Equity Award Tracker - Demo Presentation Script

1. Introduction

Good [morning/afternoon], today I'm presenting my full-stack project called Equity Award Tracker. This project is built using Angular 19 for the frontend and .NET 8 Web API for the backend. It is designed for employees to securely log in and view their equity awards, share breakdown, and real-time stock performance, all in one place.

2. Project Structure Overview

The app has a complete authentication system using JWT tokens, a responsive dashboard with dynamic info cards, charts, tables, and a live stock page. The data for users and equity is currently stored in JSON files for simplicity, but the architecture is ready to scale with a database if needed.

3. Login and Register (Authentication Flow)

The login and registration module uses a JWT-based system. I've stored credentials in a JSON file named users.json. When a user logs in, a token is generated by the backend using the user's email and password, and that token is stored in the browser's localStorage.

All protected routes are guarded by Angular's AuthGuard, which checks for token presence and validity. On the backend, I've added JWT validation middleware to intercept unauthorized requests. This way, the dashboard is only accessible after a successful login.

4. Dashboard Walkthrough

Once logged in, the user is taken to a personalized dashboard. The layout has the following components:

a. Navbar

The navbar contains:

- A centered logo using Bootstrap's grid and flex utilities.
- A notification bell for future implementation of reminders or messages.
- A theme switch to toggle between dark and light mode-this is achieved by toggling a CSS class on the <body> tag.
- And a Logout button, which clears the token and redirects to the login page.

b. Info Cards

Below the navbar, we have four key info cards:

- Total Shares: Sum of all shares awarded.
- Vested Shares: Shares that the employee already owns.
- Unvested Shares: The remaining shares.
- Current Value: Live market value calculated by multiplying total shares with current stock price.

These values are dynamically fetched from the equity data JSON file and calculated on component initialization.

c. Live Price Fetching

For the live stock price, I used the Alpha Vantage API. The backend hits this API and returns clean stock data to the frontend, keeping the API key hidden and secure. The stock value is refreshed each time the dashboard loads or when the live stock page is visited.

d. Pie Chart

I integrated ng2-charts to visualize the split between vested and unvested shares using a Pie Chart. The chart updates automatically based on the processed equity data and also adjusts based on theme changes to maintain consistency.

e. Awards Table

Below the chart, there's a dynamic table that lists all awards:

- Columns include shareType, totalShares, vestedShares, and vestingDate.

- I implemented filters so that clicking on any info card will filter this table.

The filtering is done using simple Angular logic inside the component using flags and array filtering.

f. Line Chart (Performance)

To visualize the award distribution or trends, I also added a Line Chart. It updates based on filtered data and shows insights into how shares are allocated over time or by type.

5. Live Stock Page

Apart from the dashboard, I've created a dedicated Live Stock Page. It fetches and displays live price, daily change, and percentage change using Alpha Vantage.

This page can be extended later to show historical price trends, which is very useful for employees tracking company stock performance.

6. Security Layer

Security is a key part of this project:

- Frontend Interceptor: I've implemented an HTTP interceptor to automatically attach the JWT token to all requests. If the token is expired or tampered, the interceptor logs the user out immediately.

- Backend Middleware: Every protected endpoint uses [Authorize] with JWT validation middleware in .NET, which ensures token authenticity.

7. Scalability & Future Plans

Right now, data is stored in static JSON files for simplicity and faster development. But the backend is modular and ready to connect with a SQL Server database via Entity Framework Core.

Future enhancements can include:

- Admin panel for award management

- Notifications

- Email triggers on vesting events

- Historical stock data graphs

8. Technologies Used

Layer	Tech Stack
Frontend	Angular 19, Chart.js, Bootstrap
Backend	.NET 8 Web API, JWT Auth
Data	JSON files for users and equity
Charting	ng2-charts (Chart.js)
Styling	Bootstrap + custom CSS
API Used	Alpha Vantage for live prices

9. Challenges Faced & Solutions

Some of the key challenges I faced:

- Token security: Ensured proper frontend-backend token validation using Interceptor and middleware.

- Live API rate limits: Handled using throttling on backend and efficient caching logic if needed.

- Chart updates: Ensured seamless updates of chart data when filters were applied.

10. Conclusion

This project gave me a solid hands-on experience in full-stack development, especially in integrating authentication, handling live APIs, and building a responsive UI with charts and tables.

I've kept the architecture modular and scalable, so more features like admin controls or database support can be easily integrated in the next phase.