📊 Stock Cards - Project Summary

Author: Alikhan Abzhanov
Course: INFO 390: Commercial Web App Dev 1
Date: December 2025
GitHub: github.com/AlikhanIllini/Final_Project_Alikhan_alikhan4
Deployed App: Render.com

Project Overview

Stock Cards is a web application for organizing and tracking U.S. stocks using a card-based interface. Each stock is represented as a card that stores core market data along with user-managed information such as notes, tags, and saved filters. The app is designed for clarity and structure, not real-time trading or advanced analysis.

Login Credentials

> **Internal Tester:** mohitg2 / graingerlibrary
> **Internal Guest:** infoadmins / uiucinfo

Topics & Features Attempted

Core Django Concepts

Topic	Implementation
Models & ORM	Stock, StockCard, Tag, PriceSnapshot, SavedFilter models with relationships
Views & Templates	Class-based and function-based views with template inheritance
Forms	ModelForms for card creation, tag management, manual price entry
User Authentication	Registration, login, logout with @login_required decorator

URL Routing	Namespaced URLs with RESTful patterns
Admin Customization	Custom admin views for all models
Static Files	CSS styling and JavaScript for interactivity
Migrations	Database schema evolution

Bonus Features Implemented

1. **External API Integration** - Real-time stock prices via Yahoo Finance (yfinance) with 15-minute caching
2. **Weekly Email Digest** - Django management command for automated email summaries
3. **Color-Coded Tags** - Custom tag system with user-selectable colors
4. **Saved Filters** - Persistent filter combinations users can save and reuse
5. **Price History Tracking** - Historical snapshots with 7-day and 30-day change calculations
6. **Archive System** - Soft-delete functionality without losing data
7. **Priority Levels** - High/Medium/Low classification for cards
8. **Manual Price Fallback** - Graceful degradation when API is unavailable
9. **Responsive Dashboard** - Mobile-friendly card-based UI
10. **Multi-User Support** - Complete user isolation with per-user data scoping

Technology Stack

Python 3.14 Django 5.2.6 SQLite yfinance API HTML5 CSS3 JavaScript Git/GitHub Render (Deployment)

Data Models

Model	Purpose	Key Fields
Stock	Store ticker symbols	ticker, company_name, sector
StockCard	User's stock tracking card	user, stock, notes, priority, target_price, is_archived
Tag	Custom categorization	user, name, color
PriceSnapshot	Historical price data	stock, price, volume, source, timestamp
SavedFilter	Saved filter combinations	user, name, filter_config, is_default

Project Structure

```
Final_Project_Alikhan_alikhan4/
├── appserver/          # Django project settings
├── cards/              # Main application
│   ├── models.py       # Data models
│   ├── views.py        # View logic
│   ├── forms.py        # Form definitions
│   ├── price_adapter.py # Stock API integration
│   ├── templates/      # HTML templates
│   ├── static/         # CSS/JS files
│   └── management/     # Custom commands
├── docs/               # Documentation
├── manage.py           # Django management
└── requirements.txt    # Dependencies
```

Key Learnings

- Django's MTV architecture provides excellent separation of concerns
- External API integration requires robust error handling and fallback mechanisms
- User experience matters as much as functionality
- Proper data modeling upfront saves significant refactoring time

Challenges & Solutions

Challenge	Solution
API Rate Limiting	Implemented 15-minute caching + manual price fallback
Multi-User Data Isolation	Careful queryset filtering in all views
Email Configuration	Console backend for development, SMTP ready for production

Future Improvements

- Real-time WebSocket updates for live price streaming
- Portfolio analytics with charts (Chart.js/D3.js)
- Export functionality (CSV/PDF)
- Additional data sources (news, analyst ratings)
- Mobile app using Django REST Framework