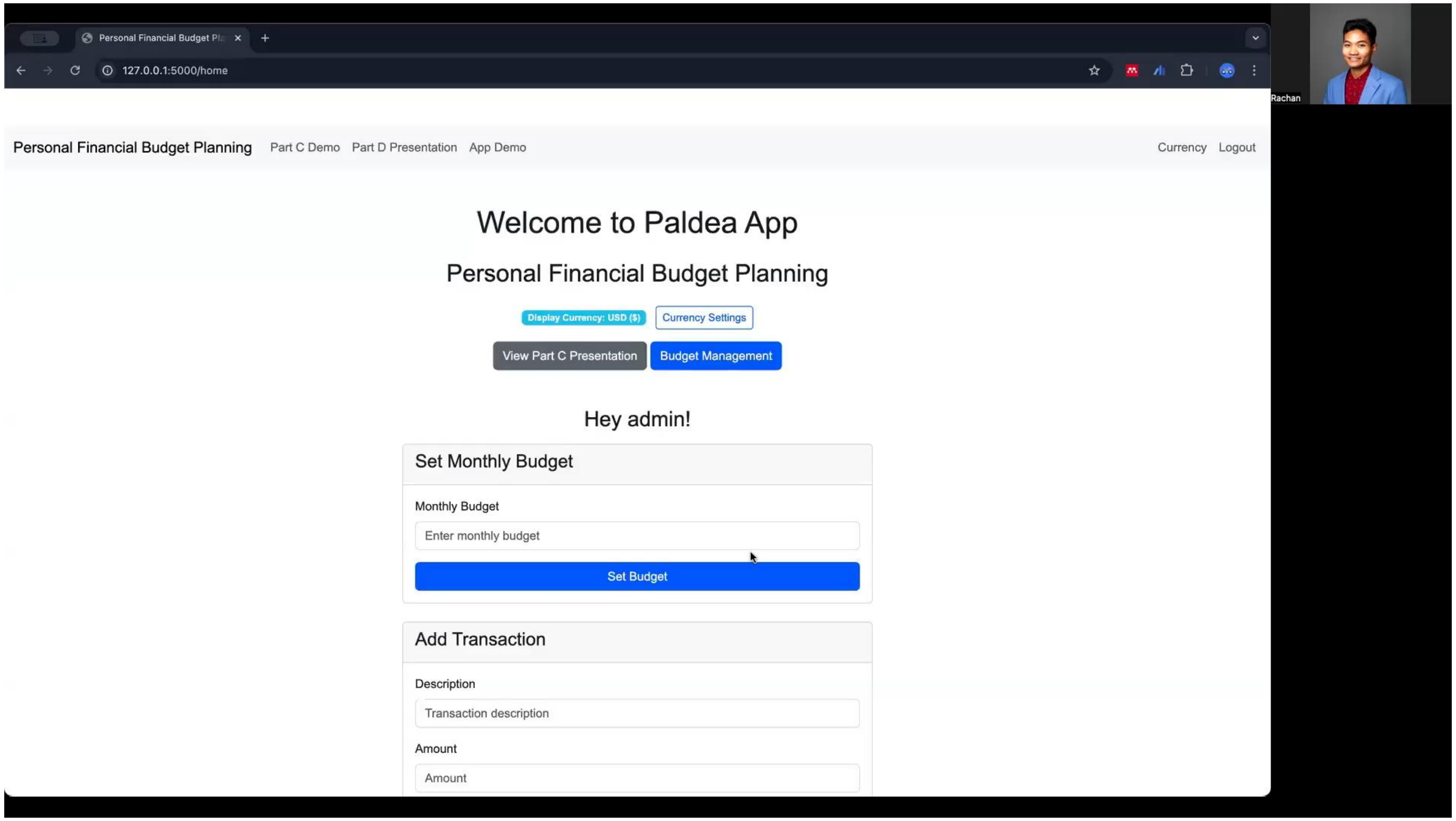
Overall:

- **Total statements:** 810
- **Missed statements:** 375
- **Overall coverage: 55%**

Per file:

- config.py – 100%
- models.py – 92%
- __init__.py – 83%
- views.py – 48%
- utlities.py – 70%
- finSystem.py – 9%

Presenter: Manish Shrivastav



Welcome to Paldea App

Personal Financial Budget Planning

Display Currency: USD ($)

Hey admin!

Set Monthly Budget

Monthly Budget: Enter monthly budget

Add Transaction

Description: Transaction description

Amount: Amount

App Demo

Key Successes

- Completed full MVC finance tracking system
- Delivered forecasting, visualization, and reporting on time
- Improved stability and modular architecture
- Strong team collaboration and communication

Key Failures & Challenges

- Early delays due to API and environment failures

- Integration of forecasting & PDF generation required extra debugging, it broke down today again

- Agile velocity inconsistent in early sprints

- UI/UX improvements limited by time constraints

Lessons Learned

- IMPORTANCE OF CLEAR COMMUNICATION IN DISTRIBUTED TEAMS

- BREAKING USER STORIES INTO SMALLER TASKS STABILIZES SPRINT VELOCITY

- EARLY TESTING PREVENTS CRITICAL ROADBLOCKS LATER

- DESIGNING FOR API FAILURE INCREASES SYSTEM RELIABILITY