# ASTRA - Topics Implementation Summary

## Basic Info

- Project Name: Astra - Women's Empowerment Platform
- GitHub: https://github.com/SwarnikaBhardwaj/Astra.git
- Deployed URL:

---

## TABLE 1: FOUNDATIONAL TOPICS (15 points)

These are the must-haves. Everyone has to do these.

### 1. GitHub Repository, Environment Setup & Project Structure (3 pts)

What I did:

- Made a Django project called "Astra" using Conda environment
- Set up two apps: core (for main stuff) and api (for JSON data)
- Configured all the settings properly
- Will push to GitHub (doing that in next steps)

Where to find it: Just look at the project structure, settings.py file

### 2. Wireframes and UI Planning (2 pts)

What I did:

- Made a wireframes document explaining the design
- Took screenshots of every major page
- Wrote down user flows (how people navigate the site)
- Explained why I chose the teal/white theme

Where to find it: docs/wireframes.md, docs/screenshots folder

### 3. Models and Database (3 pts)

What I did:

- Created 7 models: UserProfile, Hub, Post, Comment, MentorshipRequest, Skill, Badge, HelpfulVote
- Set up relationships between them (like how posts connect to hubs)
- Ran all migrations
- Set up the admin panel so you can see everything
- Added a bunch of sample data (100+ posts, 50+ users, 7 hubs)

Where to find it: core/models.py, core/admin.py

## 4. Views, Templates, and URLs (3 pts)

What I did:

- Made over 20 different views (the code that handles page requests)
- Used template inheritance so I don't repeat HTML code
- Set up all the URL routes properly
- Used Django template tags throughout

Where to find it: core/views.py, core/urls.py, templates folder

## 5. User Authentication (2 pts)

What I did:

- Used Django's built-in login system
- Added login required decorators so you can't see stuff unless you're logged in
- Made staff-only pages for the analytics dashboard
- Set up the test accounts: mohitg2/graingerlibrary and infoadmins/uiucinfo

Where to find it: core/views.py (look for @login_required and @staff_member_required)

## 6. Deployment (2 pts)

What I did:

- [Will fill this in after deploying]
- Set up production settings
- Got static files working
- Deployed to [platform name]

Where to find it: Deployed URL

---

# TABLE 2: ADD-ON TOPICS (15 points minimum, I did 21)

You only need 5 of these, but I did 7 because why not.

## 1. ORM Queries and Summaries (3 pts)

What I did:

- Used Django's query optimization (select_related and prefetch_related)
- Made aggregation queries to count stuff (like how many posts per hub)
- Created methods in models to calculate stats on the fly
- Built the whole analytics system using these queries

Where to find it: Check the models.py methods like member_count() and post_count(), also the analytics view

## 2. Static Files and CSS (3 pts)

What I did:

- Integrated Bootstrap 5
- Wrote custom CSS for the dark red/gold theme
- Set up static files properly in settings
- Added Google Fonts for better typography

Where to find it: base.html style section, settings.py STATIC_URL stuff

## 3. Vega-Lite Visualizations (3 pts)

What I did:

- Made 3 interactive charts on the analytics page:
  - Bar chart showing posts per hub
  - Horizontal bar chart for top skills
  - Pie chart for mentorship request statuses
- They're all interactive (hover to see details)
- Styled them to match the dark theme

Where to find it: analytics_dashboard.html, all the Vega-Lite JavaScript code

## 4. Forms and CRUD Operations (3 pts)

What I did:

- Created 5 different Django forms
- Full CRUD (Create, Read, Update, Delete):
  - Create: new posts, comments, user accounts, mentorship requests
  - Read: all the list and detail pages
  - Update: edit posts and profiles
  - Delete: delete posts (with a confirmation page so you don't accidentally delete)
- All forms have validation

Where to find it: core/forms.py, and views like post_create, post_edit, post_delete

## 5. JSON API Endpoints (3 pts)

What I did:

- Made 4 API endpoints that return JSON:
  - /api/hubs/ - all hubs with stats
  - /api/posts// - posts for a specific hub
  - /api/stats/platform/ - overall platform stats
  - /api/stats/skills/ - skills data
- These could be used for a mobile app later

Where to find it: api/views.py, api/urls.py

## 6. Data Export (3 pts)

What I did:

- Made a CSV export feature for user portfolios
- Shows all your posts with stats
- Download button on profile page
- Uses Python's csv module

Where to find it: export_portfolio view in core/views.py

## 7. Public User Authentication (3 pts)

What I did:

- Anyone can sign up (not just staff)
- Custom signup form that also creates a profile
- Auto-login after you sign up
- Different permission levels (regular users vs mentors vs staff)

Where to find it: signup view, SignUpForm in forms.py

---

# BONUS TOPICS (Extra credit)

## 8. Matplotlib Chart (3 pts)

What I did:

- Added another chart using Matplotlib (different from Vega-Lite)
- Shows post activity over the last 7 days as a line graph
- Saved as an image and displayed on analytics page
- Also styled to match the dark theme

Where to find it: generate_matplotlib_chart function in core/views.py

---

# OTHER STUFF I BUILT (not required but cool)

- Complete mentorship system where people can request mentors
- Voting system where you can mark posts as helpful
- Anonymous posting option
- Join/leave hubs to customize your feed
- Skills and badges on profiles

---

# TEST ACCOUNTS

Staff account (has full access):

- Username: mohitg2
- Password: graingerlibrary

Regular user:

- Username: infoadmins
- Password: uiucinfo

Other accounts (all use password: demo1234):

- sarah_mentor
- alex_learner
- priya_dev

## PROJECT STRUCTURE

The project is organized like this:

- Astra folder has all the settings
- core app has the main functionality
- api app has the JSON endpoints
- templates folder has all the HTML
- docs folder has all the documentation
- manage.py is what you use to run commands

---

## CODE QUALITY STUFF

- Used consistent naming throughout
- Added comments where code might be confusing
- Used Django's security features (CSRF tokens, login required, etc.)
- Optimized database queries so pages load fast
- Made it mobile-responsive with Bootstrap

---

This project basically shows I can build a full Django app from scratch now! I tried to make it look good, and deploy it. All the features actually work!